compAs
Grupo de capacitación e investigación pedagógica

UNIVERSIDAD TÉCNICA ESTATAL DE QUEVEDO
1984
ECUADOR

Tecnologías para el desarrollo de APLICACIONES WEB

Autor: **Gleiston Cicerón Guerrero Ulloa**

BIOGRAFÍA DEL AUTOR

Gleiston C. Guerrero Ulloa

Ingeniero en Computación por la Escuela Superior Politécnica del Litoral (ESPOL).

Master en Dirección de Negocios por el Instituto Superior de Educación, Administración y Desarrollo (ISEAD) y la Universidad Complutense de Madrid, Máster en Desarrollo de Software por la Universidad de Granada.
Estudiante de doctorado en Tecnologías de la Información y la Comunicación con el tema de Tesis "Metodología de desarrollo iterativo guiado por pruebas para sistemas basados en Internet de las cosas".

Trabaja como docente de la Universidad de Guayaquil (1998-2013) y de la Universidad Técnica Estatal de Quevedo (2000 en adelante) ha dictado cátedras relacionadas con Administración de Empresas, y con desarrollo y administración de sistemas. En los últimos 4 años ha dictado asignaturas como: Tecnologías de Internet, Programación Web, Aplicaciones Distribuidas, y Construcción de Software (2020 hasta la fecha). Además ha sido director de Proyectos de Investigación de pregrado en las carreras de Ingeniería en Sistemas. Ha sido director del proyecto de Vinculación con la Comunidad "Tele educación para el fortalecimiento de los saberes ancestrales, valores y educación en las familias de los sectores rurales y urbano marginales de la zona norte de la Provincia de Los Ríos"(2017-2019). Participante del proyecto de investigación "Uso de Internet de las cosas para proporcionar recordatorios a adultos mayores" (2018-2019). Actualmente es participante del proyecto internacional de investigación MIoT-Health - https://www.linkedin.com/company/miot-health/ (2020 - hasta la fecha).

Autor de varias publicaciones científicas como: IoT-Based System to Help Care for Dependent Elderly (2018), From a Common Chair to a Device that Issue Reminders to Seniors (2019), A Ubiquitous Photo Frame to Provide Reminders to Older Adults (2020), IoT-Based Smart Medicine Dispenser to Control and Supervise Medication Intake (2020), Lineamientos para el diseño de aplicaciones de ingreso de texto para adultos mayores (2021)

Tecnologías para el desarrollo de aplicaciones web

Gleiston C. Guerrero Ulloa

Colaboradores en la elaboración del material complementario:
Dúval R. Carvajal Suárez & Ariel Fernández Loor

Diseñadora de la Portada
Keyla de los Santos

Universidad Técnica Estatal de Quevedo

Título del libro
Tecnologías para desarrollo de aplicaciones WEB

ISBN: 978-9942-33-543-2

Publicado 2022 por acuerdo con los autores.

Guayaquil-Ecuador

@grupocompas.ec
compasacademico@icloud.com

compÁs
Grupo de capacitación e investigación pedagógica

PREFACIO

El texto académico presentado a continuación lleva el título de “Tecnologías para el desarrollo de aplicaciones web”. Este trabajo se ha desarrollado en términos de ejemplos sencillos de aplicaciones web. Se centra básicamente en el uso de software libre en las versiones más recientes y estables para la construcción de este tipo de software.

La redacción del texto se llevó a cabo bajo la presentación del distributivo de la carga horaria presentado por el Decano de la Facultad de Ciencias de la Ingeniería (FCI) Ing. Washington Chiriboga Casanova y su aprobación por parte del Órgano Colegiado Académico Superior (OCAS) de la Universidad Técnica Estatal de Quevedo (UTEQ) precedido por su Rector el Dr. Eduardo Díaz Ocampo, PhD, a quienes expreso mis sinceros agradecimientos. La principal razón para la realización de este texto es la no existencia de un texto de fácil acceso que cubra los temas considerados por el autor como más importantes en este campo. El trabajo de este tipo de texto (relacionados con tecnologías) ha exigido la actualización continua de la redacción como de los ejercicios, por ser muy cambiante. Así mismo, estos cambios continuos hacen que la actualización de los ejercicios prácticos sean tediosa.

El haberme asignado la carga horaria para la redacción del texto y la responsabilidad para impartir las asignaturas muy relacionadas con el tema, me han servido como actualización de conocimientos, base fundamental para la redacción de este texto.

Así mismo, quisiera agradecer a mis estudiantes de la asignatura de **Tecnologías de Internet** del periodo lectivo 2020-2021 primero y segundo ciclo, y del periodo lectivo 2021-2022 primer ciclo, por poner en práctica las enseñanzas impartidas, las cuales están plasmadas en este trabajo.

Siempre he dado crédito a la práctica: "Él que escribe comete errores que son difíciles de identificar por él mismo."Por esa razón, agradezco a todos los estudiantes de octavo semestre de la carrera de Ingeniería en Sistemas (2021-2022 I ciclo), en especial a Abrahan Pachay Espinoza y José Brito Casanova por los aportes en la redacción para que sea clara y entendible por los estudiantes de Ingeniería de Sistemas, y carreras afines (Ingeniería de Software, Sistemas de Información, entre otras). A Ariel Espinoza y a Duval Carvajal, quienes aparecen como autores de los videos y los códigos de los proyectos ejemplos que se encuentran en los repositorios Youtube y GitHub respectivamente, y compartidos desde el drive de la UTEQ. De manera especial a Keyla de los Santos, Ingeniera en Diseño Gráfico, por su apoyo desinteresado en el diseño de la portada y contraportada del texto.

No por ser últimos son menos mis agradecimientos al Director del Departamento de Investigación Ing. Carlos Zambrano, PhD, y al Coordinador de las carreras de Ingenierías en Software, en Sistemas, y en Diseño Gráfico y Multimedia, Ing. Cristian Zambrano Vega, PhD, por su apoyo en la redacción de este texto académico.

Espero disfruten la lectura y apliquen lo demostrado.

Gleiston Guerrero Ulloa

Quevedo, 4 de diciembre de 2021

Índice general

DEDICATORIA

Dedico este libro a mi familia y a mis estudiantes actuales y futuros.

AGRADECIMIENTOS

Aunque se han escrito muchas palabras de gratitud, en ocasiones el papel no puede plasmar a la perfección tanto agradecimiento, afecto y admiración como siento en el término de tan fructífero trabajo gracias al apoyo de mi Alma Mater la Universidad Técnica Estatal de Quevedo (UTEQ).

Quiero mostrar mis agradecimientos al Rector de la UTEQ, Dr. Eduardo Díaz Ocampo, PhD por brindarme la oportunidad de elaborar este trabajo, la redacción de mi primer texto académico. Así mismo el apoyo a la Vicerrectora Académica Sra. Ing. Yenny Torres Navarrete, PhD y al Vicerrector Administrativo Ing. Roberto Pico Saltos, MSc., y no por ser últimos son menos importantes. El apoyo del Decano de la Facultad de Ciencias de la Ingeniería Ing. Washington Chiriboga Casanova,

Así mismo quiero expresar mis agradecimientos a los estudiantes: Sres. Ariel Fernández Loor, Duval Carvajal Suárez por la elaboración del material adicional como son ejercicios prácticos y los videos sobre su desarrollo, cuya autoría se muestran sus nombres junto al autor de este trabajo. Y Anthony Pachay Espinoza por sus sugerencias en la redacción de este texto.

INTRODUCCIÓN A LAS TECNOLOGÍAS DE DESARROLLO DE APLICACIONES WEB

Desde hace algunas décadas con el advenimiento de Internet, surge la World Wide Web WWW como un sistema para intercambio de información disponible en documentos de hipertexto o hipermedia a través de Internet. Con la web aparecen los navegadores gráficos, que hacen la búsqueda de información, la presentación de resultados del procesamiento de datos en grandes servidores de la nube, de una manera amigable en diferentes formatos de datos. Todos esto gracias a las características del protocolo universal de acceso a información en la web HTTP o Hypertext Transfer Protocol (Protocolo de Transferencia de Hipertexto).

Internet y la web han hecho posible los más grandes avances en todos los campos de la ciencia. Internet hace posible compartir información logrando romper las barreras de la comunicación, sea en modo diferido o en tiempo real. Además, compartir recursos de Tecnologías de la Información (TI) para su uso más eficiente, logrando una Informática Verde (Green Computing). Hacer uso de las características potentes y múltiples de las computadoras, sin necesidad de hacer inversiones grandes ni tenerlas cerca (sin saber dónde se encuentran), reduciéndose en la realización de una solicitud por medio de una aplicación web para lograr utilizarlas.

El ámbito en el que se emplean las aplicaciones basadas en Internet, va desde el simple intercambio de información textual, gráfica, multimedia, en tiempo real o en diferido, hasta incluso la manipulación y la monitorización de dispositivos tan grandes cómo aviones, buques y más, y tan pequeños como un sensor miniatura para censar datos de un ambiente o actuar sobre él. Particularmente, Internet y la web por su parte han "abierto las tecnologías", haciendo posible desarrollar las aplicaciones sin pensar en las plataformas en las que van a ser explotadas, siendo necesario para el cliente, sólo tener instalado en un dispositivo un navegador web.

Las aplicaciones web han sido desarrolladas utilizando diversas herramientas de desarrollo, desde los en desuso de las interfaces de entrada común (CGI: *Common Gateway Interface*) hasta los más actuales marcos de desarrollo que facilitan el trabajo a los desarrolladores web. Estos marcos persiguen (o buscan) que los desarrolladores se olviden de escribir código repetitivo o común entre este tipo de aplicaciones (Inicio de

sesión, control de sesión, operaciones de crear, consultar, editar y eliminar registros de una base de datos, por mencionar algunas).

Los marcos de desarrollo para poder cumplir con este objetivo deben estar en constante actualización para incorporar las tecnologías emergentes para el desarrollo de las aplicaciones web. En este trabajo se pretende presentar los fundamentos, las recomendaciones y el uso de las tecnologías, y modelos arquitectónicos más comunes en el desarrollo web. Además, en el caso que amerite, se presentará alguna comparación entre ellas.

Este libro está organizado por capítulos. En el capítulo I presentamos una introducción a las aplicaciones web. Dejamos claro los conceptos entre página web, sitio web y aplicaciones web, presentamos un enfoque global de los principales elementos de las arquitecturas de las aplicaciones web, y de las tecnologías para su desarrollo. En el capítulo II presentamos los diferentes momentos por los que la Web ha tenido en todo su desarrollo hasta la fecha de la redacción de este texto. Se resalta sus ventajas y limitaciones de cada uno de estos momentos, así como las principales tecnologías que los diferencian.

En el capítulo III presentamos los desafíos y las consideraciones que los desarrolladores de aplicaciones web deben tomar en cuenta. Con esto hacemos referencia a la Accesibilidad Web. Dentro de la accesibilidad web se presentará la iniciativa de accesibilidad web o Web Accessibility Initiative (WAI), la lista de herramientas automáticas para la evaluación de la accesibilidad en los sitios web. En el capítulo IV, se presentan los principales entornos integrados de desarrollo, considerando la posibilidad de poder desarrollar en más de un lenguaje de programación. A continuación, en el capítulo IV se presentarán de una manera rápida, las diferentes herramientas de desarrollo, tanto del lado del cliente como del lado del servidor.

Posterior al capítulo IV se presentarán tecnologías de desarrollo específicas. En el capítulo V se trata la tecnología de los Enterprise JavaBeans, dejando para los capítulos posteriores las tecnologías para el intercambio de información entre aplicaciones, como WebSockets (capítulo VI), Web Services (capítulo VII), Java Message Services (JMS en el capítulo VIII).

Capítulo 1

INTRODUCCIÓN A LAS APLICACIONES WEB

Objetivos

Después de haber leído esta unidad, el lector será capaz de:

- Diferenciar claramente lo que es página web, sitio web, aplicación web y portal web.
- Explicar los términos y tecnologías (servidores de aplicaciones, protocolos, entre otros) que están relacionados con las aplicaciones web.
- Caracterizar a las aplicaciones web.
- Clasificar a las aplicaciones web según el grado de interacción con el usuario y según el servicio ofrecido.

Resumen

Muchos confunden los términos como página web, aplicación web, sitio web, y portal web. Básicamente se diferencias en que uno puede ser parte de otro y en las tecnologías que intervienen en su construcción y explotación. Entre las tecnologías que intervienen en su construcción y explotación están las herramientas del lado del cliente como HTML, del lado del servidor como JSP, PHP, ASP.Net, entre los más populares de nuestro medio. Estas tecnologías hacen posible la construcción de las diferentes aplicaciones web estáticas, dinámicas, y las aplicaciones de cliente interactivo, así mismo las aplicaciones que ofrecen los diferentes tipos de servicios. Otras tecnologías que están presentes en las aplicaciones web son los protocolos de comunicación, los servidores web y los servidores aplicaciones.

1.1. Introducción

En este capítulo presentamos los conceptos, definiciones y propiedades de las páginas web, sitios web y aplicaciones web. Así mismos la arquitectura mínima que las aplicaciones web cumplen y adicionalmente otras arquitecturas comunes que hacen posible que hacen posible cumplan con mayor efectividad los requisitos de calidad. Requisitos conocidos también como requisitos no funcionales [1].

1.1.1. Definiciones

Antes de definir lo que es una aplicación web, se debe tener muy claro los siguientes términos relacionados que muchos usuarios los utilizan de manera indiferente. Como lo que son página web, sitio web, portal web y aplicación web.

Página web

Se puede entender por el término **página** como un archivo o documento, y por el término **web** se hace referencia al sistema por el cuál podemos intercambiar información con el mundo a través de Internet. Por lo tanto, una **página web** es un documento capaz de ser interpretado por un navegador o browser. El lenguaje en que están escritos estos documentos es el lenguaje de marcado de hipertexto (HTML: *Hypertext Markup Language*). Estos documentos pueden contener texto, sonido, imágenes, video y otros tipos de datos. Además, pueden incluir código de programación escrito en algún lenguaje scripting, como JavaScript o Visual Basic Script [2].

Página web

Es un documento digital escrito en HTML (muchas veces contiene CSS y JavaScript) capaz de ser interpretado por un navegador o browser.

Los elementos comunes que intervienen en la construcción de una página web, se definen a continuación.

- **HTML:** Lenguaje de Marcas de Hipertexto, del inglés *HyperText Markup Language* es el componente más básico de la Web. Define el significado y la estructura del contenido web [3].
- **CSS:** *Cascade Style Sheet*, su traducción al español sería: hojas de estilo en cascada. Es el lenguaje de estilos utilizado para describir la presentación de documentos HTML o XML (en-US). CSS describe como debe ser renderizado el elemento estructurado en la pantalla, en papel, en el habla o en otros medios [4].
- **JavaScript:** Es un lenguaje de programación ligero, interpretado, o compilado justo-a-tiempo (*just-in-time*) con funciones de primera clase. Si bien es más conocido

como un lenguaje de *scripting* (secuencias de comandos) para páginas web, y es usado en muchos entornos aparte del navegador.

Al referirse a documentos de **hipertexto**, les da la posibilidad de tener en su contenido textos u otros elementos sensitivos que sirven para enlazarlos con otras partes del mismo documento o con otros documentos. A estos elementos sensitivos se les denomina anclas e hipervínculos.

Una sola página web no podría mostrar toda la información completa de algún producto, servicio, empresa u otro elemento, que se requiera dar a conocer o poner a la disposición de los internautas[1]. Para organizar la información es necesario un conjunto de páginas. Es por eso que generalmente se refiere a sitio web o aplicación web, y a un nivel un poco superior en jerarquía se trata de portal web.

Sitio web

Por el concepto del término sitio tenemos claro que es un lugar. Por lo tanto, un sitio web es un lugar en Internet (lugar virtual) en el cual se pone a disposición de los internautas información sobre una empresa, un producto o servicio. Por lo tanto, también podemos decir que, un sitio web es una colección de páginas web con información relacionada sobre un mismo tópico e interlazadas entre ellas. Entre las páginas que forman un sitio web, se encuentra la página inicial o de bienvenida (home, index, default, about, etc.), y las páginas de contenido. En la página inicial se busca plasmar la metáfora del ambiente de recepción de la empresa (local) física, buscando sea llamativa y cautivadora, y con un fácil y rápido acceso. Todas las páginas deben guardar relación no sólo en la información que muestran sino también en su apariencia, al igual que las diferentes oficinas de la empresa física (color, muebles, etc.) [5].

Sitio web

Es una colección de páginas web con información relacionada sobre un mismo tópico e interlazadas entre ellas.

La principal característica de lo sitios web es que estos muestran información que no cambia con el tiempo, y no presentan interacción con el internauta, limitándolo únicamente a dar clic sobre el hipertexto para cargar la información que requiere. A este tipo de sitios web se les denomina sitios web estáticos. Los sitios web que tienen la funcionalidad de mostrar la información según la preferencia del usuario, haciendo posible la interacción con el usuario, se les denomina sitios web dinámicos o aplicaciones web.

Cabe recalcar que no se debe confundir entre un sitio web dinámico con un sitio web animado o amigable. Un sitio web animado que le hace la estancia agradable mientras se encuentre navegando (buscando información) dentro del sitio, pero que los accesos a

[1]Persona que hace uso de Internet para buscar información

la información del sitio los tiene a la vista, y no le permite interactuar (ingresar datos para pedir información) al internauta, sigue siendo un sitio web estático.

Aplicación web

Una aplicación web es un sitio web dinámico, capaz de hacer que sus páginas web presenten la información de acuerdo a los datos proporcionados por el usuario, es decir, se da interacción entre el sitio web y el usuario [6]. Las páginas web que forman parte de este tipo de sitios web necesitan de programas especiales (servidores de aplicaciones) para que devuelvan los resultados de lo que los internautas solicitan. Además, ya no es suficiente utilizar HTML (o alguno otro lenguaje de su familia), sino que además necesitan de herramientas de programación como Java, C#, Visual Basic, Python, PHP, Ruby, Perl, etc., y por lo general de bases de datos [7, 8].

Aplicación web

Es un sitio web dinámico, capaz de hacer que sus páginas web presenten la información de acuerdo a los datos proporcionados por el usuario.

Portal web

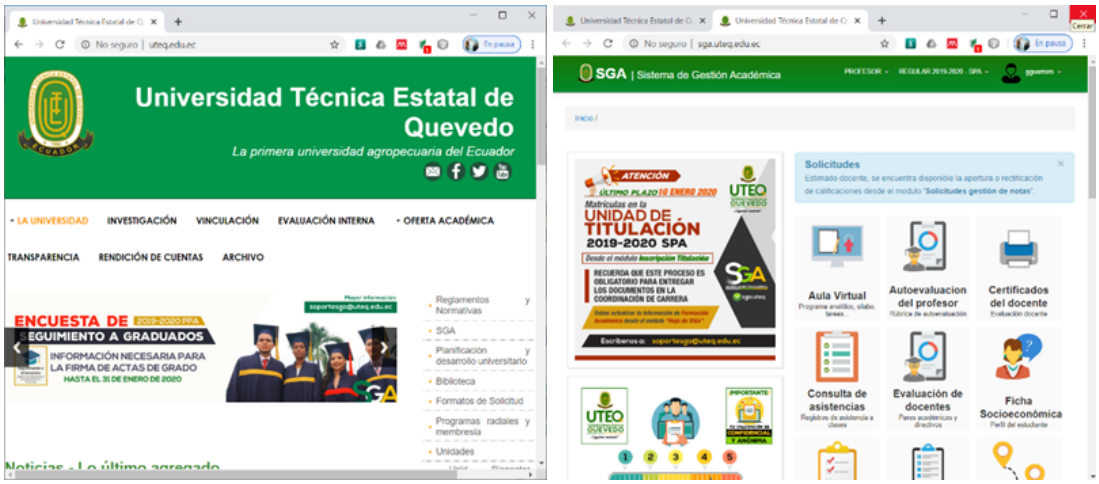
Cuando las corporaciones tienen varias empresas, o las empresas tienen varios productos u ofrecen varios servicios; la corporación (o empresa) ve la necesidad de crear un sitio web para cada uno de estos elementos. La forma como presenta al mundo estos sitios web, podemos llamarle portal web, por lo tanto, portal web es un sitio web que ofrece de manera integrada o de forma única el ingreso a los sitios web de una empresa (ver figura 1.1), de los productos o de los servicios que la corporación ofrece. Por ejemplo, dentro del portal de la UTEQ www.uteq.edu.ec como se muestra en la figura 1.1a , tenemos varios sitios web, entre ellos el portal de la aplicación web para la Gestión Académica (SGA: Sistema de Gestión Académica) sga.uteq.edu.ec como se muestra en la figura 1.1b [9].

Portal web

Es un sitio web que ofrece de manera integrada o de forma única el ingreso a los "sitios web" o "aplicaciones web" de una empresa.

En el caso del portal web de la Universidad Técnica Estatal de Quevedo (UTEQ) tiene su dominio[2] uteq.edu.ec, mientras que el sitio web (aplicación web) del Sistema de Gestión Académica tiene el subdominio sga.uteq.edu.ec. Un portal web, brinda la posibilidad de ingresar y navegar en los sitios web que engloba. Además, al ser un sitio web, también tiene páginas de información sobre la organización (en este caso de la UTEQ).

[2]Un dominio de Internet es un nombre único que identifica a una subárea de Internet.



(a) Portal de la UTEQ **(b)** Aplicación Web SGA

Figura 1.1: Portal web y aplicación web

Las páginas web (información) que podemos encontrar de manera generalizada en un sitio web son: la página inicial (portada), quiénes somos, datos de contacto o un formulario mediante el cual se puede enviar al administrador alguna novedad de la empresa, o producto(s) y/o servicio(s) que brinda, entre otras.

Un subdominio requiere que se instale en un servidor diferente (puede ser en máquinas físicas diferentes). Esto se lo hace para aumentar las seguridades para las aplicaciones web que se van a instalar en ellos. Un dominio (subdominio) puede tener varias aplicaciones web en ejecución, cuyo acceso lo hace añadiendo al final de la url del dominio el nombre de la aplicación separado por la barra inclinada /, como ejemplo: https://aplicaciones.uteq.edu.ec/tddm4iots/, donde:

- **Dominio:** uteq.edu.ec
- **Subdominio:** aplicaciones
- **Aplicación web:** tddm4iots

¡Importante recordar!

Una página web es parte de un sitio web, puede ser resultado de la ejecución de una aplicación web que devuelve código HTML o simplemente código HTML estático. Una aplicación web puede ser un sitio web pero no lo contrario. Un portal web es la entrada a las aplicaciones o sitios web de la empresa.

1.1.2. Otras definiciones importantes

Para quienes es nuevo el mundo del desarrollo de aplicaciones web, se considera que debe aclarar todos los términos que le aparezcan mientras lee y se instruye en

la obtención de las competencias como desarrollador de aplicaciones web. Se debe considerar primero obtener el conocimiento sobre la arquitectura del Protocolo de Transferencia de Hipertexto HTTP (*HyperText Transfer Protocol*) y de las aplicaciones web [10,11]. A continuación, se describen los términos que se han considerado que son relevantes para el dominio del desarrollo de este tipo de aplicaciones.

Dominio web

Un dominio en la web es un nombre único que identifica a un sitio web como único en la red. Por lo tanto, al implementarse un sitio web, se debe especificar un dominio (o subdominio) o una dirección IP[3] únicos por medio de los cuales los internautas podrán acceder a ellos por medio de Internet (web) [11,12]. De tal manera que, a todas las páginas que conforman el sitio web, se las accede a través de una URL[4] raíz común, similar a la ruta o path de los archivos en una carpeta de un medio de almacenamiento físico.

La necesidad de organizar las funcionalidades implementadas en una aplicación web o las informaciones de una empresa, hace que se creen otros dominios dentro de un dominio principal. A estos dominios se les denomina subdominios. Subdominio es el nombre de una aplicación web o sitio web que necesita del dominio para ser visible en red. Un subdominio es un subconjunto de un sitio Web, que sirve para organizar o dividir el contenido del sitio en distintas secciones [13].

Dominio web

Nombre único que identifica a un sitio web como único en la red. Subdominio es el nombre de una aplicación o sitio web que necesita del dominio para ser visible en red.

Un dominio puede dar a conocer: (1) el tipo de dominio, generalmente es www que indica que es un dominio de la web. Otros pueden ser mail[5] o ftp[6] . (2) el nombre del elemento al que hace referencia, se conoce simplemente como el nombre del dominio. (3) el tipo de elemento o de empresa (org, com, edu, mil, net, etc.). Y (4) el código del país de donde procede o fue validado (us: Estados Unidos, es: España, ec: Ecuador, etc.) [14].

Servidor web

Un servidor web o servidor HTTP es un programa de software que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales o unidireccionales y síncronas o asíncronas con el cliente.

[3]Internet Protocol (Protocolo de Internet)
[4]Uniform Resource Locator (Localizador de recursos universal)
[5]Reservado para sitios web que permiten acceder al correo electrónico
[6]Reservado para la transferencia de archivos (ftp: *File Transfer Protocol*)

Servidor web

Programa de software que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales o unidireccionales y síncronas o asíncronas con el cliente.

Un servidor web dispone de la capacidad para resolver las peticiones de contenido web estático, aunque en la actualidad algunos poseen la capacidad por medio de plugins de proveer resultados dinámicos obtenidos de la ejecución de programas escritos en lenguaje scripting como Perl, PHP, ASP, JSP, entre otros.

Servidor de aplicaciones

Es una aplicación de software que corre sobre una computadora generalmente con mayores recursos de procesamiento, memoria y almacenamiento que las demás computadoras de la red, y que ejecuta las aplicaciones (programas servidores) para brindar los servicios a los clientes (computadoras) que los soliciten. Por lo tanto, es el que proporciona los servicios que soportan la ejecución y disponibilidad de las aplicaciones desplegadas [11, 15].

Servidor de aplicaciones

Aplicación de software que corre sobre una computadora generalmente con mayores recursos de procesamiento, memoria y almacenamiento que las demás computadoras de la red, y que ejecuta las aplicaciones (programas servidores) para brindar los servicios a los clientes (computadoras) que los soliciten.

Un servidor de aplicaciones también tiene la capacidad de resolver peticiones HTTP, sin embargo, su función es la de habilitar la interacción entre los clientes (usuarios finales) y el servidor (aplicación servidor).

Existen varios servidores de aplicaciones para cada tecnología sobre la que es desarrollada la aplicación web. Por ejemplo, Java (Servidor de Páginas Java o JSP: *Java Server Pages*) tiene asociados varios servidores, algunos de ellos de código abierto y otros de pago. Entre los más importantes de **código abierto** están [16]:

- Apache Gerónimo
- Apache TomEE
- GlassFish
- JOnAS
- Payara
- WidFly (JBoss Application Server)

Entre los servidores con **licenciamiento de pago**, están:

- BEA WebLogic
- Borland AppServer
- HP Bluestone
- IBM WebSphere
- Oracle IAS
- Sun-Netscape IPlanet
- Sun One

En este tipo de servidores, es frecuente encontrarlos bajo la denominación de servidores web. En algunos casos pueden ser lo mismo, sin embargo, tienen algunas diferencias. Entonces se puede decir que: Si el servidor satisface las peticiones de los clientes sobre solicitudes de páginas web, es un **servidor web**. Mientras que, si el servidor satisface a clientes sobre **servicios web** (REST[7], SOAP[8]) o WEB API[9], WCF[10], etc., es un **servidor de aplicaciones**.

Diferencia entre servidor web y servidor de aplicaciones

Por definición se puede expresar que:

- Los servidores web son subconjunto de los servidores de aplicaciones.
- Un cliente para un servidor de aplicaciones puede ser el usuario final (GUI) de la aplicación, un navegador web o una aplicación web; mientras que, para un servidor web su cliente debe ser un navegador web o una aplicación capaz de realizar peticiones HTTP.
- La interacción cliente/servidor en un servidor web lo hace por medio del protocolo HTTP; mientras que, en un servidor de aplicaciones no se limita a HTTP.

Estos dos tipos de servidores suelen ser confundidos. Se ha intentado por lo tanto, describir las principales diferencias. En la actualidad, se puede encontrar de manera indistinta servidores de aplicaciones como servidores we, es decir, con las mismas prestaciones o capacidades. La mayoría de las aplicaciones actuales se desarrollan cada

[7] *Representational State Transfer* (Transferencia de estado representacional)

[8] *Simple Object Access Protocol* (Protocolo simple de acceso a objetos)

[9] *Application Programmiing Intreface* (Interfaz de programación de aplicaciones) disponible en la web, por lo tanto se accede por medio del protocolo HTTP

[10] *Windows Communication Foundation* (Fundación de comunicación de Windows): Modelo de programación para el desarrollo de aplicaciones con arquitectura orientada a servicios (SOA: *Service Oriented Architecture)*

vez más enriquecidas, combinan el contenido web estático con el contenido dinámico de las aplicaciones que se genera a través de una combinación de tecnologías de servidor web y servidor de aplicaciones.

1.1.3. Protocolo de transferencia de Hipertexto o HTTP

El protocolo de transferencia de hipertexto o simplemente conocido como HTTP o como el protocolo de la web. HTTP es un protocolo de la familia de protocolos TCP[11]/IP [17,18], que hace posible la transferencia de información en la web, por lo tanto, es el encargado de la transferencia de las peticiones del cliente hacia el servidor y de las páginas web y demás recursos desde el servidor hacia el navegador o cliente. Entre las características de HTTP se pueden mencionar que [10,19]:

- Es un protocolo orientado a petición-respuesta, lo que asegura que el cliente no recibirá del servidor recursos sino los ha solicitado, y viceversa, es decir, que el servidor no enviará ningún recurso a un cliente si éste no le ha solicitado.

- HTTP es sin estado. HTTP no guarda el guarda ninguna información de los clientes, tratando a cada conexión de la misma manera, es decir, cuando el cliente envía una petición, el servidor la atiende, y una vez que la petición es atendida sus datos son desechados por el servidor y éste no los recuerda más.

- Es un protocolo NO orientado a conexión. Una vez que el cliente establece la conexión con el servidor y hace la petición. En el momento que el servidor atiende la petición, la conexión con el cliente se pierde, haciendo que, para la siguiente petición vuelva a establecer una conexión (nueva) con el servidor como si fuese la primera vez.

- Acepta tipos de datos MIME (*Multipurpose Internet Mail Extensions*). Los tipos de datos MIME son especificados por partes separadas por una barra de divisor sin espacios en blanco: primera/segunda [20,21]. Primera representa el tipo o la categoría, mientras que segunda indica el tipo específico. La categoría puede especificar a un tipo de dato discreto o multiparte [18]. Los datos que envía o recibe un comando HTTP, se clasifican según la descripción MIME que se detalle. Muchos de las aplicaciones no consideran la extensión del archivo, sino el tipo de dato MIME. De esta forma, el protocolo puede intercambiar cualquier tipo de dato, sin preocuparse de su contenido. La transferencia se realiza en modo binario, byte a byte, y la identificación MIME permite que el navegador (receptor) les dé a los datos el tratamiento adecuado [22].

- Métodos HTTP. Le permiten al cliente solicitar al servidor los requerimientos (acciones) que se desean realizar sobre un recurso bajo una URL. Los métodos de

[11]Protocolo de Control de Transmisión (*Transmission Control Protocol*)

HTTP son: POST, GET, HEAD, PUT, DELETE, CONNECT, OPTIONS, TRACE y PATCH. Estos métodos se detalla a continuación [23]:

- POST. Este método se lo utiliza para realizar cambios en el estado del servidor por medio de una entidad que es enviada a un recurso específico. En el desarrollo de aplicaciones web, los formularios web utilizan el método POST para que el navegador envíe los datos en el cuerpo de la petición al servidor. Este método se debe considerar usarlo para llevar a cabo una inserción [22,24].
- GET. Se utiliza para solicitar una representación de un recurso específico. Este método debe ser usado en peticiones que solamente soliciten datos, es decir para consulta de recursos. En el desarrollo de aplicaciones web, los formularios web al utilizar este método como método de solicitud, los datos viajan como parte de la URL del recurso solicitado [22,24].
- HEAD. En aplicaciones REST, el método HEAD pide una respuesta idéntica al método GET, pero sin el cuerpo de la respuesta. En resumen, el método HEAD envía la metainformación contenida en los encabezados HTTP como si fuese la respuesta a una solicitud GET. La funcionalidad de HEAD hace que sea usado para comprobar la validez de los enlaces.
- PUT. Las solicitudes del método PUT para la entidad adjunta se almacenan en la URI de la solicitud suministrada. Si el recurso al que se refiere la URI ya existe, se procederá a realizar una operación de actualización, caso contrario, se deberá realizar una operación de creación si la URI de la solicitud es una URI de recurso válida. Se aconseja que este método se utilice para actualización de recursos [22,24].
- DELETE. Este método es el más obvio en su función. Sirve para borrar un recurso específico, es decir, identificado por la URI de la solicitud [24].
- CONNECT. Este método HTTP se utiliza para solicitar al servidor una conexión de tipo túnel TCP/IP. CONNECT es principalmente usado cuando se requiere de un proxy para una conexión segura cifrada HTTPS[12] o para comunicaciones mediante SSL[13]. Cabe enfatizar que cualquier cliente web puede hacer uso de estos métodos y además, que se puede configurar un servidor web para que reciba cualquier combinación de éstos [11].
- OPTIONS. Al existir servidores y recursos que no soportan todos los métodos HTTP, se ha desarrollado el método OPTIONS que sirve para determinar qué métodos HTTP soporta el servidor web con respecto a un recurso en concreto. Al querer determinar todos los métodos soportados por el servidor, se envía un * en la URI [22].
- TRACE. HTTP también tiene un método para monitorizar los mensajes que haya entre el cliente y el servidor web, para esto es el método TRACE. El

[12]HTTP Seguro
[13]Secure Sockets Layer (capa de Sockets seguros)

propósito principal de usar este método es para diagnosticar fallas y/o para verificar si existen servidores intermediarios.

- PATCH. Este método HTTP sirve para realizar solicitudes para la actualización parcial de un recurso. Aunque PUT y PATCH actualizan, se debe usar PUT para reemplazar completamente el recurso, mientras PATCH para una actualización parcial del recurso. Decidir usar PATCH como solicitud para actualización (envista de que la mayoría son actualizaciones parciales) depende de la compatibilidad de los navegadores y herramientas de desarrollo.

Métodos HTTP

Los métodos HTTP son: GET, POST, PUT DELETE, HEAD, CONNECT, OPTIONS, TRACE, PATCH.

1.1.4. URI

Uniform Resource Identifier (Identificador Uniforme de Recursos) conocidos simplemente como URI. Una URI está formada por una cadena de texto que sirve para identificar a un recurso único en la web. URI tiene tres especializaciones [23, 24]:

- Localizador Uniforme de Recursos (Uniform Resource Locator) o más conocidos como URL,
- nombre uniforme de recursos (Uniform Resource Name) o simplemente URN para identificar el recurso y el nombre, y
- característica uniforme de recurso (Uniform Resource Characteristic) conocido como URC. Este tipo de URI es la menos conocido, y sirve para conocer los metadatos de una URI.

Las direcciones que se colocan en la barra de dirección de los navegadores se les denomina URL, y también son ejemplos de URI. Otros ejemplos diferentes de URI se podrán observar, observando el código fuente de una página web (no en todas) esta presentará en la etiqueta <html> algo similar a lo siguiente: <html class='v2' dir='ltr' xmlns='http://www.w3.org/1999/xhtml' xmlns:b='http://www.google.com/2005/gml/b' xmlns:data=http://www.google.com/2005/gml/data' xmlns:expr=http://www.google.com/2005/gml/expr'> las expresiones que están después del signo igual ("=") de xmlns, representan las URIs de los recursos que se usarán en esa página. La URI. Adicional de la identificación del recurso, la URI puede contener datos que sirven para identificar la forma de acceso al recurso.

1.1.5. Localizador uniforme de recursos o URL

La URL es una cadena de texto que sirve para localizar un recurso en la red de Internet, por lo tanto, es análogo a la dirección postal de una persona [25]. La URL es lo

más conocido o usado, es lo que los navegadores presentan en la barra de dirección cuando han mostrado un recurso al cliente. Por lo tanto, indica lo necesario para acceder al recurso: dónde se encuentra el recurso y el protocolo con el cuál podemos acceder: HTTP, HTTPS, telnet[14], ftp[15], entre los más conocidos: http://www.uteq.edu.ec https://www.google.com.ec telnet://telnet.yahoo.com ftp://ftp.uteq.edu.ec [26].

Al usar los términos URI y URL por separado, se podría decir que URI, como tal se utiliza para referenciar a un recurso que se quiere acceder, no para visualizar o no para que el usuario (persona) interactúe con él, sino para ser tratado o utilizado por otros recursos (por ejemplo el navegador). Mientras que URL es para visualizar o cargar en una aplicación (browser) el recurso, que generalmente se trata de una página web, o un dato de internet (texto, imagen, video, animación) [22].

1.1.6. Nombre uniforme de recursos o URN

URN representa el nombre de un recurso dentro del conjunto al que pertenece, pero no representa la ubicación o localización del mismo. Al estar restringido al conjunto al que pertenece, no es seguro que sea único entre todos los conjuntos existentes. Los siguientes ejemplos se pueden considerar como URN: ISBN, ORCID, ResearchID, DOI, entre los más importantes [23].

Característica uniforme de recursos o URC

La URC corresponde a los metadatos de una URI, por lo tanto, está formada por una cadena de caracteres que identifica a un recurso en la web. Una URC enlaza a un URN con su URI, a un URN con su URL. Ejemplo:

1. view-source:http://uteq.edu.ec.com/
2. data:text/plain;charset=UTF-8

Términos relacionados con HTTP

Términos relacionados y muy parecidos (pero diferentes) son: URI, URL, URN, URC.

1.2. Aplicaciones web

El proyecto de la web nació con la iniciativa de Tim Berners-Lee en marzo de 1989 [27], este proyecto fue considerado por primera vez en 1994 y posteriormente en 1995. El

[14]Es el protocolo que sirve para acceder a otra máquina de forma remota. También se implementa un programa cliente que permite acceder a la máquina remota mediante terminal o consola.

[15]File Transfer Protocol (Protocolo de transferencia de archivos) Sirve para transferir archivos entre dos máquinas conectadas a una red TCP

documento más antiguo entre artículos de revistas y congresos indexados en la Web of Science (WoS) que trata sobre una aplicación web, data de 1995, fue presentada por Ginsburg y Kambil como parte de un proyecto de Internet denominado EDGAR [28].

Las primeras aplicaciones basadas en computadores se desarrollaron sin ninguna forma para compartir información [29]. Con el advenimiento de las redes de computadoras surge la necesidad de compartir la información entre computadoras, con ello aparecieron las aplicaciones cliente/servidor. En las primeras aplicaciones cliente/servidor, el cliente consiste en un programa con su propia interfaz, el mismo que es instalado en cada cliente por separado. Este programa cliente realiza una petición a otro programa denominado servidor, el cuál le responde satisfaciendo el requerimiento. Las actualizaciones en los clientes se harán por separado (uno a la vez). Estas aplicaciones podemos nombrarlas como Aplicaciones con arquitectura cliente/servidor dos capas.

Evolución de las aplicaciones

Las Aplicaciones de software pasaron desde aplicaciones que se ejecutaban en una computadora incapaces de compartir información hasta aplicaciones totalmente distribuida y heterogéneas que comparten información totalmente distribuida.

Las aplicaciones cliente/servidor dos capas son usadas en puntos de venta, y con la popularidad de internet y los servicios web, estas aplicaciones han alcanzado un nivel más alto al poder comunicarse con otras máquinas, como por ejemplo para requerir autorización de tarjetas de crédito.

Por las facilidades que brinda Internet y con la necesidad de agilizar los procesos, y el requisito de acercar el negocio al cliente; se han creado las aplicaciones web. Estas aplicaciones trabajan conectadas a Internet o a una Intranet [30]. Se pueden utilizar tan solo por medio de un navegador, sin necesidad de tener que instalar un programa en particular (en muchos casos) en el cliente, residiendo la aplicación en un servidor de aplicaciones y las bases de datos en el mismo o en distintas máquinas servidoras [31]. Estas aplicaciones se conocen como aplicaciones con arquitectura cliente/servidor 3 capas, algunas llegando a aumentar sus capas, denominándose como N-capas.

1.2.1. Características de las aplicaciones web

Las principales características de las aplicaciones podemos resumirlas en las siguientes [32]:

- **No requieren ser instaladas en la máquina del cliente.** Al ser accedidas por medio de un navegador, el cliente no necesita descargar e instalar la aplicación, sólo necesita tener acceso a internet (intranet) y un navegador. Esto elimina la barrera de accesibilidad del usuario.

- **Son aplicaciones multiplataforma.** En vista de que no dependen del dispositivo cliente ni de su sistema operativo. Esto hace posible que su funcionamiento sea

óptimo en un navegador actual, el usuario tendrá en su dispositivo la misma apariencia (muy parecida) y la misma interacción [33].

- **Ahorro en mantenimiento.** En este tipo de aplicaciones se hace en una sola máquina (servidor) y las actualizaciones en los clientes son instantáneas, es decir, automáticamente acceden a las nuevas características de las aplicaciones actualizadas.
- **Desarrollo económico.** El desarrollo de aplicaciones web suele ser más económico, al ser su desarrollo más sencillo y rápido al existir Frameworks que ayudan en procesos comunes que comparten estas aplicaciones, por ejemplo, el início de sesión, operaciones CRUD (*Create, read, update, delete*), entre otras [30].
- **Diseño fácil y sencillo.** El diseño de las aplicaciones web es más sencillo y fácil de realizar. Al utilizar lenguajes muy simples para su diseño, la complejidad técnica es menor, por lo que puede encontrar más desarrolladores de aplicaciones web y su desarrollo es más asequible que otro tipo de aplicaciones [30].
- **Independientes de hardware y software**. Al ser independiente del dispositivo y del sistema operativo, llega a todos los usuarios, sin ninguna limitación dependiente de la tecnología [33].
- **Independientes de los recursos del cliente.** Las aplicaciones web no pueden hacer uso fácilmente de los recursos del dispositivo del cliente. Al necesitar hacer uso de los recursos del cliente complica su desarrollo y/o debilita las seguridades.
- **Sin estado y sin conexión.** La descarga de la información se da cada vez que el cliente accede a la aplicación. Esto supone una barrera para el usuario, tanto en tiempo como en coste [23].
- **Dependientes de la red.** Las aplicaciones web son dependientes de internet (o de una Intranet) [34], es decir el usuario que necesite acceder a una aplicación web debe contar con acceso a Internet [23].
- **Almacenamiento remoto.** Las aplicaciones web están alojadas en un servidor remoto, siendo accedida por los usuarios que conocen su ubicación o URL. En consecuencia, muchas veces su acceso se ve limitado por los resultados de los buscadores [23, 30, 33].

Características de las aplicaciones web

Entre las principales características de las aplicaciones web se puede citar: Son multiplataforma, fácil y económico mantenimiento, desarrollo económico, diseño fácil y sencillo, independientes de hardware y software, independientes de los recursos del cliente, sin estado y sin conexión, dependientes de la red y almacenamiento remoto.

Para decidirse por el desarrollo de una aplicación web, no solamente se deben analizar sus características, ventajas y desventajas, sino todos los otros aspectos que vienen inmersos en ellos. El hecho de alojar la aplicación (muchas veces conjuntamente con su base de datos) en un servidor remoto, vuelve a los interesados dependientes de terceros, y, se puede estar dando el control del activo más preciado de la empresa que son los datos.

Además, para los desarrolladores (y los clientes) deben considerar que, al ser una aplicación universalmente accedida, deben ser desarrolladas aplicando normas y métricas internacionales emitidas por la World Wide Web (www.w3.org) para garantizar la accesibilidad sin importar las características de los usuarios [35].

1.2.2. Clasificación de las aplicaciones web

Para clasificar a las aplicaciones web en tipos, debemos considerar un punto de vista en particular. Por ejemplo: según la interacción son el usuario, según el servicio ofrecido por la aplicación web, entre los puntos de vista más importantes.

Criterios de clasificación de las aplicaciones web

Se pueden clasificar: Según el tipo de interacción, según el servicio ofrecido, entre los principales puntos de vista.

Según el tipo de interacción

El grado de interacción entre la web y los usuarios (interacción hombre-máquina) es uno de los factores predominantes para determinar los avances de la web. Al analizar este criterio se puede determinar tres tipos de aplicaciones web:

- **Aplicaciones web estáticas.** En realidad este tipo de **sitios web** se deberían llamar como tales: *sitios web estáticos*, en vista de que, al hablar de una *aplicación* web se está hablando de que es capaz de cambiar su contenido según las peticiones del usuario. Sin embargo, para mejorar la **mantenibilidad** del sitio web, éste podrá ser alimentado desde una base de datos sin que el usuario tan sólo lo sospeche. Es decir, la información aparentemente estática podrá ser almacenada en una base de datos y la aplicación web la carga para mostrarla al usuario. Por ejemplo: el listado de productos y/o servicios con sus características, donde el sitio web serviría simplemente para hacer publicidad [36]. Por otro lado, un **sitio web estático** es en el que los archivos solicitados por el cliente se transfieren exactamente como están almacenados en el servidor [37]

- **Aplicaciones web dinámicas.** Este tipo de aplicaciones web son las que cambian su contenido según las peticiones del usuario. Por lo tanto su despliegue está controlado por un servidor de aplicaciones, que es quién atiende las peticiones de los clientes y transmite la información que las satisface [36, 37]. Esté tipo de

aplicaciones no pueden ser desarrolladas únicamente con herramientas como HTML y Javascript. Involucran tecnologías o herramientas como PHP, ASP, ASP.NET, JSP por mencionar algunos. En la figura 1.2 se muestran algunas de las aplicaciones web dinámicas más populares.

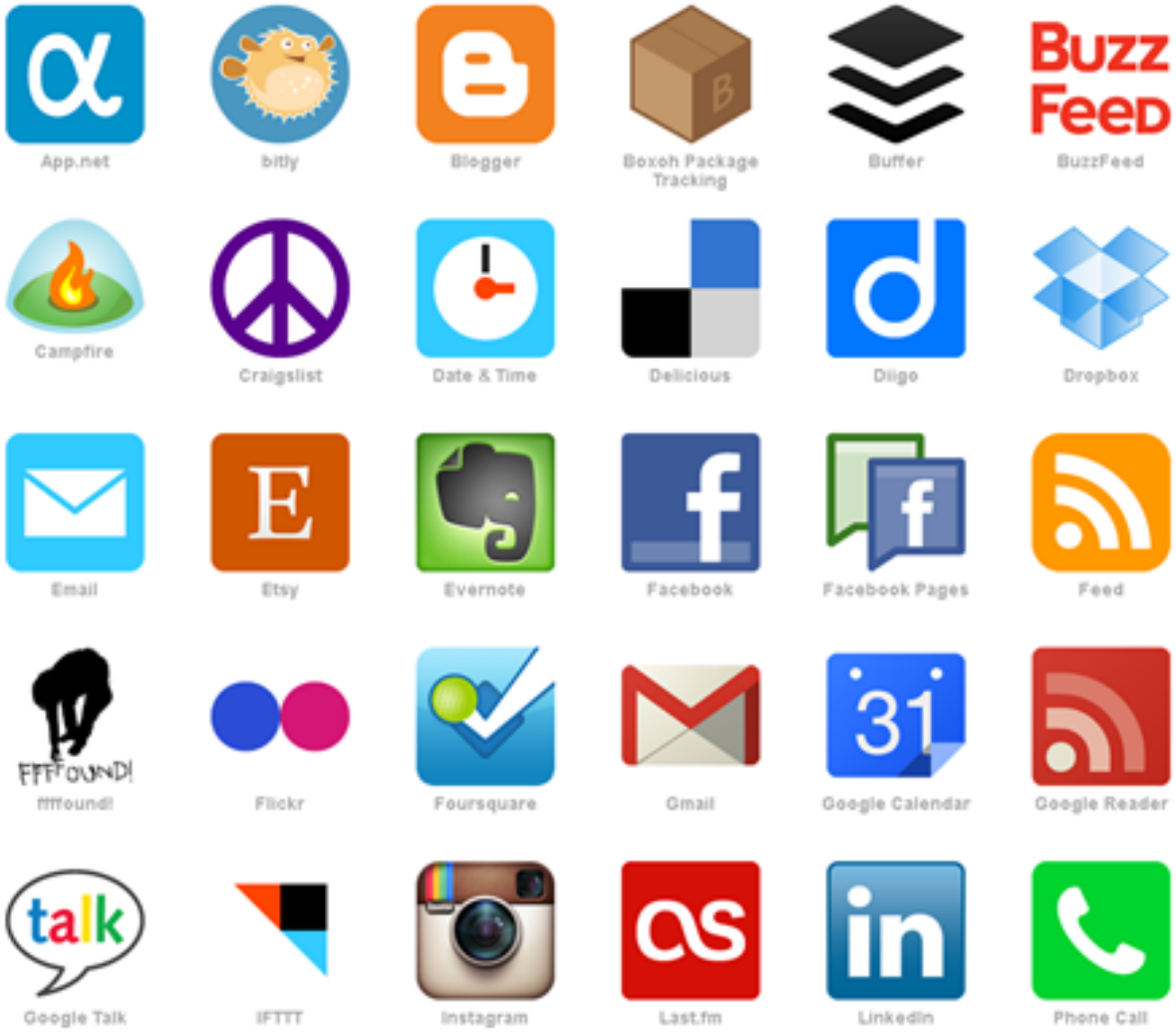


Figura 1.2: Tipos de aplicaciones web

- **Aplicaciones web cliente interactivo.** Son básicamente aplicaciones web estáticos (sitios web estáticos) que tiene un grado considerable de interacción con el cliente, sin embargo no es necesario que intervenga un servidor de aplicaciones para satisfacer las solicitudes del usuario, simplemente el browser o navegador es suficiente para ello. Es decir, la lógica de programación reside en el cliente y consiste básicamente código de lenguaje scripting (javascript, vbscript).

No se debe confundir entre una aplicación simplemente interactiva y una aplicación web dinámica. Estos dos tipos de aplicaciones tienen sus diferencias principales: El acceso a una base de datos y las herramientas con las que fueron desarrolladas.

Según el servicio ofrecido

El tipo de servicio ofrecido por una aplicación web generalmente sirve para identificarla o nombrarla. Siguiendo este criterio y basándose en [38] se pueden clasificar en:

- **Comercio electrónico:** Este tipo de aplicaciones son utilizadas para llevar a cabo las actividades del comercio de manera virtual, es decir, para compra y venta de artículos donde el usuario interactúa con la aplicación web [34].

- **Marketing:** El mecanismo para dar a conocer los productos a los clientes ha sido otro de los procesos parte del marketing (promoción de productos) que se ha visto beneficiado por las aplicaciones web, haciendo posible que la información del producto o servicio llegue a los posibles clientes sin considerar fronteras geográficas.

- **Blog:** Los blogs son sitios web donde los usuarios pueden encontrar información y pueden dejar mensajes a los autores. Dentro de esta línea podemos encontrar las aplicaciones de periódicos o noticias. Estas aplicaciones generalmente se construyen utilizando sistemas de gestión de contenidos (CMS : Content Management System) como WordPress, Prontus, BigPress, entre otros [34, 36].

- **Foros de Discusión:** Las aplicaciones web para los foros de discusión son herramientas que permiten y gestionan discusiones u opiniones sobre un tema específico [39]. Para ello los foros se clasifican por temas (hilos, asuntos), y a su vez los temas se clasifican en subtemas, hasta un nivel muy específico. Esto lo hacen con el fin de que no haya ambigüedad, y así los lectores y/o participantes no se confundan. Los foros de discusión se los puede considerar como un tipo de blogs de noticias. Foros de discusión u opinión hay un sin número, entre los principales se pueden citar a: lifeder, AMDI, Foros 24h, 3D Juegos, ADSL Zone, Amstrad.es Forum, entre otros [36, 39].

- **Alojamiento y compartición de archivos:** Este tipo de aplicaciones se caracterizan por permitirle al usuario organizar su trabajo como el correo electrónico, las agendas, los documentos, ejemplos: Gmail, Hotmail, Outlook, Calendar, Calendly, DropBox, Drive de Google, entre otros.

- **Localización:** Las aplicaciones web en base a mapas para localización de lugares y rutas. Este tipo de aplicaciones permiten la búsqueda de lugares que están clasificados por categorías, dando como resultados las rutas de llegada con cálculos de tiempo según las opciones de viaje. Entre las aplicaciones de localización se puede encontrar a: Foursquare, ViaMichelin, o los sitios como Google Maps.

- **Juegos en línea:** El entretenimiento es otro de las áreas de aplicaciones para el desarrollo web. Los juegos en línea están invadiendo la web. Muchas de estas aplicaciones requieren complementos instalados en los navegadores del cliente,

complementos como Flash y Java para funcionar. Se caracterizan por ser las aplicaciones con mayor interactividad. Entre los juegos en línea que se pueden encontrar están: Adventure Box, Juegos.com, juegajuegos.ws y flyordie.com.

- **Redes sociales:** Las redes sociales fueron primordiales para converger de la web 1.0 a la web 2.0. Las redes sociales son aplicaciones web que permiten compartir información entre los integrantes que han sido agregados o con todos los de la red. Entre las redes sociales más conocidas están linkedIn, Facebook, Twitter por mencionar algunas [34, 40].

- **Multimedia:** Las aplicaciones web que pertenecen a esta categoría se caracterizan por permitir compartir elementos multimedia (imágenes, músicas, videos) con los integrantes de la red. Estas aplicaciones en su mayoría requieren que los navegadores de los clientes tengan instalados complementos como Flash para funcionar correctamente. Los ejemplos más conocidos que pertenecen a esta categoría se pueden citar a:, Instagram, Grooveshark, YouTube entre otros [36].

- **Wikis:** El objetivo principal de este tipo de aplicaciones es ilustrar a los internautas sobre temas específicos y en general, permitiendo la consulta de información sobre estos temas. Los internautas encuentran en estas aplicaciones artículos enciclopédicos editados colectivamente (cualquiera puede editar). La confiabilidad de la información que muestran las wikis es dada por la intermediación o control de ediciones, o de las publicaciones que los internautas realizan. Las wikis son basadas en MediaWiki, Wikipedia, Wiktionary [34, 40].

- **Anuncios:** El objetivo de esta categoría de aplicaciones web es el de permitir a los internautas publicar anuncios ofreciendo o demandando un bien o un servicio, o empleos. Así los demás internautas pueden encontrar lo que necesiten y ponerse en contacto con quien lo publicó. Las aplicaciones web de anuncios se encontran a milanuncios.com, Infojobs. Además, en esta categoría se pueden enmarcar a las aplicaciones que sirven para demandar u ofertar oportunidades de trabajo de freelancer como: freelancer.com.

- **Servicios a la comunidad:** Con la aparición del término "gobierno electrónico", esta categoría de aplicaciones tubo un mayor afianzamiento. Son aplicaciones generalmente implementadas por el gobierno para poner a disposición de los ciudadanos servicios como los ofrecidos por municipios o ayuntamientos, hacienda o rentas, o por el seguro social. Al ser implementadas por los gobiernos podemos encontrar en cada país muchas aplicaciones de esta categoría. A manera de ejemplos tenemos: www.agenciatributaria.es, www.sri.gob.ec, www.guayaquil.gob.ec, www.usa.gov.

- **Banca electrónica:** La característica principal para que las personas accedan a este tipo de aplicaciones es que garanticen la seguridad y la privacidad de los datos [41]. Son aplicaciones que brindan la opción de solicitar productos ofertados (algunas la

única opción) y de realizar transacciones bancarias (como transferencias, depósitos en cheques) a sus clientes, además de consultar datos económicos. En la actualidad se puede afirmar que toda institución bancaria posee una aplicación web con mayor o menor número de opciones para sus clientes y futuros clientes. Por ejemplo: www.bancoguayaquil.com, www.bancomachala.com, www.bancosantander.es, www.usbank.com, www.bankofamerica.com, www.cajaruralgranada.es.

- **Buscadores:** Los buscadores se pueden considerar Mashups , o aplicaciones que se forman por la unión de una o varias aplicaciones. Por ejemplo las aplicaciones que brindan la opción de comparación de precios [36, 42].

1.3. Conclusiones

Las aplicaciones web tienen muchas características determinantes para ser diferenciadas de las aplicaciones de escritorio. Estas características se convierten fácilmente en ventajas frente a las otras. Sin embargo algunas de esas ventajas también pueden ser desventajas, dependiendo del punto de vista. En otras palabras, algunas de las ventajas las mantiene hilo muy delgado que al romperse las convierte en desventajas, dependiendo estas de los investigadores que defienden a uno u otro tipo de aplicaciones. Por otro lado, los tipos de aplicaciones web pueden ser tan extensos o tan reducido según el punto de vista del observador. En este capítulo se ha presentado un número considerable de tipos de aplicaciones web (aun faltan por ejemplo los *Content Management Systems*) para que los lectores tengan la posibilidad de aumentar la lista, y/o agruparlas por sus características y/o servicios que brindan.

Evaluación a los lectores

Para verificar el cumplimiento de los objetivos o resultados de su aprendizaje, el lector debe considerar resolver los siguientes problemas planteados.

- Construir una tabla con las características que reflejen las semejanzas y diferencias entre página web, sitio web, aplicación web y portal web.
- Instalar y configurar un par de servidores de aplicaciones para Java.
- Utilizar alguna aplicación web de uso restringido y de información susceptible (Aplicación bancaria, académica, entre otras.), y una aplicación o sitio web sin restricción (blogs, resultados de la búsqueda de alguno tema, entre otras). Listar las semejanzas y diferencias.
- Con el ejercicio anterior y otros ejemplos, identificar y describir cómo se cumplen las características de las aplicaciones web dictadas en este texto.

- Ejercite sus conocimiento de buscador de información en la web, busque un término a su gusto ("Desarrollo de aplicaciones web", por ejemplo) y cada resultado vaya clasificándolo como aplicaciones web según el grado de interacción con el usuario y según el servicio ofrecido.

Capítulo 2

EVOLUCIÓN DE LA WEB

Objetivos

El lector al finalizar este capítulo será capaz de:

- Caracterizar cada una de las versiones de la web.
- Listar las tecnologías que se utilizan en cada una de las versiones de la web.

Resumen

Todo en el mundo evoluciona, esto hace que cada momento se etiquete con un identificador, generalmente éste es un número. En el área de la informática, se denomina versión (en literatura edición). La Web no ha sido excepción. Hasta la actualidad, han existido 4 momentos de la web bien marcados.

2.1. Introducción

Las tecnologías de comunicación tienen como objetivo primordial producir un nivel de satisfacción en los usuarios, al poderse conectar desde cualquier lugar del mundo [29]. Las tecnologías para la comunicación que domina en el mundo es Internet. Internet se ha convertido en un recurso grande y complejo, que va más allá de todas las fronteras geográficas, políticas, administrativas y culturales, dejando un desafío nuevo e inusual para la sociedad. Internet es utilizado principalmente para la comunicación, búsqueda y compartición de información, negocios y para entretenimiento [43, 44].

Internet es la tecnología que ha hecho posible se desarrollen múltiples recursos que se utilizan para el intercambio de información, comercio electrónico, entre otros aspectos de comunicación. Entre estos recursos, el más importante es la Web, que es el canal principal para el uso de software, hardware e infraestructura de terceros como si fuesen propios.

la Web o formalmente la World Wide Web (WWW) es conocida como el medio de información global más grande del mundo, a través del cual, el usuario puede compartir, leer y escribir datos a través de dispositivos conectados a Internet [4]. la Web fue creada por Tim Berners-Lee. En el año 1989 Berners-Lee propuso los principios arquitecturales básicos de la Web, creando su primer prototipo en 1990. Uno de los objetivos claros que tenía de la Web, era la flexibilidad, por lo que primó el principio de las mínimas especificaciones. El lenguaje utilizado para la escritura de documentos de hipertexto fue HTML, que también fue otra creación de Berners-Lee. Además, Bermers-Lee, también dió solución a la no existencia de un protocolo con las funciones necesarias y lo suficientemente rápido para ser usado en la Web. En 1989 él diseñó el protocolo HTTP junto con los primeros servidores WWW [45–48].

la Web e Internet no son tan diferentes. Internet es una red de redes donde millones de computadoras están conectadas globalmente formando una **red de redes** dónde cualquier computadora puede comunicarse con otra. Mientas que la Web es una forma de acceder a la información a través navegadores de Internet, que son quienes acceden a las páginas web y las visualizan para los usuarios.La información en la Web está conectada por hipervínculos, y su contenido puede ser texto, gráficos, audio y video [29, 49].

Se debe destacar el papel fundamental de Tim Berners-Lee en todo lo que respecta a la Web. En 1994 él fundó el World Wide Web Consortium (W3C), siendo actualmente el motor del desarrollo de los estándares predominantes para trabajar en la web (http://www.w3c.org).

En este capítulo se describe la evolución de la Web; indicando su definición, características, seguridad, tecnologías y principios de cada una de las versiones de la Web.

2.2. Versiones de la Web

la Web ha pasado por una evolución sorprendente. La Web 1.0 que se identifica por visualizar contenido estático. La Web 2.0 se caracteriza por permitir un contenido dinámico o interactivo, además de la llegada masiva de navegadores web. La Web 3.0 se identifica por el contenido colaborativo, ajustándose a los intereses de los usuarios. Los sitios web pueden comunicar a los usuarios un significado entendible e interpretable por las máquinas, de tal manera que existe una gran variedad de recursos para buscar y encontrar contenidos y servicios de interés que ofrece la Web [50]. Y La Web 4.0 que se caracteriza por ser la Web emotiva, basándose en un ancho de banda de alta tecnología que permite al usuario obtener un mejor contenido [51].

2.2.1. La Web 1.0 - Web Estática

En los inicios de la Web, no se pensó en versiones, es así que, cuando aparecieron cambios sustanciales en la Web, los investigadores consideraron definir segunda versión

de la Web, dejando a sus inicios como la versión inicial o versión 1.0 (Web 1.0). Por lo tanto, su periodo de duración fue desde su creación en el año 1990 [45, 52], hasta el año 2000. Se representaba de forma de sólo lectura, la cual existían pocos webmaster o productores de páginas web, es decir, una persona que se dedica a desarrollar una página web y agregar contiendo a la misma. Los usuarios únicamente podían visualizar datos estáticos y no podían interactuar con la página. Como se ilustra en la figura 2.1, la interacción se la consideraba como una funcionalidad limitada [48, 53].

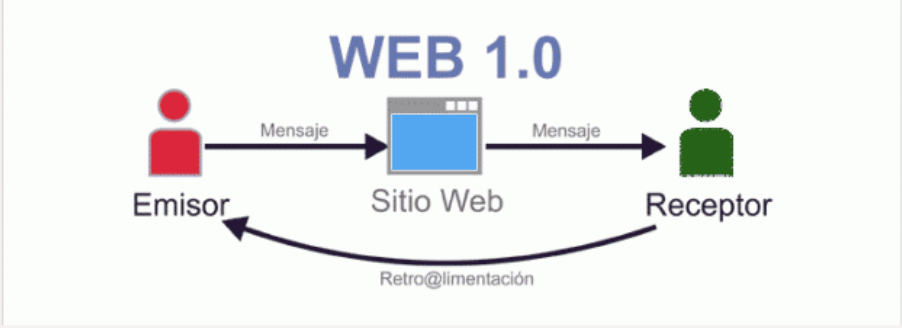


Figura 2.1: Interacción en La Web 1.0

La Web 1.0 fue utilizada principalmente para proporcionar información sobre sociedades empresariales, noticias u otros contenidos [54].

Características

Las características de La Web 1.0 están relacionadas principalmente por el tipo de contenido, la interacción con el usuario y el tipo de comunicación entre clientes (navegadores) y servidores (www). Las principales características que podemos mencionar son [48,51]:

- **Páginas estáticas:** Las páginas web se consideraban estáticas por motivo de que los usuarios sólo podían acceder a contenidos web predeterminados y sin poder interactuar con aquellos contenidos. Predominaba HTML como herramienta de desarrollo (ver figura 2.1) [53].

- **Carga rápida de las páginas:** Al ser contenido web predeterminado, y la información que se mostraban era sólo de texto, con pocas imágenes y sin ningún archivo multimedia, hacía que el tiempo en cargar el contenido en el navegador del cliente era mínimo.

- **Dispersión del consumidor:** Se refiere a que pocos usuarios podían añadir contenido o comentarios [53].

- **Contenido creado por organizaciones:** La mayoría de las empresas crearon sus sitios web, básicamente con anuncios de compras y ventas de sus productos [53].

- **Baja frecuencia de cambios:** Añadir o cambiar algo en la página web requería de tiempo y esfuerzo, debido a que las páginas web estaban escritas en código HTML sin ayuda de programas o herramientas.
- **Falta de normas:** Al no contar con normas de diseño de páginas web, las páginas creaban un desconcierto en el usuario.
- **Falta de autenticación:** Todos los usuarios eran considerados visitantes, por motivo de que no se usaban bases de datos para realizar los inicios de sesión.
- **Muy pocos buscadores y navegadores:** Los buscadores en la Web 1.0 eran Yahoo!, AltaVista y los motores de búsqueda más populares. Google no se considera como un buscador de la Web 1.0. Así mismo existían pocos navegadores. Los navegadores de esta versión de la Web era NetScape, Internet Explorer, entre los más usados.
- **Navegación poco estructurada:** la navegación, causaba que las direcciones o enlaces en las páginas no tuvieran un fin [29,55].
- **Interacción con el usuario casi nula:** al ser una web donde los contenidos estaban preestablecidos y para realizar algún cambio debía ser hecho por el webmaster, hace que en esta versión la iteración con el usuario se limite a unicamente rellenar un formulario y enviar datos por medio de correo electrónico, como se ilustra en la figura 2.1.
- **Diseño sencillo de páginas web:** El diseño consistía en el empleo de saltos de línea y líneas horizontales como separadores, se usaban listas para organizar la información, quedando los diseños similares a un documento en papel [55].

Características de la Web 1.0

Entre las principales características la Web 1.0 se puede nombrar: Carga rápida de las páginas, dispersión del consumidor, contenido creado por organizaciones, baja frecuencia de cambios, falta de normas, falta de autenticación, muy pocos buscadores y navegadores, navegación poco estructurada, interacción con el usuario casi nula, y diseño sencillo de páginas web.

Seguridades

En la Web 1.0 no se preocupaban por las seguridades. El servidor web hacía el papel de un servidor de archivos, al cual el cliente solicitaba un archivo HTML y el servidor web le entregaba el contenido para ser visualizado en el navegador (cliente). Los sitios web eran creados por las propias organizaciones para publicar documentos con información estática, sin mayor importancia y esta información estaba disponible para todos quienes accedían a la Web.

Tecnologías usadas

Las tecnologías utilizadas en la esta versión fueron HTML, HTTP y URL, siendo protocolos web básicos. HTML para el diseño o escritura de las páginas web. HTTP como el protocolo para la solicitud de información de parte del cliente (solicitud de archivos HTML) y para responder (retornar el contenido del archivo HTML solicitado). Y URL como su significado lo dice, como localizador universal de recursos. Además, está versión no soportaba comunicaciones de doble sentido, por lo tanto fue basada únicamente en una plataforma unidireccional.

Principios

Los principios que rigieron a la Web 1.0 fue el principio informativo al publicarse únicamente archivos con información que el webmaster determinaba, y el principio de acceso universal a la información, ya que la información estaba disponible para todo el mundo. En la Web 1.0 no existía la autenticación para acceder a la información.

2.2.2. La Web 2.0 - Web Social

La Web 2.0 conocida como la Web Interactiva. En este versión el usuario puede ingresar datos y recibir información. Por lo tanto, con La Web 2.0 aparecen nuevas tecnologías en Internet y las maneras de utilizarlas [29]. Sin lugar a dudas la web 2.0 inició con una nueva era de interactuar en la red [53]. Muchas de actividades cotidianas, se volcaron a la red, como por ejemplo buscar información, hacer compras, estudiar, hasta concertar citas pueden realizarse sin problemas, e incluso a menudo, más barata por la Red [56]. Esta era inició en el año 2004 [52, 57]. La Web 2.0 se dio con la aparición de las redes sociales, es por eso que es conocida como **la Web Social** [58, 59]

La Web 2.0

La Web 2.0 es conocida como la Web Social, se caracteriza por la alta interacción con el usuario y por el aporte del usuario en la alimentación y creación de aplicaciones y/o sitios web.

Las aplicaciones de la Web 2.0 ganaron importancia en la sociedad de ese entonces e incluso en la actualidad no podría decirse que la Web 1.0 o que la Web 2.0 está lejos de la sociedad. La Web a evolucionado, las nuevas tecnologías han emergido y deben y están siendo aprovechadas tanto por el gobierno como por la empresa privada. La Administración Electrónica como el comercio electrónico han tomado muchas nuevas direcciones. La Web 2.0 tiene mucho potencial para el sector público en términos de interacción, participación y transparencia (ver figura 2.2). Sin embargo, con las bondades que presenta también vienen inmersos los desafíos que se deben superar, por lo tanto, es importante tener en cuenta también los riesgos potenciales de las aplicaciones de la

Web como el aislamiento, la exclusión, la violación de la intimidad y principalmente el mal uso de la información de las personas [60].

Figura 2.2: Interacción en la Web 2.0

Características

Al seguir la misma línea de la Web 1.0, las características de la Web 2.0, se relacionan con el contenido, el grado de interacción con el usuario, el tipo de comunicación entre los navegadores y la WWW. Como características principales se pueden mencionar a las siguientes [52]:

- **Las páginas dejan de ser estáticas**. Esto significa que el contenido es creado por los usuarios, es decir, la Web dispone de aplicaciones web que permiten crear contenido, disminuyendo la dependencia de los diseñadores web para tener su propia página web. Las más utilizadas fueron las aplicaciones para crear los blogs, RSS (*Really Simple Syndication*), SNS (*Social Networking Services*) [53].

- **Carga de las páginas**. Se pude pensar que, la carga de las páginas de la Web 2.0 es más lenta que las páginas de la Web 1.0 porque incluyen elementos interactivos, lógica de programación y consultas a bases de datos que se deben ejecutar en el lado servidor para satisfacer las solicitudes de los usuarios. Sin embargo, si comparamos esta tarea en las épocas en las que tuvieron su advenimiento, eso sería falso, porque así como han surgido nuevas tecnologías para el desarrollo web, así mismo el hardware ha evolucionado potenciando sus capacidades y disminuyendo su valor económico.

- **Dispersión del consumidor**. Mientras en la Web 1.0 eran muy pocos los que podían agregar contenido, en la Web 2.0 el abanico de usuarios se extendió considerablemente hasta llegar a considerarse que el límite sería el número de personas que acceden a Internet. Esta afirmación se hace en vista a la cantidad de personas que acceden a Redes Sociales. Las redes sociales son las favoritas para agregar contenido y múltiple interacción con el usuario [61].

- **Propietarios de los sitios web**. Antes de la Web 2.0 los sitios web era exclusivos de grandes empresas y organizaciones. Eso cambió con el surgimiento de la Web 2.0. la Web es de todos. Con la facilidad para la creación de sitios web, también surgió la facilidad (técnica y económica) para la obtención de un dominio web y su administración, esto hizo imparable el crecimiento de la Web hasta la fecha.

- **Frecuencia de cambios**. Los cambios en las páginas de los sitios web ya no requieren de mucho tiempo ni de mucho esfuerzo, ya que, la existencia de aplicaciones y herramientas que ayudan en estas tareas.

- **Estándares de diseño web**. Hasta que se fundó el Proyecto de estándares Web (*Web Standards Project*) con sus siglas en inglés **WaSP** (`http://www.webstandards.org`) en 1998, cada fabricante de navegadores producía sus propios elementos y maneras para manipular los documentos web. Con esta fragmentación lo que se estaba logrando era la negación a los usuarios del acceso a la información. Es por eso que, el objetivo principal de WaSP era conseguir que los fabricantes de navegadores apoyaran los estándares establecidos por el W3C [62]. Por ende, WaSP impulsa y difunde el diseño de sitios web que puedan acceder desde la más amplia gama de tecnologías y entornos de navegación web, estos incluyen a los navegadores viejos y modernos, y a las tecnologías de asistencia; y al mismo tiempo impulsa el desarrollo de tecnologías de navegación web y de asistencia que apoyen los estándares abiertos. [62, 63].

- **Autenticación de Usuarios**. La identificación de las personas que acceden a la información fue requerida más allá de conocer el número de veces que acceden a la información (Web 1.0), se requiere conocer el número de usuarios (personas) que acceden. Además de tratar de identificar quién es el que accede a la información. Las aplicaciones de La web 2.0 piden al usuario se registre y se identifique (Facebook, Twitter, Amazon.com, etc.), otros sitios simplemente solicitan al usaurio que ingrese un correo electrónico para reconocerlo como un usuario más que accede a la información que ponen a su disposición.

- **Abundantes buscadores y navegadores**. Aún sobreviven los navegadores y los buscadores de la era de la Web 1.0 en la nueva era de la Web, en la 2.0. Sin embargo, hoy en día existen muchos navegadores y buscadores. Por ejemplo, como navegadores tenemos: Internet Explorer, Mozilla FireFox, Safari, Google Chrome, Opera, Avant Browser, NetScape Navegator, Maxthon, Flock, Seamonkey, entre otros. Por otro lado, entre los bucadores tenemos: Google, Baidu, Bing, Yahoo!, Yandex, DuckDuckGo, Ask, Naver, Ecosia, Seznam, AOL Search, Dogpile, Qwant, Good Searchz, Daum, Start Page, Goo.

- **Navegación estructurada**. Uno de los problemas en la navegación, era el uso de marcos (*frames*). Con la Web 2.0, estos problemas fueron atacados con el uso de otros elementos para la maquetación de los sitios web, como tablas, cajas (div)

entre otros; y así se eliminó los problemas de desorientación de la navegación. Estos problemas muchas veces se les escapaban de las manos a los desarrolladores web. En la actualidad, cuidan mucho la estructura de la información para brindar sitios con una navegación nada lineal, es decir, las páginas contienen redundancias de enlaces o elementos de acceso a la información que brinda el sitio web [64]. Pocos sitios web pueden ofrecer un estilo de navegación en estrella[1], teniendo como centro a la página de inicio (*home*) desde la cual van y a la que vienen el resto de páginas.

- **Interacción con el usuario**. La interactividad es la principal diferencia entre las dos primeras eras de la Web. La Web 1.0 con una interactividad casi nula, mientras que en la Web 2.0 abundante interactividad. La Web 2.0 logró una diferencia notable de Internet con cualquier otro medio de comunicación [57]. Pasa de usar sitios web para leer la información y a lo sumo enviar un mensaje o comentario rellenando un formulario a modificar el contenido de las páginas web y a crear sus propios sitios web.

- **Diseño de sitios web**. Así como la interactividad, el diseño de los sitios web pasó de diseñar sitios web sencillos, llenos de texto, con una u otra imagen, a tener que diseñar sitios web con muchas posibilidades de interacción con el usuario [53]. La maquetación de sitios web en la Web 2.0 ha ido del uso de tablas, hasta el uso de cajas, en ambos casos con el uso de las hojas de estilo en cascada o CSS (*Cascade Style Sheet*) [65]. Así mismo desde el uso de herramientas de diseño web hasta el uso de frameworks y los sistemas de gestión de contenido conocidos como CMS por sus siglas en inglés (*Content Management System*) entre otras tecnologías utilizadas.

Características de la Web 2.0

Las principales características la Web 2.0 son: Las páginas dejan de ser estáticas, la carga de las páginas es un poco lenta, dispersión del consumidor, se masifica un poco los propietarios de los sitios web, aumenta la frecuencia de cambios, aparecen los estándares de diseño web, es necesaria autenticación de usuarios, proliferan los buscadores y navegadores, se hace una navegación estructurada, aumenta el grado de interacción con el usuario, diseño de sitios web con múltiples posibilidades de interacción con el usuario.

Las aplicaciones de la Web 2.0 más utilizadas fueron (datos de 2019) los blogs, los RSS, los SNS, los medios para compartir fotos y vídeos y las herramientas de mensajería instantánea. Facebook fue la aplicación más utilizada (61,3 %), seguida de RSS (53,3 %), Twitter (46,7 %), YouTube (37,3 %), los blogs (18,7 %) e Instagram (17,3 %); el podcasting resultó ser la menos utilizada, con un 4 % [58].

[1]Tipos de mapas de navegación Multimedia: https://www.jhonurbano.com/2013/06/tipos-mapa-de-navegacion-multimedia.html

Seguridades

Mientras la tecnología web avanza, los problemas de seguridad avanzan con ella, es así que la Web 2.0 trajo consigo algunas vulnerabilidades de las que los desarrolladores e investigadores deben preocuparse. Entre las que tenemos:

- **Defectos de autenticación y autorización**. En los inicios de la Web 2.0, e incluso hasta el momento, se pueden encontrar varias debilidades en la técnica de autenticación y autorización. Entre las más importantes [29,66]:
  - Las contraseñas no caducan.
  - Al fijar la longitud y la complejidad de las contraseñas.
  - Contraseñas predecibles, es decir, se pueden adivinar (falta de protección contra la fuerza bruta).
  - ID de sesión pueden ser predecibles.
  - La mayoría de las veces no caducan.
  - No hay límites de tiempo para la vida útil de los ID de sesión (no expira la sesión).
  - Códigos Captcha con fallos.
- **Falsificación de solicitud entre sitios** o en inglés *Cross Site Request Forgery* (CSRF). Con este tipo de ataque un pirata informático pretende tener acceso a un Firewall, a un comercio electrónico, a la Intranet de la empresa u otros sitios web que la víctima ha utilizado con acceso/autenticado. Esto lo logra, si puede utiliza la dirección IP o las '*cookies*[2] de la víctima. Lo que puede obtener el atacante puede ser muy dañino para la víctima como: retirar dinero de sus cuentas bancarias, comprar cosas en sitios de comercio electrónico, robar datos de la intranet de la empresa o cambiar la configuración del enrutador o del firewall local [19,29,67,68].
- **Fuga de Información**. Cuando los desarrolladores dejan sin uso alguna páginas, sin embargo no las quitan del servidor, alguna persona (pirata informático) puede escribir un nombre de un recurso cualquier y puede coincidir con alguno que por error está en el servidor. Y si éste muestra información confidencial (archivo con código PHP pero que no pueda ser interpretado **indexphp** - index.php) el sitio e incluso otros elementos de la red pueden estar en peligro: Servidores de bases de datos, servidores de aplicaciones, etc [29,66].
- **Gusanos XSS**. Un gusano XSS es código capaz de autopropagarse. Por lo tanto, para irrumpir por medio de estos ataques, un pirata informático inyecta código XSS en las **páginas web/aplicaciones web**. Su objetivo es que se propague cuando los usuarios visiten esa página [29,69].

[2]Una *cookie* (galleta informática) es una pequeña información enviada por un sitio web y almacenada en el navegador del usuario

- **Inyección SQL**. Inyección SQL es un método para agregar código malicioso que, valiéndose de una vulnerabilidad informática presente en una aplicación a nivel de validación entradas para ejecutar transacciones sobre una base de datos [29,70]. Un atacante no sólo puede acceder a la aplicación web y a la base de datos, dependiendo de la base de datos, el atacante puede acceder al sistema operativo. [29]. De modo similar, en la inyección de Xpath, el atacante altera una consulta XML para lograr sus objetivos.

  - Cuando el cliente proporciona la entrada a un sitio web, se crea una consulta Xpath para los datos XML, por lo que un atacante puede enviar algunos datos intencionalmente para averiguar cómo están estructurados los datos XML o acceder a los datos que normalmente no tiene acceso.
  - En la inyección JSON, el atacante inyecta código JavaScript malicioso en JSON en el sitio del cliente, cuando el cliente inicia sesión en los sitios que el atacante quiere vulnerar como un usuario autenticado, el atacante envía un enlace a las víctimas y las convence para que visiten esos sitios infectados, creados por el pirata informático. Si el cliente visita los sitios infectados mientras ya está conectado a los sitios de destino, toda la información presente en esos sitios de destino se enviará al pirata informático.
  - **Formar una tautología**. Cuando con seguridad el sistema puede estar ejecutando una consulta *SELECT* con una clausula *WHERE*, con los datos ingresados, el pirata informático puede ingresar partes de la consulta que van a ser utilizados en la cláusula *WHERE*, de manera que siempre de como resultado Verdadero (*TRUE*). Una de las opciones sería en el momento de autenticarse como usuario de un sitio. Por ejemplo, en la la figura 2.3, sin importar el nombre de usuario que se ingreso (por ejemplo **"palabra"**), y en la entrada de la contraseña se ingrese **cualquier_palabra' OR '1'='1**, se estaría formando una tautología, si el desarrollador del sitio web comprobara la existencia de ese usuario con una programación como:

```
1  SELECT COUNT(*) FROM tabla
2  WHERE campo1='palabra'
3  AND campo2='cualquier_palabra' OR '1'='1'
```

Listado 2.1: Ejemplo de *SQL Injection*

- **Mashups**. Los *mashups* surgen como la combinación de todos los recursos o servicios de varios sitios web en una única experiencia de usuario. Un *mashup* puede conectarse dinámicamente a sitios web que no están necesariamente bajo el control del proveedor, lo que se convierte en un riesgo de seguridad para los proveedores de contenido [29].

- **Secuencias de comandos entre sitios**. *Cross Site Scripting* o (XSS). El ataque XSS tiene como objetivo tomar el control de la computadora del usuario atacado, para

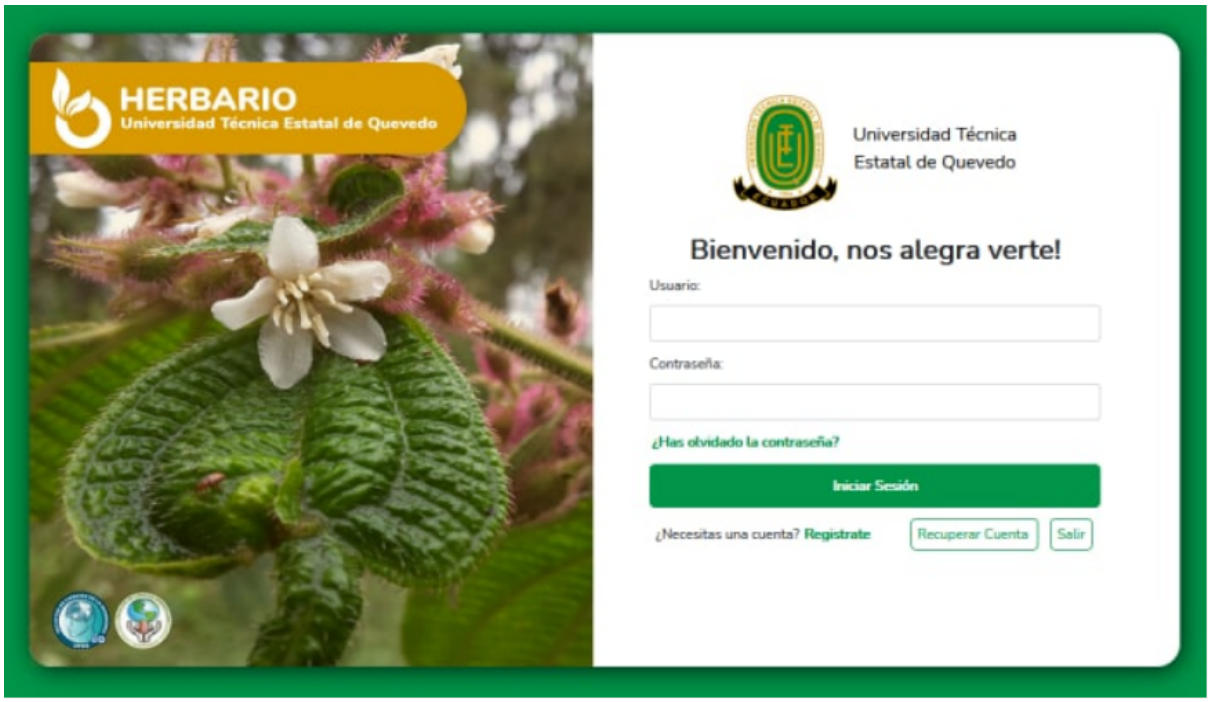


Figura 2.3: Inicio de Sesión de la Aplicación Herbario de la UTEQ - https://bioforest.uteq.edu.ec/HerbarioUTEQ/login.html

ello, los piratas informáticos inyectan su propio código ejecutable en páginas web legítimas generadas dinámicamente. Cada vez que alguien visita o descarga el contenido ofrecido en esa página web en particular, el código que reside dentro de ella (inyectado) se ejecuta en la computadora de las víctimas, de esta manera el **pirata informático** logra su objetivo [29, 69]. XSS tiene diferentes tipos, desde ataques persistentes a no persistentes, y su impacto varía desde daños en el diseño (apariencia) considerándose como una alerta de aviso, hasta daños severos como robo de cookies[3] al violar la privacidad de los usuarios. Hay tres tipos de ataques XSS reportados en la literatura: **XSS almacenado, reflejado y basado en DOM**. Sus comportamientos son similares, de hecho se superponen. En consecuencia, se considera que XSS tiene dos tipos que son XSS de cliente y XSS de servidor [29]:

- Cliente XSS ocurre cuando el DOM se actualiza en una **llamada JavaScript insegura**, a través de los datos proporcionados por el intruso. Si la intrusión es de un usuario, se denomina **"XSS de cliente reflejado"**; de lo contrario, se denomina **"XSS de cliente almacenado"** en el caso de datos maliciosos que se originan en un servidor web.
- De manera similar, el servidor XSS se debe a que los datos no confiables se

[3]Las cookies son archivos que crean los sitios que el usuario visita para guardar información de los usuarios (usuarios, contraseñas, entre los más delicados) sobre su navegación para hacer que su experiencia en línea sea más sencilla.

convierten en parte de la respuesta generada por el servidor en forma de HTML. El servidor XSS se convierte en un **"servidor XSS reflejado"** cuando la fuente de datos maliciosos es un usuario. Se denomina **"servidor XSS almacenado"** si la respuesta generada se origina en un servidor [71].

- Muchos otros casos de inyección SQL los puede encontrar en el documento [72]. A criterio del autor, los desarrolladores web deben conocerlos para prevenir los ataques.

Tecnologías o herramientas de la Web 2.0

Gracias al desarrollo de las tecnologías surge con éxito la Web 2.0. Las tecnologías que dieron lugar a esta era de la Web, y por lo tanto, las que los desarrolladores web utilizaron para crear las pequeñas y grandes aplicaciones que marcaron la diferencias entre la Web 1.0 y la Web 2.0, entre las principales se pueden mencionar [29,52,68,73,74]:

- **Distribución de contenido**. La compartición de manera adecuada de contenido como RSS, Atom[4], RDF[5] se utilizan para la creación de servicios web 2.0 [48]. Atom es otra especificación de sindicación destinada a resolver los problemas de las múltiples versiones incompatibles de RSS.

- **Tecnología de Internet basada en Ajax**[6]. AJAX significa JavaScript y XML asíncronos. Es tan popular que ya no se lo trata gramaticalmente como un acrónimo sino como una palabra. Es un enfoque para crear aplicaciones web. Enriquece la interfaz de usuario, y hace que las páginas web sean más interactivas, más rápidas y más fáciles que los sitios web tradicionales basados en otras tecnologías. Reduce la cantidad de datos que deben recargarse cada vez que hay una solicitud del cliente al servidor. AJAX En realidad, se trata de varias tecnologías que se unen de forma potente: XHTML o HTML, hojas de estilo en cascada (CSS), JavaScript y XML [75–77].

- **DOM**[7]. Modelo de Objetos de Documento o Modelo en Objetos para la Representación de Documentos [78]. DOM es una API que representa a un documento HTML o XML en una estructura de árbol. DOM permite el acceso dinámico a través de la programación para acceder, añadir y cambiar dinámicamente contenido estructurado en documentos con lenguajes como JavaScript.

- **REST**[8]. La transferencia de estados de representación es un enfoque para obtener el contenido de la información de una página web, así mismo para realizar peticiones al servidor. Aunque se creó con un fin un poco reducido, actualmente

[4]El Formato de Redifusión Atom es un fichero en formato XML usado para Redifusión web.
[5]Marco de Descripción de Recursos (del inglés Resource Description Framework, RDF)
[6]AJAX del inglés: Asynchronous JavaScript And XML
[7]*Document Object Model*
[8]*Representational State Transfer*

es ampliamente usado. Los servicios web se pueden implementar basados en esta arquitectura de software.

- **XML**[9] **y CSS**[10]. El lenguaje de marcado extensible se ha utilizado para la gestión de la información y la personalización de la descripción de cualquier tipo de dato. Por su parte las hojas de estilo en cascada se han utilizado para mejor la apariencia (el diseño) de las páginas agregándoles estilos [75,78].

- **Blogs**. Las herramientas para la creación de los blogs han hecho que usuarios inexpertos en desarrollo web puedan crear sus propios sitios web para intercambiar información con los demás usuarios. Un blog es una página web que les sirve a los autores para publicar información de su interés personal que puede ser requerida y/o analizado por otros usuarios. Generalmente estos usuarios dejan sus comentarios sobre el material compartido [57]. Las herramientas que le permiten crear sus blogs, se pueden mencionar a: *Movable Type*[11], *WordPress*[12], *Blogger*[13], entre otros [75,79]. La elección del software adecuado para la creación y gestión de un blog para aplicaciones empresariales requiere que se consideren varios factores. Estos factores se describen muy bien en [75].

- **Really Simple Syndication (RSS)**. Sindicación realmente simple o RSS es una familia de formatos de fuentes web utilizados para distribuir contenidos de blogs o páginas web. RSS consiste en un archivo XML que resume los elementos de información y enlaza con las fuentes de información. Notifica a los usuarios sobre las actualizaciones de los blogs o sitios web de su interés. Los **alimentadores RSS** de páginas web o blogs suelen estar enlazados con la palabra "subscribirse", o con las letras XML o RSS en un recuadro naranja [75,79].

- **Wikis**. Un wiki es un sistema de autoría colaborativa (o gestión de contenidos) sencillo y potente a la vez. Está basado en la web para crear y gestionar contenidos. Permite que cualquier usuario añada un nuevo artículo, o editar y complementar un artículo existente a través de un navegador web. Permite además, que los usuarios pueden hacer un seguimiento de los cambios realizados en un artículo [57]. El término wiki deriva de la palabra hawaiana wikiwiki, que significa rápido o veloz. La enciclopedia en línea generada por el usuario Wikipedia (http://en.wikipedia.org) es un wiki [75]. Entre las características de un Wiki se pueden citar [75]:

    - Un lenguaje que permita formatear texto y vincular documentos y contenidos externos. Este lenguaje es el “Wikitext”.

[9] *EXtensible Markup Language*
[10] *Cascade Style Sheet*
[11] https://www.movabletype.org/
[12] https://wordpress.com/
[13] https://www.blogger.com/

- Un sitio web con sus estructura y navegación simples. Los usuarios creadores de contenido pueden vincular sus páginas fácilmente una a otra. Debido a que la jerarquía y la estructura de estos sitios son planas, la navegación es simple.
- Plantillas simples. Cuando el usuario hace la petición al servidor de una página de wikitexto, la aplicación wiki transforma las marcas wiki a marcas HTML y crea enlaces entre páginas, y envuelve este contenido convertido en una plantilla para proporcionar un aspecto uniforme y coherente a todas las páginas del sitio wiki.
- Los enlaces son creados por la aplicación del wiki, basándose en el título de la página, liberando al autor de la tarea de buscar qué URLs usar para enlazar una página con otra dentro del wiki. Así mismo, los hipervínculos a las páginas del wiki se crean automáticamente.
- Flujo de trabajo sin control. Se puede realizar cualquier tarea de gestión de contenidos sin supervisión o aprobación de algún personal editorial. Su gestión se hace mediante la supervisión de los cambios, y la capacidad del wiki para retroceder a una versión anterior y evitar el spam [79]. Si fuese necesario se puede controlar el acceso y los privilegios de los usuarios.
- La búsqueda de información o temas específicos dentro de un wiki se hace utilizando palabras clave asociadas. Para este servicio poseen una función de búsqueda integrada [79].

- **Mashups**. La tecnología Mashup es la tecnología más utilizada para el desarrollo rápido de aplicaciones [73, 80]. Las tecnologías tradicionales para el desarrollo rápido de aplicaciones (RAD: *Rapid Application Development*) están siendo reemplazadas por la tecnología Mashup y otras relacionadas. Mientras los usuarios colaboren compartiendo más componentes, APIs y Widgets, seguiran siendo las tecnologías más usadas en materia RAD. La tecnología Mashup es una de las representaciones más populares de las aplicaciones de software, que se generan mediante la combinación de diferentes tipos de APIs y recursos de datos.

Tecnologías usadas en las aplicaciones de la Web 2.0

Las principales tecnologías de la Web 2.0 son: Las que permiten la distribución de contenido, tecnología de Internet basada en Ajax, DOM, REST, XML, Blogs, Really Simple Syndication (RSS), Wikis, Mashups.

Principios de la Web 2.0

Una de las características de la Web 2.0, es la gran interactividad del usuario. La interactividad de los usuarios va desde el ingreso de datos para modificar la página del sitio web (indirectamente), hasta la creación de nuevos sitios web (directamente). Los

usuarios interactúan en línea de muchas múltiples maneras, añadiendo valor y creando nuevas oportunidades [81].

Según [81] La Web 2.0 refleja la maduración de Internet como medio de comunicación, centrado en el usuario, descentralizado y colaborativo. Además Musser y O'Reilly, establecen cinco principios y mejores prácticas que se deben aplicar para que la Web 2.0 alcance el éxito. Los principios y mejores prácticas que Musser y O'Reilly plantean son los siguientes [81]:

- Los usuarios añaden valor. Los usuarios añaden valor directamente a través de la participación activa e indirectamente como efecto secundario de sus acciones. Los usuarios crean contenidos, comentan, chatean, suben, comparten, recomiendan, enlazan, agregan, filtran, buscan e interactúan en línea de muchas otras maneras. Cada una de estas acciones añade valor y crea nuevas oportunidades.

- Detalla los problemas que resuelve cada patrón o las oportunidades que crea.

- Explica las tendencias del mercado tecnológico, económico y demográfico que impulsan el desarrollo y la adopción de la Web 2.0. Cada día aumenta el número de usuarios de Internet, con un crecimiento de casi el 900 % de 400 millones en el año 2000 a más de 4352 millones de usuarios para el 2019 [82, 83]. Internet ha tenido un impacto sin precedentes en las economías y sociedades de todo el mundo [82].

- Ilustra las mejores prácticas a través de estudios de casos de líderes de la industria. Las 5 principales compañías que se encuentran en la capa de aplicaciones en la actualidad son Alphabet [14] (compañía matriz de Google), Amazon[15], Tencent[16], Facebook[17] y Alibaba[18], son entre otras, las responsables de la masificación del uso de Internet y del cambio en la forma de hacer negocios [82].

- Proporciona herramientas para la autoevaluación práctica. La autoevaluación en el camino al mejoramiento continuo. Ponga en práctica la autoevaluación como obtención de datos para su análisis.

- El software no limitado a un solo dispositivo. Debe estar consciente que los usuarios están utilizando una gran variedad de dispositivos para acceder a Internet, como las tradicionales pantallas de computadores de sobremesa, teléfonos inteligentes, tabletas, relojes inteligentes, entre otros. Y,

- Las experiencias enriquecedoras para los usuarios. Las características con las que nació la Web 2.0 hacen que el usuario tenga las mejores experiencias en la Web.

[14] http://www.google.com
[15] https://www.amazon.com
[16] https://www.tencent.com/
[17] https://www.facebook.com
[18] https://www.alibaba.com/

2.2.3. La Web 3.0 o ¿la Web Semántica?

La Web 3.0 es una nueva versión de la Web. Aunque la versión 2.0 se puede considerar muy diferente a su predecesora, no sucede eso entre la Web 2.0 y La 3.0. Sin desentonar y guardando la armonía, se puede decir que la Web 3.0 es una Web extendida, con el mayor significado, brindándole al usuario la posibilidad de encontrar respuestas a sus preguntas de formas más significativa y rápida [29, 57]. Los expertos en Internet no han unificado sus enfoques y opiniones sobre la actual versión de la Web, La Web 3.0 o ¿Web Semántica? [53]. Tampoco se ponen de acuerdo en su fecha de aparición. Según Latorre en [52], Web 3.0 como concepto apareció en 2006 y considera que en la práctica, como una versión de la Web lo hizo en el año 2010. Los principales expertos en tecnologías de la información consideran la Web 3.0 como una web semántica y de personalización [29].

La Web 3.0

Las principales tecnologías de la Web 2.0 son: Las que permiten la distribución de contenido, tecnología de Internet basada en Ajax, DOM, REST, XML, Blogs, Really Simple Syndication (RSS), Wikis, Mashups.

Eric Schmidt, el director general de Google, definió (Alrededor de 2008) a la era de la Web 3.0, como la era cuando la construcción de las aplicaciones sea de una forma diferente. Su predicción era que la Web 3.0 se vería en última instancia como aplicaciones que se construyen en conjunto, que serían aplicaciones relativamente pequeñas, los datos estarán en la nube, las aplicaciones podrán funcionar en cualquier dispositivo, PC o teléfono móvil, las aplicaciones serán muy rápidas y serán muy personalizables. Además, se distribuirán viralmente, por medio de redes sociales [29, 53, 84, 85].

Aunque muchos investigadores confunden a **la Web 3.0** con **la Web Semántica** [52,86], estás son diferentes pero estrechamente relacionadas. Según Teem-Berners Lee, La web Semántica es parte de la Web 3.0. De manera resumida, la Web Semántica trata de la integración de datos, convirtíendolos no sólo en datos visuales sino en información significativa mediante el uso de metadatos.

Sin duda, el objetivo que mueve a la WWW es el de permitir el acceso universal a la información, sin importar las habilidades o limitaciones de los usuarios. Esto considera a los diferentes entornos en los cuales la información debe ser entregada. Por lo tanto la tecnología debe ser discreta, debe integrarse al ambiente en el que se desenvuelve el usuario, la respuesta debe adaptarse a las personas y a su contexto, y debe ser predictiva para que ayude a los usuarios (humanos). En otras palabras, el ambiente debe ser inteligente.

Elementos clave de la Web 3.0

La Web 3.0 se basa por que las aplicaciones web son clientes de aplicaciones web, es decir, la comunicación entre aplicaciones [85]. Claramente se nota que esta es la

era en la que las aplicaciones web son muy fáciles de desarrollar, pueden ser usadas desde cualquier dispositivo, permiten a los usuarios compartir audio y video en directo (video streaming). Los motores de búsquedas dan como resultados información más completa y específica, los usuarios no deben especificar claramente lo que buscan [57]. Los usuarios de redes sociales se enfrentan a aplicaciones más inteligentes, con mayores posibilidades de compartir con sus amigos. Se ha vivido desde algunos años atrás la era de los datos e información masiva. Todo esto gracias a los elementos claves de la Web 3.0. A continuación presentamos estos elementos (los más importantes) [53]:

- **Web Social.** Aunque las redes sociales están presentes desde la Web 2.0, se considera que son el elemento que va a evolucionar con la Web. Utilizando las tecnologías de la Web 3.0, en lugar de enlazar únicamente documentos. La web social se considera una forma eficaz y atractiva de conectar a personas de todo el mundo. Gracias a las facilidades que brindan de compartir no sólo textos, sino audios, videos y la transmisión de vídeo directamente cuando se está produciendo [29].

- **La web semántica.** La web semántica es una extensión en evolución de la web [86], que permite a las personas encontrar la información a un nivel mucho más profundo del significado de los términos de búsqueda y el contexto en el que se utilizan [52, 86, 87]. La información está estructurada de tal manera que las máquinas pueden leerla y entenderla tanto como los humanos, sin ambigüedades [29]. la interacción de la Web 3.0 se ilustra en la figura 2.4.

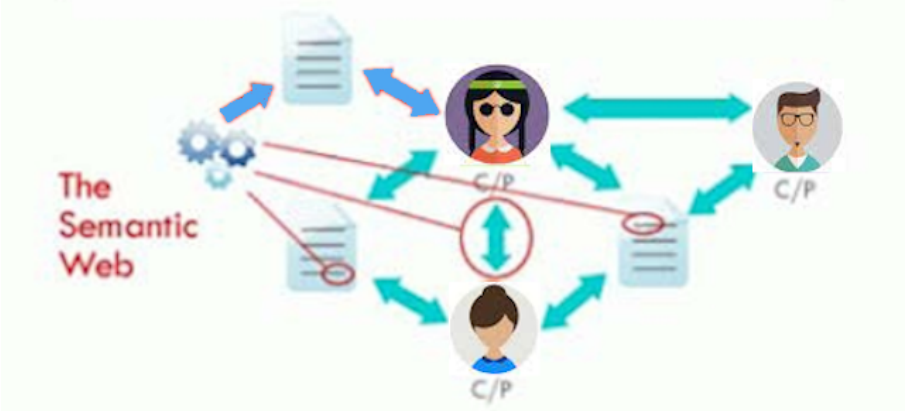


Figura 2.4: Interacción en la Web 3.0

- **Web 3D.** Los mundos virtuales en 3D, han llamado la atención de todo tipo de usuarios, desde curiosos hasta investigadores para apoyar al avance de la creación de **vidas paralelas** [87]. Los sitios web para la creación de mundos virtuales como los mencionados en [53] (Second Life, IMVU, Active Worlds, y Red Light Center) han ganado una gran popularidad entre el público. la Web3D permite a la gente vivir dos mundos paralelos (real y virtual). En el virtual como un avatar en su

nombre para explorar, conocer a otros habitantes. Llevan una vida igual a la real, participando en actividades individuales o de grupo. Para mejorar el estilo de vida virtual, sería posible con la aplicación de la háptica para mejorar la comunicación interpersonal para expresar afecto, intención o emoción a través del contacto físico, como un apretón de manos o un abrazo. Por lo tanto, añadir la háptica a los mundos virtuales sociales a través de la Web aumenta los beneficios y abre nuevas posibilidades para los participantes [29].

- **La Web centrada en los medios.** Con medios se refiere a los medios para la comunicación o información, como el texto, sonido, imagen, video [87]. Por lo tanto, Web centrada en los medios se refiere a que la Web debe valerse de los medios multimedia para encontrar otros medios similares, pero no sólo valiéndose en los metadatos. Es decir, para buscar imágenes sobre algo físico, lo único que deberíamos hacer es proporcionar al motor de búsqueda como entrada una imagen de lo físico que buscamos, y, basándose en las características del objeto físico de la imagen proporcionada, el buscador debería estar capacitado para devolver las imágenes de objetos físicos con características similares [29].

- **La Web Omnipresente y Ubicua (WOU)**. Con omnipresente y ubicua se refiere que está presente en todas partes. Es decir que el usuario puede interactuar con la Web sin necesidad de él acercarse a los dispositivos sino que ellos están siempre cerca del usuario. Entonces, la WOU está presente en todos los computadores, todos los dispositivos, y todos los objetivos físicos en general incluyendo a las personas. Es así que la persona puede solicitar cualquier servicio disponible a la red sin ser consciente de hacerlo. Este concepto lleva a ***Web of Things o WoT*** en español la **Web de las Cosas** [88, 89].

- **Inteligencia Artificial.** El uso de la robótica y la inteligencia artificial marcan la era de la Web 3.0 en comunicación bidireccional, primero entre humanos y computadoras y segundo entre humanos entre sí a través de computadoras, así mismo la comunicación entre computadoras o entre dispositivos en general [90].

Los elementos de la Web 3.0

Los elementos clave de la Web 3.0 son: Web Social, La web semántica, Web 3D, la Web centrada en los medios, la Web omnipresente y ubicua (WOU), Inteligencia Artificial, entre otros.

2.2.4. La Web 4.0 o ¿Internet de las Cosas?

Las fechas o años de inicio de las versiones de la Web no se encuentran facilmente entre las publicaciones al respecto. Cuando estos datos se encuentran entre las publicaciones no coinciden, aunque su variación no es importante. Por ejemplo, La web 4.0 segúl

Latorre [52], tuvo sus inicios en el año 2016, mientras [91] afirman que inició en el año 2015.

La Web 4.0, aunque aún no llega a su momento más importante, ha sido el gran avance centrado en ofrecer un comportamiento más inteligente, más predictivo, de modo que, con sólo realizar una afirmación o petición se pueda poner ejecutar acciones que tendrán como resultando aquello que se ha expresado [52,91]. Es la que ha abierto la posibilidad de interactuar con el medio físico, y que éste responda a nuestras necesidades según el contexto. Esto se conoce como parte de **"Internet de las cosas"** o IoT por sus siglas en inglés (*Internet of Things*). IoT es el mundo interconectado, es la posibilidad de que los objetos físicos estén presentes en el mundo digital, es la oportunidad para mejorar el estilo de vida (cuidado, trabajo, ocio, estado emocional) de las personas. Un sistema basado en IoT interconecta objetos físicos, personas, procesos y datos. Objetos físicos dotados de inteligencia para apoyar al ser humano en la toma de decisiones de su día a día [92]

La Web 4.0 e IoT

La Web 4.0 es la que da la posibilidad de interactuar con el mundo físico. Un mundo físico (objetos físicos, personas, procesos y datos) interconectado dotado de inteligencia se conoce como IoT.

La era actual de la Web llamada la Web 4.0, ha tomado algunos nombres por parte de los investigadores en [57] le da el nombre de la **"Web Activa"**. En [91], la nombran como **Web Simbiótica**[19] o **Web de Conexiones Inteligentes**, aunque también la denominan **Internet de las cosas**. Por otro lado, en [93] a esta era de la Web la denomina la Web Pragmática[20].

En la Web Simbiótica se conectan personas, procesos, cosas y datos. Crean nuevas capacidades, experiencias más satisfactorias y nuevas oportunidades nunca antes percibidas, todo esto gracias a los datos y su tratamiento para convertirlos en acciones sin o con mínima intervención humana. Ya en la Web 3.0 se generaba datos e información que podía ser entendida sin ambigüedades tanto por los humanos como por las máquinas (usuarios = humanos + máquinas). En la Web 4.0 la interacción entre personas y máquinas se da en una simbiosis perfecta, gracias a las tecnologías que permiten el acceso a la red por cualquier usuario, en cualquier lugar, en cualquier momento, sin importar el dispositivo por medio del cual lo hagan [91].

Si se juntan las tres denominaciones como parte de las características de la Web 4.0, es decir, lo activo, lo simbiótico y lo pragmático, se puede definir a la Web 4.0 como la Web en la que el usuario humano se interrelaciona con las cosas del mundo físico y mundo virtual para obtener un mayores ventajas, utilizando un lenguaje acorde con el

[19] Asociación de individuos animales o vegetales de diferentes especies, sobre todo si los simbiontes sacan provecho de la vida en común https://dle.rae.es/simbiosis

[20] Disciplina que estudia el lenguaje en su relación con los hablantes, así como los enunciados que estos profieren y las diversas circunstancias que concurren en la comunicación https://dle.rae.es/pragmático

colectivo relacionado.

Algunos elementos de la Web 4.0

Mucho de lo que hay ya es de la Web 4.0, sin embargo será aún más diferente. Cuando esté completamente desarrollado, eliminará varios de los pasos necesarios al usar la Web 3.0, de esta manera su uso será más directo e "invisible".

- **Los motores de búsqueda** aún sufrirán cambios para estar acorde a la Web 4.0. Lo propio de la Web 4.0 serán los asistente virtuales, a quienes les pides lo que deseas y ellos te lo sirven. Actualmente ya se vive algo de esto. Los asistentes virtuales como Cortana en Windows, como Alexa en Amazon, Siri en dispositivos iOS de Apple, Google Now, Google Assistant en aplicaciones y entre dispositivos Android respectivamente; cada vez responden mejor a las preguntas de sus usuarios [94]. Estos asistentes virtuales cada día van comprendiendo mejor el lenguaje natural, tanto hablado como escrito, con la capacidad de poder responder las preguntas como si fueran un asistente humano [57].

- Una **Red activa** al ser capaz de ella misma proponer cosas según la situación, el contexto, el ambiente en el que el usuario se esté desenvolviendo. Por ejemplo, un turista está cerca de un lugar emblemático, el asistente de la Web le debe sugerir lo visite dándole a conocer la información del lugar. Así mismo si está cerca (hablando espacio-temporalmente) de un evento el asistente le sugerirá asistir. Todo esto si es del agrado o preferencias del usuario. De esto ya hay algunos trabajos, sin embargo aún estamos en medio camino [57].

- **Big Data** entre nosotros. Toda la inteligencia de las aplicaciones es gracias a los datos que se obtienen del mundo real. Los avances en el campo de la inteligencia artificial o AI (*Artifitial Intelligence*) y en los modelos y algoritmos de aprendizaje automático, unidos a la llegada de big data, facilitan el procesamiento y el aprendizaje de la información a una escala sin precedentes [95]. Se procesa de manera más efectiva, vinculando toda la información obtenida a través de múltiples fuentes [57].

- **Aplicaciones web Inteligentes**. la Web Inteligente es la próxima generación de la Web Semántica. Las aplicaciones inteligentes tienen tres ingredientes principales como se muestra en la figura 2.5.

Elementos de la Web 4.0

Entre los elementos de la Web 4.0, se pueden listar a: Los motores de búsqueda y los asistentes virtuales, la red activa, Big Data, aplicaciones web inteligentes.

Figura 2.5: Interacción en la Web 4.0

2.3. Conclusiones

La Web 1.0 marcó el inicio inimaginable para el intercambio de información traspasando todas las barreras del mundo físico, desde las distancias, diferencias horarias, idiomas, capacidades de las personas, etc. la Web es el **lugar** donde todas las personas logran aprovechar el derecho universal de la **igualdad**.

La Web 2.0 es el camino mediante el cual, aplicando estándares y lineamientos en el desarrollo de las aplicaciones web se logran los objetivos de la WWW. Además de los efectos sociales y económicos que ha logrado en el mundo. Claro está que no se debe descuidar, y menos olvidar los inconvenientes que las tecnologías de la Web traen consigo.

La Web Semántica conjuntamente con las tecnologías que han emergido para crear y gestionar nuevas aplicaciones web, dirigidas a nuevos tipos de emisores y receptores de información hacen posible que La Web 2.0 haya sido repotenciada y den lugar a La Web 3.0.

Todo cambia con el tiempo, incluso la ciencia, y así cómo la tecnología. Internet ha evolucionado con el tiempo, no se está seguro si lo hizo para lograr una fase más de la Web, pasando de 3 fases anteriores, que implicaron cambios significativos en su funcionamiento y características para los usuarios. Lo que sí es seguro que llegó a suceder, la aparición de La Web 4.0.

Evaluación a los lectores

Como parte del control de cumplimiento de los objetivos planteados en este capítulo, se debe evaluar a los lectores. Para ello debe resolver las siguientes preguntas.

- Realizar una tabla con cada una de las versiones de la web y sus características, y ejemplos esclarecedores.

- En un cuadro sinóptico ubique las tecnologías (y sus correspondientes objetivos) que se utilizaron (utilizan) en cada una de las versiones de la web.

Capítulo 3

ACCESIBILIDAD WEB

Objetivos

Al finalizar este capítulo, el lector será capas de:

- Comprender la importancia de la accesibilidad web.
- Reconocer los esfuerzos hechos por los organismos como el W3C respecto a la inclusión de los grupos minoritarios de personas como las personas con discapacidades y personas de la tercera edad.
- Identificar los criterios de accesibilidad web que posee una página web.
- Interpretar cada uno de los principios y pautas que son emitidos para que una página web (contenido web), herramienta de autor o un agente de usuario debe cumplir para ser inclusivo.

Resumen

La accesibilidad Web da la posibilidad a todas las personas sin importar las diferencias en sus habilidades. La web es la fuente de información que proporciona la posibilidad de mejorar los distintos aspectos de la vida. La WAI trabaja para establecer estándares internacionales que garanticen la accesibilidad web. Existen muchos indicadores para garantizar la Accesibilidad Web, por lo que en este capítulo se consideran los más importantes.

3.1. Introducción

Lograr que la Web sea accesible para todos, sin importar las discapacidades o problemas tecnológicos que presenten las personas. La accesibilidad web universal es uno de los objetivos de **World Wide Web Consortium**. W3C es una comunidad internacional

donde las organizaciones miembros, el personal a tiempo completo y el público trabajan juntos para desarrollar estándares web. Dirigida por el inventor y director web Tim Berners-Lee y el director ejecutivo Jeffrey Jaffe, la misión del W3C es llevar la Web a su máximo potencial [96].

3.1.1. Definición

La accesibilidad Web da la posibilidad a todas las personas sin importar las diferencias en sus habilidades, sean estas disminuciones congénitas, causadas por algún accidente o que se presentan con el tiempo. Un sitio web es accesible cuando ha sido diseñado para que el usuario navegue sin inconvenientes y obtenga la información con facilidad y rapidez. La facilidad de la navegación se ve directamente referida a un diseño Web que permite que las personas logren percibir, entender, navegar e interactuar la Web [97]. Y la obtención de la información con rapidez se refiere a que la información que la Web proporciona al usuario, éste la encuentre a una distancia no mayor de dos o tres clics [98]. En el diseño del sitio web debe considerarse que sus usuarios serán adicionalmente las personas con disminuciones en sus capacidades físicas y mentales, y las personas de avanzada [99].

Accesibilidad web

La accesibilidad Web significa que personas con algún tipo de discapacidad y/o con limitaciones tecnológicas van a poder hacer uso de la Web.

Otro aspecto a considerar en la accesibilidad es las características del entorno del usuario. Considerado como entorno del usuario, desde la conexión a internet que tenga a su alcance hasta el equipo por medio del cual acceda. Buscando con la accesibilidad que el mayor número de personas acceda al sitio web (o página web) de manera independiente de sus capacidades y de su entorno [99]. La accesibilidad web no se trata únicamente de los sitios web sino también de las herramientas y las tecnologías utilizadas para acceder a la información disponible en la web. Estas conjuntamente con los sitios web se deben diseñar y desarrollar para que puedan ser usadas por las personas con capacidades diferentes [100]

3.1.2. Importancia de la Accesibilidad Web

La web es la fuente de información que proporciona la posibilidad de mejorar los distintos aspectos de la vida, aspectos como educación (formación y/o capacitación), empleo, gobernabilidad, servicios, comercio, entretenimiento, por mencionar algunos. Un portal web accesible brinda la posibilidad a las personas con capacidades diferentes, a las poblaciones rurales, y a los habitantes de países en desarrollo, mejorar su estilo de vida. Brindándoles, de manera especial a estos grupos de personas la oportunidad de acceder a la información e interactuar con la web. Un sitio web accesible asegura la

equidad y la igualdad de oportunidades a las personas con capacidades diferentes. La accesibilidad web cierra la brecha digital para este grupo de personas [101].

Accesibilidad web

Permite a todas las personas sin importar sus limitaciones sentirse iguales, con las mismas posibilidades de mejorar su estilo de vida.

A finales de 2019 e inicios del año 2020, con la aparición de COVID-19 o SARS-CoV-2, y luego en marzo de 2020, al convertirse en una pandemia, con ella la necesidad de confinamiento de la población a nivel mundial [102], la web se ha convertido en un recurso universal de vital importancia para la realización de las actividades cotidianas, como mantenerse comunicados, el trabajo de oficina que se convirtió en teletrabajo, la educación presencial se convirtió en educación en línea o aprendizaje electrónico (e-learning), los comercios se volcaron al comercio electrónico, por mencionar algunas [103]. Por esto podemos concluir que la Web ha facilitado el trabajo en estos días de pandemia.

3.2. Iniciativa de Accesibilidad Web (WAI)

La WAI trabaja para establecer estándares internacionales que garanticen la accesibilidad web. Entre estos estándares tenemos por ejemplo las Pautas de Accesibilidad al Contenido Web (WCAG 1.0, WCAG 2.0, WCAG 2.1). Otro estándar ISO es ISO/IEC40500 [104, 105]. Los estándares internacionales involucran a todos los componentes que trabajan juntos para lograr el éxito de la Web. Entre estos componentes están las tecnologías web, los agentes de usuario como navegadores web y otros "agentes de usuario" , herramientas de creación (gestores de contenido, diseñadores web), y sitios web. La WAI del W3C desarrolla especificaciones técnicas, pautas, técnicas y recursos de apoyo que describen soluciones de accesibilidad [100].

La web al ser accesible para un rango diverso de personas, es decir, es accesible para personas con capacidades diferentes respecto a motricidad, audición, visión y capacidad cognitiva. La Web cumple con uno de sus objetivos, en vista de que está diseñada para que funcione para todas las personas, independientemente de su hardware, software, idioma, ubicación o capacidad cognitiva [99]. Esto hace que las barreras que las personas enfrentan en el mundo físico por sus diferentes capacidades, en el mundo de la Web desaparezcan. Por lo tanto, algunos de los productos (sitios web, aplicaciones web, tecnologías, herramientas) al estar mal diseñados, excluyen a las personas del uso de la Web [96].

Para la presentación de la accesibilidad web no bastaría un libro para plasmar todo sobre el tema, es por eso se tomará en cuenta los aspectos más comunes y fáciles de implementar.

3.3. Aspectos de la Accesibilidad

Existen muchos indicadores para garantizar la Accesibilidad Web, por lo que en este apartado se presentan los que se han considerado más importantes. Si los desarrolladores no conocen o tienen dudas sobre algunos aspectos de la accesibilidad, W3C en [106] brinda sugerencias para verificar muchos de los aspectos de la accesibilidad para las aplicaciones web que estén en desarrollo. A continuación, se mencionan las más relevantes.

3.3.1. Título de las páginas web

Los títulos de las páginas web son elementos fundamentales para la orientación de las personas, por lo que un buen título tiene que permitirle al usuario identificar rápidamente donde se encuentra. Para las personas con discapacidades visuales se puede incluir un lector de títulos. Esta es una de las razones que se debe identificar como un buen título es que sea capaz de describir rápida y adecuadamente el contenido que el usuario encontrará en la página, por lo tanto los títulos de las páginas debes ser diferentes. No se espera tener en mismo contenido en páginas diferentes del mismo sitio web [107, 108].

3.3.2. Texto alternativo para imágenes

La mejor función de los textos alternativos en las imágenes es para que las personas con discapacidad visual se enteren de lo que este elemento de una página web muestra como información a los usuarios [109]. Para lograr este objetivo los usuarios deben utilizar lectores pantalla. El lector de pantalla es el encargado de leerle a las personas con discapacidad visual el texto alternativo que se encuentra en la etiqueta HTML img en la opción alt="" [97, 110, 111]. Además, es de mucha ayuda para los usuarios que no desean cargar las imágenes, ya sea por limitaciones de conectividad o por costos. Por lo tanto, se debe verificar que todas las imágenes de las páginas web de las aplicaciones tengan texto alternativo. Por ejemplo:

<img src="image url" style="css attributes" class="css class" id=" css id" alt="explanatory text about what the image shows">

3.3.3. Relación de contraste

Los navegadores y las páginas web deben permitir a los usuarios poder cambiar el color de fondo y de las letras, debido a que existen personas que se les dificulta leer el contenido por el contraste que existe entre el fondo y las letras, por ejemplo: las personas con poca sensibilidad visual necesitan fondos oscuros y letras brillantes para poder leerlos, por otra parte, los disléxicos no pueden leer texto que tengan colores muy brillantes [106, 112].

3.3.4. Acceso por teclado y enfoque visual

Las personas ciegas y algunas personas videntes con impedimentos de movilidad no pueden usar un *mouse* (ratón) por lo que para interactuar con la Web dependen de los comandos del teclado, como la entrada de voz, por lo que el sitio web debe permitirles a los usuarios con estas discapacidades poder acceder a todas las funciones del sitio mediante el teclado [106].

3.3.5. Alternativas multimedia (vídeo, audio)

La información de la página web tiene que estar disponible para todas las personas, si un usuario tiene una discapacidad visual la información debe estar en podcasts u otro tipo de audio, si el usuario es sordo o tiene disminución auditiva la información puede estar en forma de vídeo muy visual o en texto [106, 113].

3.3.6. Estándares de accesibilidad WEB

Para asegurar que todas las aplicaciones pueden ser accesibles se den cumplir los estándares que se encuentran dividido en tres secciones [114, 115]:

- Accesibilidad al contenido Web o WCAG: Ofrece normas para cualquier contenido web, también ofrece estándares para la estructura de la página [114].
- Accesibilidad a los Agentes de usuario o UAAG: Se enfoca en toda herramienta que le permite al usuario acceder a los contenidos en la Web [115].
- Accesibilidad de las Herramientas de autor o ATAG: Esta sección se centra en las herramientas que crear contenido para la Web [114].

3.3.7. Niveles o Criterios de éxito

Cabe resaltar que las sugerencias de inclusión que presentan las Pautas están divididas por niveles o criterios de éxito el nivel A el más básico, el nivel AA es medio y el AAA es el nivel más exigente de las pautas y que cumple con todas las normativas de los niveles anteriores y agregando nuevas [116].

Nivel de conformidad

Un sitio web o una aplicación web alcanza un nivel de conformidad, cuando uno de los siguientes niveles de conformidad se satisface por completo [117].

- **Nivel A:** Para lograr conformidad con el Nivel A (el mínimo), la página web satisface todos los Criterios de Conformidad del Nivel A, o proporciona una versión alternativa conforme.

- Nivel AA: Para lograr conformidad con el Nivel AA, la página web satisface todos los Criterios de Conformidad de los Niveles A y AA, o proporciona una versión alternativa conforme al Nivel AA.
- Nivel AAA: Para lograr conformidad con el Nivel AAA, la página web satisface todos los Criterios de Conformidad de los Niveles A, AA y AAA, o proporciona una versión alternativa conforme al Nivel AAA.

Nivel A: Para lograr conformidad con el Nivel A (el mínimo), la página web satisface todos los Criterios de Conformidad del Nivel A, o proporciona una versión alternativa conforme. Nivel AA: Para lograr conformidad con el Nivel AA, la página web satisface todos los Criterios de Conformidad de los Niveles A y AA, o proporciona una versión alternativa conforme al Nivel AA. Nivel AAA: Para lograr conformidad con el Nivel AAA, la página web satisface todos los Criterios de Conformidad de los Niveles A, AA y AAA, o proporciona una versión alternativa conforme al Nivel AAA.

Aspectos de la accesibilidad web

Entre los más importantes se pueden citar:

- Título de las páginas web
- Texto alternativo para imágenes
- Relación de contraste
- Acceso por teclado y enfoque visual
- Alternativas multimedia (vídeo, audio)
- Estándares de accesibilidad WEB
- Niveles o Criterios de éxito

.

3.4. Directrices de Accesibilidad al Contenido Web (WCAG)

Las Pautas de accesibilidad al contenido web o WCAG (*Web Content Accessibility Guidelines*), se desarrollan a través del proceso de W3C en cooperación con otras organizaciones de todo el mundo, con el fin de proveer recomendaciones para el contenido en la Web, definiendo así un único estándar para que cubra las necesidades de todas las personas, organizaciones y países. El contenido de las WCAG expone cómo hacer para que el contenido sea más accesible para las personas con discapacidades, el contenido incluye imágenes, texto y sonidos, también incluye normas para la estructura y presentación de los sitios web [118].

3.5. Principios y pautas WCAG

La versiones más usadas de WCAG son la 2.0 que cuenta con 12 pautas y 2.1 que tiene 13 pautas y se encuentran organizada bajo cuatro principios de igualdad lo cuales son: perceptible, operable, comprensible y robusto [118].

3.5.1. Principio Perceptible

El objetivo de este principio es buscar alternativas para transmitir la información de la aplicación web a todos los usuarios, incluyendo a los usuarios con discapacidades. Las pautas de este principio son [119]:

Pauta 1- La Alternativa de texto

Una gran cantidad de contenido no textual tienen alternativas de texto que permite transmitir la información a personas con discapacidades visuales o cognitivas [120].

- **Controles, entrada:** Si el contenido es un control que no es texto, se le debe mostrar al usuario su nombre, la función y el estado en el que se encuentras, estas características se pueden definir y determinar mediante la programación, según el criterio de éxito 4.1.2 definido por [120] estas propiedades ya vienen disponibles en los controles HTML, por lo que estas sugerencias van dirigidos a los desarrolladores que crean sus propios componente para la interfaz de usuario de sus aplicaciones.
- **Medio basados en el tiempo:** Si el contenido es un medio basado en el tiempo (audio, vídeos, animaciones y música), los de texto no pueden describir todo el contenido, pero sirve como ayuda para identificar de forma básica el sentido del contenido no textual que presenta la Web [121, 122].
- **CAPTCHA:** Si el contendió es un captcha que verifica si el usuario es una persona y no un robot, se darán alternativas de acceso que se adapten a las diferentes discapacidades, por ejemplo: sumar dos números o escribir las palabras de un audio [123].

Pauta 2-Alternativas para medios basados en el tiempo:

En esta pauta se detallara más las alternativas para los medios basados en el tiempo [119]:

- **Sólo audio (pregrabado)-Nivel A:** Cuando el contenido es solo audio, la alternativa se presenta en de texto descriptivo, no solo describiendo lo que el audio está informando o lo que los autores dicen, también se detallan acciones y expresión como risas, aplausos, gritos y transiciones, por ejemplo [124]: **Episodio 1:** Introducción, temas: Inteligencia artificial, el público se levantó y aplaudió, luego el comentarista con una carcajada empezó hablar.

- **Sólo vídeo (pregrabado)-Nivel A:** Si el contenido es solo vídeo, se sugiere poner una pista de solo audio que describa el contenido del vídeo, este tipo de alternativa es útil para las personas con discapacidades visuales, otro ejemplo que se puede aplicar es presentar un texto descriptivo del vídeo para las personas con problemas de captación o auditivos.

- **Subtítulos-Nivel AA:** Los subtítulos en los vídeos tienen como objetivo ayudar a las personas sordas o con problemas auditivos, estos describen todo el contenido y se encuentran incrustados en el mismo siendo siempre visible y claro para el usuario.

- **Lenguajes de señas-Nivel AA:** El objetivo del lenguaje de señas es permitir a las personas sorda que no puedan leer el texto y los subtítulos de forma rápida acceder a la información de un audio o vídeo de forma rápido [125].

- **Descripción de audio extendida-Nivel AAA:** El objetivo de esta técnica es brindarles a los usuarios una segunda pista de audio que describa el contenido, siempre es bueno tener más de una versión explicativa para el contenido de las aplicaciones web por si a los usuarios no les queda claro con una explicación.

- **Audio en vivo-Nivel AAA:** Para los audios en vivo existen algunas recomendaciones, si el vídeo en vivo es de alguna obra, discurso, curso o conferencias, se debe dejar el enlace que redirija al usuario al guión o la descripción del contenido que se hablara en el vídeo en vivo, otra alternativa es incluir a una persona que sepa lenguaje de señas para pueda transmitir lo que sucede en el vídeo [126].

Pauta 3 - Sitios web adaptables:

Al hablar de sitios adaptable se centran en las aplicaciones web puedan ser vistos desde cualquier dispositivo sin que la información pierda su significado [119]:

- **Información y relaciones-Nivel A:** El objetivo de este criterio es asegurarse que la información visual o auditiva y las relaciones no pierdan significado o sentido al cambiarse el formato, una persona use lector de pantalla tiene que tener la misma información que un usuario normal.

- **Características sensoriales-Nivel A:** Este criterio tiene como objetivo garantizar que todos los usuarios puedan acceder a las instrucciones de las páginas web, para aquellas personas con problemas cognitivos que no pueden reconocer formas (flechas, cuadrados o triángulos) se les brindan informaciones extras en forma de texto o audio por ejemplo: “Botón buscar” o “Botón aceptar” [127].

- **Orientación-Nivel AA:** Este criterio busca asegurar que todas las aplicaciones web puedan funcionar independientemente de su orientación (Horizontal o vertical) y sin perder funcionalidad [128].

- **Identificar Propósito-Nivel AA:** El objetivo de esta regla es asegurar que el propósito de los elementos de una página web se puedan identificar mediante la programación. Al momento de desarrollar una aplicación web se puede programar una opción que les permita a los usuarios personalizar los elementos de la página acorde a sus necesidades, según lo que ellos crean que es el propósito real del componente.

Pauta 4 – Distinguible:

Esta pauta se centra en ayudar a los usuarios con discapacidades a captar el contenido de una aplicación web [119]:

- **Uso del color-Nivel A:** Este criterio busca asegurar que la información transmitida por los diferentes colores pueda llegar a los usuarios. Si un usuario tiene problemas para distinguir los colores, se puede utilizar otros medios visuales para lograr el propósito de este criterio, por ejemplo: si el color verde representa a todo los botones "aceptar" de una aplicación, entonces se puede usar un icono de visto para hacerle entender al usuario el significado el botón [129].

- **Contraste(mínimo) -Nivel AA:** Este criterio tiene como objetivo asegurarse que los usuarios se sientan cómodos con el contraste que existe entre el texto y el fondo, para mejorar el contraste debe existir un panel de configuración para el texto [112].

- **Cambiar el tamaño del texto-Nivel AA:** Este criterio busca la comodidad del usuario con el tamaño del texto, incluyendo el tamaño del texto de los botones y otros controles.

- **Audio de fondo bajo o nulo-Nivel AAA:** Busca prevenir confusiones entre los diferentes audios, busca que cualquier ruido de fondo sea separado o silenciando del mensaje principal [130].

- **Presentación visual-Nivel AAA:** Busca que el texto presentado en la página sea apropiado por el usuario, por lo que él debe poder cambiar y configurar el color de texto y del fondo, también es importante describir a los textos visuales o caracteres especiales para que las personas con problemas cognitivos puedan entenderla el contenido [131].

- **Espacio del texto-Nivel AA:** La experiencia de lectura de los usuarios es muy importante por lo que el objetivo de este criterio es que los usuarios puedan controlar los espacios del texto y el resto de configuraciones para el texto, debió a que existen usuarios a quienes el espacio afecta a su velocidad de lectura [132–134].

- **Contenido en Hover o Focus-Nivel AA:** Este criterio busca reducir los problemas que aparecen con los contenidos emergentes como modales, menú de opciones

desplegables y submenús, ya que estos contenidos pueden afectar a la navegación de todos los usuarios [135].

3.5.2. Principio Operable

El principio operable se enfoca en que todo los componentes del sitio deben ser operables, es decir que todos los usuarios puedan usar la página y que esta se adapte a las necesidades y capacidades de navegación [119]:

Pauta 5 - Teclado accesible:

Esta pauta se centra en el teclado o los teclados adaptados, tiene como objetivo garantizar a los usuarios poder ejecutar toda la función de la página por medio del teclado.

- **Teclado-Nivel A:** Garantiza que toda la navegación y funciones de la página se puedan realizar mediante el teclado o por teclados alternativos especiales para personas con discapacidad [136].
- **Sin trampa de teclado-Nivel A:** este criterio busca evitar que las funciones del teclado propias de la páginas causen problemas con las funciones del teclado de los programas de navegación que usen los usuarios.
- **Atajos de teclados de caracteres-Nivel A:** Los atajos de teclado son herramientas importantes para los usuarios, pero para los usuarios con discapacidades puede ser un problema y más para las personas que navegan gracias al teclado específicos, por lo que es importante poderle permitir a los usuarios reconfigurar o desactivar estos atajos para prevenir confusiones en la navegación.

Pauta 6 - Suficiente Tiempo:

Esta pauta tiene como objetivo brindarle a los usuarios el tiempo suficiente que les permita leer y utilizar todo el contenido de la página.

- **Pausar, parar, esconderse-Nivel A:** Este criterio busca eliminar las distracciones para el usuario durante el uso de la página. Las distracciones incluyen otros contenidos de la página como animaciones, vídeos y juegos, también se puede incluir las actualizaciones automáticas de la página [137].
- **Interrupciones-Nivel AAA:** Evitar interrupciones por parte del servidor que sean innecesarias para el usuario. Por supuesto, en algunas ocasiones son necesarias.
- **Tiempo de espera-Nivel AAA:** Este criterio garantiza que cuando se acabe el tiempo de espera de algún proceso se les comunicará por diferentes medios según sus necesidades especiales y así los usuarios no perderán los datos.

Pauta 7- Convulsiones y reacciones físicas:

Esta pauta busca garantizar que los desarrolladores no diseñen páginas web que puedan provocar problemas a las personas, como convulsiones o reacciones físicas perjudiciales.

- **Tres parpadeos-Nivel A:** Este criterio evita que las personas sufran convulsiones fotosensibles causadas por distintas luces que se pueden encontrar en animaciones o video de la página, estas luces deben de estar por debajo del umbral de paradero permitido que son 3 parpadeos por segundo [138].
- **Animaciones de interacciones-Nivel AAA:** Este criterio evita que algunos usuarios tengan malas experiencias con las animaciones, ya que esta le pueden causar fatiga visual, malestares y mareos [139].

Pauta 8: navegable

Esta pauta proporciona ayudas a los usuarios para que puedan navegar entre los controles de las aplicaciones web y encontrar e identificar el contenido de la misma.

- **Página Titulada-Nivel A:** Este criterio se cumple cuando todas las páginas web tienen un título descriptivo que ayude al usuario a orientarse en la aplicación [108].
- **Propósito de enlace-Nivel A:** Cuando los enlaces en la páginas o en un contenido textual brinda información al usuario sobre ellos, entonces podemos concluir que cumplen con este criterio [140].
- **Enfoque visible-Nivel AA:** El objetivo de este elemento es hacer que el usuario identifique qué elemento está seleccionado por el teclado [141].
- **Propósito del enlace-Nivel AAA:** Garantiza que la funcionalidad de los enlaces coincida con descripción de su función [35].

Pauta 9 -Modalidad de entrada

La pauta especificada como la modalidad de entrada se enfoca en bríndale al usuario varias alternativas para que puedan entender y utilizar todas las entradas del sistema.

- **Gesto de punteros-Nivel A:** garantiza que el usuario pueda utilizar otros dispositivos para poder navegar aparte del ratón y del teclado.
- **Cancelación del puntero-Nivel A:** busca evitar errores al seleccionar elementos no deseados. En las páginas web se debe implementar la utilización del doble clic o del clic presionado para asegurarse de que el usuario ha seleccionado el elemento deseado [142].

- **Actuación de movimiento- Nivel A:** garantiza que se tomará en cuenta todos los comandos que se realizan por movimiento del dispositivo móviles como sacudidas [143].
- **Mecanismos de entrada-Nivel AAA:** garantiza que el usuario puede cambiar y entrar entre las diferentes modalidades para navegar en la Web.

3.5.3. Principio Comprensible

Todos los desarrolladores deben buscar que la información y el funcionamiento de la interfaz de usuario sean comprensibles sin importar el método de navegación y tomando en cuenta las necesidades del usuario [119, 144]:

Pauta 10-Legible:

Esta pauta busca que el contenido de la página que se encuentre en forma de texto sea legible y se comprenda de todas las formas.

- **Idioma -Nivel A:** La información de la aplicación web debe poder traducirse en diferentes idiomas [145].
- **Palabras inusuales-Nivel AA:** este criterio busca que todo el contenido sea entendible para los usuario, ya que existen personas que no pueden entender palabras complicadas, abreviatura y símbolos especiales [146].
- **Pronunciación-Nivel AAA:** busca garantizar que el texto de la página sea simple y claro, también incluye alternativas explicativas para comprender el texto.

Pauta 11- Predecible:

La predecibilidad se refiere que todas las páginas web del sitio deben tener la misma metáfora de navegación. Por lo tanto, tiene como objetivo que la página web sea predecible para el usuario y así evitar confusiones.

- **Enfocado-Nivel A:** busca garantizar que todas las funciones del sitio web sean visibles mientras el usuario navega, todo los componente tiene que tener foco y poder ser visibles para los usuarios [147].
- **Navegación consistente-Nivel AA:** garantiza un único sistema de navegación que sea igual para todas las páginas.
- **Identificación consistente-Nivel AA:** garantiza que la funcionalidad sea consistente, que existan tantas variaciones en la funcionalidad de los elementos de la página web.
- **Cambiar solicitud-Nivel AAA:** brinda a los usuarios la oportunidad de solicitar cambios en el sistema web [148].

Pauta 12-Asistencia de entrada:

Esta pauta ayuda a los usuarios a evitar posibles errores y en caso de que existan errores le ayudar a corregirlos.

- **Error de identificación-Nivel A:** garantiza que el usuario sepa que ha ocurrido algunos errores en el sistema web por medio de notificaciones [149].
- **Prevención de errores (legal, financiero, datos)-Nivel AA:** evita perjudicar a las personas con discapacidades por posibles errores, los sitios web deben garantizar que los datos puedan ser reversibles, siempre comprobar los procesos y confirmarlos [150].
- **Ayuda-Nivel AAA:** ayuda a los usuarios con discapacidad a entender el contexto de las operaciones que se están realizando en la página web [151].

3.5.4. Principio Robusto

Este principio define que el contenido de un sitio web debe ser confiable y que, a medida que avance la tecnología este contenido debe seguir siendo interpretable para los usuarios [119]:

Pauta 13- Compatibilidad con usuarios actuales y del futuro

Esta pauta busca garantizar que todos los elementos de la aplicación web sean compatibles con los futuros usuarios.

- **Análisis-Nivel A:** Garantiza que las tecnologías actuales de asistencia puedan analizar y entender el contenido de la página web.
- **Nombre, Rol, valor-Nivel A:** Este criterio busca de las tecnologías de asistencias puedan recopilar información sobre las configuraciones para mantenerse actualizado sobre las necesidades del usuario [152].
- **Mensaje de estado-Nivel AA:** Busca que los usuarios siempre tengan conocimiento de los cambios realizados en el sistema.

3.6. Pautas de Accesibilidad de las Herramientas de Autor (ATAG)

ATAG (*Authoring Tool Accessibility Guidelines*) son un conjunto de pautas o normas dirigidas a las herramientas de autor (software y servicios) que son utilizados por desarrolladores para generar contenido para los sitios web. Las pautas de las herramientas de autor van dirigidas principalmente [153]:

- Editores HTML como Atom, Sublime Text, Adobe Dreamweaver, etc.
- Softwares para crear sitios web como WordPress, Joomla, Wix, entre otros.
- Procesadores de texto como NotePad, WordPad, etc.

El principal objetivo de las ATAG es permitir que las herramientas de autor puedan ser utilizada por las personas con discapacidad, de esta manera, este grupo minoritario de personas también puedan generar contenido para la Web. Las ATAG también se aseguran que cumplan las normas y pautas de WCAG para que los contenidos web sean accesibles [154].

3.6.1. Pautas de ATAG

ATAG se divide en dos partes, la parte A tiene pautas dirigidas a las herramientas de creación y la parte B sugiere pautas dirigidas al contenido web generado por las herramientas de autor, a continuación se describirán las pautas de ATAG creadas por W3C y descritas en [155].

Parte A

Principio A1: Este principio se centra en la interfaz de las herramientas de autor que cumplan con las pautas de accesibilidad [155–157]:

- Pauta 1: Si existen funciones basadas en la Web en las herramientas de autor, estas tienen que seguir las normas WCAG para incluir a los usuarios que utilicen tecnologías de asistencia.
- Pauta 2: Si no es basada en la Web, se tiene que asegurar que se siga las pautas de accesibilidad existente de la plataforma, es decir seguir las opciones que los desarrolladores implementaron en las herramientas.

Principio A2: Este principio busca que la plataforma y sus elementos sean perceptibles para el usuario [155–157]:

- **Pauta 3:** Esta pauta se enfoca en poner contenido alternativo a la información de la página web siguiendo las WCAG sobre la alternativa de texto y de los medios basado en el tiempo.
- **Pauta 4:** Esta pauta busca garantizar que el usuario sepa sobre el estado y las propiedades del contenido que se está editando en la plataforma.

Principio A3: Este principio busca garantizar que los elementos de la plataforma sean operables por cualquier usuario [155–157]:

- **Pauta 5:** Busca asegurar que todos los elementos del sistema sean operables mediante el teclado o los teclados adaptados para personas con discapacidad.

- **Pauta 6:** Garantiza a los usuarios el guardar automáticamente y que éste pueda configurar el tiempo de autoguardado.
- **Pauta 7:** Esta pauta busca evitar las convulsiones provocadas por luces parpadeantes y pone como regla que el parpadeo no puede supera los 3 por segundo en las animaciones o vídeo de la plataforma.
- **Pauta 8:** La navegación entre los componentes y los elementos de la plataforma debe tener una forma, sin importar el recurso que se está utilizando para navegar: teclado, ratón o asistente; con esto se evitan confusiones para el usuario.
- **Pauta 9:** El objetivo de esta pauta es proporcionar al usuario un panel de búsqueda de texto.
- **Pauta 10:** Tiene como objetivo darle al usuario la opción de configurar la plataforma incluyendo aspectos visuales, y guardar eso cambios.
- **Pauta 11:** Garantiza que la plataforma tenga vistas previas, ya que la mayoría de desarrolladores siempre tienen que verificar el código.

Principio A4: La vista de la edición del contenido que se encuentra en la plataforma tiene que ser comprensible para todos los usuario [155–157]:

- **Pauta 12:** Esta pauta garantiza al usuario que los cambios hechos en la configuración son reversibles y que el usuario pueda cambiar la configuración de la interfaz del software.
- **Pauta 13:** Dar una guía de ayuda para el usuario, es decir dejar una documentación que describa las funciones de cada elemento de la plataforma.

Parte B

Principio B1: Si en la plataforma existen funciones automatizadas, entonces el usuario debe poder configurar estas funciones [155, 156]:

- **Pauta 14:** Esta pauta asegura que los contenidos automáticos sean accesibles para el usuario.
- **Pauta 15:** Esta busca asegurar la información de accesibilidad que existe en la plataforma, siguiendo las normas descrita en WCAG.

Principios B2: Los autores son compatibles con la producción de contenido accesible, por lo que ninguna función de autor puede bloquear la accesibilidad de los usuarios [155, 156]:

- **Pauta 16:** Esta pauta busca garantizar que la herramienta de creación no impone restricciones en el contenido web de los autores, cumpliendo los criterios WCAG.

- **Pauta 17:** Consiste en informar a los autores sobre el contenido web accesible para evitar problemas de accesibilidad, es decir dar información sobre que parte de la plataforma necesita acceso a internet para funcionar correctamente.
- **Pauta 18:** Esta pauta ayuda a los desarrolladores a gestionar los contenidos alternativos, siguiendo las normas de WCAG, evita que los medios alternativos incorrectos generen problemas en la accesibilidad del contenido web WCAG.
- **Pauta 19:** Consiste en que las plataformas deben proporcionar plantillas de accesibilidad web, para reducir los riesgos y el esfuerzo.
- **Pauta 20:** Proporciona contenido extra creado por la plataforma.

Principio B3: Este principio se basa en apoyar a los usuarios con el contenido que se encuentra creado [155, 156]:

- **Pauta 21:** El objetivo de esta pauta es Mostrar el estado de los elementos de accesibilidad de la plataforma, da una sugerencia o asistencia de verificación (WCAG) y ayuda a los autores en la toma de decisiones.
- **Pauta 22:** Ofrecer soluciones a los usuarios sobre los problemas de accesibilidad presentes en la plataforma.

Principio B4: Las herramientas de autor promueven el uso de normas de accesibilidad e integran sus propias pautas o funciones de accesibilidad [155, 156]:

- **Pauta 23:** Busca garantizar el soporte a las plataformas y garantiza que se ejecuten las nomas de validación y de errores.
- **Pauta 24:** Se enfoca en asegurar que los usuarios o autores sepan utilizar las herramientas de accesibilidad del contenido como ejemplo: las alternativas de texto y verificación de accesibilidad.

3.7. Pautas de Accesibilidad de Agentes de Usuario - (UAAG)

Las UAAG (*User Agent Accessibility Guidelines*) brindan direccionan a cómo hacer que los agentes de usuario sean accesibles para personas con discapacidad. Al hablar de los agentes de usuario se refiere a los navegadores web, reproductores multimedia, extensiones de navegador, asistentes y todos elementos que permita acceder al contenido web, por lo que las normas van dirigidas a sus desarrolladores, un agente de usuario puede mejorar su accesibilidad mediante la aplicación de las pautas UAAG, no sólo dará una mejor experiencia a las personas discapacitadas, sino también para todos los usuarios en general [114, 158].

3.7.1. Pautas de UAAG

Las pautas de UAAG se aplican a diferentes partes de los agentes de usuario como la interfaz de usuario, interfaz de contenido, servidores de comunicación, ajuste de configuración y configuraciones opcional. Muchas de estas pautas tienen las mismas bases y conceptos que WCAG y ATAG. Las pautas que se describen a continuación son las más destacadas de UAAG y que tiene conceptos específicos de los agentes de usuario. A continuación las pautas descritas por W3C [101]:

- **Pauta 2:** Los usuarios pueden solicitarle al navegador que busque la información faltante de la alternativa de texto de algún contenido no textual [159].
- **Pauta 3:** El usuario tiene que ser capaz de distinguir todos los elementos y en qué estado se encuentran (habilitado, visibles, deshabilitado etc.), el navegador también tiene que mostrarle al usuario los enlaces que ya el selecciono o hizo clic con anterioridad.
- **Pauta 6:** Si el sistema genera voz sintetizada, tiene que darle la opción al usuario de poder cambiar y configurar los parámetros de voz a sus gustos.
- **Pauta 7:** El usuario tiene que poder cambiar, guardar o deshabilitar las hojas de estilo o el mecanismo de estilo del autor como las hojas de estilo CSS. Al permitirle a los usuarios cambiar los estilos CSS se aumenta la posibilidad de que el sitio web sea accesible a las necesidades del usuario.
- **Pauta 8:** Las ventanas de los navegadores tienen que ser ajustables y personalizables, las ventanas tienen que tener elementos resaltantes para mantener a los usuarios orientados y ser enfocadas automáticamente al ser usadas.
- **Pauta 22:** Esta pauta evita perjudicar a los usuarios y evitar procesos erróneos, por lo que las herramientas de autor deben permitirles a los usuarios poder corregir errores, recuperar procesos y tener opción de cancelar o retorcer.
- **Pauta 16:** Las herramientas de autor deben permitir a los usuarios guardar las configuraciones según como lo prefieran, también les deben permitir restaurar las configuraciones predeterminadas cuando lo deseen.
- **Pauta 26:** Si las herramientas de autor están diseñadas en base a HTML se deben cumplir todas las normas de WCAG para asegurarse que el contenido web sea accesible para todos usuarios [101].

3.8. Ejemplos cómo aplicar la Accesibilidad Web

3.8.1. Ejemplo 1:

Con el advenimiento de las aplicaciones que son capaces de intercomunicarse, han hecho que los desarrolladores creen métodos para asegurarse que quién interactúa con sus aplicaciones es un humano. Un método creado para esta aseveración es el uso de **código captcha** (ver figura 3.1). Su implementación es una de las características de una página web claramente incluyente de las personas con discapacidad, al crear métodos alternativos para verificar que quién accede a una página web no es un robot, por ejemplo, las personas con problemas visuales no pueden realizar un CAPTCHA normal, por lo que el método de verificación para ellos será [123, 160]:

- escribir la palabra del audio por comando de voz.
- sumar dos números, para lo cual tendrían la ayuda de herramientas de autor (lectores de pantalla).
- al acercar el ratón a una imagen le muestre en texto grande y/o le exprese en comando de voz el código a ingresar o repetir.

Figura 3.1: Uso de Captcha y sus alternativas

3.8.2. Ejemplo 2: Describir los elementos de la página

Describir los elementos de una página web también es fundamental, por ejemplo, cada botón tiene que tener una descripción de su nombre y función, mediante el atributo **alt** de las etiquetas HTML. Cuando un usuario utiliza un lector de pantalla, es importante que todos los elementos tengan una descripción (valor para el atributo alt) porque este valor es interpretado y comunicado mediante voz a las personas que consultan la información de la página [106, 161].

3.8.3. Ejemplo 3: Vídeos subtitulados

Otro caso para la accesibilidad al contenido web es el uso de los subtítulos en diferentes idiomas, ya sean personalizados o generados automáticamente por plataformas web como YouTube. De esta manera, los vídeos pueden llegar a más personas de diferentes países y también a las personas con problemas auditivos. Gracias al uso de estas tecnologías, muchos desarrolladores de páginas web pueden subir sus vídeos en YouTube y luego compartirlos en sus páginas propias, facilitándoles el proceso de subtitulación [162].

3.8.4. Ejemplo 4: Uso de complementos como aditamento a los periféricos (ratón)

En [162] detallan el uso de un complemento para el navegador, con el cual, los usuarios pueden navegar usando el ratón. Este complemento controla la posición y detecta todos los objetos sobre los que se encuentra ratón, e inmediatamente le informara al usuario por medio de audio la función y las características del objeto. Yu et al., mencionan que al poner aprueba el complemento muchos de los participantes pudieron detallar e identificar correctamente los elemento de la página. Este trabajo claramente mejoraría la accesibilidad de las aplicaciones web, por supuesto utilizando el atributo **alt** de los elementos de la página, salvo otras implementaciones propias.

3.8.5. Ejemplo 5:Herramientas para las pantallas táctiles braille multilínea

MathML es el lenguaje que tiene como objetivo expresar comandos que cualquier maquina pueda entender y es usado en combinación con XHTML. Este es un lenguaje fundamental para que la información pueda llegar a las personas ciegas utilizando las pantallas táctiles braille multilínea. Gracias a estas herramientas se pueden transmitir incluso imágenes a través de múltiples líneas a personas ciegas [163].

3.8.6. Ejemplo 6: Combinaciones de colores

Para ayudar a las personas con problemas de lectura debido al contraste, se puede implementar de una forma sencilla, un panel de configuraciones para el color del texto y el color de fondo. Hoy en día varias plataformas, herramientas de autor y agentes de usuario han implementado el modo oscuro, este no sólo ayuda a las personas con dislexia, sino que también ayuda a los usuarios considerados normales ha leer con facilidad especialmente en la noche [112].

Esfuerzos para cumplir con la accesibilidad web

El W3C para ayudar a que los sitios web sean accesibles, dictamina: Directrices, principios y pautas para diseñar sitios web accesibles.

3.9. Evaluación de sitios web

La creación de una aplicación web de empresa tiene como objetivo inicial ocupar y ampliar rápidamente el mercado y ofrecer a los clientes productos, información y servicios de forma más rápida y eficaz utilizando técnicas y métodos de información avanzados [164]. En difícil reconocer el éxito de la aplicación, sino hasta cuando al aplicación este desplegada y usándose por los usuario. Las razones del fracaso de los sitios web han sido estudiados por diversos autores. El trabajo [164] las resume en ocho razones:

1. El desconocimiento sobre lo que conlleva después de desplegar una aplicación web en Internet, hace que no se considere el posicionamiento del sitio web. Muchas empresas limitan la acción de una aplicación web a un anuncio, sin alcanzar su verdadera producción.

2. Una aplicación web (Sitio web) con información desordenada, y abundante, hace que los usuarios no obtengan la información que están buscando. Teniendo la información que el usuario busca, sin embargo **no está disponible**.

3. La creación de una aplicación web sin planificación hace que en algunas de sus páginas existan bloques sin información.

4. La poca consciencia del mantenimiento y la gestión hacen que los usuarios no puedan obtener información actualizada o un servicio adecuado a los cambios del momento.

5. La falta de un diseño interactivo hace que el sitio carezca de medios para comunicarse con los usuarios y responda con lentitud sin respuesta.

6. En un sitio de comercio electrónico, la aplicación web no puede explotar su valor de producción sin los módulos de pedido, pago y retroalimentación de información (parte de las directrices de la W3C).

7. Cuando uno de los objetivos del diseño de la página web es la apropiación del efecto de visión, se utiliza cantidades exageradas de imágenes, audio y animaciones. En lugar de lograr efectos positivos en los usuarios, logran efectos negativos (antipatía) y entorpecen la velocidad de navegación entre las páginas.

8. La no participación de los directivos de la organización en el desarrollo de la aplicación web, específicamente en el diseño, el diseñador no podrá comprender la situación de la empresa, obteniendo una estructura difícil de entender y contenidos caóticos.

Pese a la revisión que han realizado los autores de [164] de cara a explicar las razones del fracaso de los sitios web, no es suficiente lo que mencionan a la accesibilidad como factor dominante del fracaso de las aplicaciones web. Sin embargo en materia de accesibilidad los sitios web se evalúan, y es decisión de los propietarios implementar los cambios necesarios para que estos sean accesibles a todas las personas.

En trabajos como [165,166] evalúan sitios web frente a los 3 principios que deben cumplir para que una aplicación web sea accesible: perceptible, operable, entendible y robusto [100,164–167]. Arasid, et. al [165], analizaron los sitios web de 13 universidades de Java Occidental (Indonesia) utilizando la herramienta automática TAW. Por su lado Domínguez et. al [166], con la misma herramienta que Arasid et. al [165], analizaron los sitios web oficiales de turismo de 190 países de los 216 incluidos en el último informe de la Organización Mundial del Turismolos criterios de cumplimiento de cada directriz y de cada principio, para los niveles de conformidad AA y AAA. Ambos estudios buscaban informar a las partes interesadas para que mejoren en los aspectos que están débiles.

La herramienta de evaluación de la accesibilidad de aplicaciones web es un programa de software o servicios en línea que ayuda a determinar si un sitio web (aplicación web) cumple o no con las pautas de accesibilidad web.

La evaluación de la accesibilidad de los sitios web ha sido de preocupación de investigadores y/o desarrolladores. Evidencia de esto es la existencia de una amplia gama de herramientas con este objetivo. Cada una con sus respectivos alcances y limitaciones, como AChecker, Amp, A-Prompt 1.0, Bobby, EvalAccess 2.0, eXaminato, HERA, MAGENTA 2.0, TAW, WAVE, entre otras [166].

A estas herramientas se las puede clasificar según el ámbito de la evaluación, en dos tipos:

- Herramientas de evaluación general, que evalúan casi cualquier pauta. Entre ellas se pueden nombrar por ejemplo, TAW y AChecker.
- Herramientas de evaluación específicas que evalúan temas concretos cubiertos por las pautas de accesibilidad web, como Color Checker.

La accesibilidad debe evaluarse para garantizar que el sitio web proporciona el objetivo deseado. La evaluación de la accesibilidad del sitio web puede ayudar a los diseñadores y desarrolladores a mejorar su sitio web basándose en los resultados de la evaluación.

Por lo tanto, la evaluación de la accesibilidad tiene dos momentos importantes: (1) Evaluación continua durante el desarrollo de la aplicación web o sitio web, y (2) la evaluación que puede ser programada o en momentos dispersos cuando se

formulan nuevas directrices de accesibilidad o al recibir retroalimentación de los usuarios (internautas).

3.9.1. Herramientas de evaluación automática de la accesibilidad web

en el sitio oficial del W3C muestran una lista de 159 herramientas. Para hacer un filtro considerable se ha escogido las herramientas que muestran información actualizada al año 2021.

A11y-sitechecker por Martín Forstner

A11y-sitecheker es una herramienta para verificar un sitio completo según los criterios de accesibilidad. Utiliza axe-core[1] para verificar sitios completos en busca de problemas de accesibilidad. Rastrea el primer dominio dado e intenta encontrar todos los enlaces de forma recursiva. Además, es posible hacer clic en elementos que tienen un tabindex >= 0.

Información sobre la herramienta:

- https://www.npmjs.com/package/a11y-sitechecker,
- Lanzamiento: 2021-feb-27
- Información detallada sobre "A11y-sitechecker":
  - **Pautas que verifica:** WCAG 2.1 y 2.0 — Pautas de accesibilidad al contenido web 2.X del W3C.
  - **Asistente para:** Generación de informes de resultados de evaluación.
  - **Comprueba automáticamente:** Grupos de páginas web o sitios web.
  - **Idiomas:** Inglés, Alemán (Deutsch)
  - **Formatos compatibles:** WAI-ARIA, CSS, HTML.
  - **Licencia:** Fuente abierta
  - **Información de accesibilidad:** (Ninguno proporcionado)
  - **Información actualizada:** 2021-Mar-15

A11yWatch por A11yWatch LLC

Herramienta para la automatización de pruebas de accesibilidad web, ayuda a mejorar la accesibilidad de un sitio web con herramientas útiles para mejorar la productividad en todas sus páginas de manera rápida y confiable.

Información sobre la herramienta:

[1] https://www.deque.com/axe/core-documentation/

- https://www.a11ywatch.com, Versión: 0.1.0, Lanzamiento: 2020-ene-29.
- Información detallada sobre “A11yWatch”:
  - **Pautas que verifica:** WCAG 2.1, 2.0, 1.0 — Pautas de accesibilidad al contenido web X.X del W3C.
  - **Asistente para:** Generar informes de los resultados de la evaluación Proporcionar una guía de evaluación paso a paso Mostrar información dentro de las páginas web Modificar la presentación de las páginas web.
  - **Comprueba automáticamente:** Páginas web individuales, Grupos de páginas web o sitios web, Páginas restringidas o protegidas con contraseña.
  - **Idioma:** inglés.
  - **Formatos compatibles:** WAI-ARIA, CSS, HTML, XHTML, SVG, Imágenes, Texto/Idioma.
  - **Servicio en línea:** Comprobador en línea, servicio alojado, instalación del servidor.
  - **Licencia:** Fuente abierta
  - **Información de accesibilidad:** (Ninguno proporcionado)
  - **Información actualizada:** 2021-may-27

Hugr por Dig Inclusion

Hugr es un marco de pruebas de accesibilidad que ayuda a los usuarios a probar los sitios web según los criterios de las WCAG. Ayuda a los evaluadores a identificar problemas y, al mismo tiempo, sugiere soluciones. La versión principal es de uso gratuito, pero también hay una versión comercial y empresarial.

- https://hugr.app/, Versión: 1.0 beta,
- Lanzamiento: 23 de septiembre de 2020
- Información detalladasobre .Abrazo"
  - **Pautas:** WCAG 2.1, 2.0 — Pautas de accesibilidad al contenido web 2.X del W3C.
  - **Asistente para:** Generar informes de los resultados de la evaluación, proporcionar orientación de evaluación paso a paso.
  - **Idioma:** inglés.
  - **Servicio en línea:** Comprobador en línea, servicio alojado, instalación del servidor.
  - **Licencia:** Software Libre, Comercial, Empresarial.
  - **Información de accesibilidad:** (Ninguno proporcionado)
  - **Información actualizada:** 2021-feb-15

Accessi.org por Adán

Inspirado por la creciente necesidad de aumentar el conocimiento sobre cómo se deben diseñar y construir los sitios web para que sean accesibles para personas con discapacidades, el verificador de cumplimiento de accesibilidad de sitios web único de Accessi es a la vez simple de entender y completo en su resultado de prueba contra WCAG 2.1.

Información sobre la herramienta:

- https://www.accessi.org/,
- Lanzamiento: 2021-ene-02.
- Información detalladasobre "Accessi.org"
  - **Pautas:** WCAG 2.1, 2.0 — Pautas de accesibilidad al contenido web 2.X del W3C.
  - **Asistente para:** Generar informes de los resultados de la evaluación, Proporcionar orientación de evaluación paso a paso.
  - **Comprobación automáticamente:** páginas web individuales.
  - **Idioma:** inglés.
  - **Formatos compatibles:** CSS, HTML, Imágenes, Texto/Idioma, Video/Animaciones.
  - **Servicio en línea:** Comprobador en línea.
  - **Licencia:** Software libre.
  - **Información de accesibilidad:** (Ninguno proporcionado)
  - **Información actualizada:**2021-Mar-15

AccessMonitor por Fundação para a Ciência ea Tecnologia, IP

Está siendo utilizado por la Administración Pública portuguesa. Cualquiera puede obtener un informe gratuito de una página Web. Las agencias públicas pueden crear una muestra de páginas de un sitio web para monitorear; también pueden monitorear a las páginas nuevas que sean agregadas. El sistema tiene un logo dinámico que resume los resultados basados en la muestra.

Información sobre la aplicación:

- https://accessmonitor.accesibilidade.gov.pt/, Versión: 1.0,
- Lanzamiento: 2011-feb-02
- Información detallada sobre "AccessMonitor"

- **Pautas:** WCAG 2.0, 1.0 — Pautas de accesibilidad al contenido web del W3C X.0 del W3C.
- **Asistente para:** Generación de informes de resultados de evaluación.
- **Comprueba automáticamente:** Páginas web individuales, Grupos de páginas web o sitios web.
- **Idioma:** Portugués (Português)
- **Formatos compatibles:** CSS, HTML, XHTML.
- **Servicio en línea:** Comprobador en línea.
- **Licencia:** Software libre
- **Información de accesibilidad:** (Ninguno proporcionado)
- **Información actualizada:** 2021-may-27

AChecker por Cantan Group

Verificador interactivo, internacional, personalizable, de accesibilidad de contenido Web. Permite a los usuarios crear sus propias pautas y crear sus propias comprobaciones de accesibilidad. Basado en las Verificaciones de Accesibilidad Abiertas (OAC)

Información sobre la herramienta:

- https://achecker.achecks.ca/checker/index.php,
- **Lanzamiento:** 2008-jul-19
- Información detallada sobre "AChecker"
  - **Pautas:** WCAG 2.0 — W3C Web Content Accessibility Guidelines 2.0, Sección 508, estándar federal de adquisiciones de EE. UU., BITV, estándar del gobierno alemán, Ley Stanca, legislación de accesibilidad italiana.
  - **Asistente para:** Generación de informes de resultados de evaluación.
  - **Comprueba automáticamente:** Páginas web únicas, páginas restringidas o protegidas con contraseña
  - **Idiomas:** Inglés, Alemán (Deutsch), Italiano (Italiano).
  - **Formatos compatibles:** CSS, HTML, XHTML.
  - **Servicio en línea:** servicio alojado.
  - **Licencia:** Software Libre, Código Abierto.
  - **Información de accesibilidad:** (Ninguno proporcionado)
  - **Información actualizada:** 2021-jul-07.

ACHEQUES por Cantan Group

ACHECKS supervisa el cumplimiento de WCAG 2 AA para sus dominios a través de paneles intuitivos de resultados de AChecker y Lighthouse que generan informes en sus sitios web. La verificación de PDF se proporciona a través de Tingtun. ACHECKS ahonda en su dominio y llama a los servicios web para generar resultados agregados y página por página.

Información sobre la herramienta:

- https://achecks.ca,
- Lanzamiento: 2021-ene-01
- Información detallada sobre "CHEQUEOS":
  - Pautas:WCAG 2.1, 2.0 — Pautas de accesibilidad al contenido web del W3C 2.X, Sección 508, estándar federal de adquisiciones de EE. UU., BITV, estándar del gobierno alemán, Ley Stanca, legislación de accesibilidad italiana.
  - **Asistente para:** Generar informes de los resultados de la evaluación, Proporcionar orientación de evaluación paso a paso.
  - **Comprueba automáticamente:** Páginas web individuales, Grupos de páginas web o sitios web.
  - **Idioma:** inglés.
  - **Formatos compatibles:** Documentos WAI-ARIA, CSS, HTML, XHTML, PDF.
  - **Servicio en línea:** Servicio alojado.
  - **Licencia:** Comercial
  - **Información de accesibilidad:** (Ninguno proporcionado)
  - **Información actualizada:** 2021-ago-03

AInspector WCAG por La Universidad of Illinois at Urbana-Champaign

AInspector Sidebar es un complemento de Firefox que evalúa el DOM de una página web para los requisitos WCAG 2.0 Nivel A y AA. Hay dos conjuntos de reglas disponibles para usar en la evaluación: técnicas HTML4 y técnicas HTML5 + ARIA. Proporciona resúmenes y resultados a nivel de elemento.

Información de la herramienta:

- https://addons.mozilla.org/en-US/firefox/addon/ainspector-wcag/,
- **Versión:** 1.0.0,
- **Lanzamiento:** 2016-ago-24

- Información detallada sobre "AInspector WCAG"
  - **Pautas:** WCAG 2.0 — Directrices de accesibilidad al contenido web 2.0 del W3C.
  - **Asiste por:** Generar informes de resultados de evaluación, Mostrar información dentro de páginas web.
  - **Comprueba automáticamente:** páginas web individuales.
  - **Idioma:** inglés.
  - **Plugin para el navegador:** Firefox.
  - **Formatos compatibles:** CSS, HTML, XHTML, Imágenes.
  - **Licencia:** Fuente abierta
  - **Información de accesibilidad:** (Ninguno proporcionado)
  - **Información actualizada:** 2021-jun-28

Compliance Sheriff por Cryptzone

El Sheriff de Cumplimiento garantiza que el contenido en línea cumpla con los estándares y regulaciones de accesibilidad, privacidad, calidad y personalizados. Permite a los clientes identificar y corregir problemas rápidamente y realizar un seguimiento de las mejoras en todas sus propiedades web.

Información sobre la herramienta:

- https://www.compliancesheriff.com/,
- **Versión:** 5.1,
- **Lanzamiento:** 12 de agosto de 2016
- Información detalladasobre "Compliance Sheriff"
  - **Pautas:** WCAG 2.0, 1.0: Pautas de accesibilidad al contenido web del W3C X.0, Sección 508, estándar federal de adquisiciones de EE. UU.
  - **Asiste por:** Generar informes de resultados de evaluación, Mostrar información dentro de páginas web.
  - **Comprueba automáticamente:** Páginas web individuales, Grupos de páginas web o sitios web, Páginas restringidas o protegidas con contraseña.
  - **Idioma:** inglés.
  - **Complementos del navegador:** Chrome, Firefox, Safari, Internet Explorer $\leq$ 8.
  - **Formatos compatibles:** HTML, XHTML, imágenes, documentos de Microsoft Office.

- **Servicio en línea:** Comprobador en línea, servicio alojado, instalación del servidor
- **Licencia:** Prueba o demostración, comercial, empresarial.
- **Información de accesibilidad:** (Ninguno proporcionado)
- **Información actualizada:** 2021-may-27.

Access Assistant Community Edition por Level Access

Sirve para probar la accesibilidad de las páginas web de forma instantánea, directamente en el navegador. La mayoría de las herramientas de prueba de accesibilidad probarán solo el código estático. Con Access Assistant puede probar todas las iteraciones del código, incluidos los componentes web. Los elementos con violaciones se resaltan en la página.

Información de la herramienta:

- https://www.webaccessibility.com/,
- Versión: 5.0,
- Lanzamiento: 2017-dic-15
- Información detallada acerca de "Access Assistant Community Edition"
  - **Pautas:** WCAG 2.1, 2.0: Pautas de accesibilidad al contenido web del W3C 2.X, Sección 508, estándar federal de adquisiciones de EE. UU.
  - **Asistente para:** Generar informes de los resultados de la evaluación Proporcionar una guía de evaluación paso a paso. Muestra información dentro de las páginas web.
  - **Comprueba automáticamente:** Páginas web únicas, páginas restringidas o protegidas con contraseña.
  - **Idioma:** inglés.
  - **Complementos del navegador:** cromo, firefox.
  - **Formatos compatibles:** WAI-ARIA, HTML.
  - **Licencia:** Software libre.
  - **Información de accesibilidad:** (Ninguno proporcionado)
  - **Información actualizada:** 2021-may-27

Readability Grader por Jellymetrics

Readability Grader es una herramienta que permite a los desarrolladores verificar si su contenido es fácil de leer. Genera 7 puntuaciones diferentes.

Información sobre la herramienta:

- https://jellymetrics.com/readability-grader,
- **Versión:** 1.0,
- **Lanzamiento:** 2016-Jul-15.
- Información detalladasobre "Calificador de legibilidad"
  - **Pautas:** WCAG 2.0 — Directrices de accesibilidad al contenido web 2.0 del W3C.
  - **Idioma:** inglés.
  - **Formato compatible:** Texto / Idioma
  - **Servicio en línea:** Comprobador en línea.
  - **Licencia:** Software libre.
  - **Información de accesibilidad:** (Ninguno proporcionado)
  - **Información actualizada:** 2021-may-27

Accessibility Checker por Intent Based

Una herramienta de auditoría gratuita basada en la web que escanea un sitio web en busca de los estándares WCAG actuales. Su singularidad es la presentación de los errores que encuentra en el sitio web analizado. Por cada error que encuentre el escáner, recibirá una explicación detallada al respecto, a quién afecta y múltiples opciones sobre cómo solucionarlo.

Información de la herramienta:

- https://www.accessibilitychecker.org/,
- Versión: 1,
- Lanzamiento: 2021-jun-20
- Información detallada sobre el "Comprobador de Accesibilidad"
  - **Pautas:** WCAG 2.1, 2.0 — Pautas de accesibilidad al contenido web 2.X del W3C.
  - **Asiste por:** Generar informes de los resultados de la evaluación, proporcionar orientación de evaluación paso a paso.
  - **Comprueba automáticamente:** páginas web individuales.
  - **Idioma:** inglés.
  - **Formatos compatibles:** CSS, HTML.
  - **Servicio en línea:** Comprobador en línea.

- **Licencia:** Software libre.
- **Información de accesibilidad:** https://www.accessibilitychecker.org/accessibility-statement/
- **Información actualizada:** 2021-ago-03

PDF Accessibility Checker (PAC) por PDF/UA Foundation

PDF Accessibility Checker (PAC) es una herramienta gratuita y fácil de usar para comprobar la accesibilidad de los documentos PDF. Se lanzó inicialmente en 2010 y fue la primera herramienta automatizada de validación de cumplimiento de PDF/UA. PAC 2021 también es compatible con todos los requisitos verificables por máquina de WCAG 2.1 AA.

Información de la herramienta:

- Directrices: WCAG 2.1 - Directrices de Accesibilidad al Contenido Web 2.1 del W3C, Sección 508, norma de contratación federal de EE.UU., EN 301 549, norma de accesibilidad europea, BITV, norma del gobierno alemán
- Ayuda a: Generación de informes de los resultados de la evaluación
- Idiomas: Danés (Dansk), inglés, francés (Français), alemán (Deutsch), español (Castellano)
- Formato soportado: Documentos PDF
- Licencia: Software libre
- Información sobre la accesibilidad
- Información actualizada: 2021-Sep-02

Siteimprove Accessibility Checker por Siteimprove

Basado en las capacidades de Siteimprove Intelligence Platform, Siteimprove Accessibility Checker escanea páginas web individuales en busca de problemas de accesibilidad. La extensión destaca los problemas en la página, proporciona explicaciones claras de cómo afectan a los usuarios y recomendaciones sobre cómo solucionarlos.

Información sobre la herramienta:

- https://acortar.link/hFlNBI,
- Versión: 2.0,
- Lanzamiento: 2021-jun-02
- Información detallada sobre “Siteimprove Accessibility Checker”

- **Pautas:** WCAG 2.1, 2.0: Pautas de accesibilidad al contenido web 2.X del W3C, Sección 508, Estándar federal de adquisiciones de EE. Estándar de la industria japonesa, RGAA, estándar del gobierno francés, SI 5568, pautas de accesibilidad web israelíes, Ley Stanca, legislación de accesibilidad italiana.
- **Asistente para:** Proporcionar orientación de evaluación paso a paso, mostrar información dentro de las páginas web.
- **Comprueba automáticamente:** Páginas web únicas, páginas restringidas o protegidas con contraseña.
- **Idioma:** inglés.
- **Complementos del navegador:** Chrome, Firefox, Edge/Explorer, Ópera.
- Formatos compatibles:WAI-ARIA, CSS, HTML, XHTML, Imágenes.
- **Licencia:** Software libre.
- **Información de accesibilidad:** https://siteimprove.com/en/vpat/
- **Información actualizada:** 2021-jul-05

examinator (examinador) por Carlos Benavidez

Una herramienta de evaluación automática en línea que prueba técnicas y fallas de WCAG 2.0 utilizando una métrica para reflejar los resultados de manera cuantitativa.

Información sobre la herramienta:

- http://examinador.net/,
- Versión: 2.0,
- Lanzamiento: 2005-Sep-01
- Información detallada sobre .examinador"
  - **Pautas:** WCAG 2.0 — Directrices de accesibilidad al contenido web 2.0 del W3C.
  - **Asistente para:** Generar informes de resultados de evaluación, mostrar información dentro de páginas web.
  - Comprueba automáticamente:páginas web individuales.
  - Idioma:Español (Castellano).
  - Formatos compatibles:CSS, HTML, XHTML.
  - Servicio en línea:Comprobador en línea.
  - Licencia:Software libre.
  - Información de accesibilidad:(Ninguno proporcionado).
  - Información actualizada:2021-may-27

UCDmanager by jordisan

UCDmanager es una aplicación web colaborativa GRATUITA destinada a gestionar y documentar diferentes técnicas de Diseño Centrado en el Usuario y usabilidad, de manera integrada: roles de usuario, personas, evaluaciones heurísticas, pruebas de usabilidad del usuario, etc.

Información de la herramienta:

- https://uxmanager.net/,
- Lanzamiento: 2013-ene-01
- Información detallada sobre "UCDmanager"
  - **Pautas:** WCAG 2.0, 1.0 — Pautas de accesibilidad al contenido web del W3C X.0.
  - **Asiste por:** Generar informes de los resultados de la evaluación, Proporcionar orientación de evaluación paso a paso.
  - **Idioma:** inglés.
  - **Servicio en línea:** Instalación del servidor.
  - **Información de accesibilidad:** (Ninguno proporcionado)
  - **Información actualizada:** 2021-may-27

axe Accessibility Linter for Visual Studio Code by Deque Systems, Inc.

Filtro de accesibilidad para HTML, JSX, TSX, Vue y Markdown.

Información de la herramienta:

- https://acortar.link/ygPRqz,
- **Versión:** 4.2.1,
- **Lanzamiento:** 2021-Abr-01
- Información detallada sobre "axe Accessibility Linter for Visual Studio Code"
  - **Pautas:** WCAG 2.1, 2.0 — Pautas de accesibilidad al contenido web 2.X del W3C.
  - **Asistente para:** Generación de informes de resultados de evaluación.
  - **Herramienta de autor:** estudio visual de microsoft
  - **Comprueba automáticamente:** páginas web individuales.
  - **Idioma:** inglés.

- **Formatos compatibles:** WAI-ARIA, CSS, HTML, XHTML, SVG, Markdown, Texto/Idioma.
- **Licencia:** Software libre.
- **Información de accesibilidad:** (Ninguno proporcionado)
- **Información actualizada:** 2021-sep-30

OzART por AccessibilityOz

Pruebas por lotes automatizadas de páginas/sitio con guías de flujo de trabajo para manejar problemas que no se pueden determinar automáticamente.

Información sobre la herramienta:

- http://www.accesibilidadoz.com/ozart/,
- Versión: 4.2,
- Lanzamiento: 2014-Ene-01
- Información detallada sobre "OzART"
  - **Pautas:** WCAG 2.1, 2.0 — Pautas de accesibilidad al contenido web 2.X del W3C.
  - **Asistente para:** Generación de informes de resultados de evaluación.
  - **Comprueba automáticamente:** Grupos de páginas web o sitios web, páginas restringidas o protegidas con contraseña.
  - **Idioma:** inglés.
  - **Formatos compatibles:** CSS, HTML, XHTML, documentos PDF, Imágenes.
  - **Servicio en línea:** servicio alojado.
  - **Licencia:** Comercial.
  - **Información de accesibilidad:** (Ninguno proporcionado)
  - **Información actualizada:** 2021-ago-03

Siteimprove Intelligence Platform por Siteimprove

Monitoreo de WCAG a gran escala basado en el formato ACT 1.0 y las mejores prácticas. Desde 2007, Siteimprove ha estado probando y educando a los equipos sobre accesibilidad digital, ofreciendo cursos de accesibilidad aprobados por IAAP, capacitación y el mejor soporte de su clase. Los Siteimprovers están contribuyendo activamente dentro del W3C.

Información sobre la herramienta:

- https://siteimprove.com/es/accesibilidad/,

- Lanzamiento: 2021-feb-10
- Información detallada sobre “Siteimprove Intelligence Platform”:
  - **Pautas:** WCAG 2.1, 2.0: Pautas de accesibilidad al contenido web 2.x del W3C, Sección 508, Estándar federal de adquisiciones de EE. Estándar de la industria japonesa, RGAA, estándar del gobierno francés, SI 5568, pautas de accesibilidad web israelíes, Ley Stanca, legislación de accesibilidad italiana.
  - Asiste por:Generar informes de los resultados de la evaluación Proporcionar una guía de evaluación paso a paso Mostrar información dentro de las páginas web Modificar la presentación de las páginas web.
  - Herramientas de autor:Adobe Experience Manager, Sitecore, WordPress, Drupal, TYPO3, Optimizely, Cascade CMS, Magnolia, Percussion, i:create, RWS, Jahia, Granicus, Antilles, Sitebox, iproxcms, Enonic, Xperience, Sitefinity, Umbraco, Sitevision, Plone, FirstSpirit
  - Comprueba automáticamente:Páginas web individuales, Grupos de páginas web o sitios web, Páginas restringidas o protegidas con contraseña.
  - **Idiomas:** Danés (Dansk), holandés (Nederlands), inglés, finlandés (Suomi), francés (Français), alemán (Deutsch), italiano (Italiano), japonés, noruego (Norge), portugués (Português), español (Castellano), sueco (Svenska).
  - **Formatos compatibles:** WAI-ARIA, CSS, HTML, XHTML, Documentos PDF, Imágenes, Texto/Idioma, Video/Animaciones.
  - **Servicio en línea:** servicio alojado.
  - **Licencia:** Empresa comercial.
  - **Información de accesibilidad:** https://siteimprove.com/en/vpat/
  - Información actualizada:2021-jul-05

Polypane por Polypane B.V.

Polypane es un navegador web para crear sitios web receptivos y accesibles con una multitud de herramientas de prueba de accesibilidad, descripciones detalladas de páginas y simuladores de discapacidad.

Información sobre la herramienta:

- https://polypane.app,
- Versión: 4.0,
- Lanzamiento: 14 de mayo de 2019
- Información detallada sobre "Polypane"

- **Pautas:** WCAG 2.1, 2.0 — Pautas de accesibilidad al contenido web del W3C 2.X, Sección 508, estándar federal de adquisiciones de EE. UU., BITV, estándar del gobierno alemán, Pautas nacionales irlandesas de accesibilidad de TI, JIS, estándar industrial japonés.
- **Asistente para:** Generar informes de los resultados de la evaluación, proporcionar una guía de evaluación paso a paso, mostrar información dentro de las páginas web, modificar la presentación de las páginas web.
- **Comprueba automáticamente:** Páginas web únicas, páginas restringidas o protegidas con contraseña.
- **Idioma:** inglés.
- **Formatos compatibles:** WAI-ARIA, CSS, HTML, XHTML, SVG.
- **Licencia:** Comercial.
- **Información de accesibilidad:** (Ninguno proporcionado)
- **Información actualizada:** 2021-ene-07

Ghost Inspector Accessibility Testing por Ghost Inspector, Inc.

Ghost Inspector le permite crear o registrar pruebas de navegador automatizadas para un sitio web. Pruebas que verifican tanto la funcionalidad como la accesibilidad en todo momento, y luego las ejecuta continuamente desde la nube. Sirve para realizar auditorías WCAG automatizadas con informes detallados en cualquier página o pantalla, incluso dentro de una aplicación web iniciada.

Información sobre la herramienta:

- https://ghostinspector.com/docs/accessibility-testing/,
- Lanzamiento: 2021-abr-12
- Información detalladasobre “Ghost Inspector Accessibility Testing”
  - **Pautas:** WCAG 2.1, 2.0 — Pautas de accesibilidad al contenido web del W3C 2.X, Sección 508, estándar federal de adquisiciones de EE. UU., BITV, estándar del gobierno alemán, Pautas nacionales irlandesas de accesibilidad de TI, JIS, estándar industrial japonés.
  - **Asistente para:** Generar informes de los resultados de la evaluación, Proporcionar orientación de evaluación paso a paso.
  - **Comprueba automáticamente:** Páginas web individuales, Grupos de páginas web o sitios web, páginas restringidas o protegidas con contraseña.
  - **Idioma:** inglés.
  - **Formatos compatibles:** CSS, HTML, XHTML.

- **Servicio en línea:** Comprobador en línea, servicio alojado.
- **Licencia:** Prueba o Demostración, Comercial.
- **Información de accesibilidad:** (Ninguno proporcionado)
- **Información actualizada:** 2021-may-27

Accessibility Score por AudioEye

Live Monitoring de AudioEye escanea los elementos del sitio web cada vez que un usuario carga una página. Los prueba basándose en los criterios de éxito de WCAG. En promedio, se escanea un sitio web casi 1000 veces al día. Proporcionando una puntuación de accesibilidad única para realizar un seguimiento del progreso.

Información sobre la herramienta:

- https://www.audioeye.com,
- Versión: 1.0,
- Lanzamiento: 21 de abril de 2021
- Información detallada sobre .Accessibility Score"
  - **Pautas:** WCAG 2.1, 2.0, 1.0 — Pautas de accesibilidad al contenido web X.X del W3C, EPUB Accesibilidad 1.0, Sección 508, estándar federal de adquisiciones de EE. UU., EN 301 549, estándar europeo de accesibilidad, BITV, estándar del gobierno alemán, Directrices nacionales de accesibilidad de TI de Irlanda, JIS, estándar de la industria japonesa, RGAA, estándar del gobierno francés, SI 5568, directrices de accesibilidad web de Israel, Ley Stanca, legislación de accesibilidad italiana.
  - **Asiste por:** Generar informes de resultados de evaluación, mostrar información dentro de páginas web, modificar la presentación de páginas web
  - **Comprueba automáticamente:** Páginas web individuales, grupos de páginas web o sitios web, páginas restringidas o protegidas con contraseña.
  - **Idioma:** inglés.
  - **Plugin para el navegador:** Todo.
  - **Formatos compatibles:** WAI-ARIA, CSS, HTML, XHTML, SVG, Documentos PDF, Imágenes, SMIL, Texto/Idioma, Video/Animaciones.
  - **Servicio en línea:** Comprobador en línea, servicio alojado.
  - **Licencia:** Software libre, prueba o demostración, comercial, empresarial.
  - **Información de accesibilidad:** https://acortar.link/q1Yl9N
  - **Información actualizada:** 2021-jun-28.

Accessibility Suite por Online ADA

El complemento Accessibility Suite audita los sitios web en busca de problemas de accesibilidad WCAG/Sección 508. Escanea imágenes, publicaciones, páginas y objetos en un sitio web y señala los elementos de código exactos que no cumplen. Los desarrolladores pueden usar informes detallados para actualizar su sitio web y configurar escaneos recurrentes.

Información sobre la herramienta:

- https://adaplugin.com/,
- Versión: 4.8,
- Lanzamiento: 2018-dic-04
- Información detallada sobre "Accessibility Suite"
  - **Pautas:** WCAG 2.1, 2.0: Pautas de accesibilidad al contenido web del W3C 2.X, Sección 508, estándar federal de adquisiciones de EE. UU.
  - **Asiste por:**Generación de informes de resultados de evaluación.
  - Herramienta de autor:WordPress.
  - **Comprueba automáticamente:** Páginas web individuales, grupos de páginas web o sitios web, páginas restringidas o protegidas con contraseña.
  - **Idioma:** inglés.
  - **Formatos compatibles:** WAI-ARIA, CSS, HTML, Imágenes.
  - **Licencia:** Fuente abierta.
  - **Información de accesibilidad:** (Ninguno proporcionado)
  - **Información actualizada:** 2021-sep-30.

ReachDeck por Texthelp

ReachDeck permite proteger las comunicaciones, contenido y reputación. Se puede mejorar la accesibilidad, la legibilidad y el alcance de un contenido en línea. Permite que todo el equipo de desarrollo se comunique con confianza.

Información acerca de la herramienta:

- https://www.texthelp.com/products/reachdeck/,
- Lanzamiento: 2021-Mar-10
- Información detallada sobre "ReachDeck"
  - **Pautas:** WCAG 2.1 — Directrices de accesibilidad al contenido web del W3C 2.1

- **Asiste por:** Generar informes de los resultados de la evaluación, proporcionar una guía de evaluación paso a paso, mostrar información dentro de las páginas web.
- **Comprueba automáticamente:** Grupos de páginas web o sitios web
- **Idioma:** inglés.
- **Formatos compatibles:** WAI-ARIA, CSS, HTML, XHTML, Texto/Idioma.
- **Licencia:** Prueba o demostración, comercial, empresarial. **Información de accesibilidad:** (Ninguno proporcionado)
- **Información actualizada:** 2021-jun-28.

uTester por Usuario1st

Una herramienta de prueba de regresión de accesibilidad automatizada y fácil de usar que detecta violaciones de accesibilidad, permite:

- Administrar los casos de prueba a lo largo del tiempo.
- Registrar los flujos de usuarios finales y la actividad de los usuarios.
- Programar pruebas e informes automatizados en todos los casos de prueba.
- Tableros fáciles de usar
- Exportar informes de cumplimiento

Información acerca de la herramienta:

- https://www.user1st.com/check1st,
- Versión: 1.0018,
- Lanzamiento: 2018-Abr-19
- Información detallada sobre “uTester”:
  - **Pautas:** WCAG 2.1, 2.0, 1.0 — Pautas de accesibilidad al contenido web X.X del W3C.
  - **Asiste por:** Generar informes de los resultados de la evaluación, proporcionar una guía de evaluación paso a paso, mostrar información dentro de las páginas web.
  - **Comprueba automáticamente:** Páginas web individuales, grupos de páginas web o sitios web, páginas restringidas o protegidas con contraseña.
  - **Idioma:** inglés.

- **Plugin para el navegador:** Cromo
- **Formatos compatibles:** WAI-ARIA, CSS, HTML, XHTML, documentos PDF, Imágenes
- **Servicio en línea:** Comprobador en línea, servicio alojado, instalación del servidor.
- **Licencia:** Prueba o demostración, comercial, empresarial.
- **Información de accesibilidad:** (Ninguno proporcionado)
- **Información actualizada:** 2021-may-27

www.forapp.org por SCE

forApp es un servicio en línea que verifica automáticamente la accesibilidad de las aplicaciones móviles. Los resultados de la inspección se pueden descargar en formato PDF. Puede consultar los resultados de la inspección de accesibilidad para cada equipo móvil y versión de la aplicación en cada pantalla de menú.

Información sobre la herramienta:

- https://www.forapp.org/,
- Lanzamiento: 2015-Abr-13
- Información detalladasobre "www.forapp.org"
  - **Pautas:** Sección 508, estándar federal de adquisiciones de EE. UU., MAAG 1.0 - Estándar del gobierno de Corea.
  - **Asiste por:**Generar informes de resultados de evaluación, Mostrar información dentro de páginas web.
  - **Idiomas:** inglés, coreano.
  - **Servicio en línea:** Comprobador en línea.
  - **Licencia:** Prueba o Demostración, Comercial.
  - **Información de accesibilidad:** https://www.forapp.org/
  - **Información actualizada:** 2021-may-27.

3.9.2. Frecuencia de uso de las herramientas de evaluación automática de la accesibilidad web

Las herramientas automatizadas para la evaluación de la accesibilidad de sitios web que se han utilizados y cuyos trabajos se han publicados en fuentes indexadas en la *Web of Science*, se muestran en la tabla con su número de ocurrencias:

Tabla 3.1: Frecuencia de uso de las herramientas de evaluación de accesibilidad.

Herramienta	Frecuencia
WAVE	8
Achecker	8
TAW	5
HERA	2
EIII[2]	2
EvalAccess[3]	2
SortSite	2
A-Tester	1
AInspector Sidebar	1
Tenom[4]	1
Web Access[5]	1
Siteimprove Accessibility	1
UDOIT	1
DaSilva[6]	1
Total Validator	1
W3C Markup Validation Service	1

Herramientas más utilizadas

Cono la breve búsqueda y filtrando los documentos indexadaos en una de las mejores bases de datos científicas, encontramos que las herramientas más utilizadas para la evaluación de la accesibilidad fueron las herramientas WAVE, TAW y AChecker.

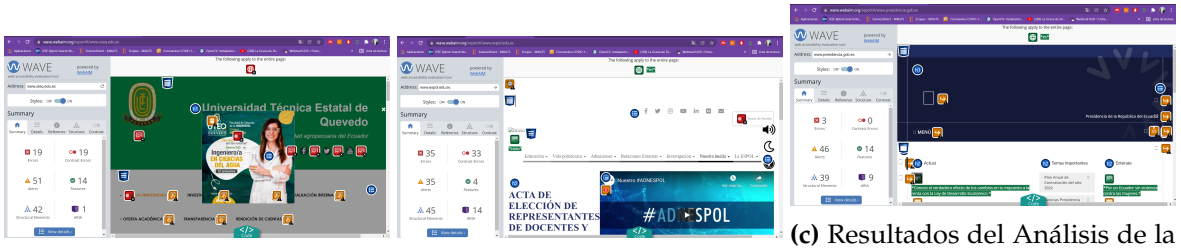
Herramienta WAVE:

Herramienta de evaluación de la accesibilidad web. Comprueba el cumplimiento de muchas de las pautas WCAG 2.0, también proporciona referencias de errores con los niveles [168]. WAVE fue desarrollado como un servicio comunitario gratuito por Web Accessibility In Mind (WebAIM). Lanzada inicialmente en 2001, esta herramienta se utiliza para evaluar la accesibilidad de millones de páginas web. WAVE cuenta con

[3]No se encuentra en el listado del W3C
[4]No se encuentra en el listado del W3C
[5]No se encuentra en el listado del W3C
[6]No se encuentra en el listado del W3C

el servicio de evaluación en línea, además de las extensiones para los navegadores Chrome y Firefox [169].

Los resultados del análisis de la accesibilidad de sitios web usando WAVE, se muestran en las figuras 3.2. En la figuraas 3.2a, 3.2b de la UTEQ (`www.uteq.edu.ec`) y de la Escuela Superior Politécnica del Litoral (ESPOL: `www.espol.edu.ec`) respectivamente, y en la figura 3.2c los resultados del sitio de la Presidencia de la República del Ecuador (`www.presidencia.gob.ec`).



(a) Resultados del Análisis de la Accesibilidad del Sitio Web de la UTEQ usando WAVE

(b) Resultados del Análisis de la Accesibilidad del Sitio Web de la ESPOL usando WAVE

(c) Resultados del Análisis de la Accesibilidad del Sitio Web de la Presidencia de Ecuador usando WAVE

Figura 3.2: Resultados de la evaluación de la accesibilidad usando Achecker

Herramienta AChecker:

Hace uso de las Comprobaciones Abiertas de Accesibilidad (OAC), que es una colección de comprobaciones basadas en todas las pautas de accesibilidad web disponibles a nivel mundial [168].

La herramienta Achecker fue otra de las herramientas que se utilizaron para evaluar a los tres sitios web institucionales analizados anteriormente con WAVE. Los resultados de esta evaluación se muestran en la figura 3.3. Los resultados de la UTEQ se muestran en la figura 3.3a, de la ESPOL en la figura 3.3b y del sitio de la Presidencia de la República en la figura 3.3c.

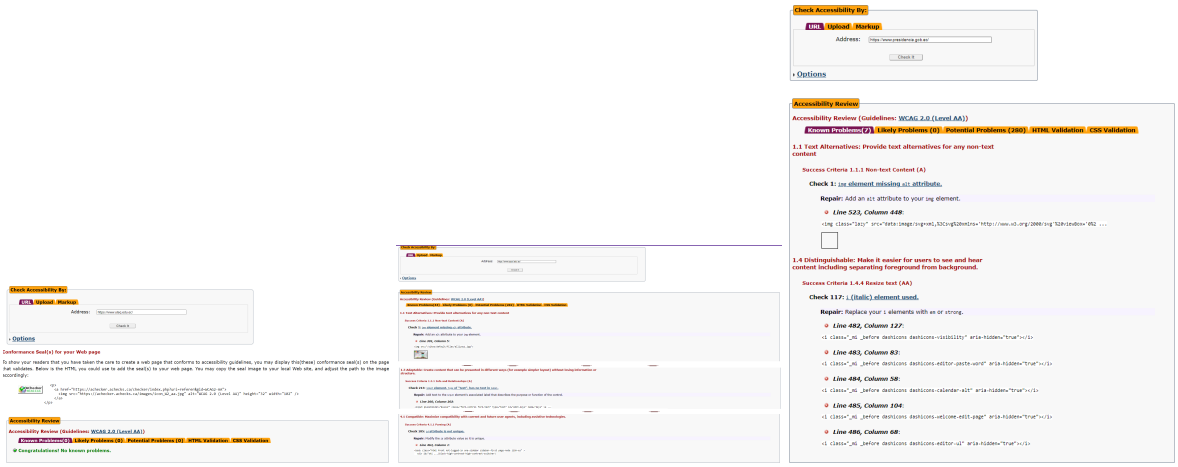
Herramienta TAW:

Test de Accesibilidad Web. Es una herramienta online para determinar la accesibilidad de la web de acuerdo con las Directrices de Accesibilidad Web del W3C y muestra los problemas de accesibilidad [168]. Ha sido desarrollado por la Fundación española CTIC[7] y es multiplataforma para la evaluación del nivel de conformidad A, AA y AAA de las WCAG 2.0 [170].

Al intentar evaluar con TAW las tres urls de los tres sitios evaluados con las otras dos herramientas aparece como error al analizar la url, tal como se muestra en la figura

Se probó de todas las formas posibles de URL: uteq.edu.ec, www.uteq.edu.ec, http://www.uteq.edu.ec, https://www.uteq.edu.ec. De la misma forma se probó para el

[7]Centro Tecnológico de la Información y la Comunicación: https://www.fundacionctic.org/



(a) Análisis del Sitio Web de la UTEQ **(b)** Análisis del Sitio Web de la ESPOL **(c)** Análisis del Sitio Web de la Presidencia de Ecuador

Figura 3.3: Resultados de la evaluación de la accesibilidad usando Achecker

Figura 3.4: Error al analizar la url de la UTEQ con TAW

sitio web de la ESPOL y de la Presidencia de la República de Ecuador. El resultado fue el mismo mensaje de error. Sin embargo, se probó un subdominio de la UTEQ en la que se tiene alojadas algunas aplicaciones, para el cual sí arrojó resultados del análisis de la accesibilidad, los cuales se muestran el la figura 3.5 (https://aplicaciones.uteq.edu.ec). Al obtener estos resultados se comprobó que se debe ingresar la URL del sitio web anteponiendo el protocolo HTTPS, en caso que sea el de su sitio web. Por defecto considera el protocolo HTTP.

Herramientas para la evaluación de la accesibilidad web

El W3C presenta un listado de 159 herramientas, siendo la más antigua TAW de *CTIC Technology Centre* lanzada el 01 de febrero de 2000, y la más reciente ACHECKS de *Cantan Group* lanzada el 01 de enero de 2021.
Entre las herramientas más usadas están: WAVE, AChecker y TAW.

Figura 3.5: Resultados del análisis de accesibilidad del subdominio **aplicaciones** de la UTEQ usando TAW

3.10. Conclusiones

La accesibilidad es importante para el desarrollo de una sociedad más inclusiva, que permita que más personas puedan acceder a la información disponible en la Web. Esto aporta a aumentar el desarrollo individual y social de las personas vulnerables. Por lo tanto, es importante que los desarrolladores de aplicaciones web, herramientas de autor y contenido web conozcan y cumplan con los estándares proporcionados por W3C para generar tecnologías que puedan utilizar las personas con discapacidades, problemas cognitivos y personas afectadas por barreras externas.

En el sitio oficial de la accesibilidad web (https://www.w3.org) existe mucha información sobre este tema, desde los objetivos de cada lineamiento, pauta y criterio de éxito, hasta como lograr cada uno de ellos. Es importante hacer notar que, el desarrollo de una aplicación web planeada como una accesible, no es más costoso que al no ser accesible. Claro está que lo que aumentaría el costo es la formación o capacitación que los desarrolladores deben invertir para lograr el desarrollo inclusivo universalmente.

Evaluación a los lectores

Una vez finalizado el estudio de este capítulo, el lector debe dar respuesta a los siguientes planteamientos:

- Explicar con sus propias palabras (referenciado a trabajos de investigación y otros textos) la importancia de la accesibilidad web hoy en día. Debe enfocarse a todas las personas, sin excepción.

- Elaborar una tabla para ubicar los organismos de la W3C y los logros alcanzados por cada uno de ellos en materia de accesibilidad web.

- Realice una tabla indicando las pautas, los principios, niveles de conformidad y su aplicación (contenido, herramientas de autor o agentes de usuario), el problema que soluciona, y exponer un ejemplo explicativo de su aplicación.
- Tomar como ejemplo el sitio web de tres universidades, por ejemplo: de la UTEQ `www.uteq.edu.ec`, de la Universidad de Granada `www.ugr.es`, y de la universidad de Hardvard `https://www.harvard.edu/` y examinar aspectos de accesibilidad con tres herramientas automáticas a su elección. Verifique los errores que presenten y deles una solución según su criterio. Debe documentar con las pautas de la accesibilidad que se deben aplicar para su solución.

Capítulo 4

ENTORNOS DE DESARROLLO INTEGRADO PARA JAVA

Objetivos

Al finalizar la lectura y la práctica de este capítulo, el estudiante será capaz de:

- Reconocer las interfazs y funcionalidades de los IDE de desarrollo más conocidos.
- Seleccionar al IDE idóneo para el desarrollo de una aplicación, por las bondades que se requieran de él para que el equipo de desarrollo pueda ser más productivo.
- Utilizar de manera productiva el IDE de desarrollo que escoja.

Resumen

La selección del IDE es muy importante cuando se requiere productividad del equipo de desarrollo. Muchos IDE de desarrollo existen en el mercado. Los dos más populares en Ecuador son Eclipse y Netbeans. Ambos no disponen de herramientas WYSIWYG, y en cuanto a sus potencialidades pare el desarrollo de aplicaciones web son muy similares.

4.1. Introducción

Un Entorno de Desarrollo Integrado o IDE por sus siglas en inglés *Integrated Development Environment*, es una aplicación para computadoras usada por los desarrolladores de software en la creación, depuración, integración de programas. Un IDE tiene como objetivo facilitar el desarrollo del software, con el uso de múltiples herramientas desde compiladores y depuradores de código de software, hasta inclusive herramientas de pruebas de los aspectos fundamentales de las aplicaciones: cómputo, comunicaciones, e intercambio de datos e información con otros sistemas [171, 172]. Además muchos IDE tienen la ventaja que permiten diseñar de interfazs para la interacción con el usuario

con la técnica conocida con el acrónimo WYSIWYG, derivado de ***W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et* ("o que ves es lo que obtienes").

Este libro está orientado a guiar a los desarrolladores en la toma de decisiones en el uso de las tecnologías de desarrollo, mas el diseño de interfazs no es parte de su contenido. Por esta razón, Los IDE que vamos a presentar es Apache Netbeans (Netbeans), y eclipse, por ser los de mayor interés en el Ecuador [173]. En sus versiones actuales no disponen de herramientas WYSIWYG. Además se considera a Windows como la palataforma de desarrollo.

Entornos de desarrollo integrado

Un IDE es un conjunto de programas que permiten a los desarrolladores crear, compilar y depurar software, y la configuración en aspectos de comunicaciones e intercambio de datos con otros sistemas.

4.2. Apache NetBeans

El propósito de un entorno de desarrollo integrado (IDE) es maximizar la productividad y respaldar el desarrollo sin problemas desde una sola herramienta. Uno de los criterios de selección de un IDE para el desarrollo un proyecto, es el apoyo que éste brinda al equipo de desarrollo en cuanto a ser más productivo. Entre los entornos de desarrollo para Java está NetBeans.

NetBeans es un entorno de desarrollo integrado libre, se considera como el editor principal del lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

La versión última de Netbeans fue la 8.2 que aún se está usando en el mercado y en la academia. Posterior a esa versión surgió el IDE **Apache Netbeans** y la actualización de este trabajo cuenta con la versión 12.

A continuación se describen las funciones de asistencia para el código, las opciones de personalización y las capacidades de navegación del editor Java del IDE Apache NetBeans 12.0. Aunque no hay cambios significantes en las prestaciones del editor de Netbeans 8.2 a las de Apache Netbeans 12.0.

4.2.1. Funciones generales del editor

Conocer la interfaz del IDE es de suma importancia de cara a obtener la mayor productividad posible del equipo de desarrollo. Así mismo con la configuración de su editor de código debe ser de tal manera que se saque provecho a sus prestaciones, como por ejemplo, reconocer las palabras reservadas, los identificadores, los tipos de constantes, entre otros. Entre las funciones generales del editor, se pueden nombrar los siguientes [174].

Formato del texto de código

Para formatear el texto del código puede hacerlo de las siguientes maneras:

- Presionando las teclas Alt+Shift+F
- Accediendo al menú **Herramientas** luego seleccione **Opciones** posteriormente **Editor** y seleccione **Formato**.
- Seleccione **Fuente** y luego seleccione **Formato** en la barra de menú.
- En el editor de código fuente, presione el botón secundario del ratón y seleccione **Formato**.

En la figura 4.1 se muestran las diferentes opciones para configurar el editor de código fuente de Netbeans.

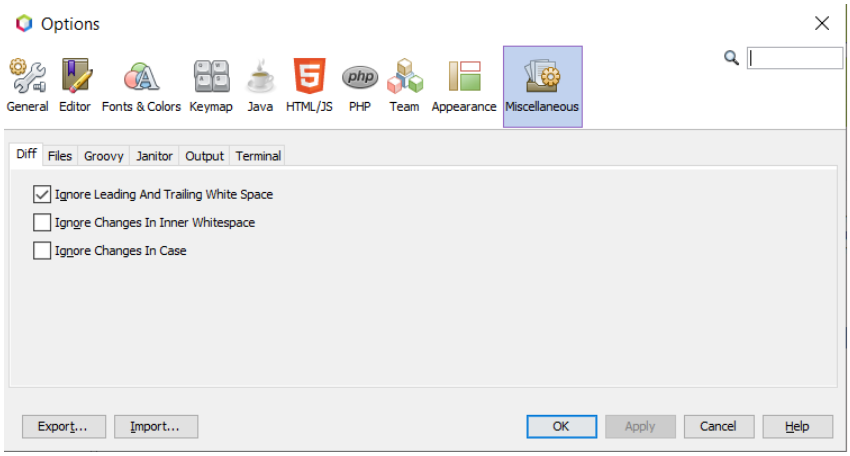


Figura 4.1: Ventana de las opciones de configuración del editor

Insertar y resaltar llaves, corchetes y comillas

El editor inteligente de NetBeans, cuando el programador inserta un elemento que necesita de su pareja coincidente, éste es insertado automáticamente. Pares coincidentes como: (, {, [, ", y '. La ventana con las opciones para habilitar esta función, si fuese el caso, se muestra en la figura 4.2. Para mostrar esta ventana puede hacerlo desde el menú **Herramientas** >**Opciones** >**Editor** >**Completar código**.

Colores de resaltado

Para personalizar los colores de resaltado, seleccione desde el menú **Herramientas** >**Opciones** >**Fuentes y colores** >**resaltado**. La ventana que permite hacer los cambios de manera interactiva se muestra en la figura 4.3

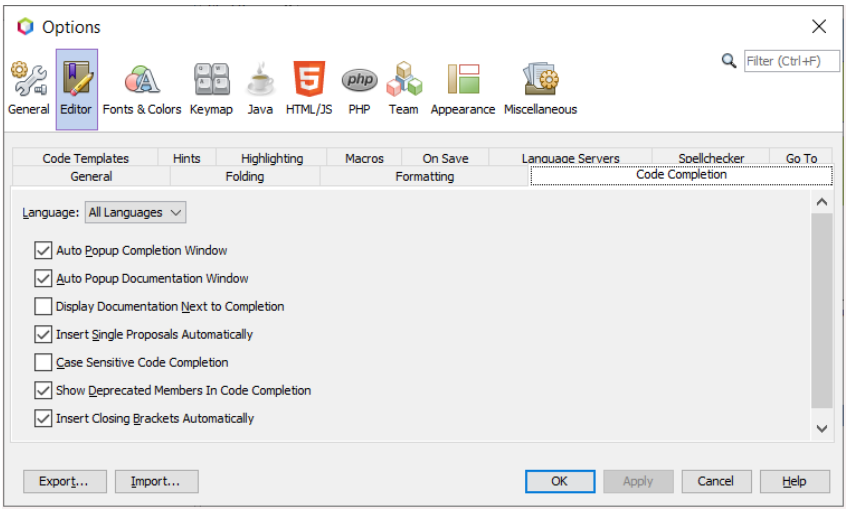


Figura 4.2: Opciones de autocompletar código

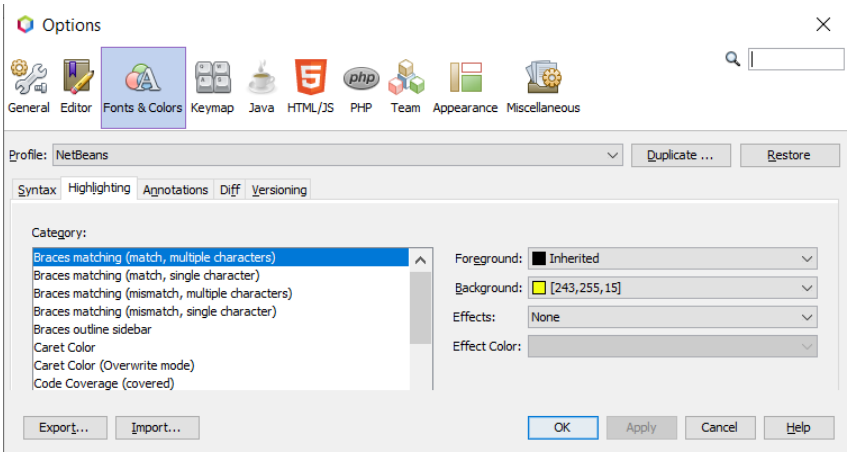


Figura 4.3: Ventana de las opciones para las fuentes y colores del editor

Plegado de código

El plegado de código permite al desarrollador identificar fragmentos de código sin necesidad de tenerlo todo el código en la pantalla, desde el código de una clase hasta el código de una estructura de control.

Netbeans permite crear pliegues (bloques) de código, como declaraciones de métodos, comentarios de Javadoc, declaraciones de importación, etc.. Estos pliegues de código pueden ser contraídos o expandidos de manera fácil. Un pliegue de código al estar expandido, este puede ser contraído. Esta posibilidad se muestra mediante una línea gris (signo menos: -) en un cuadro adjunto a la parte superior de la línea en el margen izquierdo del editor. Así mismo, los pliegues que pueden ser expandibles, se muestran mediante un cuadro con un signo más (+) en el mismo margen. Observe la figura 4.4.

Otras maneras para contraer y expandir bloques de código es seleccionando en la barra de menú **Ver** >**Pliegues de código** o, con el puntero del ratón señalando el

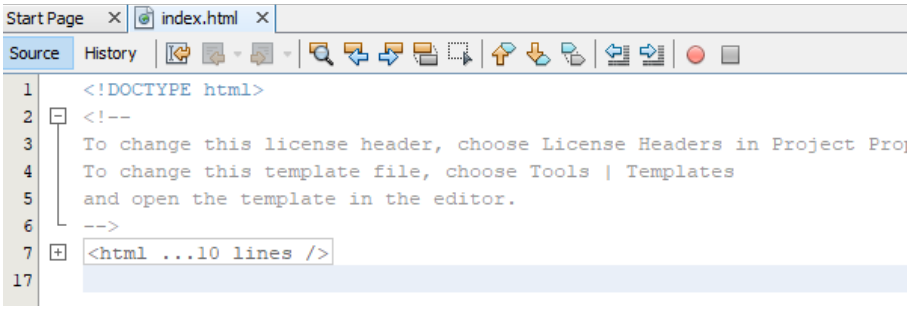


Figura 4.4: Editor con bloques de código para contraer (-) para expandir (-)

pliegue de código sobre el cuál se desea accionar se hace clic en el botón secundario y se selecciona Pliegues de código, luego expandir pliegues o contraer pliegues, dependiendo de lo que se requiera hacer. Otra manera de hacerlo es mediante los atajos de teclado. Las opciones para la personalización de plegado de códigos se encuentran en la ventana opciones de configuración del editor, para ello se debe seleccionar en la barra de menú: **Herramientas** >**Opciones** >**Editor** >**Plegado**. Todas las opciones se muestran en la figura 4.5.

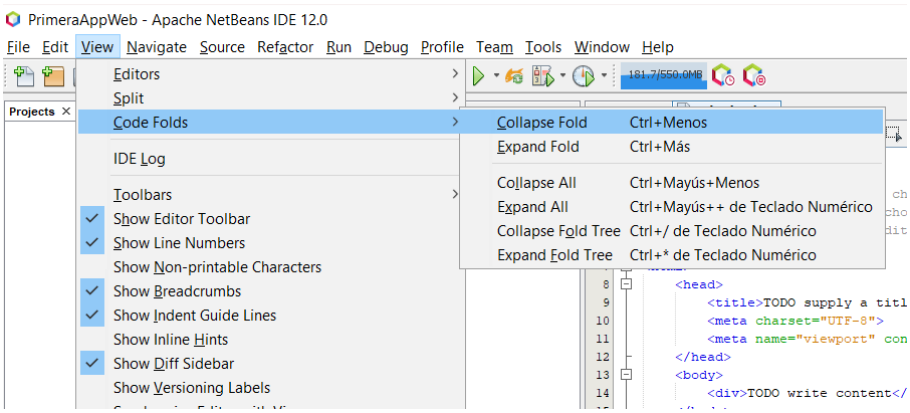


Figura 4.5: Opciones de acceso para contraer y expandir pliegues de código

Personalización de atajos de teclado

NetBeans IDE permite personalizar atajos de teclas diferentes en perfiles diferentes según las preferencias del(os) desarrollador(es). Esta es una opción que se puede utilizar para personalizarlos, ya sea para cada lenguaje de programación, o para cada desarrollador cuando comparten el equipo entre varios desarrolladores.

El atajo de teclas o accesos rápidos, en inglés *shortcuts* son otro factor importante que apoya a los desarrolladores para la edición rápida de código e inclusive ayuda a disminuir los errores de digitalización, como por ejemplo al usar la opción de insertar código.

Se puede personalizar los atajos de teclas desde la barra de menú, seleccionando **Herramientas** >**Opciones** >**Mapa de teclas**. La ventana que cargará en la pantalla con las opciones se muestra en la figura 4.6.

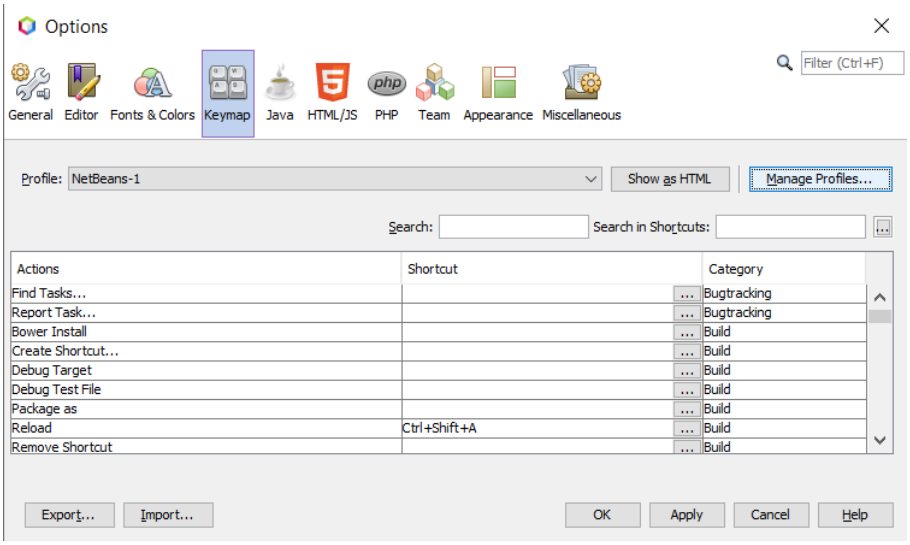
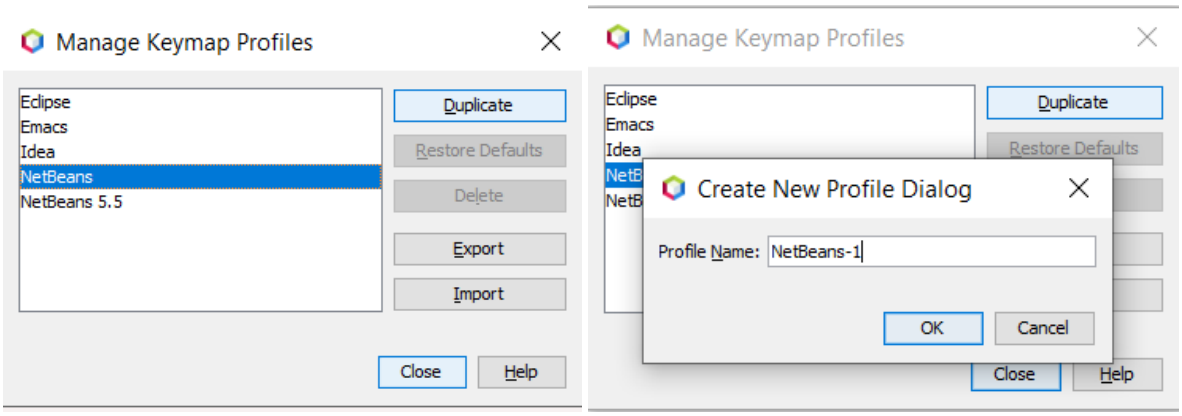


Figura 4.6: Ventana para acceder a cambiar los atajos del teclado

Al tener en pantalla a la ventana de la figura 4.6 debe dar clic en el botón **Administrador de perfiles**. En la figura 4.7 se muestra los pasos que se deben ejecutar. Primero en la ventana de Administración de perfiles que se muestra en la figura 4.7a, debe seleccionar el perfil que se desea duplicar. En este caso, se especifica duplicar el perfil NetBeans. Después de seleccionar Netbeans, se debe dar clic en el botón **Duplicar**, como se muestra en la figura 4.7a.



(a) Para seleccionar el perfil a duplicar para personalizar los atajos del teclado

(b) Ventana donde se escribe el nombre del nuevo perfil

Figura 4.7: Personalizar perfiles para atajos de teclas en el editor inteligente de Apache NetBeans

En la figura 4.7b debe escribir el nombre para el nuevo perfil, posteriormente en el botón Ok (Aceptar) y posteriormente (en la ventana anterior 4.7a) clic en el botón

Cerrar, luego nuevamente estará en su pantalla la ventana de la figura 4.6, en la cual puede dar clic en el botón *Ok*, si así se requiere.

Sugerencia al escribir código

Aunque Netbeans es un editor para múltiples lenguajes de programación, los ejemplos que se mostrarán a continuación como casos ilustrativos están basados en lenguaje de programación Java.

Este editor inteligente puede reducir en gran parte el tiempo en la digitación de código al presentarle al desarrollador la opción de escoger el código para iniciar la escritura con tan sólo el atajo de **[Cntrl + Barra espaciadora]**. Este editor le sugiere desde los tipos de datos hasta los nombres de variables.

Cuando haya escrito parte de la instrucción (tipo de dato, nombre de variable, nombre de clase, o de cualquier elemento de programación) el editor le desplaza las alternativas para completar la escritura con tan sólo digitando la tecla **Tab**. Además si escribe el nombre de una clase o elemento que se necesita importar (clase o interfaz por ejemplo), el desarrollador únicamente debe usar el atajo de teclas **[Cntrl + Barra espaciadora]** y el escoger el origen del elemento y el editor se encarga de importarlo del origen escogido.

Detalle de atajos de teclado para el desarrollo en Java

Los grupos de atajos de teclado que se han considerado bajo el criterio de la experiencia como los más utilizados se muestran a continuación como una alternativa más rápida o como cumplimiento de la inclusiva para los desarrolladores que tienen problemas motrices, entre otros.

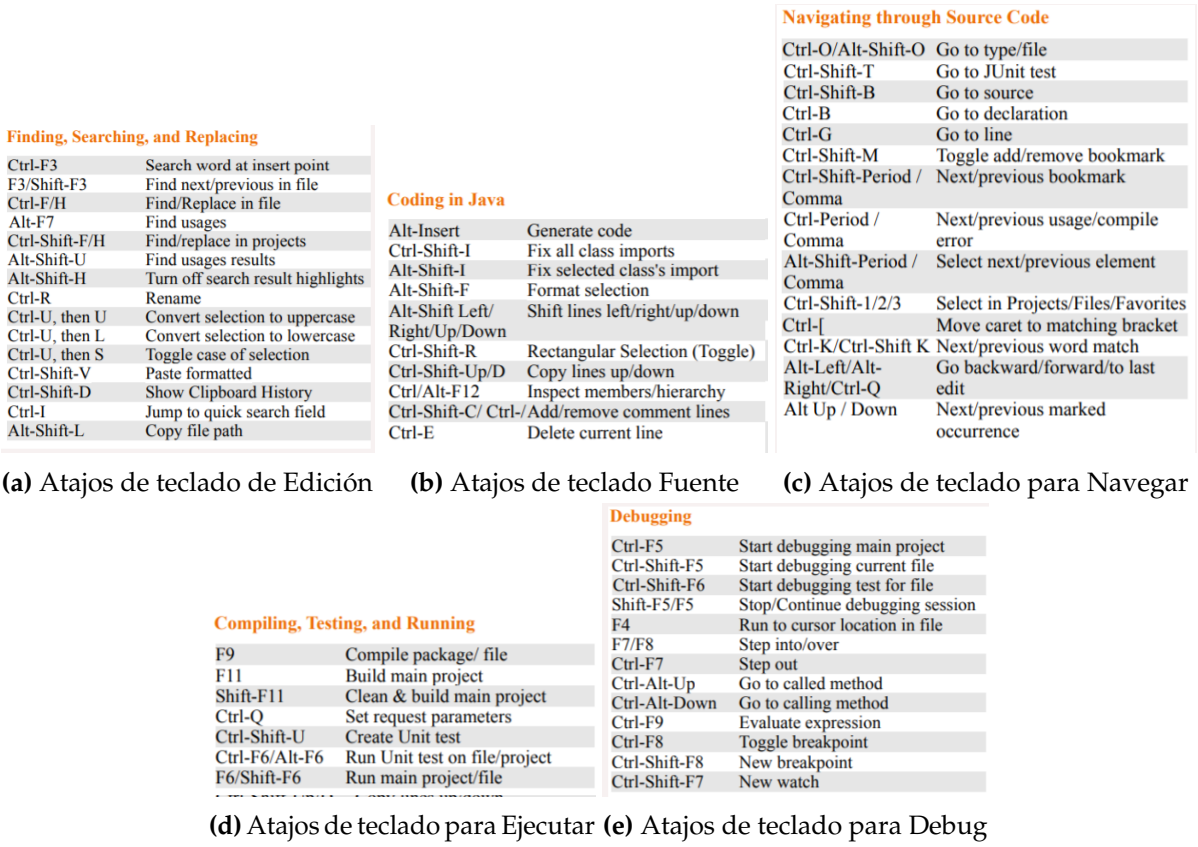
En la figura 4.8 muestra algunos de los atajos de teclas que implementa el editor de Apache NetBeans. En la 4.8a muestras la combinación de teclas para las opciones que se encuentran en el menú **Editar**, y en la figura 4.8b muestras mediante la combinación del teclado cómo ejecutar las acciones que están entre las opciones del menú **Fuente**. Las opciones del menú fuente le pueden ayudar al desarrollador a ser más productivo en su trabajo, en vista de que, le permiten generar códigos repetitivos como: funciones `getter` y `setter`, constructores, funciones que se deben sobre-escribir como la función `toString` y la función `equals`, por mencionar algunas.

Cuando un desarrollador está revisando el código o en su defecto necesita reestructurar el código, las opciones del menú **Navegar** le resultan muy convenientes. En la figura 4.8c le muestra un listado de combinación de teclas que le ayudarán a ser del trabajo del desarrollador más productivo respecto a navegar por el código del proyecto.

El IDE Netbeans tiene las opciones de compilación, pruebas y ejecución de aplicaciones. Estás opciones los desarrolladores pueden encontrarlas en el menú **Ejecutar** y en el menú **Depurar**. En la figura 4.8d muestra un resumen de las opciones de uso más frecuente para las tareas relacionadas con la compilación. Así mismo la combinación de teclas que ayudan a las tareas de depuración del código de las aplicaciones se muestran

en el listado de la figura 4.8e. Estas opciones últimas las pueden encontrar en el menú **Depurar**.

Aunque existen muchas otras combinaciones de teclas especialmente para la edición de código, no se especificarán de lleno en este trabajo, sino más bien poco a poco más adelante como se vayan necesitando.



Finding, Searching, and Replacing

Ctrl-F3	Search word at insert point
F3/Shift-F3	Find next/previous in file
Ctrl-F/H	Find/Replace in file
Alt-F7	Find usages
Ctrl-Shift-F/H	Find/replace in projects
Alt-Shift-U	Find usages results
Alt-Shift-H	Turn off search result highlights
Ctrl-R	Rename
Ctrl-U, then U	Convert selection to uppercase
Ctrl-U, then L	Convert selection to lowercase
Ctrl-U, then S	Toggle case of selection
Ctrl-Shift-V	Paste formatted
Ctrl-Shift-D	Show Clipboard History
Ctrl-I	Jump to quick search field
Alt-Shift-L	Copy file path

(a) Atajos de teclado de Edición

Coding in Java

Alt-Insert	Generate code
Ctrl-Shift-I	Fix all class imports
Alt-Shift-I	Fix selected class's import
Alt-Shift-F	Format selection
Alt-Shift Left/ Right/Up/Down	Shift lines left/right/up/down
Ctrl-Shift-R	Rectangular Selection (Toggle)
Ctrl-Shift-Up/D	Copy lines up/down
Ctrl/Alt-F12	Inspect members/hierarchy
Ctrl-Shift-C/ Ctrl-/	Add/remove comment lines
Ctrl-E	Delete current line

(b) Atajos de teclado Fuente

Navigating through Source Code

Ctrl-O/Alt-Shift-O	Go to type/file
Ctrl-Shift-T	Go to JUnit test
Ctrl-Shift-B	Go to source
Ctrl-B	Go to declaration
Ctrl-G	Go to line
Ctrl-Shift-M	Toggle add/remove bookmark
Ctrl-Shift-Period / Comma	Next/previous bookmark
Ctrl-Period / Comma	Next/previous usage/compile error
Alt-Shift-Period / Comma	Select next/previous element
Ctrl-Shift-1/2/3	Select in Projects/Files/Favorites
Ctrl-[	Move caret to matching bracket
Ctrl-K/Ctrl-Shift K	Next/previous word match
Alt-Left/Alt-Right/Ctrl-Q	Go backward/forward/to last edit
Alt Up / Down	Next/previous marked occurrence

(c) Atajos de teclado para Navegar

Compiling, Testing, and Running

F9	Compile package/ file
F11	Build main project
Shift-F11	Clean & build main project
Ctrl-Q	Set request parameters
Ctrl-Shift-U	Create Unit test
Ctrl-F6/Alt-F6	Run Unit test on file/project
F6/Shift-F6	Run main project/file

(d) Atajos de teclado para Ejecutar

Debugging

Ctrl-F5	Start debugging main project
Ctrl-Shift-F5	Start debugging current file
Ctrl-Shift-F6	Start debugging test for file
Shift-F5/F5	Stop/Continue debugging session
F4	Run to cursor location in file
F7/F8	Step into/over
Ctrl-F7	Step out
Ctrl-Alt-Up	Go to called method
Ctrl-Alt-Down	Go to calling method
Ctrl-F9	Evaluate expression
Ctrl-F8	Toggle breakpoint
Ctrl-Shift-F8	New breakpoint
Ctrl-Shift-F7	New watch

(e) Atajos de teclado para Debug

Figura 4.8: Atajos de teclas del editor inteligente de Apache NetBeans [175]

Funciones generales del editos de código de Apache NetBeans

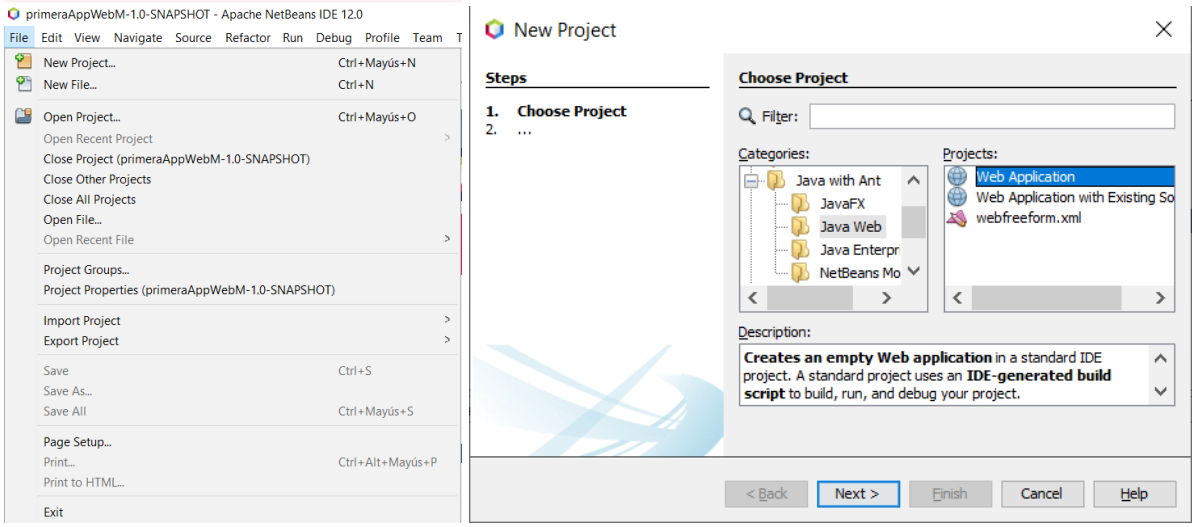
Entre las funciones que cumple el editor de Apache NetBeans IDE, están las siguientes:

- Formato del texto de código, colores de resaltado, plegado de código
- Sugerencia al escribir código
- Insertar y resaltar llaves, corchetes y comillas
- Personalización de atajos de teclado
- Detalle de atajos de teclado para el desarrollo en Java

4.2.2. Opciones de desarrollo

Aunque Apache Netbeans se ha centrado para el desarrollo de aplicaciones web con Maven[1]. Netbeans tiene muchas opciones de desarrollo. En este libro se especificará lo que concierne al desarrollo de aplicaciones web en Java.

Entre los diferentes tipos de proyectos, están los proyectos para el desarrollo de aplicaciones web. Para crear un proyecto de este tipo, se debe seleccionar desde el menú **Archivo** la opción **Nuevo proyecto** (ver figura 4.9a), le aparecerá la ventana de la figura 4.9b.



(a) Opciones del Menú [Archivo] **(b)** Ventana para escoger el tipo de proyecto a crear.

Figura 4.9: Creación de nuevo proyecto.

Entre las maneras existentes para crear una nueva aplicación web, se puede mencionar las siguientes:

- Desde la barra de herramientas seleccionando el icono amarillo en forma de carpeta con un signo más (segundo icono de izquierda a derecha) de la figura 4.10.
- Con el atajo de teclado, presionado **[Cntrl + Shift + N]**

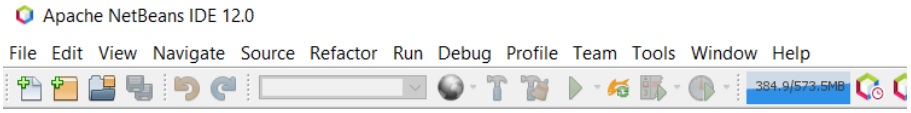


Figura 4.10: Barra de tareas de Netbeans.

Sin importar la forma de crear el nuevo proyecto, las ventanas que se le mostrarán al usuario son las mismas que se muestran en la figura 4.9.

[1]Maven es un gestor de bibliotecas, dejando para el desarrollador que copie y pegue las dependencias de las bibliotecas de clase a usar en el archivo pom.xml

En la ventana de la figura 4.9b se escoge la categoría y el tipo de proyecto que se va a crear. En este caso, se escogerá categoría Java Web y de tipo Aplicación Web. Al dar clic en siguiente se muestra la ventana de la figura 4.11. Netbeans crea una carpeta para almacenar los archivos de aplicación web. Por defecto crea algunos archivos para que la aplicación web creada sea funcional. Entre ellos, la carpeta para las páginas web entre ellas una index cuya extensión depende del framework que se vaya a usar (htm, xhtml). Además, la carpeta para el código fuente, los archivos de configuración (como web.xml), entre otros.

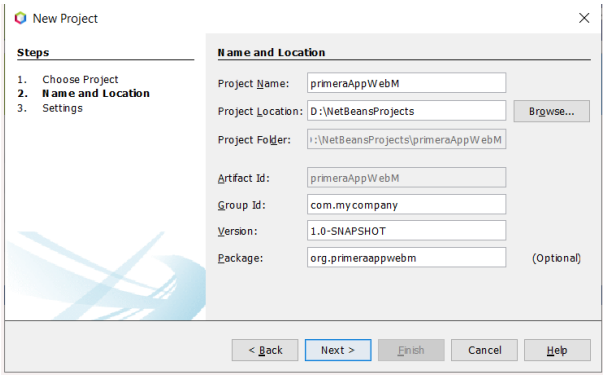


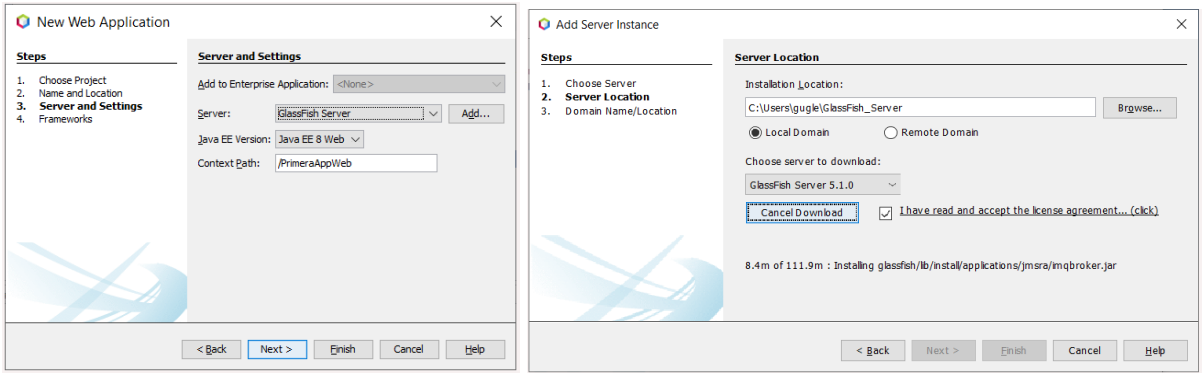
Figura 4.11: Especificación nombre y ubicación de almacenamiento del proyecto

Existen otros datos que se generan al crear el proyecto y que se pueden modificar, entre ellos: Artefacto (*Artifact Id*), que sirve como identificador único del proyecto, que en su momento podrá ser una dependencia que se puede incluir en un proyecto. Cada artefacto puede tener dependencias entre sí, por lo tanto, si se incluye un artefacto en un proyecto, Maven proporciona sus dependencias. Otro de los datos es *Group Id*. Este identificadores para el conjunto de artefactos. Este identificador se recomienda que se escriba un nombre de dominio inverso, por ejemplo si se tiene un dominio midominio.org el nombre sería org.midominio. Por último está la versión (*Version*). Versión especifica la versión del artefacto generado por el proyecto. Un artefacto puede tener varias versiones. Maven agrega SNAPSHOT como una versión que sirve para identificar a un proyecto cuando se encuentra aún en la fase de desarrollo.

Si el desarrollador tiene una carpeta para el proyecto creado, ésta debe estar completamente vacía, caso contrario Netbeans no le permitirá crear el proyecto. Una vez ya especificado los datos del de almacenamiento del proyecto, se da clic en siguiente. En este instante está listo para especificar el servidor con el que va a trabajar (ver figura 4.12)La ventana que debe mostrar es la de la figura 4.12a en la que el desarrollador selecciona el servidor con el que desea trabajar y la versión de JEE [2]. Si no tiene ningún servidor local instalado, puede ser descargado e instalado en este momento, como se muestra en la figura 4.12b.

Como antes se especificó existen muchos servidores de aplicaciones para Java. La mayoría de ellos se descargan desde sus sitios oficiales y se descomprimen en la

[2] *Java Enterprise Edition*



(a) Selección del entorno **(b)** Descargando e instalando un nuevo servidor

Figura 4.12: Creación de nuevo proyecto.

máquina del usuario, con eso es suficiente para ser usado. Para la redacción de este trabajo se usará GassFish por ser uno de los servidores gratuitos y con muy buenas prestaciones, suficientes para los objetivos del resto del texto.

Una vez especificado el servidor, podría finalizar para un proyecto básico. Sin embargo, al dar clic en siguiente le permite seleccionar el framework con el que desee trabajar. Entre los framework que viene con la instalación de Netbeans está JavaServer Faces con sus componentes PrimeFaces, IceFases y ReachFaces; Spring Web MVC y Struts (ver figura 4.13). Según la versión de Netbeans pueden variar en el tipo y las versiones de ellos.

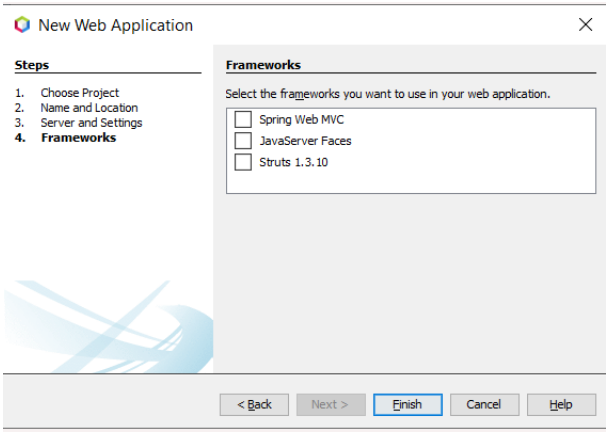


Figura 4.13: Especificación del o de los frameworks a usar

Una vez creado el proyecto para la aplicación web, es cuestión de agregar la lógica de negocios, los controladores y las vistas o interfazs necesarias, además de los elementos que el desarrollador necesite para dar solución al problema que se haya planteado. Para ello lo puede hacer desde el menú **Archivo**, luego seleccionando **Archivo nuevo** o desde la barra de herramientas, dando clic en el primer ícono de la barra de herramientas (ver figura 4.10) ícono blanco con el signo más, o directamente haciendo uso del teclado,

presionando la combinación de teclas **[Cntrl + N]**.

Para una aplicación web hay muchos tipos de elementos que se pueden agregar. A continuación se listan los elementos usados con más frecuencia.

- Clase java
- Paquete Java
- Clases Entidad
- Clases Entidad desde una Base de Datos
- Servicios Web
- Cliente Servicios Web

En la figura 4.14 se muestra el listado de elementos más comunes y los que se han usado recientemente en el editor de Netbeans. Esta lista de elementos es accedida desde el menú contextual del proyecto (clic secundario en el nombre del proyecto) luego escoge la opción **Nuevo**, luego puede seleccionar cualquier elemento de la lista. Si el elemento que desea agregar no se muestra en ese listado, escoge la última opción **Otro**.

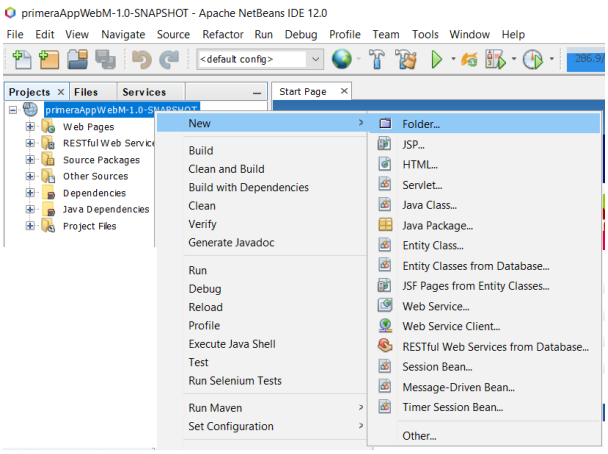


Figura 4.14: Tipos de elementos a agregar a un proyecto de Aplicación Web

La opción de **Otro** le muestra por categorías los elementos que se pueden agregar al proyecto de la aplicación web. En la figura 4.15 se muestran las categorías (la lista del lado izquierdo) y los elementos de la categoría que se seleccione (la lista del lado derecho)

En la figura 4.15 se encuentra seleccionada la categoría Web, por lo tanto los elementos que se muestran son los de esa categoría, y son los elementos generales de las aplicaciones web. Los elementos que se utilizan en el desarrollo de aplicaciones web así como en otro tipo de proyectos se encuentran agrupados en categorías.

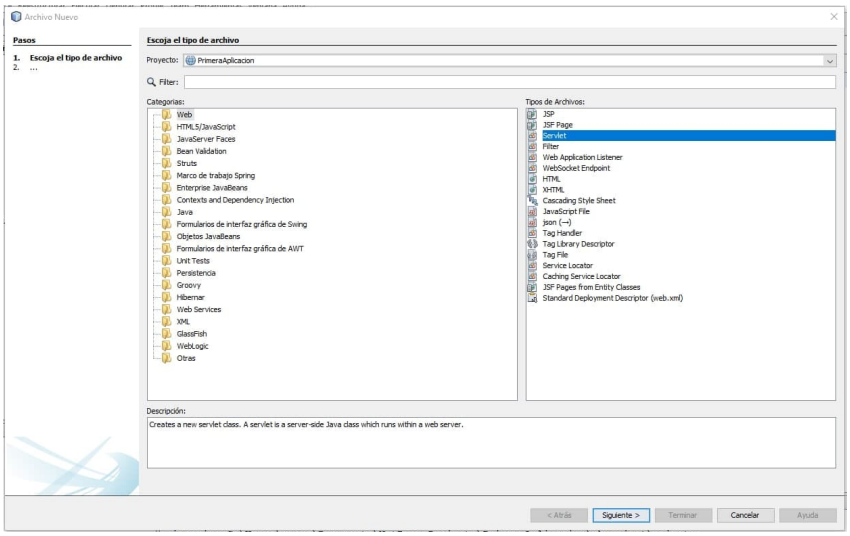
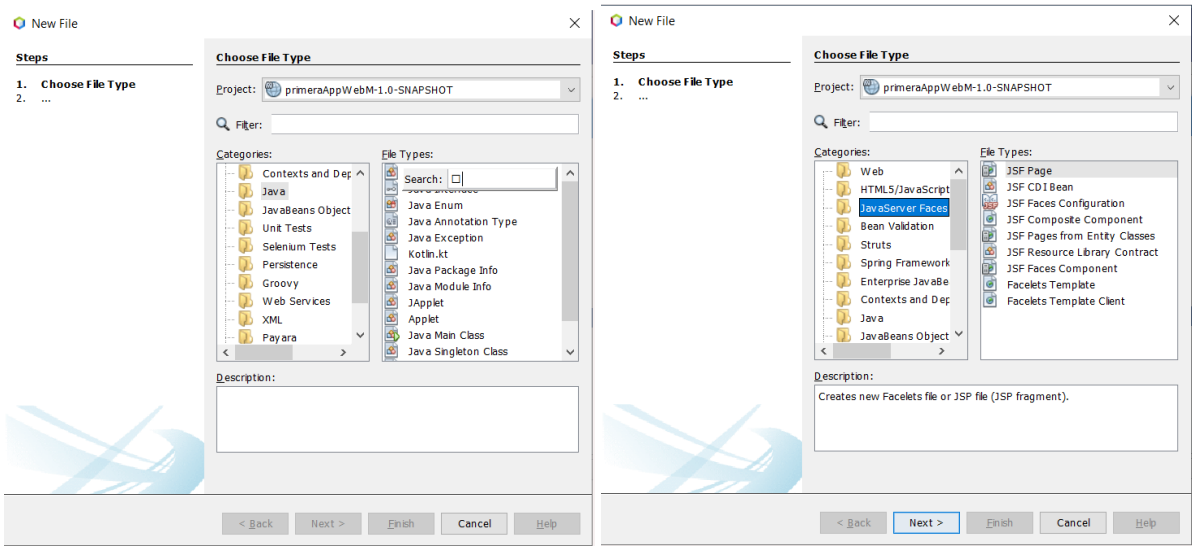


Figura 4.15: Categorías de los elementos a agregar, elementos de la categoría Web

Siguiendo esta ruta, lo que se debe hacer para continuar con el desarrollo es agregar nuevos elementos al proyecto. En la figura 4.16 muestra las opciones. En la figura 4.16a se muestran los elementos de la categoría Java. En esta categoría muestra los elementos Java que se pueden utilizar sin importar el tipo de proyecto que se esté desarrollando. Otra de las categorías de elementos más usados según la experiencia del autor es la categoría **JavaServer Faces** que se muestran en la figura 4.16b.

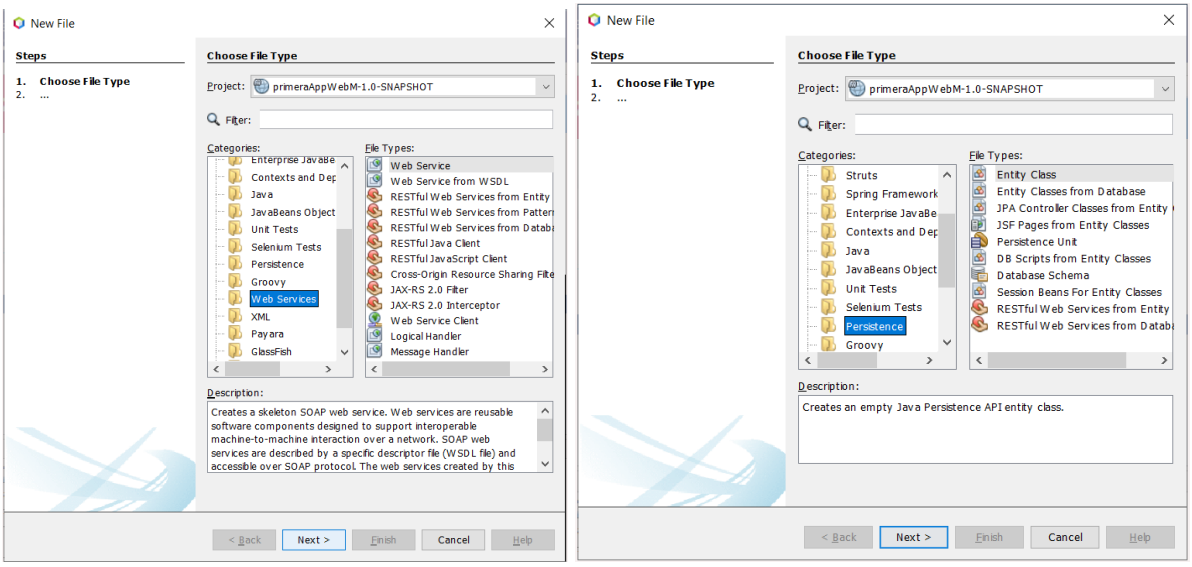


(a) Elementos de la categoría Java **(b)** Elementos de la categoría JavaServer Faces

Figura 4.16: Creación de nuevo proyecto.

Cabe recalcar que se debe considerar las novedades de cada versión de los IDE. Por ejemplo, desde NetBeans 7.0 el archivo de configuración de JavaServer Faces no es

creado al momento que se agrega este framework a un proyecto web, por lo tanto se debe agregar manualmente, y suele ser el primero. En este archivo se suelen anotar las reglas de navegación que son importantes para enlazar un archivo con otro usando controles JSF como <h:commandButton>, <h:commandLink>, así mismo para responder a las acciones del usuario, como el evento click sobre un botón (propiedad `action`). Los servicios web y la persistencia de datos serán motivo de estudio en capítulos posteriores. En la figura 4.17 se muestra como agregar esas categorías de elementos. Para la creación de servicios web, como también para la creación de los clientes de los servicios web (clientes = consumidores) se encuentran agrupados en la categoría **Servicios Web** como se muestra en la figura 4.17a, y en la figura 4.17b se muestran los elementos para la gestión de la persistencia de datos.



(a) Elementos de la categoría Servicios Web **(b)** Elementos de la categoría Persistencia

Figura 4.17: Creación de nuevo proyecto.

Pruebas unitarias

En el ambiente de desarrollo se están utilizando las pruebas o **Test**, que ayudan a producir código de software más limpio y libre de errores. Además, las pruebas ayudan a los desarrolladores a obtener el código de programación que cumple con los requisitos del cliente, eliminando la pérdida de tiempo al tratar de obtener lo que supuestamente el cliente desea [176].

Con las pruebas unitarias todos los interesados en el proyecto resultan beneficiados. El trabajo del desarrollador será mucho más fácil, obteniendo un código de mayor calidad. Además, disminuirá el tiempo dedicado a la depuración y la corrección de incidencias. Con estos resultados, el cliente estará más contento porque la aplicación hace lo que él ha especificado (lo esperado) que haga, **por lo que ha pagado**.

Las pruebas fomentan las adecuaciones y la refactorización. Si el cambio no estuviera realizado correctamente las pruebas nos avisarán de ello. La frase **"funcionó, no lo toques más"** les resultará familiar a muchos desarrolladores. Al aplicar las pruebas unitarias, esa frase no existiría, o al menos, no se necesitaría pronunciar. Si se considera que el código es mejorable, se puede refactorar sin ningún problema.

Además, las pruebas reducen considerablemente los problemas de la integración y los tiempo dedicados a ella. En las pruebas se simulan las dependencias, lo que permite que se pueda probar el código sin disponer del resto de módulos. Sin las pruebas, los problemas de la integración serían más de una vez traumáticos, dejándolos habitualmente para el final del proyecto. Aquí cabe mencionar otra frase que les resultará igual o más común que la de "funcionó", y es la frase: **"sólo queda integrar"**, que hace referencia a que el proyecto está cerca de terminar, lo que suele ser engañoso.

Las opciones que el editor de NetBeans tiene para ayudar con este tema se muestran en la figura 4.18.

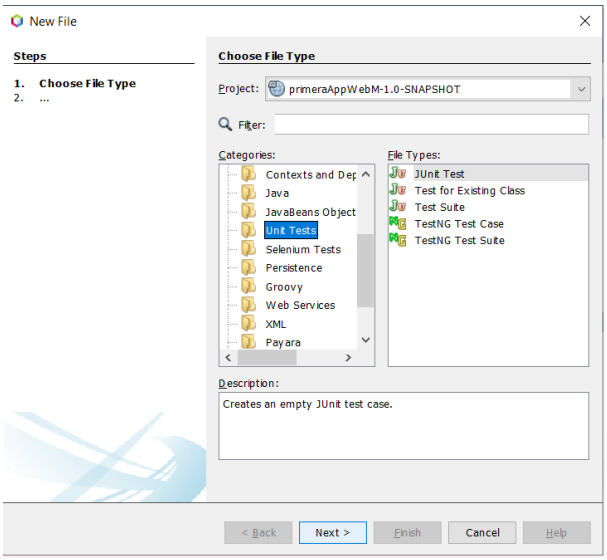


Figura 4.18: Elementos de la categoría Unidad de Pruebas (Test Unit)

4.3. IDE Eclipse 2020-06

Eclipse es un IDE multi-lenguaje construido alrededor de un espacio de trabajo (*workspace*) al que pueden incorporarse un gran número de plug-ins que proporcionan funcionalidades concretas relacionadas con lenguajes de programación específicos o con la interacción con otras herramientas implicadas en el desarrollo de una aplicación. Está desarrollado en Java, y soporta algunos lenguajes de programación tales como `Java (JSP)`, `ANSI C`, `C++`, `sh`, `Perl`, `PHP`, `sed`. Su aplicación principal en lenguaje Java.

4.3.1. Usando Eclipse para el entorno de desarrollo de JAVA

En la figura 4.19 se muestra la pantalla inicial de Eclipse (después de ser ejecutado). Se muestran los proyectos en los cuales se está trabajando. Con un sólo clic en uno de ellos, eclipse abrirá el proyecto y mostrará su contenido.

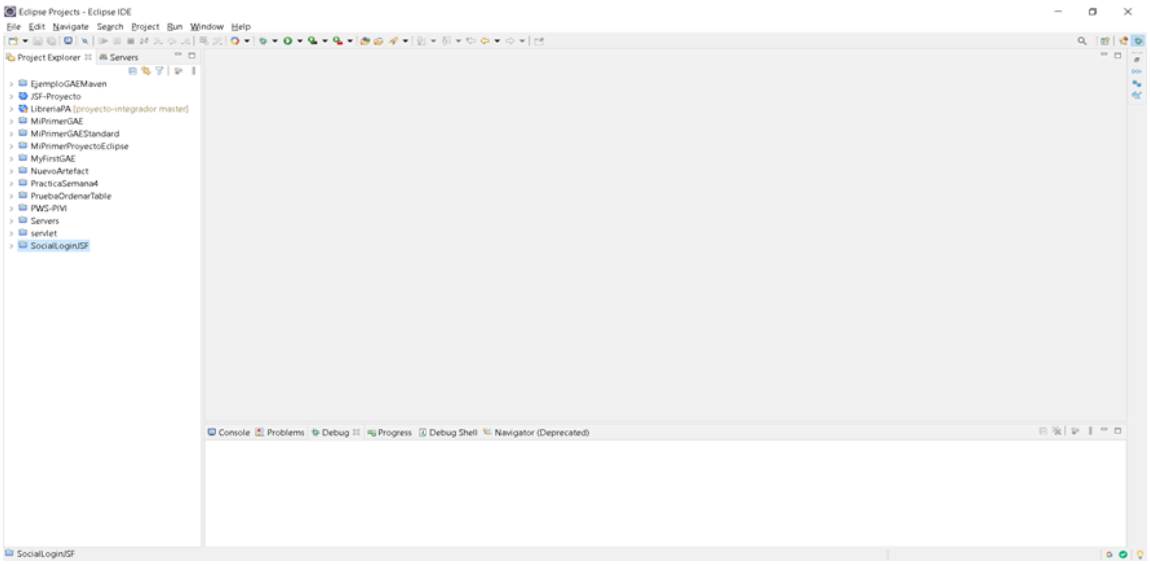


Figura 4.19: Pantalla inicial de Eclipse

En la figura 4.20 muestra como Eclipse organiza el contenido de un proyecto de una aplicación web. Eclipse como otros IDE permite trabajar directamente con plataformas en la nube. En este caso está listo para conectarse con App Engine que es una plataforma de Google. Organiza el contenido como *Java Resources*, *Deployed Resources*, *src* (fuentes), además de *target* y los archivos de configuración según el proyecto al que pertenece la aplicación web.

Dentro de la carpeta ***Java Resources*** (aunque no es una carpeta física) organiza el contenido como fuentes principales y fuentes de prueba (test), además de las librerías. Todos su contenido se traduce a clases y paquetes de clases. En la carpeta ***src*** de fuentes encontramos las páginas web (HTML, JSP, XHTML, entre otras). Además de las clases java que se han creado en el proyecto, y están listas para ser editadas o adecuadas para que cumplan con su objetivo. En la carpeta ***Deployed Resource*** como su nombre lo expresa, se guardarán los elementos necesarios para desplegar la aplicación sin ningún problema.

En la actualidad se recomienda trabajar con proyectos MAVEN para liberar al desarrollador de la descarga de las librerías, traduciéndose este trabajo a copiar y pegar elementos XML que corresponden a las dependencias de las tecnologías o herramientas a usar en el desarrollo del proyecto. Este tipo de proyecto trabajan con una archivo con formato XML denominado pom (pom.xml).

En la figura 4.21 se encuentra la consola, lista de problemas en el proyecto, debug, entre otras pestañas que tienen que ver con la ejecución y depuración de los programas.

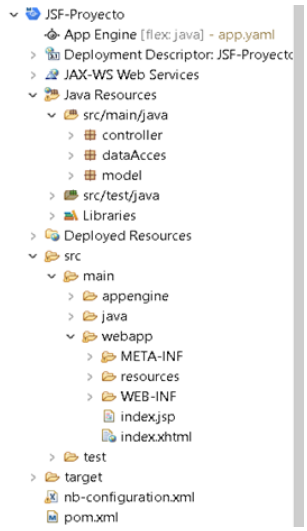


Figura 4.20: Organización del contenido de un Proyecto en Eclipse

Esta ventana con todas estás pestañas se encuentran por lo general en la parte inferior de la ventana de Eclipse.

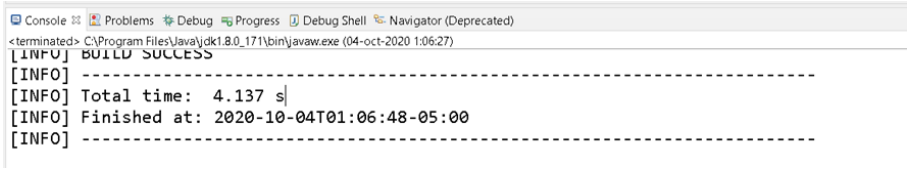


Figura 4.21: Ventanas para el seguimiento del proyecto en depuración

La figura 4.22 muestra los paneles con las herramientas necesarias para el diseño y programación de las aplicaciones agrupadas en paletas de componentes. Además, en el caso de que se esté depurando se podrá ver los valores de los objetos en la sección Variables, los puntos de quiebre y expresiones. Esta sección se muestra por defecto en la parte derecha de la ventana de Eclipse.

4.3.2. Tipos de proyectos disponibles en Eclipse IDE con Java

Al decidir utilizar Java como herramienta de desarrollo y Eclipse como IDE, podemos crear varios tipos de proyectos, como se muestra en la figura 4.23. Como ejemplos se creará un proyecto **Maven**[3] con JSF (JavaServer Faces).

[3]Maven, una palabra *yiddish* que significa acumulador de conocimiento , comenzó como un intento de simplificar los procesos de construcción en el proyecto *Jakarta Turbine*. Había varios proyectos, cada uno con sus propios archivos de compilación de **Ant**, que eran todos ligeramente diferentes. Los JAR se registraron en CVS. Deseando una forma estándar de construir los proyectos, una definición clara de en qué consistía el proyecto, una forma fácil de publicar información del proyecto y una forma de compartir los archivos JAR en varios proyectos.

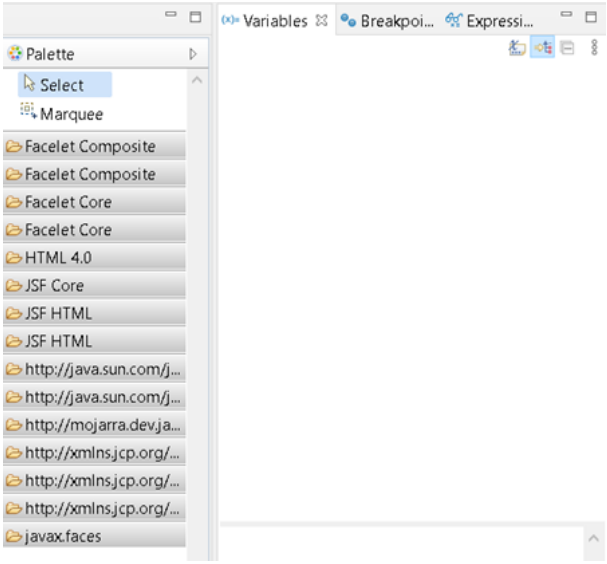


Figura 4.22: Paneles con las herramientas para modo de diseño

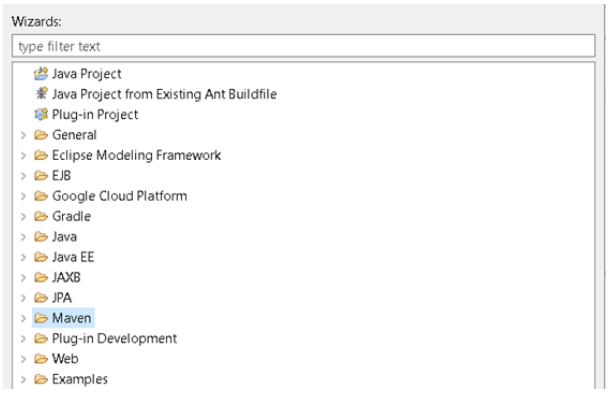


Figura 4.23: Lista de tipos de proyectos para trabajar en Eclipse con Java

4.3.3. Crear proyecto JAVA Web usando Maven con el Framework JSF

Para crear un proyecto se procede a acceder al menú *File*, escoger la opción *New* luego *Project*... Al seleccionar está opción le aparece la ventana de la figura 4.23. debe expandir la pestaña inteligente del elemento Maven, y le mostrará los tipos de proyectos Maven como se muestra en la figura 4.24.

Al dar clic en el botón *Next* (Siguiente) muestra la ventana de las figura 4.25 En esta ventana se escogerá un arquetipo [4] para trabajar. Artefacto maven-archetype-webapp del grupo org.apache.maven.archetypes.

Eclipse IDE muestra una interfaz gráfica para la gestión de las dependencias como se muestra en la figura 4.26. En el caso del ejemplo propuesto se usarán las dependencias

[4] *Archetype* (Arquetipo) es un conjunto de herramientas de plantillas de proyectos de Maven. Un arquetipo se define como un patrón o modelo original a partir del cual se hacen todas las demás cosas del mismo tipo.

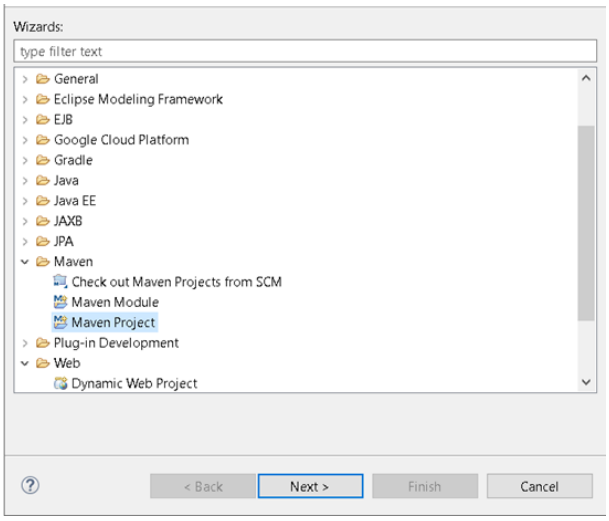


Figura 4.24: Tipos de proyectos Maven

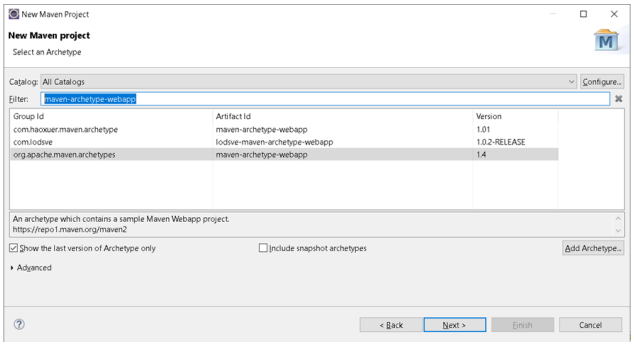


Figura 4.25: Ventana para seleccionar el Arquetipo para el proyecto Maven

de Servlet y JSF[5].

- jsf-api
- jsf-impl
- javax.servlet-api

Una vez terminado el proceso de crear el proyecto, se debe actualizar las dependencias para resolver los aparentes errores que aparecen (observar en la imagen los nombres con x color rojo). Para actualizar las dependencias, suficiente es con el acceso rápido haciendo clic secundario en el **nombre del proyecto**, luego escoger ***Maven*** finalmente ***Update Project***. Respecto al nombre, siempre se hay que recordar que, aunque los IDE modernos soportan con espacios en blanco, se debe seguir con el esquema antiguo, es decir, en minúsculas, sin espacios en blanco y sólo letras del alfabeto inglés, pudiendo adicionar números, especialmente para proyecto de aplicaciones web.

[5]JavaServer Faces (JSF) es una tecnología con un framework para el desarrollo de aplicaciones web con Java que simplifica el desarrollo de interfazs de usuario en aplicaciones Java EE.

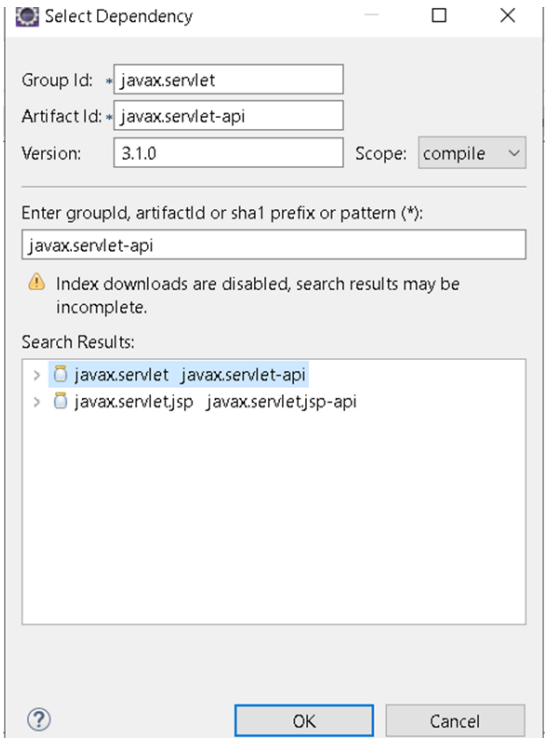


Figura 4.26: Gestión de dependencias en proyectos Maven (archivo pom.xml)

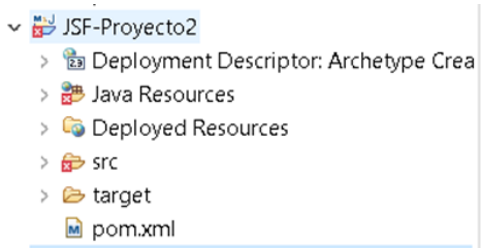


Figura 4.27: Proyecto creado con errores aparentes

Este trabajo en ningún momento tiene la intensión de presentar un manual o tutorial de los IDE de desarrollo, lo que pretende es dar la oportunidad a los desarrolladores contemplar un panorama general del alcance de las aplicaciones que se pueden desarrollar usando Java en Netbeans. Una guía completa de este editor puede encontrar en su sitio web oficial https://netbeans.apache.org/

4.4. Conclusiones

Muchos de los IDE de desarrollo actuales presentan muchas funcionalidades que esconden de los desarrolladores aspectos que son considerados **trabajo sucio** como la configuración de las aplicaciones web y aspectos de programación rutinaria, como por ejemplo clases entidad (*entity class*), uso de web services, e incluso la creación de estos.

Eclipse es el más utilizado en el ámbito de la investigación mientras que Netbeans es el más utilizado en la academía, entre los más populares [177].

Según *Google Trends*, en el Ecuador [178] entre los tres IDE expuestos en este capítulo, existe mayor interés por NetBeans. En la figura 4.28 se muestra la comparación de los IDE más populares, pudiendo claramente darse cuenta que, el mayor interés recae sobre NetBeans.

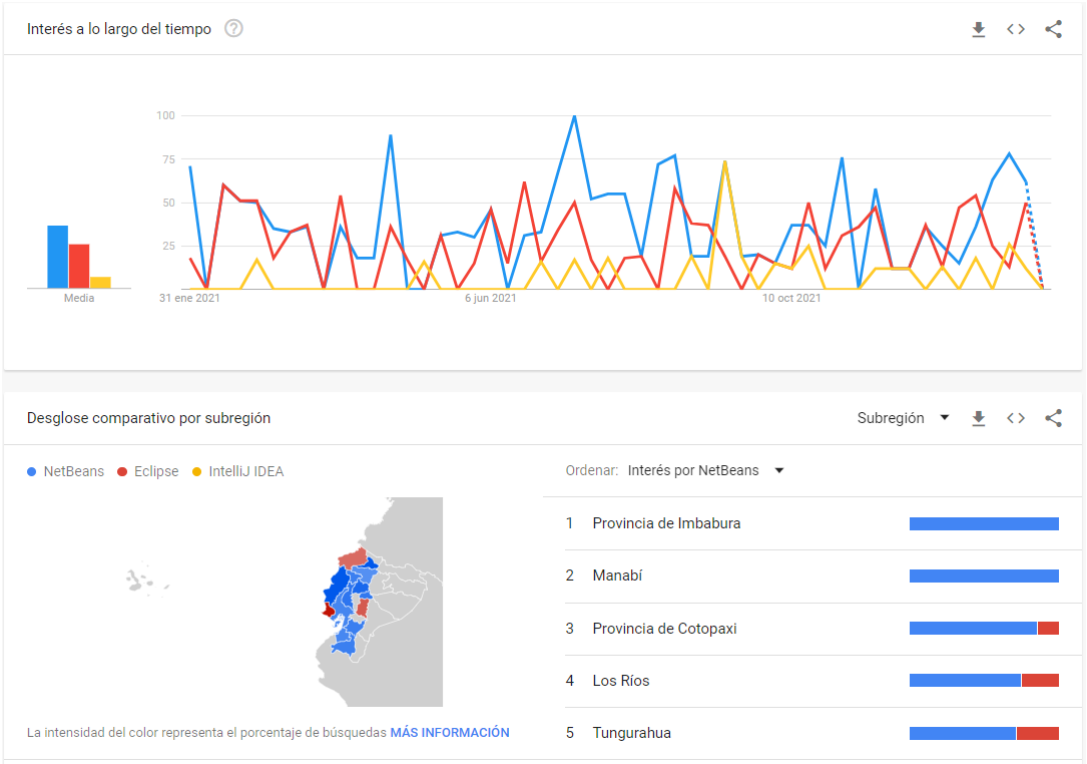


Figura 4.28: Interés en el Ecuador sobre los tres IDE

Evaluación a los lectores

El proceso de enseñanza-aprendizaje siempre debe aterrizar en la adquisición de nuevos conocimientos y el desarrollo de habilidades preexistentes o nuevas. Para tratar de medir estos objetivos, el lector debe responder y resolver las siguientes preguntas:

- Desarrollar una pequeña aplicación usando los tres IDE de desarrollo expuestos en este capítulo. Comparar de manera personal la facilidad, las prestaciones de cada uno de los IDE usados. Para ello, debe desarrollar la misma aplicación en los tres IDE.

- Buscar tres IDE para Java adicionales y desarrollar la misma aplicación anterior, y añadir a la comparativa hecha en la tarea anterior.

- Tomar como objetivo desarrollar algún sistema y bajo criterios bien establecidos (facilidades y prestaciones) escoja el IDE óptimo para su desarrollo.

Capítulo 5

ENTERPRISE JAVABEANS (EJB)

Objetivos

El lector al finalizar este capítulo será capaz de:

- Identificar cada uno de los tipos de EJB conjuntamente con sus ventajas y desventajas.
- Crear aplicaciones con estándares de la plataforma para agilizar el proceso de desarrollo.

Resumen

Hoy en día, muchas aplicaciones implementan lógica de negocios como tercera capa como componentes estándar J2EE (*Java™ 2 Platform, Enterprise Edition*). La tecnología EJB es una propuesta del mundo Java para el desarrollo de aplicación empresarial. Un *Enterprise Java Bean* (EJB) es un componente que debe ejecutarse de un contenedor de EJBs. El contenedor EJB es un programa Java que trabaja en el servidor, con GassFish. El objetivo de la plataforma Java EE es proporcionar a los desarrolladores un potente conjunto de API a la vez que acorta el tiempo de desarrollo, reduce la complejidad de la aplicación y mejora el rendimiento de la aplicación. Entre las ventajas que obtendrán los desarrolladores de aplicaciones y los clientes finales se pueden citar: Simplicidad, portabilidad de la aplicación, reusabilidad de componentes, posibilidad de construcción de aplicaciones complejas, separación de la lógica de presentación de la lógica de negocios, despliegue en muchos entornos operativos, despliegue distribuido, interoperabilidad entre aplicaciones, integración con sistemas no-Java, recursos educativos y herramientas de desarrollo. Todo no es sólo ventajas. Entre las desventajas se puede decir que: El desarrollar un Sistema con EJB's es sumamente complejo. EJB's es uno de los principales componentes de J2EE y por esta razón también depende fuertemente de otras partes de J2EE. Entre los tipos de *beans* de sesión: *Beans* de sesión con estado o Stateful, *Beans* de sesión sin estado o *stateless*, *Beans* de Sesión

Singleton o *Singleton Session Beans*. Con NetBeans como IDE de desarrollo ayuda a los desarrolladores a crear aplicación empresarial de manera fácil y rápida.

5.1. Introducción

Hoy en día, muchas aplicaciones implementan lógica de negocios como tercera capa como componentes estándar J2EE (*Java™ 2 Platform, Enterprise Edition*). Exponer estos componentes como Servicios Web SOAP, los hace accesibles casi universalmente y proporcionan un simple mecanismo para integrar estos componentes [179]. La arquitectura modular J2EE hace que este proceso sea relativamente fácil. Un *Enterprise Java Bean* (EJB) es un componente que debe ejecutarse de un contenedor de EJBs y se diferencia bastante de un JavaBean normal. Un JavaBean es un objeto Java al cual accedemos de forma directa desde nuestro programa [180]. En la figura 5.1 se muestra un esquema de trabajo de los EJB básico.

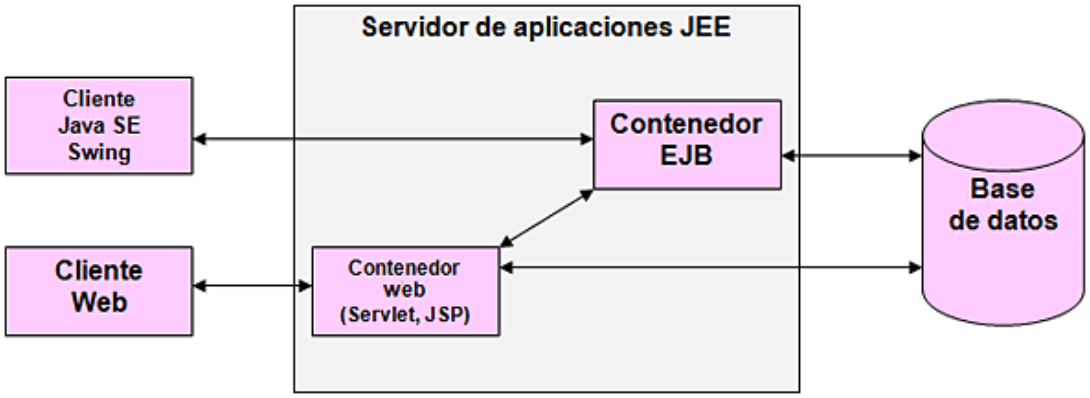


Figura 5.1: Datos del proyecto para la nueva Aplicación Web

La tecnología EJB es una propuesta del mundo Java para el desarrollo de aplicación empresarial. Vinculada con las aplicación empresarial como aplicaciones informáticas de gestión implementadas en grandes empresas con múltiples oficinas. Por lo general, son aplicaciones distribuidas utilizadas por varios clientes y utilizan transaccións y bases de datos, implementando políticas de seguridad en gran medida [179].

La tecnología EJB, proporciona un modelo de componentes distribuidos para que los desarrolladores de aplicaciones no se preocupen en los problemas a nivel de sistema que resultan complejos, y puedan centrarse en los problemas específicos del negocio. Esta partición permite el desarrollo rápido de aplicaciones (RAD), al mismo tiempo que la aplicación resulte escalable, robusta y segura [181].

Los *Enterprise JavaBeans* son componentes del lado del servidor que encapsulan la lógica de negocios de una aplicación. Los *Enterprise JavaBeans* simplifican el desarrollo de aplicaciones gestionando automáticamente la gestión de las transaccións y la seguridad.

Enterprise JavaBeans

La tecnología EJB:

- Es un componente que debe ejecutarse de un contenedor de EJBs.
- Es del mundo Java para el desarrollo de aplicación empresarial.
- Proporciona un modelo de componentes distribuidos para que los desarrolladores de aplicaciones no se preocupen en los problemas a nivel de sistema.
- Los *Enterprise JavaBeans* son componentes del lado del servidor que encapsulan la lógica de negocios de una aplicación.
- Los *Enterprise JavaBeans* simplifican el desarrollo de aplicaciones gestionando automáticamente la gestión de las transaccións y la seguridad.

5.2. Contenedor EJB

El contenedor EJB es un programa Java que trabaja en el servidor y que contiene todas las clases y objetos necesarios para el correcto funcionamiento de los EJBs. En la figura 5.2 se muestra el esquema básico de los EJB y su instaciación en GassFish.

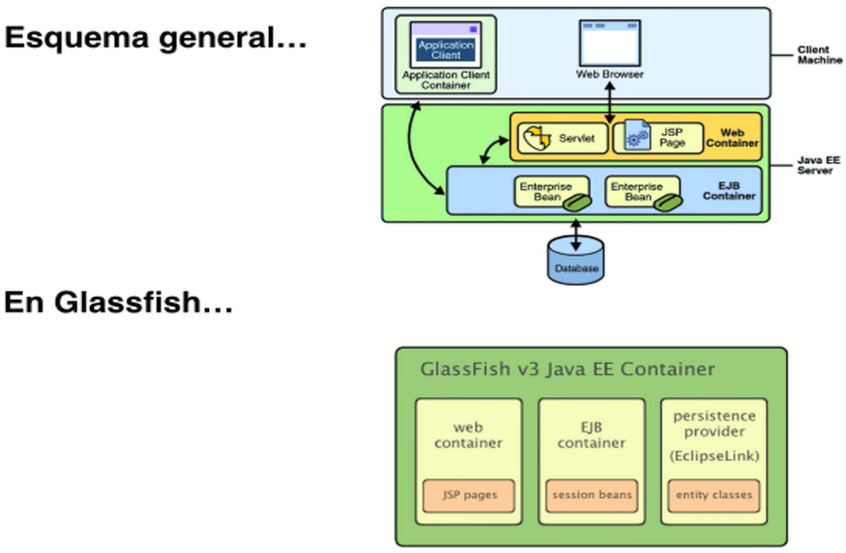


Figura 5.2: *Arquitectira de Enterprise Java Beans*

Los servicios que ofrece un contenedor de EJBs son los siguientes [180, 182]:

- Integración: Proveen una forma de acoplar en tiempo de ejecución diferentes componentes, mediante la simple configuración de anotaciones xml. La integración

es un servicio que proveen los *beans* de sesión y los *beans* administrados por mensajes o MDBs (*Message-Driven Beans*).

- Pooling: El contenedor de EJBs crea para componentes EJB un pool de instancias que es compartido por los diferentes clientes. Aunque cada cliente visualiza el entorno como si recibiera siempre instancias diferentes de los EJB, el contenedor está constantemente reutilizando objetos para optimizar la memoria. El pooling es un servicio que se aplica a los Stateless Session Beans y a los MDBs.

- Seguridad en hilos (*Thread-safely*): El programador puede escribir componentes del lado del servidor como si estuviera trabajando en una aplicación sencilla con un solo thread (hilo). El contenedor se encarga de que los EJBs tengan el soporte adecuado para una aplicación multi-usuario (como son en general las aplicación empresarial) de forma transparente, asegurando el acceso seguro, consistente y alto rendimiento. Se aplican a los *beans* de sesión y a los MDBs.

- Administración de Estados: El contenedor de EJBs almacena y maneja el estado de un Stateful Session Bean de forma transparente, lo que significa que el programador puede mantener el estado de los miembros de una clase como si estuviera desarrollando una aplicación de escritorio ordinaria. El contenedor manipula los detalles de las sesiones.

- Mensajería: Mediante los MDBs es posible desacoplar por completo dos componentes para que se comuniquen de forma asíncrona, sin reparar demasiado en los mecanismos de la API de servicios de mensajería Java (*Java Message Services API*) que los MDBs encapsulan.

- Glspltransation: EJB soporta el manejo de transaccións declarativas que permiten agregar comportamiento transaccional a un componente simplemente usando anotaciones xml de configuración. Esto significa que cuando un método de un EJB (*Session Bean* o MDB) se completa normalmente, el contenedor se encargará de finalizar la transacción y validar los cambios que se realizaron en los datos de forma permanente. Si algo fallara durante la ejecución del método (una excepción o cualquier otro problema), la transacción haría un retroceso (rollback) y es como si el método jamás se hubiera invocado.

- Seguridad: EJB soporta integración con la API *Java Authentication and Authorization Service* (JAAS), haciendo casi transparente el manejo transversal de la seguridad. Se aplica a todos los *Session Beans*.

- Interceptores: EJB introduce un framework liviano y simple para la programación orientada a aspectos o AOP por sus siglas en inglés (Aspect Oriented Programming). No es tan robusto y completo como otros, pero es lo suficientemente útil para que sea utilizado por los demás servicios del contenedor para brindar

de forma invisible las preocupaciones transversales (*crosscutting concerns*) de seguridad, transaccións, seguridad en hilos.

- Acceso Remoto: Es posible acceder de forma remota a distintos EJBs de forma sencilla, simplemente mediante la Inyección de Dependencia. El procedimiento para inyectar un componente local o uno remoto es exactamente el mismo, abstrayéndonos de las complicaciones específicas de la invocación de métodos remotos o RMI (*Remote Method Invocation*) o similares. Este servicio aplica únicamente a los *Session Beans*.
- *Web Services*: Un *Stateless Session Bean* puede publicar sus métodos como servicios web mediante una sencilla anotación.
- Persistencia: EJB provee la especificación de la API para la persistencia en Java o JPA (Java Persistence API) para el mapeo de objetos llamados Entidades a tablas u objetos de Java simple (POJO: *Plain Old Java Object POJOs*).

Contenedor EJB

Programa Java que trabaja en el servidor y que contiene todas las clases y objetos para el correcto funcionamiento de los EJBs. Entre sus servicios están:

- Integración: Acopla en tiempo de ejecución diferentes componentes.
- Pooling: Crea un pool de instancias para compartirlo con los clientes.
- Seguridad en hilos (*Thread-safely*): Escritura de componentes de manera sencilla.
- Gestión de Estados: Gestiona el estado de un Bean de sesión de manera transparente.
- Mensajería: Desacopla componentes para comunicación asincrónica.
- Transaccións: Finaliza la transacción y valida los cambios en los datos de forma permanente.
- Seguridad: EJB soporta integración con la API JAAS.
- Interceptores: EJB introduce un framework liviano y simple para la programación orientada a aspectos.
- Acceso Remoto: Accede de forma remota a distintos EJBs de forma sencilla.
- *Web Services*: Los beans de sesión sin estado pueden publicar sus métodos como servicios web mediante una sencilla anotación.
- Persistencia: EJB provee la especificación de API para persistencia (JPA).

5.3. Características de los EJBs

Los EJBs son componentes desarrollados en lenguaje Java que implementan la tecnología de los EJB. Los *enterprise beans* se ejecutan en el contenedor EJB. Éste es un entorno de ejecución dentro del servidor como GassFish. El contenedor EJB proporciona servicios a nivel de sistema, como transaccións y seguridad a los *enterprise beans* de manera transparente para sus desarrolladores. Estos servicios le permiten crear e implementar de manera rápidamente EJBs, que forman el núcleo de las aplicaciones transaccionals Java EE[1] [183–185].

Los *EJB* simplifican el desarrollo de grandes aplicaciones distribuidas por varias razones, entre ellas [185, 186]:

- Al ser contenedor EJB quien proporciona servicios a nivel de sistema para los *Enterprise Beans*, los desarrolladores de estos componentes solo deben enfocarse en resolver problemas de la lógica de negocios. Por su parte, el contenedor EJB es responsable de los servicios a nivel de sistema como el procesamiento de transaccións y la seguridad, siendo esto transparente para los desarrolladores.
- Los EJBs (en lugar del cliente) contiene la lógica de negocios de la aplicación, por lo tanto, el desarrollador del cliente puede centrarse únicamente en la representación de la información para el cliente.
- Dado que los EJB son componentes portátiles, la persona responsable del ensamblaje de componentes puede crear nuevas aplicaciones a partir de componentes existentes, logrando de esta forma la reutilización del software.

Característocas de los EJB y sus Contenedores

Los *EJB* simplifican el desarrollo de aplicaciones distribuidas por varias razones:

- Los desarrolladores de componentes EJB implementan la lógica de negocios
- Los contenedores EJB proporcionan servicios a nivel de sistema.
- Los contenedores EJB hacen transparente para el desarrollador: Procesamiento de transaccións, seguridads entre otras.
- Los desarrolladores sólo pueden centrarse en representar la información.
- EJB son componentes portátiles.
- Los EJB facilitan la reutilización de código.

[1] *Java Platform, Enterprise Edition* (Java EE) es el estándar en software empresarial impulsado por la comunidad. Java EE se desarrolla utilizando *Java Community Process*, con contribuciones de expertos de la industria, organizaciones comerciales y de código abierto, grupos de usuarios de Java e innumerables personas.

Un resumen gráfico de las características de los EJB y sus Contenedores se puede observar en la en la figura 5.3.

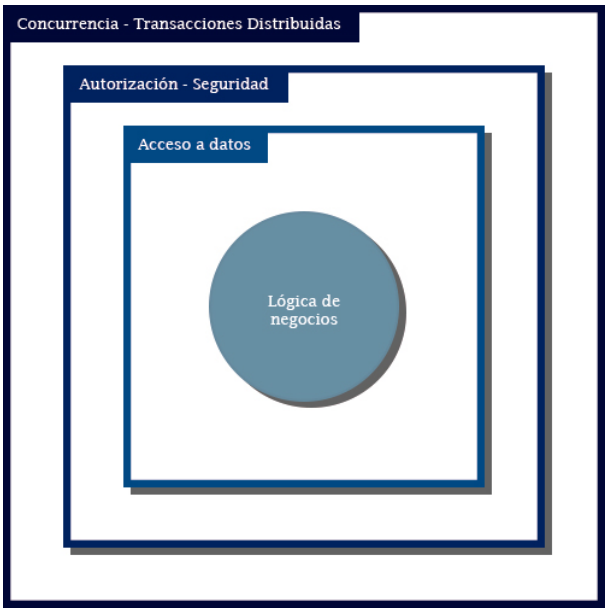


Figura 5.3: *Enterprise Java Beans*

5.3.1. Uso de *Enterprise Beans*

EL uso de los *Enterprise Beans* como cualquier otra tecnología tiene su nicho de problemas en los que se debe aplicar. Para que se aprovechen los beneficios ofrecidos por esta tecnología, el problema a resolver debe cumplir ciertos requisitos. En este caso, para considerar implementar *enterprise beans* en la solución del problema, este debe cumplir con los siguientes requisitos.

- **Las solicitudes de requisitos va en aumento**. Para adaptarse a un número creciente de usuarios, se espera que la aplicación que de solución al problema sea escalable. Es posible que se deba distribuir los componentes de una aplicación en varias computadoras. Los *enterprise beans* de una aplicación no sólo pueden ejecutarse en diferentes computadoras, sino que también pueden estar distribuidos en entre múltiples computadoras. Estos aspectos son transparentes[2] para los usuarios.

- **Susceptible a la pérdida de datos**. La solución debe garantizar la integridad de los datos en las transaccións. Los *enterprise beans* admiten transaccións, por lo tanto, proporcionan los mecanismos que administran el acceso concurrente de objetos compartidos.

[2]Para el usuario, el sistema se encuentra instalado en una sola computadora.

- **Heterogeneidad de clientes a atender.** Cuando uno de los requisitos de los usuarios es atender la petición de multitud de clientes y estos a su vez son heterogéneos, entonces se está seguro que es el tipo de problema correcto. Con sólo unas pocas líneas de código, los clientes remotos pueden localizar fácilmente los *enterprise beans*. Estos clientes pueden ser delgados[3], diversos y numerosos.

Uso de los EJB

No todos los requisitos de un sistema se deben implementar con EJB. Para considerar implementar una solución con EJB, esta debe cumplir con los siguientes requisitos:

- Las solicitudes de requisitos va en aumento
- Susceptible a la pérdida de datos
- Heterogeneidad de clientes a atender

5.4. Ventajas de la tecnología EJB

La arquitectura EJB aporta beneficios a todos los roles (desarrollador, ensamblador de aplicaciones, administrador, implementador, fabricante del servidor). Las ventajas que obtendrán los desarrolladores de aplicaciones y los clientes finales son las siguientes [14]:

- **Simplicidad.** Debido a que el contenedor de aplicaciones libera al programador de realizar las tareas del nivel del sistema la escritura de un *enterprise bean* es casi tan sencilla como la escritura de una clase Java. El desarrollador no tiene que preocuparse de temas de nivel de sistema como la seguridad, transaccións, multi-threading o la programación distribuida. Por lo tanto, el desarrollador de aplicaciones se concentra en la lógica de negocios y en el dominio específico de la aplicación.
- **Portabilidad de la aplicación.** Una aplicación EJB puede ser desplegada en cualquier servidor de aplicaciones que soporte J2EE.
- Reusabilidad de componentes. Una aplicación EJB está formada por componentes *enterprise beans*. Cada *enterprise bean* es un bloque de construcción reusable. Hay dos formas esenciales de reusar un *enterprise bean* a nivel de desarrollo y a nivel de aplicación cliente. Un *JavaBean* desarrollado puede desplegarse en distintas aplicaciones, adaptando sus características a las necesidades de las mismas. También un *bean* desplegado puede ser usado por múltiples aplicaciones cliente.

[3]Es el cliente en una arquitectura Cliente/servidor, donde la mayor carga de procesamiento es asignada al servidor, quedando el cliente (delgado) únicamente como interfaz con la cual el usuario interactúa con el sistema.

- **Posibilidad de construcción de aplicaciones complejas.** La arquitectura EJB simplifica la construcción de aplicaciones complejas. Al estar basada en componentes y en un conjunto claro y bien establecido de interfazs, se facilita el desarrollo en equipo de la aplicación.

- **Separación de la lógica de presentación de la lógica de negocios.** Un *enterprise bean* encapsula típicamente un proceso o una entidad de negocio (un objeto que representa datos del negocio), haciéndolo independiente de la lógica de presentación. El programador de la lógica de negocios no necesita preocuparse de cómo formatear la salida; quien se preocupe será el programador que desarrolle la interfaz usando los datos de salida que proporcionará el *bean*. Esta separación hace posible desarrollar distintas lógicas de presentación para la misma lógica de negocios o cambiar la lógica de presentación sin modificar el código que implementa el proceso de negocio. En otras palabras, facilita las tareas de mantenimiento[4] de la aplicación web.

- **Despliegue en muchos entornos operativos.** Se entiende por entornos operativos el conjunto de aplicaciones y sistemas (bases de datos, sistemas operativos, aplicaciones ya en marcha, etc.) que están instalados en una empresa. Al detallarse claramente todas las posibilidades de despliegue de las aplicaciones, se facilita el desarrollo de herramientas que asistan y automatizan este proceso.

- **Despliegue distribuido.** La arquitectura EJB hace posible que las aplicaciones se desplieguen de forma distribuida entre distintos servidors de una red. El desarrollador de *beans* no necesita considerar la topología del despliegue. Escribe el mismo código independientemente de si el *bean* se va a desplegar en una máquina o en otra (cuidado: con la especificación 2.0 esto se modifica ligeramente, al introducirse la posibilidad de las interfazs locales).

- **Interoperabilidad[5] entre aplicaciones**. La arquitectura EJB hace más fácil la integración de múltiples aplicaciones de diferentes vendedores. La interfaz del *enterprise bean* con el cliente sirve como un punto bien definido de integración entre aplicaciones.

- **Integración[6] con sistemas no-Java.** Las APIs relacionadas, como las especificaciones *Connection* y *Java Message Service* (JMS), así como los *beans* manejados por mensajes, hacen posible la integración de los *enterprise beans* con sistemas no

[4]Mantenimiento es el conjunto de operaciones que se deben realizar para que el software (y hardware en algunos casos) del sistema (en este caso Aplicación Web) pueda cumplir con los nuevos requerimientos del cliente.

[5]El Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) define a la interoperabilidad como: la capacidad de dos o más sistemas o componentes para intercambiar información y utilizarla.

[6]La integración de sistemas es la practica de conectar y unificar diferentes partes o subsistemas, con el fin de obtener un sistema unificado.

Java, como sistemas de planificación de recursos empresariales o ERPs (*Enterprise Resource Planning*) o aplicaciones *mainframes*[7].

- **Recursos educativos y herramientas de desarrollo.** El hecho de que la especificación EJB sea un estándar hace que exista una creciente oferta de herramientas y formación que facilita el trabajo del desarrollador de aplicaciones EJB.

Ventajas y desventajas de los EJB

Aunque las ventajas de los EJB son numerosas, también tiene desventajas. Las ventajas se pueden resumir:

- Simplicidad
- Portabilidad de la aplicación
- Reutilización de componentes
- Posibilidad de construcción de aplicaciones complejas
- Separación de la lógica de presentación de la lógica de negocios
- Despliegue en muchos entornos operativos
- Despliegue distribuido.
- Interoperabilidad entre aplicaciones
- Integración con sistemas no-Java
- Recursos educativos y herramientas de desarrollo

5.5. Desventajas de la tecnología EJB

Así mismo como la tecnología EJB ofrece diferentes beneficios, también es necesario dar a conocer sus desventajas, las cuales son [187]:

- **Tiempo de Desarrollo:** El desarrollar un Sistema con EJB's es sumamente complejo, aunque para ciertas empresas puede presentar una solución ideal, debido a la complejidad-tiempo (costo) para muchas corporaciones los EJBs resultan una solución sobrada, denominada en Ingles: *"overkill"*.

[7] Un Mainframe es un computador de grandes prestaciones para la captura, almacenamiento, procesamiento y distribución de grandes volúmenes de datos. Usados principalmente en ámbitos como: aplicaciones para la banca, gobierno y mercado de valores, aerolíneas y tráfico aéreo, así como para aplicación empresarial con requisitos de robustez elevados

- **Conocimiento exhausto de Java:** EJB's es uno de los principales componentes de J2EE y por esta razón también depende fuertemente de otras partes de J2EE: Como RMI, JNDI[8] y JDBC[9].

Ventajas y desventajas de los EJB

Sus desventajas se pueden resumir:

- Tiempo de Desarrollo
- Conocimiento exhausto de Java

5.5.1. Tipos de *Enterprise Beans*

La tabla 5.1 se resume los dos tipos de *enterprise beans*. Más adelante se exponen con más detalle algunas de sus cuestiones importantes.

Tabla 5.1: Tipos de *Enterprise Beans*

Tipo de *Enterprise Java Beans*	**Propósito**
Session	Realiza una tarea para un cliente; opcionalmente, puede implementar un servicio web.
Message-Driven	Actúa como oyente para un tipo de mensajería particular, como la API del servicio de mensajes de Java.

5.6. *Session Beans*

Un ***bean* de sesión** o ***session Bean*** encapsula la lógica de negocios que es implementada para que los clientes invoquen mediante programación a través de vistas de cliente de servicios web, locales o remotos. Para acceder a una aplicación que está desplegada en un servidor, el cliente invoca los métodos del *bean* de sesión. El *bean* de sesión realiza ciertas tareas para su cliente, entre ellas lo protege de la complejidad mediante la ejecución de tareas de negocios en el servidor [188].

[8] *Java Naming and Directory Interface* o Interfaz de Nombrado y Directorio Java es una Interfaz de Programación de Aplicaciones de Java para servicios de directorio. Permite a localizar objetos y datos a través de un nombre.

[9] *Java Database Connectivity* o Conectividad a bases de datos de Java es una API que permite ejecutar operaciones sobre una base de datos desde aplicaciones Java, sin importar el motor de la base de datos

Los *beans* de sesión no son persistentes, es decir, no son recordados en el tiempo (sus datos no se guardan en una base de datos ni en un sistema de archivos).

5.6.1. Tipos de *beans* de sesión

Los *beans* de sesión son de tres tipos: (1) con estado o stateful, (2) sin estado stateless y (3) singleton [183, 188].

Beans de sesión con estado o Stateful

El estado de un objeto esta dado por los valores de atributos o datos de la instancia (objeto en memoria). En un *bean* de sesión con estado, los atributos de la instancia representan el estado de un cliente/*bean* de sesión único. Debido a la relación entre el cliente y su *bean* de sesión (interactúan o "hablan"), por lo general al estado se denomina estado conversacional [183, 189].

Beans de sesión con estado

Los atributos de la instancia representan el estado de un cliente/*bean* de sesión único. Su estado se conserva durante el tiempo que está activo el cliente/*bean* de sesión.

Como se muestra en la figura 5.4, en alusión a su nombre, un *bean* de sesión equivale a una sesión interactiva. Los *bean* de sesión no se comparte; tiene un solo cliente, de la misma manera que una sesión interactiva tiene sólo un usuario. Cuando el cliente termina la sesión, su *bean* de sesión termina con él, dejando a un lado su relación con el cliente [189].

Su estado se conserva durante el tiempo que está activo el cliente/*bean* de sesión. Si el cliente decide eliminar el *bean*, la sesión finaliza y con ella su estado desaparece. Sin embargo, no hay necesidad de retener el estado cuando finaliza la conversación entre el cliente y el *bean*.

Beans de sesión sin estado o *Stateless*

Su nombre indica que los *beans* de sesión no guardan su estado. Es decir, los valores de los atributos existen sólo en el tiempo específico de invocación de parte del cliente. Una vez que el método termina su ejecución, no se debe conservar el estado específico del cliente. Sin embargo, los clientes son capaces de cambiar el estado de los atributos de la instancia de los *beans* sin estado agrupados, preservando su estado hasta que sea invocado nuevamente este *bean* de sesión sin estado agrupado. Mientras los métodos de los *beans* de sesión sin estado no sean invocados, todas las instancias de un *bean* sin estado son equivalentes, lo que permite que el contenedor EJB asigne una instancia a cualquier cliente. Es decir, el estado de un *bean* de sesión sin estado es para todos los clientes [189].

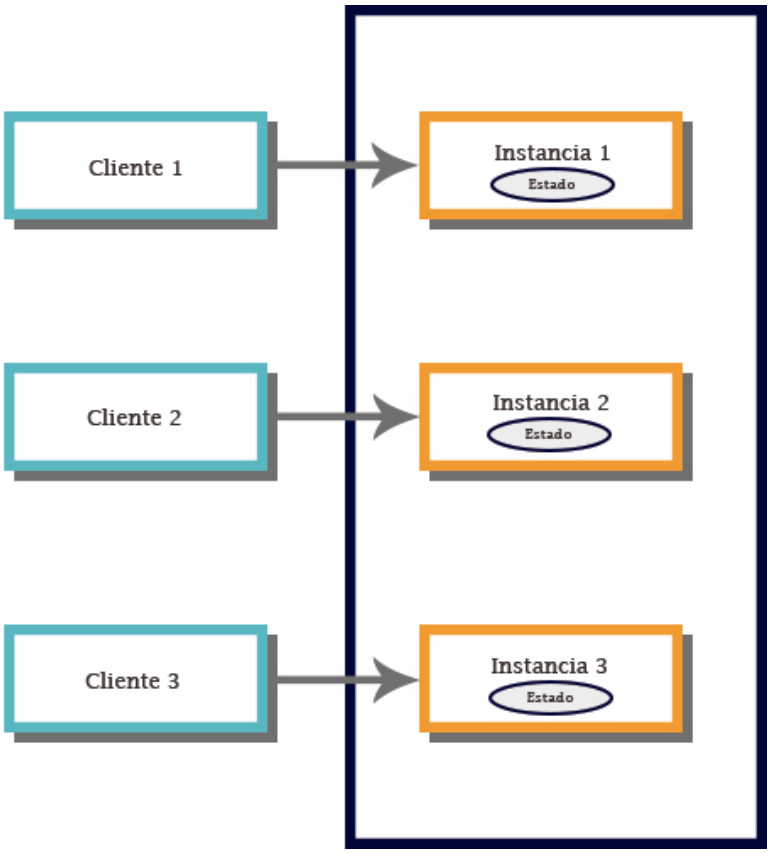


Figura 5.4: *Enterprise Java Beans*

Algunos problemas de ingeniería de software que se plantean, los desarrolladores pueden concluir que son ideales para resolverlos con el uso de *enterprise beans*. Una de las razones sería que su naturaleza muestra que los datos de los *beans* son únicos para todos los clientes que soliciten los servicios. Como por ejemplo, obtener los resultados matemáticos como las medidas de dispersión, la compresión de archivos de imágenes, de videos, pdf, verificación de tarjetas de crédito, etc. Para el cálculo de la media se necesitarían los datos a partir de los cuáles se va a calcular. Por lo tanto cada cliente debe proporcionarlos. Para la compresión de archivos el cliente proporcionará el buffer de memoria con el contenido del archivo, y con él, el formato del archivo, y en algunos casos podrán proporcionar el factor de compresión (aunque puede ser el mismo para todos). El *bean* correspondiente devolverá el resultado (el valor de la media, o el buffer con el contenido comprimido) [183, 189].

En todos los casos planteados, el proceso de negocio ha abarcado una solicitud de un método que, asemejando a la programación tradicional, es semejante a los métodos estáticos de una clase. Necesitan únicamente los parámetros (y algunas constantes) para brindar el servicio. El *bean* no tiene ninguna necesidad de retener ningún estado de las solicitudes anteriores. Por lo tanto, la naturaleza de los *beans* planteados se presentan un paradigma de solicitud única, o *beans* sin estado o proveedores de métodos anónimos,

en vista de que, no necesitan conocer al cliente con anterioridad [189].

Bean de sesion sin Estado

"Un bean de sesión sin estado puede implementar un servicio web, pero un bean de sesión con estado no puede" [189].

Beans de Sesión *Singleton* o *Singleton Session Beans*

Se crea una instancia de un *bean* de sesión singleton una vez por aplicación y existe durante el ciclo de vida de la aplicación. Los *beans* de sesión Singleton están diseñados para circunstancias en las que los clientes comparten una única instancia de *Enterprise Bean* y acceden a ella simultáneamente.

La funcionalidad de los *beans* de sesión singleton es semejante a la de los *beans* de sesión sin estado, diferenciándose de ellos en la cantidad de *beans* existentes en una aplicación. Mientras en una aplicación pueden coexistir varios *beans* de sesión sin estado, por el contrario sólo hay un *bean* de sesión singleton por aplicación. Cualquiera de los *beans* de sesión sin estado puede responder a una solicitud del cliente. Así mismo, una de sus similitudes entre ellos es que, tanto los *beans* de sesión sin estado como los *beans* de sesión singleton pueden implementar puntos finales de servicios web.

Los *beans* de sesión *singleton* mantienen su estado entre invocaciones de clientes, perdiendo su estado, como es de esperarse, en caso de bloqueos o cierres del servidor.

Las aplicaciones que utilizan un *bean* de sesión *singleton* pueden especificar que se debe crear una instancia del *singleton* al iniciar la aplicación, lo que permite que el *singleton* realice tareas de inicialización para la aplicación. El *singleton* también puede realizar tareas de limpieza cuando se apaga la aplicación, porque el *singleton* funcionará durante todo el ciclo de vida de la aplicación.

Tipos de beans de sesion

Los beans de sesión pueden ser:

- Sin estado (*Stateless*)
- Con estado (*Stateful*)
- Singleton (*Universales o globales*)

Cuándo utilizar *beans* de sesión

Los *beans* de sesión con estado son apropiados si se cumple alguna de las siguientes condiciones [189]:

- Los datos de un *bean* son la información de la interacción entre el *bean* y un cliente específico.

- El *bean* necesita contener información sobre el cliente a través de las invocaciones de métodos.
- El *bean* es un agente que se encuentra entre el cliente y los otros componentes de la aplicación, con el fin de simplificarle la vista al cliente.
- Detrás de escena, el *bean* gestiona el flujo de trabajo de varios *enterprise beans*.

Los *bean* de sesión sin estado pueden mejorar el rendimiento, bajo las siguientes circunstancias:

- El estado del *bean* no tiene datos para un cliente específico.
- En una invocación de un solo método, el *bean* realiza una tarea genérica para todos los clientes. Por ejemplo, puede utilizar un *bean* de sesión sin estado para enviar un correo electrónico que confirme un pedido en línea.
- El *bean* implementa un servicio web.

Se debe utilizar *beans* de sesión *singleton* en el desarrollo de una aplicación cuando se determinen los siguientes requisitos:

- Cuando los datos que definen el estado del *bean* se requiere tener (debe ser necesario) acceso en toda la aplicación.
- Se prevé que van a existir varios subprocesos accediendo simultáneamente a un único *enterprise bean*.
- La aplicación debe realizar tareas al inicio y al final de su ejecución. Esto se puede implementar en un *enterprise bean*.
- Se requiere implementar servicios web. Esto se lo puede hacer mediante un *bean*.

5.7. *Enterprise Java Beans* en la práctica

En esta sección se presentarán ejemplos básicos desarrollando con EJB. Para la realización de estos ejemplos vamos a utilizar el IDE Apache NetBeans 12, aunque la diferencia de usar alguna versión anterior (NetBeans 8.2) no es significativa. Se usará conjuntamente con el JDK 1.8 y JEE 8.0, aunque con las versiones actuales no hay mayores cambios.

5.7.1. Ejemplo 1:

Session Beans de Tipo *Stateless* con Interfaz Remota. Para crear un *Session Bean*, se lo puede hacer en 3 tipos de aplicaciones:

- *Enterprise application*: contiene EJBs y sus clientes,

- Un módulo EJB: sólo contienen EJBs y,
- Una aplicación Web

Para este ejemplo se utilizarán 3 proyectos diferentes, todos enlazados entre sí, los cuales son: un módulo EJB donde residan los *Session Beans*, una aplicación web que sería la aplicación cliente y un proyecto de librería de clases donde se aloja la interfaz remota. Para lo cual se deben seguir los siguientes pasos:

1. Crear un proyecto para un módulo EJB (*EJB module*) de la categoría Java EE. En la figura 5.5 muestra los datos que se deben rellenar al crear un módulo EJB.

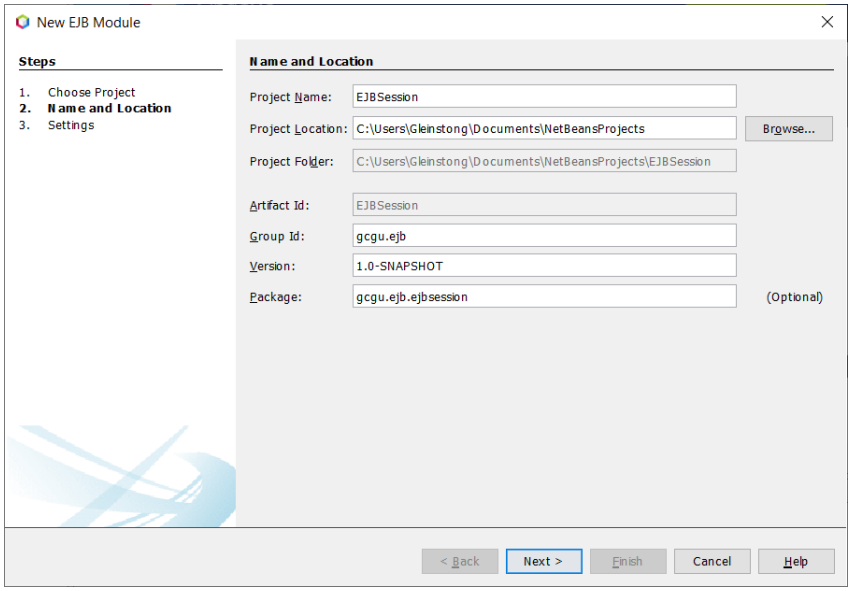
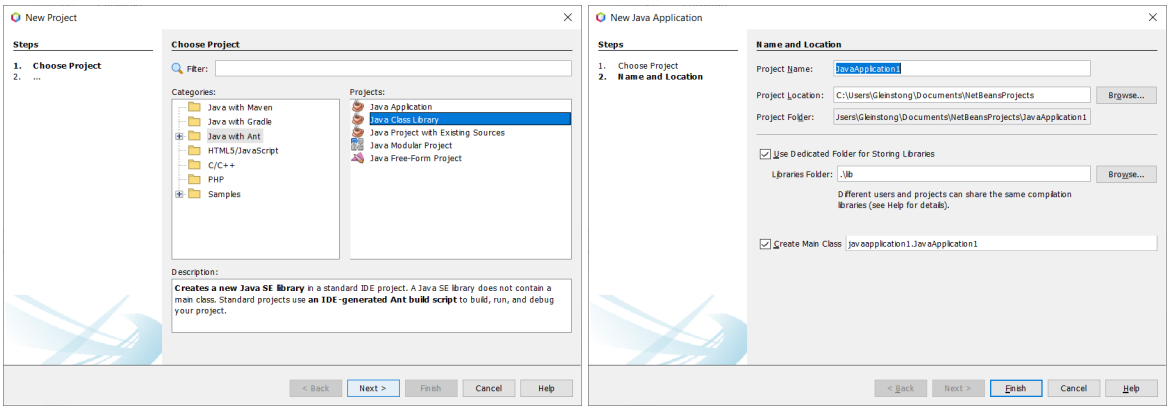


Figura 5.5: Datos que rellenar al crear un Módulo EJB

2. Crear un proyecto Java para biblioteca de clases. Los pasos para crearlo en Apache NetBeans 12 se muestran en la figura 5.6, primero se debe escoger la categoría de proyectos a crear. Para una biblioteca de clases se debe escoger en **categorías** *Java With Ant* y luego escoge el proyecto *Java Class Library*. En la figura 5.6a se muestra la ventana en la que el asistente permite escoger la categoría de proyectos. Al dar clic en siguiente aparece la ventana que se muestra en la figura 5.6b, permite ingresar los datos que se deben rellenar para su correcta creación (inicial).

3. Ahora crearemos el nuevo proyecto para la aplicación web. Ya lo hemos hecho antes en el versión 12 del IDE Apache NetBeans (ver 4.9b). La figura 5.7 muestra como crear un nuevo proyecto para una aplicación web en el IDE Apache Netbeans, como se detalló, será la versión usada para el desarrollo de estos ejemplos.

4. En este punto se procederá a crear la Interfaz Remota con la notación `@Remote` del paquete `javax.ejb.Remote`, cuyo código se muestra en el listado de código 5.1.



(a) Seleccionar la categoría de proyecto a crear. **(b)** Datos del proyecto biblioteca de clases.

Figura 5.6: Pasos para crear un proyecto para una biblioteca de clases.

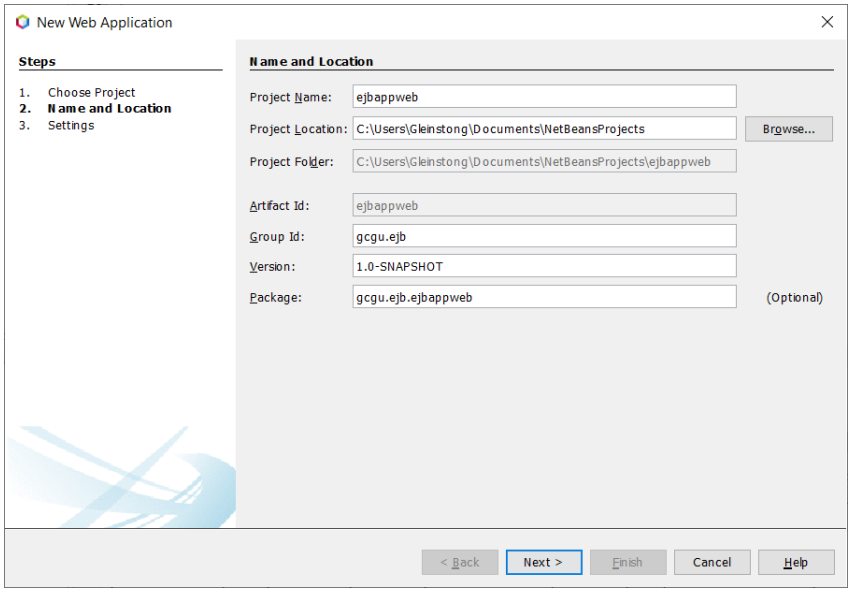


Figura 5.7: Datos del proyecto para la nueva Aplicación Web

```
1  package ejbremote.remoteinter;
2
3  import javax.ejb.Remote;
4
5  /**
6   *
7   * @author Gleinton
8   */
9  @Remote
10 public interface RemoteMessage{
11     String getMenssage(String message);
12     Float suma (Float first, Float second);
13 }
```

Listado 5.1: Código de la interfaz remota (*Remote*)

5. Una vez creado el .jar de la biblioteca de clases (compilar y construir el respectivo proyecto), se procede a agregar la dependencia del proyecto de la biblioteca de clases al proyecto de la aplicación web. Para ello, en la carpeta de las dependencias del proyecto Maven (EJBSession), se presiona el botón secundario del ratón y se escoge la opción de añadir dependencia (*Add Dependency*), y se procede a rellenar los campos del formulario (ver figura 5.8).

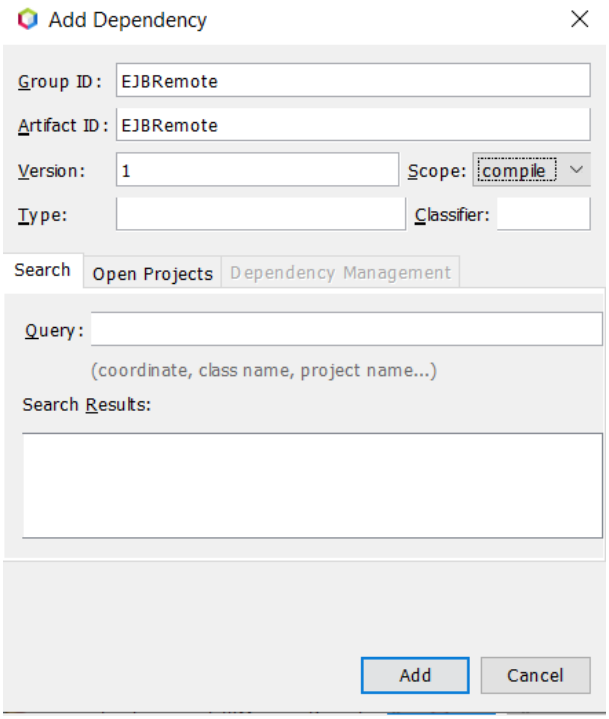


Figura 5.8: Añadir una Dependencia

Ninguno de los datos es puntual (puede ser cualquiera), al final se da clic en el botón Añadir (*Add*). Finalmente se procede a instalar la dependencia, para ello dando clic secundario en el nombre de la dependencia y se escoge la correcta. El proyecto finalmente debe tener una estructura parecida a la que se muestra en la figura 5.9.

6. Se procederá a añadir un *bean* de sesión en el proyecto para el módulo EJB (ver 5.10). Es recomendable (obligatorio) se agregue este *bean* a un paquete del proyecto. Puede ser a un paquete existente o simplemente crearlo conjuntamente con la clase del *bean* de sesión. Para crear el paquete con la clase, se puede anteponer al nombre de la clase el nombre del paquete (paquete.NombreClase), o simplemente en la caja de texto paquete (*package*) colocar el nombre de la ruta del paquete

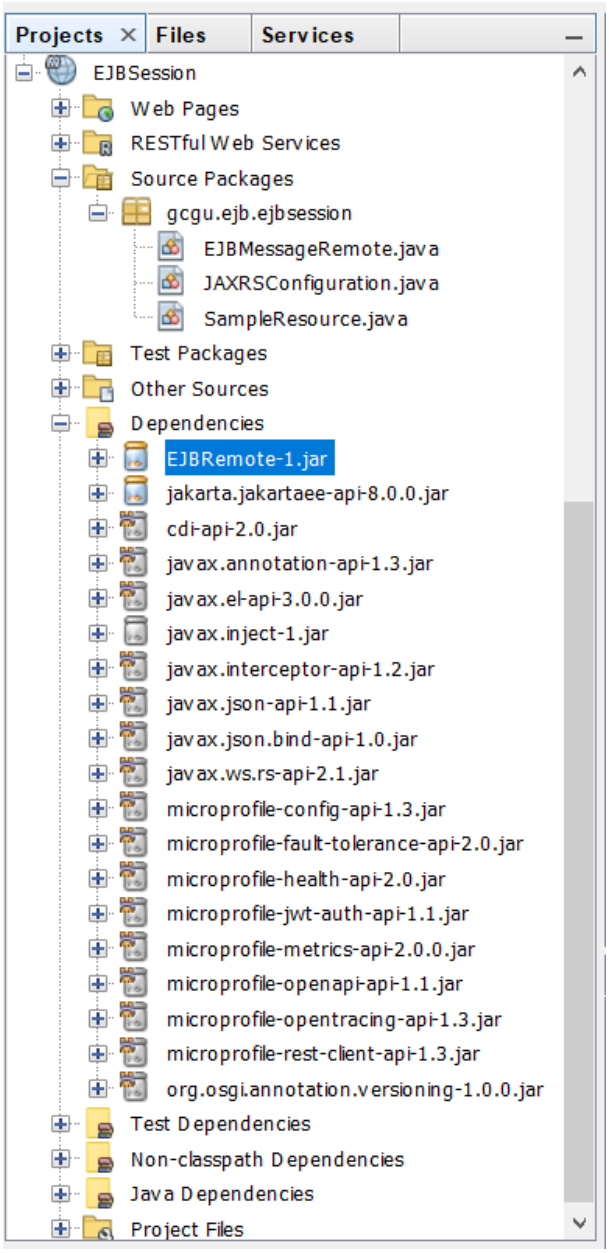


Figura 5.9: Estructura del proyecto de la aplicación web

contenedor. Luego se procede a especificar que tipo de *bean* de sesión se desea crear, así como también el tipo de interfaz a implementar. Para este ejemplo se escogerá el tipo Stateless. En NetBens 8.2, se puede especificar la interfaz remota a implementar. Sin embargo, en Apache NetBeans en este caso, se debe hacer la importación del `jar` del proyecto *Java Class Library* que se creó anteriormente (ver figura 5.6b).

7. Una vez creado el *bean* de sesión, se procede a añadirle los métodos de negocios definidos en la interfaz remota, es tan simple como dar clic en el foco amarillo que

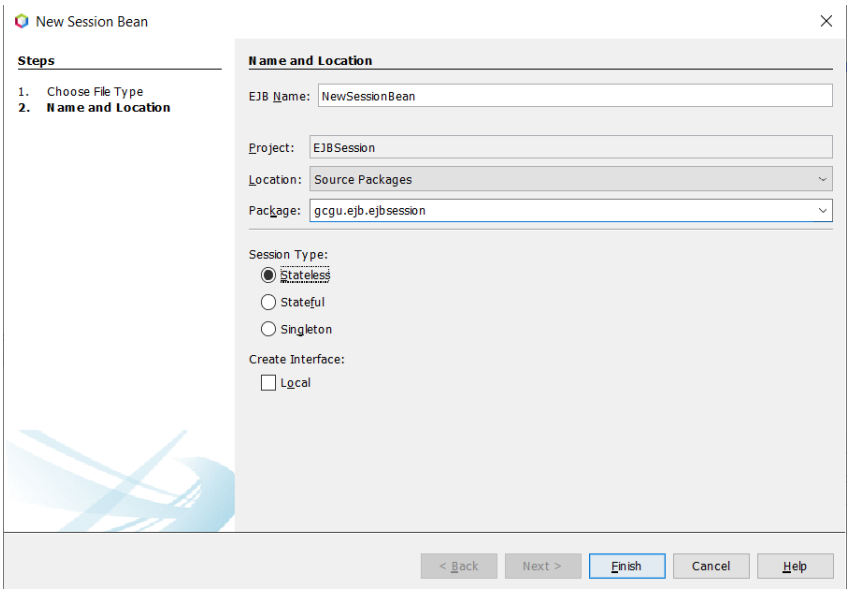


Figura 5.10: Datos del proyecto para la nueva Aplicación Web

muestra el editor (ver figura 5.11), y escoger la primera opción **Implementar todos los métodos abstractos**. Este error es propio al definir una clase **no abstracta** en la que se está implementando una interfaz que se desee. Si se desea implementar nuevos métodos de negocios (no aconsejable, deben ser definidos en una interfaz), la forma más sencilla es usando el botón secundario del ratón dentro del cuerpo de la clase *bean* (en este caso), se escoge **Insertar código** luego **Añadir método de negocios (*Add Business Method*)**, se procede a establecer el nombre del método, el tipo de retorno, y los parámetros que necesite el método para lograr su objetivo.

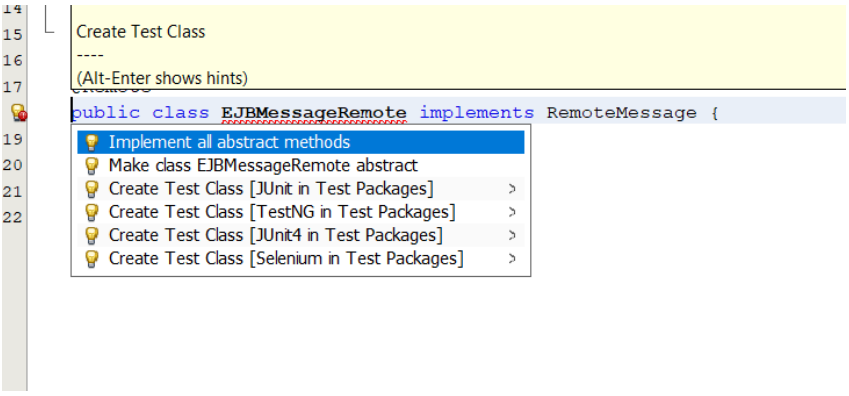


Figura 5.11: Implementación de los métodos de la interfaz en una clase.

En la figura 5.12 se muestra la ventana que permite agregar de manera fácil un método de negocios para el *bean* de sesión. Los datos que pide corresponden a la definición del método de la clase: nombre del método, tipo de valor de retorno, y los parámtros con sus tipos. Los tipos de retorno deberían ser objetos. Al ser

valores enteros, o reales, java define clases para cada uno de estos tipos: float: Float, int: Integer, double: Double, etc.

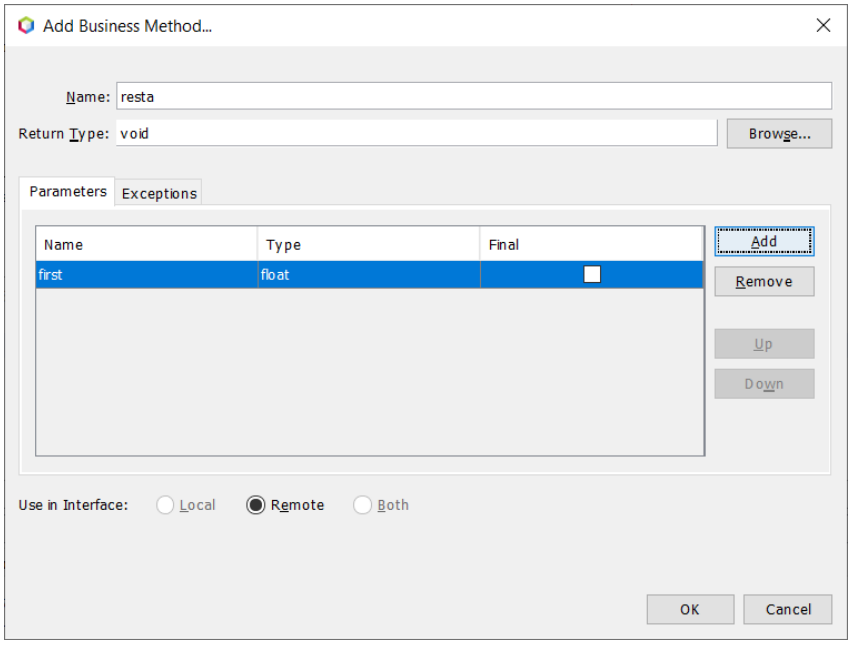


Figura 5.12: Datos del proyecto para la nueva Aplicación Web

De esta manera, se pueden agregar tantos métodos como se desee, así mismo los tipos de parámetros pueden ser tan diferentes como las necesidades del método de negocios lo determinen. El resultado de añadir el método suma a la clase del *bean* de sesión se muestra en el listado de código 5.2

```
1  package gcgu.ejb.ejbsession;
2
3  import ejbremote.remoteinter.RemoteMessage;
4  import javax.ejb.Stateless;
5  import javax.ejb.Remote;
6
7  /**
8   *
9   * @author Gleiston
10  */
11 @Stateless
12 @Remote
13 public class EJBMessageRemote implements RemoteMessage {
14
15     @Override
16     public String getMenssage(String message) {
17         return message;
18     }
19
20     @Override
21     public Float suma(Float first, Float second) {
```

```
22          return first + second;
23      }
24
25      // Add business logic below. (Right-click in editor and choose
26      // "Insert Code > Add Business Method")
27  }
```

Listado 5.2: Código de la clase *bean* de sesión

8. Todo listo para consumir el ***Session Bean*** o *Bean* de Sesión. Para ello se creará un *Servlet*[10] (en el proyecto de aplicación web), sobre el proyecto (o sobre *Web Pages*) se presiona el botón secundario y se escoge **Nuevo ->Servlet**. Si es la primera vez que va a usar *Servlets*, seguro no lo tendrá entre las opciones rápidas, podrá entonces seleccionar Otros **Nuevo ->Otros** y en la ventana que se despliega escoja en categorías Web y en tipo de archivo *Servlet*, tal como se muestra en la figura 4.15. Para consumir el EJB, se procede a presionar el botón secundario del ratón en el cuerpo del *Servlet* (método **processRequest**), escogemos **Llamada a *Enterprise Bean*** y se selecciona el *Enterprise Bean* correspondiente, en este ejemplo debe ser **MensajeRemoto**. El código de la clase *Servlet* creada se muestra en el listado de código 5.3.

   El consumo del EJB se puede hacer desde cualquier clase Java. Por ejemplo, en una clase entidad o *Entity*, o en una clase controladora como un `ManageBean`, entre las más comunes. Aunque se está reduciendo el número de aplicaciones de escritorio, en ellas también se pueden consumir los EJB.

```
1  package gcgu.ejb.ejbappweb.servlet;
2
3  import java.io.IOException;
4  import java.io.PrintWriter;
5  import javax.ejb.EJB;
6  import javax.servlet.ServletException;
7  import javax.servlet.http.HttpServlet;
8  import javax.servlet.http.HttpServletRequest;
9  import javax.servlet.http.HttpServletResponse;
10
11 /**
12  *
13  * @author Gleinton
14  */
15 public class MessageServlet extends HttpServlet {
16
17     @EJB
18     private ejbremote.remoteinter.RemoteMessage eJBMessageRemote;
19
20     /**
```

[10]Un *servlet* es un programa Java que se ejecuta en un servidor Web y construye o sirve páginas web solicitadas por algún cliente.

```
21      * Processes requests for both HTTP <code>GET</code> and <code>
    POST</code>
22      * methods.
23      *
24      * @param request servlet request
25      * @param response servlet response
26      * @throws ServletException if a servlet-specific error occurs
27      * @throws IOException if an I/O error occurs
28      */
29     protected void processRequest(HttpServletRequest request,
    HttpServletResponse response)
30             throws ServletException, IOException {
31         Float uno, dos;
32         String sendedMessage;
33         response.setContentType("text/html;charset=UTF-8");
34         try ( PrintWriter out = response.getWriter()) {
35             /* TODO output your page here. You may use following
    sample code. */
36             out.println("<!DOCTYPE html>");
37             out.println("<html>");
38             out.println("<head>");
39             out.println("<title>Servlet MessageServlet</title>");
40             out.println("</head>");
41             out.println("<body>");
42
43             uno = Float.parseFloat(request.getParameter("first"));
44             dos = Float.parseFloat(request.getParameter("second"));
45
46             sendedMessage = eJBMessageRemote.suma(uno, dos).toString()
    ;
47
48             out.println(" <strong > First = " + uno + "<br />");
49             out.println(" <strong > Second = " + dos + "<br />");
50             out.println(" <strong > The sum = " + sendedMessage + "<br
    />");
51
52             out.println("</body>");
53             out.println("</html>");
54         }
55     }
56
57     // <editor-fold defaultstate="collapsed" desc="HttpServlet methods
    . Click on the + sign on the left to edit the code.">
58     /**
59      * Handles the HTTP <code>GET</code> method.
60      *
61      * @param request servlet request
62      * @param response servlet response
63      * @throws ServletException if a servlet-specific error occurs
64      * @throws IOException if an I/O error occurs
65      */
66     @Override
67     protected void doGet(HttpServletRequest request,
```

```
    HttpServletResponse response)
68              throws ServletException, IOException {
69          processRequest(request, response);
70      }
71
72      /**
73       * Handles the HTTP <code>POST</code> method.
74       *
75       * @param request servlet request
76       * @param response servlet response
77       * @throws ServletException if a servlet-specific error occurs
78       * @throws IOException if an I/O error occurs
79       */
80      @Override
81      protected void doPost(HttpServletRequest request,
    HttpServletResponse response)
82              throws ServletException, IOException {
83          processRequest(request, response);
84      }
85
86      /**
87       * Returns a short description of the servlet.
88       *
89       * @return a String containing servlet description
90       */
91      @Override
92      public String getServletInfo() {
93          return "Short description";
94      }// </editor-fold>
95
96  }
```

Listado 5.3: Código de la clase *Servlet* (Consumo del EJB)

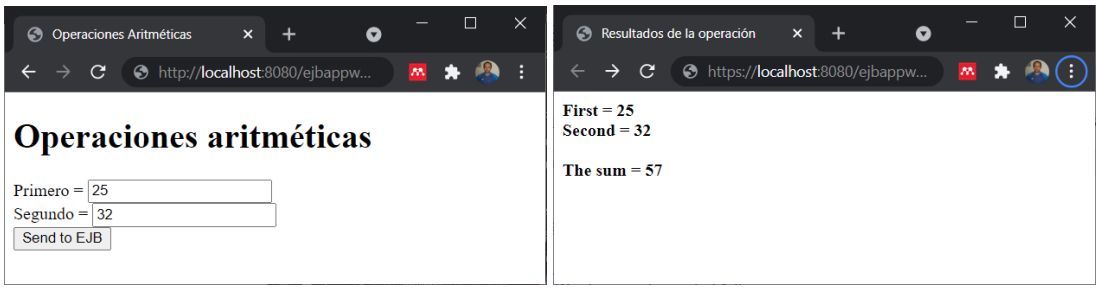
9. Es tiempo de diseñar al interfaz para que el usuario interactúe con la aplicación. Será algo sencillo. Una página HTML con un formulario que contendrá dos cajas de texto para ingresar los dos valores punto flotante que serán procesados por el *Servlet*. Sino se configuró ningún atributo en el *Servlet*, su nombre para especificar en el formulario será igual que el nombre de la clase sin la extensión, como se muestra en el listado de código 5.4.

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Operaciones Aritméticas</title>
5          <meta charset="UTF-8" />
6          <meta name="viewport" content="width=device-width , initial-
       scale=1.0" />
7      </head>
8      <body>
9          <h1>Operaciones aritméticas</h1>
10         <form action="ServletMensaje" method="POST">
```

```
11              Primero = <input type="text" name="first" maxlength="8"
12                                required /> <br />
13              Segundo = <input type="text" name="second" maxlength="8"
14                                required /> <br />
15              <input type="submit" value="Send to EJB"/>
16          </form>
17      </body>
18  </html>
```

Listado 5.4: Código HTML de la página **index.html**

10. Resultados. Una vez con todas las páginas creadas, se debe ejecutar el proyecto para mostrar la correspondiente interfaz entre el usuario y la aplicación web. En la figura 5.13 se muestra ambas interfazs de usuario. En la figura 5.13a muestra la página index.html, en la cual, el usuario ha ingresados los valores a ser procesados por el servidor (*Servlet*), y el la figura 5.13b muestra los resultados obtenidos en el *Servlet* consumiendo el EJB.



(a) Interfaz de Usuario - página **index.html** **(b)** Interfaz de Usuario - Resultados del **Servlet**

Figura 5.13: Ejecución de la aplicación con Servlets

5.7.2. Ejemplo 2

Si los requisitos de un cliente son que se pueda gestionar los pedidos que sus clientes hacen a su tienda. Se presenta la solución de abastecer los productos a su, se trabajará para ello con productos, clientes, proveedores, órdenes y empleados. Otro de los requisitos solicitados es que se pueda visualizar los ingresos en tiempo real. Para ello, el equipo de desarrollo decide usar EBJ para un mejor control.

1. Una vez analizado el problema, se procede a crear la base de datos en PostgreSQL[11] con sus respectivas tablas. El listado de código en SQL (*Structured Query Language* - Lenguaje Estructurado de Consultas) se muestra en el listado 5.5.

```
1  /* Si existe la base de datos, se elimina y se crea nuevamente */
2  drop database if exists PEDIDOS;
3
```

[11]Es un sistema de gestión de bases de datos relacional orientado a objetos y de código abierto.

```
4  -- Se crea la base de datos
5  create database PEDIDOS;
6
7  CREATE TABLE empleados(
8  id int NOT NULL,
9  nombre char(30) NULL,
10 apellido char(30) NULL,
11 fechanacimiento date NULL,
12 jefeinmediato int NULL,
13 extension int NULL,
14 CONSTRAINT pk_empleados PRIMARY KEY (id));
15
16 CREATE TABLE proveedores(
17 id int NOT NULL,
18 razonsocial char(50) NOT NULL,
19 contacto char(50) NOT NULL,
20 nummovil char(12) NULL,
21 numconvencional char(12) NULL,
22 CONSTRAINT pk_proveedores PRIMARY KEY
23 (id));
24
25 CREATE TABLE categorias(
26 id int NOT NULL,
27 nombre char(50) NOT NULL,
28 CONSTRAINT PK_categorias PRIMARY KEY
29 (id) ) ;
30
31 CREATE TABLE clientes(
32 id int NOT NULL,
33 documentoid char(10) NOT NULL,
34 nombreCIA char(30) NOT NULL,
35 nombrecontacto char(50) NOT NULL,
36 direccion char(50) NOT NULL,
37 fax char(12) NULL,
38 email char(50) NULL,
39 nummovil char(12) NULL,
40 numconvencional char(12) NULL,
41 CONSTRAINT pk_clientes PRIMARY KEY
42 (id));
43
44 CREATE TABLE ordenes(
45 id int NOT NULL,
46 idempleado int NOT NULL,
47 idcliente int NOT NULL,
48 fechaorden date NOT NULL,
49 descuento int NULL,
50 CONSTRAINT pk_ordenes PRIMARY KEY
51 (id) );
52
53
54 CREATE TABLE detalle_ordenes(
55 id int NOT NULL,
56 idorden int NOT NULL,
```

```
idproducto int NOT NULL,
CANTIDAD int NOT NULL,
CONSTRAINT pk_detalle_ordenes PRIMARY KEY
(idorden,id));

CREATE TABLE productos(
id int NOT NULL,
idproveedor int NOT NULL,
idcategoria int NOT NULL,
descripcion char(50) NULL,
preciounitario numeric NOT NULL,
stock int NOT NULL,
CONSTRAINT PK_productos PRIMARY KEY
(idproducto )) ;

ALTER TABLE ordenes
ADD CONSTRAINT fk_ordenes_clien_ord_clientes FOREIGN KEY(idcliente)
REFERENCES clientes (id)
on delete restrict on update restrict;

ALTER TABLE ordenes ADD CONSTRAINT fk_ordenes_emple_ord_empleado
    FOREIGN KEY(idempleado)
REFERENCES empleados (id)
on delete restrict on update restrict;

ALTER TABLE detalle_ordenes ADD CONSTRAINT
    fk_detalle__orden_det_ordenes FOREIGN KEY(idorden)
REFERENCES ordenes (id)
on delete restrict on update restrict;

ALTER TABLE detalle_ordenes ADD CONSTRAINT
    fk_detalle__prod_deta_producto FOREIGN KEY(idproducto)
REFERENCES productos (id)
on delete restrict on update restrict;

ALTER TABLE productos ADD CONSTRAINT fk_producto_cate_prod_categori
    FOREIGN KEY(idcategoria)
REFERENCES categorias (id)
on delete restrict on update restrict;

ALTER TABLE productos ADD CONSTRAINT fk_producto_prov_prod_proveedo
    FOREIGN KEY(idproveedor)
REFERENCES proveedores (id)
on delete restrict on update restrict;

ALTER TABLE empleados ADD CONSTRAINT fk_empleado_reporta FOREIGN KEY(
    jefeinmediato)
REFERENCES empleados (id)
on delete restrict on update restrict;
```

Listado 5.5: Sentencias SQL para la creación de la base de datos

La inserción de datos para las pruebas, se muestra en el código SQL del listado

```
1  INSERT INTO categorias (id, nombrecat) VALUES (100, 'CARNICOS');
2  INSERT INTO categorias (id, nombrecat) VALUES (200, 'LACTEOS');
3  INSERT INTO categorias (id, nombrecat) VALUES (300, 'LIMPIEZA');
4  INSERT INTO categorias (id, nombrecat) VALUES (400, 'HIGINE PERSONAL')
       ;
5  INSERT INTO categorias (id, nombrecat) VALUES (500, 'MEDICINAS');
6  INSERT INTO categorias (id, nombrecat) VALUES (600, 'COSMETICOS');
7  INSERT INTO categorias (id, nombrecat) VALUES (700, 'REVISTAS');
8
9  INSERT INTO proveedores (id, razonsocial, contacto,nummovil,
       numconvencional) VALUES
10 (10, 'DON DIEGO', 'MANUEL ANDRADE', '099234567','2124456');
11 INSERT INTO proveedores (id, razonsocial, contacto,nummovil,
       numconvencional) VALUES
12 (20, 'PRONACA', 'JUAN PEREZ', '0923434467','2124456');
13 INSERT INTO proveedores (id, razonsocial, contacto,nummovil,
       numconvencional) VALUES
14 (30, 'TONY', 'JORGE BRITO', '099234567','2124456');
15 INSERT INTO proveedores (id, razonsocial, contacto,nummovil,
       numconvencional) VALUES
16 (40, 'MIRAFLORES', 'MARIA PAZ', '098124498','2458799');
17 INSERT INTO proveedores (id, razonsocial, contacto,nummovil,
       numconvencional) VALUES
18 (50, 'ALMAY', 'PEDRO GONZALEZ', '097654567','2507190');
19 INSERT INTO proveedores (id, razonsocial, contacto,nummovil,
       numconvencional) VALUES
20 (60, 'REVLON', 'MONICA SALAS', '099245678','2609876');
21 INSERT INTO proveedores (id, razonsocial, contacto,nummovil,
       numconvencional) VALUES
22 (70, 'YANBAL', 'BETY ARIAS', '098124458','2450887');
23 INSERT INTO proveedores (id, razonsocial, contacto,nummovil,
       numconvencional) VALUES
24 (120, 'JURIS', 'MANUEL ANDRADE', '099234567','2124456');
25 INSERT INTO proveedores (id, razonsocial, contacto,nummovil,
       numconvencional) VALUES
26 (80, 'CLEANER', 'MANUEL ANDRADE', '099234567','2124456');
27 INSERT INTO proveedores (id, razonsocial, contacto,nummovil,
       numconvencional) VALUES
28 (90, 'BAYER', 'MANUEL ANDRADE', '099234567','2124456');
29 INSERT INTO proveedores (id, razonsocial, contacto,nummovil,
       numconvencional) VALUES
30 (110, 'PALMOLIVE', 'MANUEL ANDRADE', '099234567','2124456');
31
32 INSERT INTO productos VALUES (1,10,100,'SALCHICHAS VIENESAS',2.60,200)
       ;
33 INSERT INTO productos VALUES (2,10,100,'SALAMI DE AJO',3.60,300);
34 INSERT INTO productos VALUES (3,10,100,'BOTON PARA ASADO',4.70,400);
35 INSERT INTO productos VALUES (4,20,100,'SALCHICHAS DE POLLO',2.90,200)
       ;
36 INSERT INTO productos VALUES (5,20,100,'JAMON DE POLLO',2.80,100);
37 INSERT INTO productos VALUES (6,30,200,'YOGURT NATURAL',4.30,80);
```

```
38 INSERT INTO productos VALUES (7,30,200,'LECHE CHOCOLATE',1.60,90);
39 INSERT INTO productos VALUES (8,40,200,'YOGURT DE SABORES',1.60,200);
40 INSERT INTO productos VALUES (9,40,200,'CREMA DE LECHE',3.60,30);
41 INSERT INTO productos VALUES (10,50,600,'BASE DE MAQUILLAJE',14.70,40)
       ;
42 INSERT INTO productos VALUES (11,50,600,'RIMMEL',12.90,20);
43 INSERT INTO productos VALUES (13,60,600,'SOMBRA DE OJOS',9.80,100);
44
45 set datestyle to dmy;
46
47 INSERT INTO empleados VALUES (1,'JUAN', 'CRUZ', '18/01/67',null, 231);
48 INSERT INTO empleados VALUES (2,'MARIO', 'SANCHEZ', '01/03/79',1,144);
49 INSERT INTO empleados VALUES (3,'VERONICA', 'ARIAS', '23/06/77',1,
       234);
50 INSERT INTO empleados VALUES (4,'PABLO', 'CELY', '28/01/77',2, 567);
51 INSERT INTO empleados VALUES (5,'DIEGO', 'ANDRADE', '15/05/70',2, 890)
       ;
52 INSERT INTO empleados VALUES (6,'JUAN', 'ANDRADE', '17/11/76',3, 230);
53 INSERT INTO empleados VALUES (7,'MARIA', 'NOBOA', '21/12/79',3, 261);
54
55 INSERT INTO clientes VALUES (1,'1890786576','SUPERMERCADO ESTRELLA','
       JUAN ALBAN','AV.AMAZONAS',NULL,NULL,NULL,NULL);
56 INSERT INTO clientes VALUES (2,'1298765477','EL ROSADO','MARIA CORDERO
       ','AV.AEL INCA',NULL,NULL,NULL,NULL);
57 INSERT INTO clientes VALUES (3,'1009876567','DISTRIBUIDORA PRENSA','
       PEDRO PINTO','EL PINAR',NULL,NULL,NULL,NULL);
58 INSERT INTO clientes VALUES (4,'1876090006','SU TIENDA','PABLO PONCE',
       'AV.AMAZONAS',NULL,NULL,NULL,NULL);
59 INSERT INTO clientes VALUES (5,'1893456776','SUPERMERCADO DORADO','
       LORENA PAZ','AV.6 DICIEMBRE',NULL,NULL,NULL,NULL);
60 INSERT INTO clientes VALUES (6,'1678999891','MI COMISARIATO','ROSARIO
       UTRERAS','AV.AMAZONAS',NULL,NULL,NULL,NULL);
61 INSERT INTO clientes VALUES (7,'1244567888','SUPERMERCADO descuento','
       LETICIA ORTEGA','AV.LA PRENSA',NULL,NULL,NULL,NULL);
62 INSERT INTO clientes VALUES (8,'1456799022','EL descuento','JUAN
       TORRES','AV.PATRIA',NULL,NULL,NULL,NULL);
63 INSERT INTO clientes VALUES (9,'1845677777','DE LUISE','JORGE PARRA','
       AV.AMAZONAS',NULL,NULL,NULL,NULL);
64 INSERT INTO clientes VALUES (10,'183445667','YARBANTRELLA','PABLO
       POLIT','AV.REPUBLICA',NULL,NULL,NULL,NULL);
65
66 INSERT INTO ordenes VALUES(1,3,4,'17/06/21', 5);
67 INSERT INTO ordenes VALUES(2,3,4,'02/06/21', 10);
68 INSERT INTO ordenes VALUES(3,4,5,'05/06/21', 6);
69 INSERT INTO ordenes VALUES(4,2,6,'06/06/21', 2);
70 INSERT INTO ordenes VALUES(5,2,7,'09/06/21', NULL);
71 INSERT INTO ordenes VALUES(6,4,5,'12/06/21', 10);
72 INSERT INTO ordenes VALUES(7,2,5,'14/06/21', 10);
73 INSERT INTO ordenes VALUES(8,3,2,'13/06/21', 10);
74 INSERT INTO ordenes VALUES(9,3,2,'17/06/21', 3);
75 INSERT INTO ordenes VALUES(10,2,2,'18/06/21', 2);
76
77 INSERT INTO detalle_ordenes VALUES(1,1,1,2);
```

```
78 INSERT INTO detalle_ordenes VALUES(1,2,4,1);
79 INSERT INTO detalle_ordenes VALUES(1,3,6,1);
80 INSERT INTO detalle_ordenes VALUES(1,4,9,1);
81 INSERT INTO detalle_ordenes VALUES(2,1,10,10);
82 INSERT INTO detalle_ordenes VALUES(2,2,13,20);
83 INSERT INTO detalle_ordenes VALUES(3,1,3,10);
84 INSERT INTO detalle_ordenes VALUES(4,1,9,12);
85 INSERT INTO detalle_ordenes VALUES(5,1,1,14);
86 INSERT INTO detalle_ordenes VALUES(5,2,4,20);
87 INSERT INTO detalle_ordenes VALUES(6,1,3,12);
88 INSERT INTO detalle_ordenes VALUES(7,1,11,10);
89 INSERT INTO detalle_ordenes VALUES(8,1,2,10);
90 INSERT INTO detalle_ordenes VALUES(8,2,5,14);
91 INSERT INTO detalle_ordenes VALUES(8,3,7,10);
92 INSERT INTO detalle_ordenes VALUES(9,1,11,10);
93 INSERT INTO detalle_ordenes VALUES(10,1,1,5);
```

Listado 5.6: Sentencias SQL para la inserción de registros en las tablas de la base de datos

2. **Crear el *Pool* de conexiones JDBC** Se procede a crear el origen de datos JDBC. Para ello debe estar levantado el servidor con el que se desea trabajar, en este caso se ha escogido **GassFish 5.0**. Desde la pestaña de prestaciones de NetBeans se expande **servidors**, seleccionado el servidor adecuado se presiona el botón secundario del ratón y se seleccionar ejecutar. Una vez ejecutado se procede nuevamente a seleccionar el servidor y se escoge la opción **Ver consola del administrador de dominio (*View Domain Admin Console*)**. Enseguida se va a desplegar el servidor en el navegador que esté por defecto.

3. Desplegada la consola de administrador, se expande **JDBC**, y luego ***JDBC Connection Pool***. En la figura 5.14 se muestra este proceso.

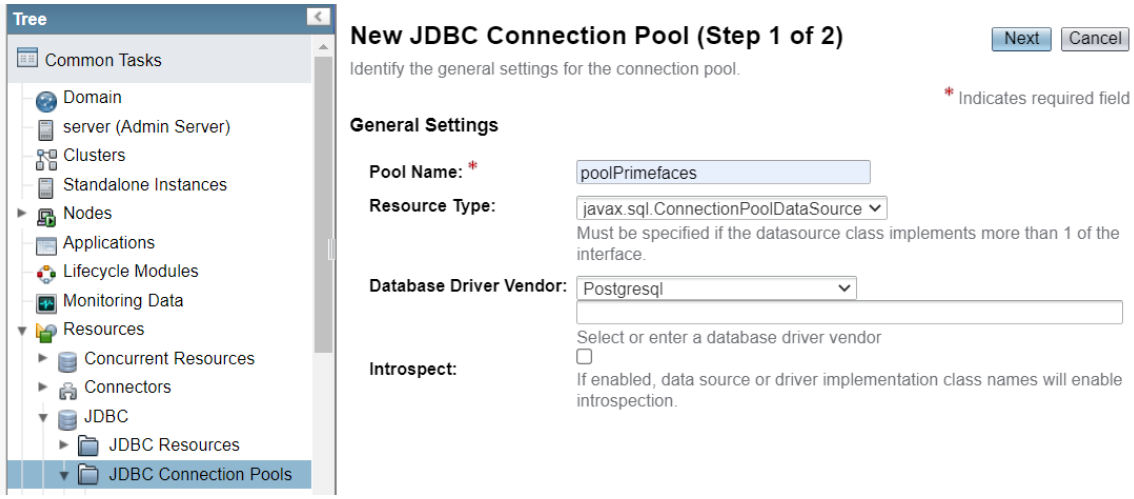


Figura 5.14: Creación del *Pool* de conexiones y su recurso (Paso 1)

Los datos que se ingresan en esta primera parte son fundamentales. Para continuar

debe dar clic en el botón ***Next*** (parte superior derecha de la ventana). Se debe desplegar la ventana de la figura 5.15.

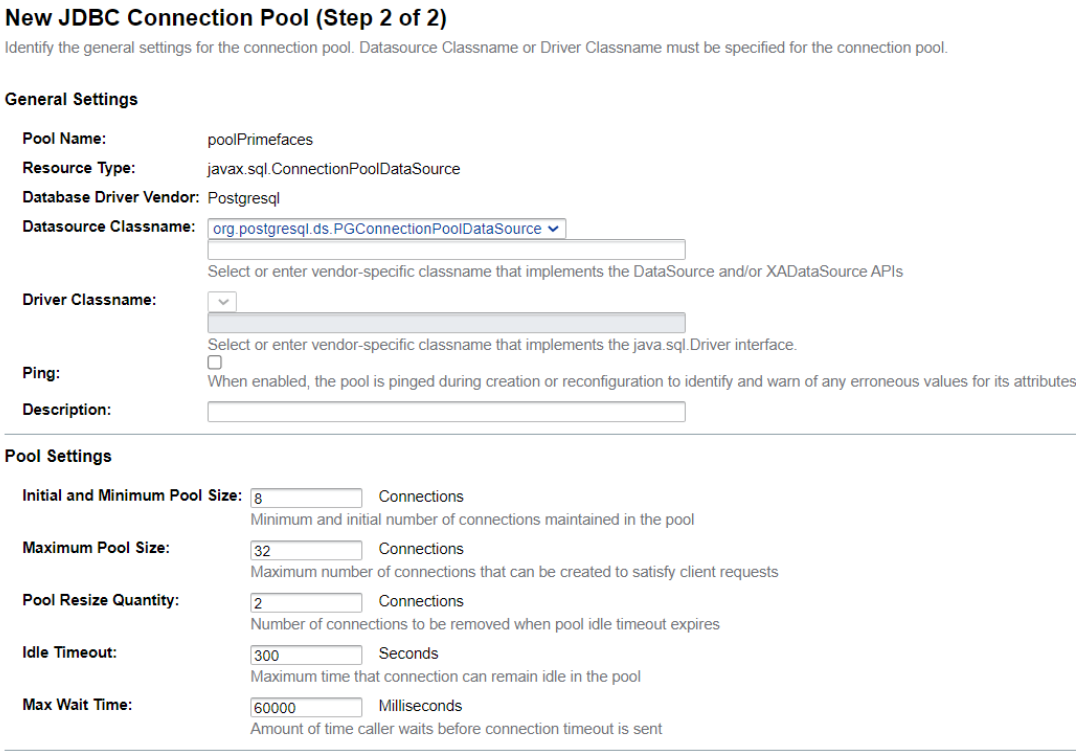


Figura 5.15: Creación del *Pool* de conexiones y su recurso (Paso 2)

Los datos que muestra la figura 5.15 son resultado de los ingresados en el primer paso. Si se despliega la página más abajo, se muestra una lista grade de datos, de los cuáles sólo se recomienda manipular los datos que se muestran en la figura 5.16.

Aunque los datos como el nombre del servidor y el puerto están como propiedades (separadas) se deben colocar estos datos adicionando a la propiedad url y URL.

Una vez creada, se puede editar al *pool* de conexiones simplemente dando clic sobre el nombre del que desea editar. Una vez editado se puede probar si existe conexión (los datos correctos y el servidor acepta peticiones TC/IP), para ello se da clic en el botón Ping, como resultado aparecerá un ícono de color amarillo con un visto verde tal como se muestra en la figura 5.17.

4. **Crear el recurso JDBC**. Así como se creó el *pool* de conexiones se procede a crear el recurso, expandiendo ***JDBC Resources*** **o Recursos JDBC** (ver figura 5.18a). Debe hacer clic en el botón *New* o Nuevo para ingresar los datos como se muestra en la figura 5.18b.

5. **Crear la unidad de persistencia.** En NetBeans, con el ratón sobre el nombre del proyecto presionar el botón secundario y seleccionar **Nuevo** y seleccionar

Additional Properties (78)

Add Property | Delete Properties

Select	Name	Value
☑	ReadOnly	false
☐	LogUnclosedConnections	false
☐	SocketFactory	
☑	DatabaseName	ejemplo1
☑	PortNumber	5432
☐	DefaultRowFetchSize	0
☐	Sslfactoryarg	
☐	Options	
☐	SslPassword	
☐	Sslhostnameverifier	
☑	User	postgres
☐	SslCert	
☐	DatabaseMetadataCacheFieldsMiB	5
☐	DefaultAutoCommit	
☐	MaxResultBuffer	
☐	RecvBufferSize	-1
☑	URL	jdbc:postgresql://localhost:5432/
☑	Url	jdbc:postgresql://localhost:5432/
☑	Password	clave
☐	SocketFactoryArg	
☑	ConnectTimeout	10
☑	LoginTimeout	0
☑	ServerName	localhost

Figura 5.16: Datos para la conexión del *Pool* de conexiones a la base de datos (Paso 3)

General | Advanced | Additional Properties

Ping Succeeded

Edit JDBC Connection Pool

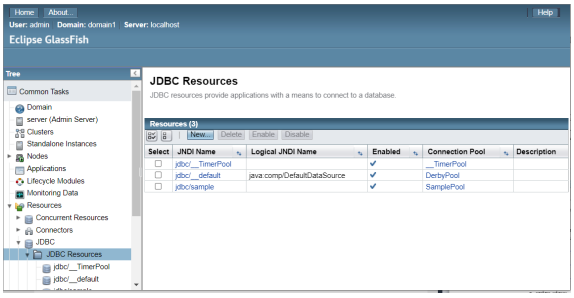
Save | Cancel

Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database.

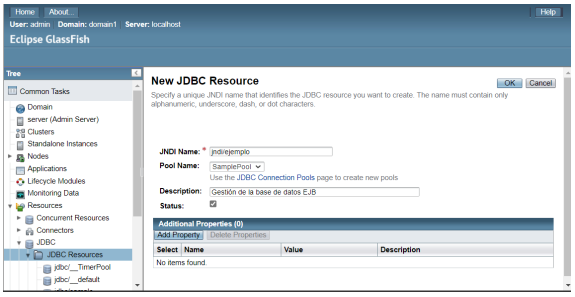
Load Defaults | Flush | Ping

* Indicates required field

Figura 5.17: Parámetros de conexión a la base de datos con el *Pool* de conexiones (Paso 4)



(a) Crear nuevo recurso



(b) Datos para el recurso

Figura 5.18: Creación del recurso JDBC (Paso 5)

***Persistence Unit* o Unidad de persistencia**, sino tiene esta opción debe seleccionar Otros, luego en categoría seleccionar Persistencia y en tipo de archivo ***Persistence Unit* o Unidad de persistencia**. Al dar clic en el botón aceptar debe aparecer la ventana para ingresar los datos de la unidad de persistencia como se muestra en la figura 5.19.

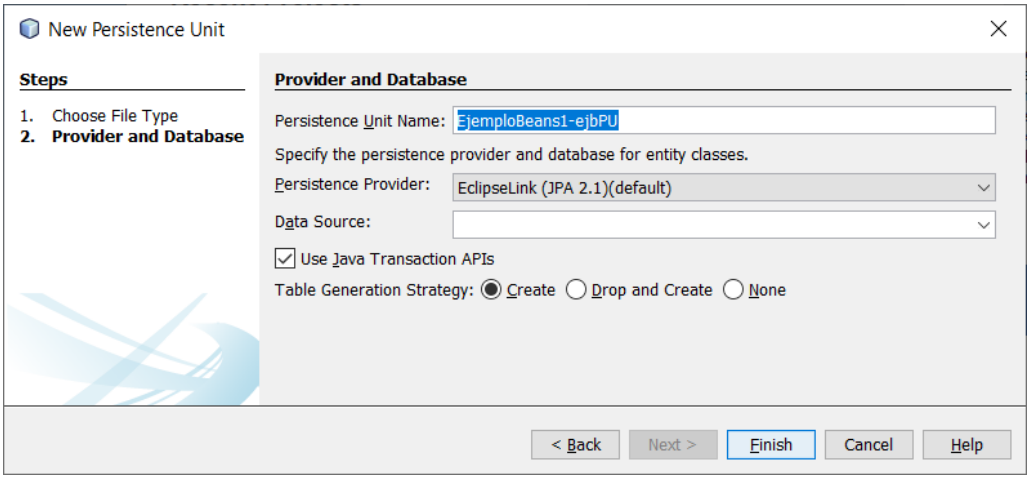


Figura 5.19: Parámetros de conexión a la base de datos con el *Pool* de conexiones

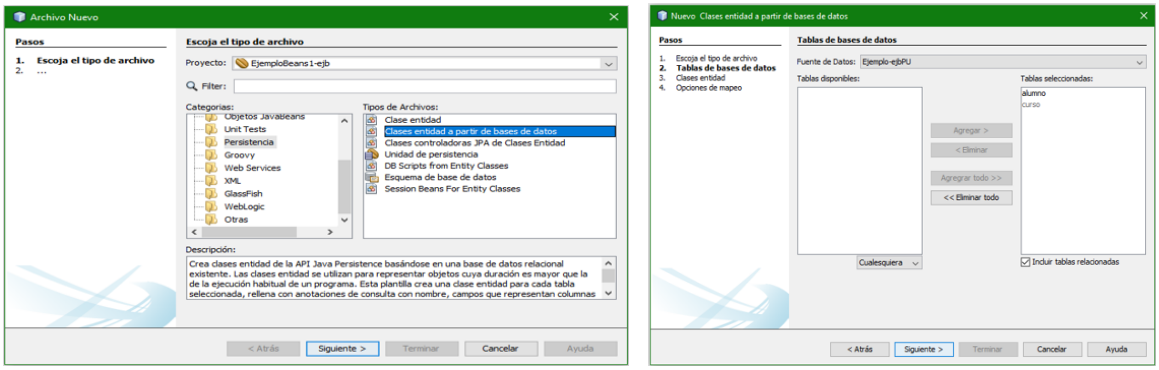
El ***Persistence Provider* o proveedor de persistencia**, se puede trabajar con **eclipselink**[12] o con **hibernate**[13]. mientras que la ***Data Source* o Fuente de datos**, en la lista desplegable debe aparecer el nombre del **recurso JDBC** que se creó en la consola del administrador de dominio de GassFish. Se debe seleccionar (no cambiar) para utilizar el **JTA *Java Transaction APIs*** y como estrategía crear (si la tabla para la **clase entidad**[14] no existe, la crea)

6. **Crear las clases entidad (*Entity Classes*)**. Una de las formas más rápidas para crear las clases entidad es a partir de la base de datos. Para ello, seleccionando el proyecto (-ejb) presionar el botón secundario del ratón y seleccionar **Otros** y escoger en categoría **Persistencia** y en tipo de archivos **Clases entidad a partir de base de datos** (ver figura 5.20a). Luego clic en el botón **Siguiente**, se le presentarán las tablas disponibles, es decir, las que pueden ser seleccionadas para trabajar (leer muy bien los mensajes y etiquetas de esta y de todas las ventanas de los asistentes). Seleccionadas las tablas deseadas presiona el botón agregar, le mostrará la ventana de la figura 5.20b.

[12]EclipseLink es una API de código abierto de la Fundación Eclipse para la persistencia.

[13]Hibernate es un Framework para el mapeo objeto-relacional (ORM) para Java y también disponible para .Net con el nombre de NHibernate. Facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación utilizando archivos (XML) o decoraciones en las clases *beans* de las entidades que permiten establecer estas relaciones.

[14]Los *beans* de entidad representan un objeto concreto que tiene existencia en alguna base de datos. Corresponde generalmente a una fila en una tabla de la base de datos.



(a) Tipos de archivos a crear **(b)** Selección de tablas

Figura 5.20: Creación de Entity Classes

El código de las clases entidad es similar en todas. En el listado de código 5.7 muestra parte del código (se debe crear algunos de los métodos gets y sets) de una de las clases a modo de ejemplo (Customer.java).

```
1  package gcgu.ejb.ejemploejb.entities;
2
3  import java.io.Serializable;
4  import java.util.List;
5  import javax.persistence.Basic;
6  import javax.persistence.CascadeType;
7  import javax.persistence.Column;
8  import javax.persistence.Entity;
9  import javax.persistence.Id;
10 import javax.persistence.JoinColumn;
11 import javax.persistence.ManyToOne;
12 import javax.persistence.NamedQueries;
13 import javax.persistence.NamedQuery;
14 import javax.persistence.OneToMany;
15 import javax.persistence.Table;
16 import javax.validation.constraints.NotNull;
17 import javax.validation.constraints.Size;
18
19 /**
20  *
21  * @author Gleinstong
22  */
23 @Entity
24 @Table(name = "CUSTOMER", catalog = "", schema = "APP")
25 @NamedQueries({
26     @NamedQuery(name = "Customer.findAll", query = "SELECT c FROM
    Customer c"),
27     @NamedQuery(name = "Customer.findByCustomerId", query = "SELECT c
    FROM Customer c WHERE c.customerId = :customerId"),
28     @NamedQuery(name = "Customer.findByName", query = "SELECT c FROM
    Customer c WHERE c.name = :name"),
29     @NamedQuery(name = "Customer.findByAddressline1", query = "SELECT
    c FROM Customer c WHERE c.addressline1 = :addressline1"),
```

```
30      @NamedQuery(name = "Customer.findByAddressline2", query = "SELECT
        c FROM Customer c WHERE c.addressline2 = :addressline2"),
31      @NamedQuery(name = "Customer.findByCity", query = "SELECT c FROM
        Customer c WHERE c.city = :city"),
32      @NamedQuery(name = "Customer.findByState", query = "SELECT c FROM
        Customer c WHERE c.state = :state"),
33      @NamedQuery(name = "Customer.findByPhone", query = "SELECT c FROM
        Customer c WHERE c.phone = :phone"),
34      @NamedQuery(name = "Customer.findByFax", query = "SELECT c FROM
        Customer c WHERE c.fax = :fax"),
35      @NamedQuery(name = "Customer.findByEmail", query = "SELECT c FROM
        Customer c WHERE c.email = :email"),
36      @NamedQuery(name = "Customer.findByCreditLimit", query = "SELECT c
         FROM Customer c WHERE c.creditLimit = :creditLimit")})
37  public class Customer implements Serializable {
38
39      private static final long serialVersionUID = 1L;
40      @Id
41      @Basic(optional = false)
42      @NotNull
43      @Column(name = "CUSTOMER_ID", nullable = false)
44      private Integer customerId;
45      @Size(max = 30)
46      @Column(name = "NAME", length = 30)
47      private String name;
48      @Size(max = 30)
49      @Column(name = "ADDRESSLINE1", length = 30)
50      private String addressline1;
51      @Size(max = 30)
52      @Column(name = "ADDRESSLINE2", length = 30)
53      private String addressline2;
54      @Size(max = 25)
55      @Column(name = "CITY", length = 25)
56      private String city;
57      @Size(max = 2)
58      @Column(name = "STATE", length = 2)
59      private String state;
60      // @Pattern(regexp="^\\(?(\\d{3})\\)?[- ]?(\\d{3})[- ]?(\\d{4})$",
         message="Invalid phone/fax format, should be as xxx-xxx-xxxx")//if
         the field contains phone or fax number consider using this
        annotation to enforce field validation
61      @Size(max = 12)
62      @Column(name = "PHONE", length = 12)
63      private String phone;
64      // @Pattern(regexp="^\\(?(\\d{3})\\)?[- ]?(\\d{3})[- ]?(\\d{4})$",
         message="Invalid phone/fax format, should be as xxx-xxx-xxxx")//if
         the field contains phone or fax number consider using this
        annotation to enforce field validation
65      @Size(max = 12)
66      @Column(name = "FAX", length = 12)
67      private String fax;
68      // @Pattern(regexp="[a-z0-9!#$%&'*+/=?^_`{|}~-]+(?:\\.[a-z0-9!#$
        %&'*+/=?^_`{|}~-]+)*@(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\\.)+[a-z0
```

```
    -9](?:[a-z0-9-]*[a-z0-9])?", message="Invalid email")//if the field
     contains email address consider using this annotation to enforce
    field validation
69      @Size(max = 40)
70      @Column(name = "EMAIL", length = 40)
71      private String email;
72      @Column(name = "CREDIT_LIMIT")
73      private Integer creditLimit;
74      @OneToMany(cascade = CascadeType.ALL, mappedBy = "customerId")
75      private List<PurchaseOrder> purchaseOrderList;
76      @JoinColumn(name = "DISCOUNT_CODE", referencedColumnName = "
    DISCOUNT_CODE", nullable = false)
77      @ManyToOne(optional = false)
78      private DiscountCode discountCode;
79      @JoinColumn(name = "ZIP", referencedColumnName = "ZIP_CODE",
    nullable = false)
80      @ManyToOne(optional = false)
81      private MicroMarket zip;
82
83      public Customer() {
84      }
85
86      public Customer(Integer customerId) {
87          this.customerId = customerId;
88      }
89
90
91      public List<PurchaseOrder> getPurchaseOrderList() {
92          return purchaseOrderList;
93      }
94
95      public void setPurchaseOrderList(List<PurchaseOrder>
    purchaseOrderList) {
96          this.purchaseOrderList = purchaseOrderList;
97      }
98
99      public DiscountCode getDiscountCode() {
100         return discountCode;
101     }
102
103     public void setDiscountCode(DiscountCode discountCode) {
104         this.discountCode = discountCode;
105     }
106
107     public MicroMarket getZip() {
108         return zip;
109     }
110
111     public void setZip(MicroMarket zip) {
112         this.zip = zip;
113     }
114
115     @Override
```

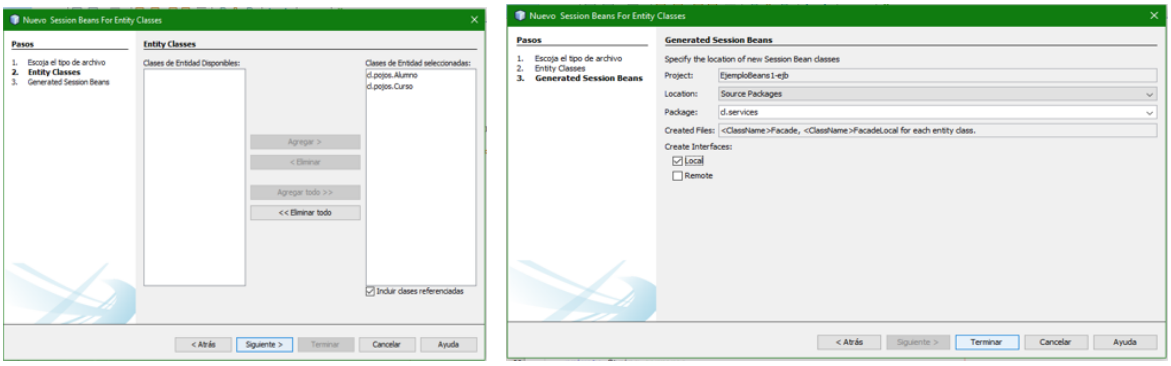
```
116     public int hashCode() {
117         int hash = 0;
118         hash += (customerId != null ? customerId.hashCode() : 0);
119         return hash;
120     }
121 
122     @Override
123     public boolean equals(Object object) {
124         // TODO: Warning - this method will not work in the case the
    id fields are not set
125         if (!(object instanceof Customer)) {
126             return false;
127         }
128         Customer other = (Customer) object;
129         if ((this.customerId == null && other.customerId != null) || (
    this.customerId != null && !this.customerId.equals(other.customerId
    ))) {
130             return false;
131         }
132         return true;
133     }
134 
135     @Override
136     public String toString() {
137         return "gcgu.ejb.ejemploejb.entities.Customer[ customerId=" +
    customerId + " ]";
138     }
139 
140 }
```

Listado 5.7: Código de la clase entidad Customer (**Customer.java**)

7. **Crear los *Session Beans* o *Beans* de Sesión**. Así como se crearon las clases entity, se procede a crear los *beans* de sesión. De la misma categoría, se escoge el tipo de archivos ***Session Beans*** a partir de ***Entity Classes***. Al dar clic en el botón **Siguiente**, mostrará la ventana que permitirá seleccionar las clases entidad disponibles. Seleccionadas estas clases, se da clic en el botón agregar y la ventana quedará como la que se muestra en la figura 5.21a.

   EL asistente le genera una clase Abstracta con el mayor código necesario para las clases específicas. En el listado de código 5.8 muestra la mayor cantidad del código de la clase **AbstractFacade**.

```
1 package gcgu.ejb.ejemploejb.sessionBeans;
2 
3 import java.util.List;
4 import javax.persistence.EntityManager;
5 
6 /**
7  *
8  * @author Gleinstong
9  */
```



(a) Selección de clases *Entity* del proyecto **(b)** Especificación de almacenamiento de las clases

Figura 5.21: Creación de *Session Beans*

```
10 public abstract class AbstractFacade<T> {
11
12     private Class<T> entityClass;
13
14     public AbstractFacade(Class<T> entityClass) {
15         this.entityClass = entityClass;
16     }
17
18     protected abstract EntityManager getEntityManager();
19
20     public void create(T entity) {
21         getEntityManager().persist(entity);
22     }
23
24     public void edit(T entity) {
25         getEntityManager().merge(entity);
26     }
27
28     public void remove(T entity) {
29         getEntityManager().remove(getEntityManager().merge(entity));
30     }
31
32     public T find(Object id) {
33         return getEntityManager().find(entityClass, id);
34     }
35
36     public List<T> findAll() {
37         javax.persistence.criteria.CriteriaQuery cq = getEntityManager
    ().getCriteriaBuilder().createQuery();
38         cq.select(cq.from(entityClass));
39         return getEntityManager().createQuery(cq).getResultList();
40     }
41
42     public List<T> findRange(int[] range) {
43         javax.persistence.criteria.CriteriaQuery cq = getEntityManager
    ().getCriteriaBuilder().createQuery();
```

```
44          cq.select(cq.from(entityClass));
45          javax.persistence.Query q = getEntityManager().createQuery(cq)
        ;
46          q.setMaxResults(range[1] - range[0] + 1);
47          q.setFirstResult(range[0]);
48          return q.getResultList();
49      }
50
51      public int count() {
52          javax.persistence.criteria.CriteriaQuery cq = getEntityManager
        ().getCriteriaBuilder().createQuery();
53          javax.persistence.criteria.Root<T> rt = cq.from(entityClass);
54          cq.select(getEntityManager().getCriteriaBuilder().count(rt));
55          javax.persistence.Query q = getEntityManager().createQuery(cq)
        ;
56          return ((Long) q.getSingleResult()).intValue();
57      }
58  }
```

Listado 5.8: Código de la clase AbstractFacade **AbstractFacade.java**

Las clases fachada (*Facade*) derivadas de la clase abstracta `AbstractFacade` se reduce a agregar un dato `EntityManager` con la decoración `@PersistenceContext` indicando una unidad de persistencia. En Apache Netbeans esa unidad de persistencia esta dentro del ámbito del proyecto. Además, agrega un constructor que hace el llamado al constructor de la clase abstracta como se muestra en el código 5.9. Y la modificación del método `getEntityManager`. Esto es lo que le da la particularidad a cada clase.

```
1   package gcgu.ejb.ejemploejb.sessionBeans;
2
3   import gcgu.ejb.ejemploejb.entities.Customer;
4   import javax.ejb.Stateless;
5   import javax.persistence.EntityManager;
6   import javax.persistence.PersistenceContext;
7
8   /**
9    *
10   * @author Gleinstong
11   */
12  @Stateless
13  public class CustomerFacade extends AbstractFacade<Customer> {
14
15      @PersistenceContext(unitName = "prod")
16      private EntityManager em;
17
18      @Override
19      protected EntityManager getEntityManager() {
20          return em;
21      }
22
23      public CustomerFacade() {
```

```
24          super(Customer.class);
25      }
26  }
```

Listado 5.9: Código de la clase CustomerFacade **CustomerFacade.java**

8. **Crear el controlador**. El controlador se creará en el módulo **war** o de la aplicación web. Se crea una clase java simple (aunque también se pueden crear con asistente), se le agrega las notaciones adecuadas y el código necesario, como consta en la figura 5.10

```
1  package gcgu.ejb.ejemploejb.controllers;
2
3  import gcgu.ejb.ejemploejb.entities.Customer;
4  import gcgu.ejb.ejemploejb.controllers.util.JsfUtil;
5  import gcgu.ejb.ejemploejb.controllers.util.JsfUtil.PersistAction;
6  import gcgu.ejb.ejemploejb.sessionBeans.CustomerFacade;
7
8  import java.io.Serializable;
9  import java.util.List;
10 import java.util.ResourceBundle;
11 import java.util.logging.Level;
12 import java.util.logging.Logger;
13 import javax.ejb.EJB;
14 import javax.ejb.EJBException;
15 import javax.inject.Named;
16 import javax.enterprise.context.SessionScoped;
17 import javax.faces.component.UIComponent;
18 import javax.faces.context.FacesContext;
19 import javax.faces.convert.Converter;
20 import javax.faces.convert.FacesConverter;
21
22 @Named("customerController")
23 @SessionScoped
24 public class CustomerController implements Serializable {
25
26     @EJB
27     private gcgu.ejb.ejemploejb.sessionBeans.CustomerFacade ejbFacade;
28     private List<Customer> items = null;
29     private Customer selected;
30
31     public CustomerController() {
32     }
33
34     public Customer getSelected() {
35         return selected;
36     }
37
38     public void setSelected(Customer selected) {
39         this.selected = selected;
40     }
41
```

```
42      protected void setEmbeddableKeys() {
43      }
44
45      protected void initializeEmbeddableKey() {
46      }
47
48      private CustomerFacade getFacade() {
49          return ejbFacade;
50      }
51
52      public Customer prepareCreate() {
53          selected = new Customer();
54          initializeEmbeddableKey();
55          return selected;
56      }
57
58      public void create() {
59          persist(PersistAction.CREATE, ResourceBundle.getBundle("/
    Bundle").getString("CustomerCreated"));
60          if (!JsfUtil.isValidationFailed()) {
61              items = null;    // Invalidate list of items to trigger re
    -query.
62          }
63      }
64
65      public void update() {
66          persist(PersistAction.UPDATE, ResourceBundle.getBundle("/
    Bundle").getString("CustomerUpdated"));
67      }
68
69      public void destroy() {
70          persist(PersistAction.DELETE, ResourceBundle.getBundle("/
    Bundle").getString("CustomerDeleted"));
71          if (!JsfUtil.isValidationFailed()) {
72              selected = null; // Remove selection
73              items = null;    // Invalidate list of items to trigger re
    -query.
74          }
75      }
76
77      public List<Customer> getItems() {
78          if (items == null) {
79              items = getFacade().findAll();
80          }
81          return items;
82      }
83
84      private void persist(PersistAction persistAction, String
    successMessage) {
85          if (selected != null) {
86              setEmbeddableKeys();
87              try {
88                  if (persistAction != PersistAction.DELETE) {
```

```
89                      getFacade().edit(selected);
90                  } else {
91                      getFacade().remove(selected);
92                  }
93                  JsfUtil.addSuccessMessage(successMessage);
94              } catch (EJBException ex) {
95                  String msg = "";
96                  Throwable cause = ex.getCause();
97                  if (cause != null) {
98                      msg = cause.getLocalizedMessage();
99                  }
100                 if (msg.length() > 0) {
101                     JsfUtil.addErrorMessage(msg);
102                 } else {
103                     JsfUtil.addErrorMessage(ex, ResourceBundle.
    getBundle("/Bundle").getString("PersistenceErrorOccured"));
104                 }
105             } catch (Exception ex) {
106                 Logger.getLogger(this.getClass().getName()).log(Level.
    SEVERE, null, ex);
107                 JsfUtil.addErrorMessage(ex, ResourceBundle.getBundle("
    /Bundle").getString("PersistenceErrorOccured"));
108             }
109         }
110     }
111
112     public Customer getCustomer(java.lang.Integer id) {
113         return getFacade().find(id);
114     }
115
116     public List<Customer> getItemsAvailableSelectMany() {
117         return getFacade().findAll();
118     }
119
120     public List<Customer> getItemsAvailableSelectOne() {
121         return getFacade().findAll();
122     }
123
124     @FacesConverter(forClass = Customer.class)
125     public static class CustomerControllerConverter implements
    Converter {
126
127         @Override
128         public Object getAsObject(FacesContext facesContext,
    UIComponent component, String value) {
129             if (value == null || value.length() == 0) {
130                 return null;
131             }
132             CustomerController controller = (CustomerController)
    facesContext.getApplication().getELResolver().
133                     getValue(facesContext.getELContext(), null, "
    customerController");
134             return controller.getCustomer(getKey(value));
```

```
135         }
136
137         java.lang.Integer getKey(String value) {
138             java.lang.Integer key;
139             key = Integer.valueOf(value);
140             return key;
141         }
142
143         String getStringKey(java.lang.Integer value) {
144             StringBuilder sb = new StringBuilder();
145             sb.append(value);
146             return sb.toString();
147         }
148
149         @Override
150         public String getAsString(FacesContext facesContext,
    UIComponent component, Object object) {
151             if (object == null) {
152                 return null;
153             }
154             if (object instanceof Customer) {
155                 Customer o = (Customer) object;
156                 return getStringKey(o.getCustomerId());
157             } else {
158                 Logger.getLogger(this.getClass().getName()).log(Level.
    SEVERE, "object {0} is of type {1}; expected type: {2}", new Object
    []{object, object.getClass().getName(), Customer.class.getName()});
159                 return null;
160             }
161         }
162     }
163 }
```

Listado 5.10: Código de la clase CustomerController **CustomerController.java**

Los métodos de la clase **ManageBean**, para este ejemplo (y para la mayoría) consisten en llamar a los métodos de la clase **Facade** con la que se esté trabajando, además de las cuestiones de la gestión de la vista. En la listado de código 5.10 se muestra el código de la clase *ManageBean Customer (CustomerController)*.

9. **Crear la interfaz de usuario.** La interfaz de usuario serán las página web Create.xhtml, List.xhtml, Edit.xhtml y View.xhtml de **JavaServer Faces** creadas con la opción de crear páginas *JavaServer Faces* desde las clases *Entity* con su componente **PrimeFaces**. El código XHTML de la página de la vista para agregar un nuevo cliente (Customer) se muestra en el listado de código 5.11.

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http:/
    /www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml"
4       xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
5       xmlns:h="http://xmlns.jcp.org/jsf/html"
```

```
6       xmlns:f="http://xmlns.jcp.org/jsf/core">
7
8     <ui:composition template="/template.xhtml">
9         <ui:define name="title">
10            <h:outputText value="#{bundle.CreateCustomersTitle}"></h:
      outputText>
11        </ui:define>
12        <ui:define name="body">
13            <h:panelGroup id="messagePanel" layout="block">
14                <h:messages errorStyle="color: red" infoStyle="color:
      green" layout="table"/>
15            </h:panelGroup>
16            <h:form>
17                <h:panelGrid columns="2">
18                    <h:outputLabel value="#{bundle.
      CreateCustomersLabel_customerId}" for="customerId" />
19                    <h:inputText id="customerId" value="#{
      customersController.selected.customerId}" title="#{bundle.
      CreateCustomersTitle_customerId}" required="true" requiredMessage
      ="#{bundle.CreateCustomersRequiredMessage_customerId}"/>
20                    <h:outputLabel value="#{bundle.
      CreateCustomersLabel_companyName}" for="companyName" />
21                    <h:inputText id="companyName" value="#{
      customersController.selected.companyName}" title="#{bundle.
      CreateCustomersTitle_companyName}" required="true" requiredMessage
      ="#{bundle.CreateCustomersRequiredMessage_companyName}"/>
22                    <h:outputLabel value="#{bundle.
      CreateCustomersLabel_contactName}" for="contactName" />
23                    <h:inputText id="contactName" value="#{
      customersController.selected.contactName}" title="#{bundle.
      CreateCustomersTitle_contactName}" />
24                    <h:outputLabel value="#{bundle.
      CreateCustomersLabel_contactTitle}" for="contactTitle" />
25                    <h:inputText id="contactTitle" value="#{
      customersController.selected.contactTitle}" title="#{bundle.
      CreateCustomersTitle_contactTitle}" />
26                    <h:outputLabel value="#{bundle.
      CreateCustomersLabel_address}" for="address" />
27                    <h:inputText id="address" value="#{
      customersController.selected.address}" title="#{bundle.
      CreateCustomersTitle_address}" />
28                    <h:outputLabel value="#{bundle.
      CreateCustomersLabel_city}" for="city" />
29                    <h:inputText id="city" value="#{
      customersController.selected.city}" title="#{bundle.
      CreateCustomersTitle_city}" />
30                    <h:outputLabel value="#{bundle.
      CreateCustomersLabel_region}" for="region" />
31                    <h:inputText id="region" value="#{
      customersController.selected.region}" title="#{bundle.
      CreateCustomersTitle_region}" />
32                    <h:outputLabel value="#{bundle.
      CreateCustomersLabel_postalCode}" for="postalCode" />
```

```
33                      <h:inputText id="postalCode" value="#{
    customersController.selected.postalCode}" title="#{bundle.
    CreateCustomersTitle_postalCode}" />
34                      <h:outputLabel value="#{bundle.
    CreateCustomersLabel_country}" for="country" />
35                      <h:inputText id="country" value="#{
    customersController.selected.country}" title="#{bundle.
    CreateCustomersTitle_country}" />
36                      <h:outputLabel value="#{bundle.
    CreateCustomersLabel_phone}" for="phone" />
37                      <h:inputText id="phone" value="#{
    customersController.selected.phone}" title="#{bundle.
    CreateCustomersTitle_phone}" />
38                      <h:outputLabel value="#{bundle.
    CreateCustomersLabel_fax}" for="fax" />
39                      <h:inputText id="fax" value="#{customersController
    .selected.fax}" title="#{bundle.CreateCustomersTitle_fax}" />
40                  </h:panelGrid>
41                  <br />
42                  <h:commandLink action="#{customersController.create}"
    value="#{bundle.CreateCustomersSaveLink}" />
43                  <br />
44                  <br />
45                  <h:commandLink action="#{customersController.
    prepareList}" value="#{bundle.CreateCustomersShowAllLink}"
    immediate="true"/>
46                  <br />
47                  <br />
48                  <h:link outcome="/index" value="#{bundle.
    CreateCustomersIndexLink}"/>
49              </h:form>
50          </ui:define>
51      </ui:composition>
52
53  </html>
```

Listado 5.11: Código de la página *Create.xhtml* para la clase Customer

Y el código de la página para listar los clientes que se encuentran registrados en la base de datos, se muestra en el listado 5.12.

```
1  <?xml version="1.0" encoding="UTF-8" ?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http:/
       /www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml"
4        xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
5        xmlns:h="http://xmlns.jcp.org/jsf/html"
6        xmlns:f="http://xmlns.jcp.org/jsf/core">
7
8      <ui:composition template="/template.xhtml">
9          <ui:define name="title">
10             <h:outputText value="#{bundle.ListCustomersTitle}"></h:
       outputText>
```

```
11          </ui:define>
12          <ui:define name="body">
13              <h:form styleClass="jsfcrud_list_form">
14                  <h:panelGroup id="messagePanel" layout="block">
15                      <h:messages errorStyle="color: red" infoStyle="
    color: green" layout="table"/>
16                  </h:panelGroup>
17                  <h:outputText escape="false" value="#{bundle.
    ListCustomersEmpty}" rendered="#{customersController.items.rowCount
     == 0}"/>
18                  <h:panelGroup rendered="#{customersController.items.
    rowCount > 0}">
19                      <h:outputText value="#{customersController.
    pagination.pageFirstItem + 1}..#{customersController.pagination.
    pageLastItem + 1}/#{customersController.pagination.itemsCount}"/>&
    nbsp;
20                      <h:commandLink action="#{customersController.
    previous}" value="#{bundle.Previous} #{customersController.
    pagination.pageSize}" rendered="#{customersController.pagination.
    hasPreviousPage}"/> 
21                      <h:commandLink action="#{customersController.next
    }" value="#{bundle.Next} #{customersController.pagination.pageSize
    }" rendered="#{customersController.pagination.hasNextPage}"/> 
22                      <h:dataTable value="#{customersController.items}"
    var="item" border="0" cellpadding="2" cellspacing="0" rowClasses="
    jsfcrud_odd_row,jsfcrud_even_row" rules="all" style="border:solid 1
    px">
23                          <h:column>
24                              <f:facet name="header">
25                                  <h:outputText value="#{bundle.
    ListCustomersTitle_customerId}"/>
26                              </f:facet>
27                              <h:outputText value="#{item.customerId}"/>
28                          </h:column>
29                          <h:column>
30                              <f:facet name="header">
31                                  <h:outputText value="#{bundle.
    ListCustomersTitle_companyName}"/>
32                              </f:facet>
33                              <h:outputText value="#{item.companyName}"/
    >
34                          </h:column>
35                          <h:column>
36                              <f:facet name="header">
37                                  <h:outputText value="#{bundle.
    ListCustomersTitle_contactName}"/>
38                              </f:facet>
39                              <h:outputText value="#{item.contactName}"/
    >
40                          </h:column>
41                          <h:column>
42                              <f:facet name="header">
43                                  <h:outputText value="#{bundle.
```

```
    ListCustomersTitle_contactTitle}"/>
44                          </f:facet>
45                          <h:outputText value="#{item.contactTitle}"
    />
46                      </h:column>
47                      <h:column>
48                          <f:facet name="header">
49                              <h:outputText value="#{bundle.
    ListCustomersTitle_address}"/>
50                          </f:facet>
51                          <h:outputText value="#{item.address}"/>
52                      </h:column>
53                      <h:column>
54                          <f:facet name="header">
55                              <h:outputText value="#{bundle.
    ListCustomersTitle_city}"/>
56                          </f:facet>
57                          <h:outputText value="#{item.city}"/>
58                      </h:column>
59                      <h:column>
60                          <f:facet name="header">
61                              <h:outputText value="#{bundle.
    ListCustomersTitle_region}"/>
62                          </f:facet>
63                          <h:outputText value="#{item.region}"/>
64                      </h:column>
65                      <h:column>
66                          <f:facet name="header">
67                              <h:outputText value="#{bundle.
    ListCustomersTitle_postalCode}"/>
68                          </f:facet>
69                          <h:outputText value="#{item.postalCode}"/>
70                      </h:column>
71                      <h:column>
72                          <f:facet name="header">
73                              <h:outputText value="#{bundle.
    ListCustomersTitle_country}"/>
74                          </f:facet>
75                          <h:outputText value="#{item.country}"/>
76                      </h:column>
77                      <h:column>
78                          <f:facet name="header">
79                              <h:outputText value="#{bundle.
    ListCustomersTitle_phone}"/>
80                          </f:facet>
81                          <h:outputText value="#{item.phone}"/>
82                      </h:column>
83                      <h:column>
84                          <f:facet name="header">
85                              <h:outputText value="#{bundle.
    ListCustomersTitle_fax}"/>
86                          </f:facet>
87                          <h:outputText value="#{item.fax}"/>
```

```
88                          </h:column>
89                          <h:column>
90                              <f:facet name="header">
91                                  <h:outputText value=" "/>
92                              </f:facet>
93                              <h:commandLink action="#{
    customersController.prepareView}" value="#{bundle.
    ListCustomersViewLink}"/>
94                              <h:outputText value=" "/>
95                              <h:commandLink action="#{
    customersController.prepareEdit}" value="#{bundle.
    ListCustomersEditLink}"/>
96                              <h:outputText value=" "/>
97                              <h:commandLink action="#{
    customersController.destroy}" value="#{bundle.
    ListCustomersDestroyLink}"/>
98                          </h:column>
99                      </h:dataTable>
100                 </h:panelGroup>
101                 <br />
102                 <h:commandLink action="#{customersController.
    prepareCreate}" value="#{bundle.ListCustomersCreateLink}"/>
103                 <br />
104                 <br />
105                 <h:link outcome="/index" value="#{bundle.
    ListCustomersIndexLink}"/>
106             </h:form>
107         </ui:define>
108     </ui:composition>
109
110 </html>
```

Listado 5.12: Código de la página *List.xhtml* para la clase Customer

10. **Resultados.** La figura 5.23 se muestra el resultado de la ejecución de la página **List.xhtml** (de `Customer`) para crear un nuevo cliente, y en la figura 5.22 se muestra el resultado de la ejecución de la página **Create.xhtml** (de `Customer`).

Ejercicios con un grado de complejidad e importancia mayor desarrollados se puede encontrar en github, entre ellos por ejemplo:

- https://github.com/javaee/javax.ejb,
- https://github.com/WASdev/sample.ejb.secure,
- https://github.com/cicsdev/cics-java-liberty-ejb,
- entre otros.

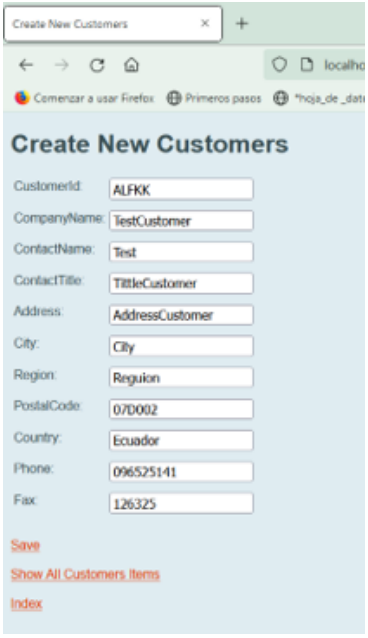


Figura 5.22: Página para ingresar los datos de un nuevo cliente (customer/Create.xhtml)

Figura 5.23: Página para listar los clientes (customer/List.xhtml)

5.8. Conclusiones

J2EE ha tenido una evolución rápida, por lo que hoy en día es una solución recomendable para desarrollar aplicación empresarial que requieran escalabilidad, disponibilidad, consistencia, seguridad, entre otras necesidades. La cantidad y calidad de software vinculado a J2EE ha aumentado gradualmente, de la misma manera ha aumentado el número de empresas y sitios que lo utilizan y ha aumentado la demanda de mercado por el desarrollo J2EE.

EJB es el componente que maneja la lógica de negocios dentro de la arquitectura J2EE por lo cual tiene gran importancia dentro de la misma. Aunque no es tan fácil su diseño, ya que requiere una base de conocimientos, lo cual es motivo del incremento en

su costo de desarrollo, su uso debe ser justificado para realizar una inversión de ese tipo.

Los *Enterprise JavaBeans* presentan una forma de construir componentes, así como un medio para hacer que estos componentes existan en un entorno transaccional, seguro y distribuido. Sin embargo, un solo *bean* representa sólo un componente y, en consecuencia, sólo una parte de una aplicación completa. EJB proporciona a los desarrolladores flexibilidad para determinar cómo se debe hacer que estos componentes funcionen juntos. Hay varias maneras de hacer que los EJB funcionen juntos para formar una aplicación empresarial completa. Se puede integrar en cada variedad de arquitectura de aplicación EJB para proporcionar tanto la tecnología que permite estas arquitecturas como las características que les añaden valor.

Evaluación a los lectores

El lector para demostrar la adquisición y desarrollo de los nuevos conocimientos y habilidades, debe resolver los problemas planteados:

- Dibuje un cuadro sinóptico con los tipos de EJB conjuntamente con sus características, ventajas y desventajas.
- Desarrollar una aplicación CRUD con EJB para el acceso a datos a un motor de base de datos de software libre.

Capítulo 6

SERVICIOS WEB O *WEB SERVICES* (WS)

Objetivos

Al finalizar la lectura y la práctica del contenido de está unidad, el lector será capaz de:

- Identificar y aprovechar las ventajas y beneficios de la arquitectura orientada a servicios (servicios web) en el desarrollo de aplicaciones.
- Explicar el proceso que se lleva a cabo para el desarrollo, publicación y consumo de un servicio web.
- Diferenciar entre los servicios web REST y SOAP.
- Emplear servicios web REST y SOAP en el desarrollo de aplicaciones web.

Resumen

Uno de los problemas que enfrenta el desarrollo de software hoy en día es la heterogeneidad de las aplicaciones con las que se tiene el software que se desarrolla. Una de las soluciones es desarrollar software orientado a servicios. El desarrollo orientado a servicio, apoya el desarrollo de aplicaciones escalables y por supuesto al la reutilización de software. Por lo general, los servicio web se pueden encontrar a disposición como un componente accesible por medio del Protocolo HTTP y por medio del Protocolo SOAP. Las actividades o etapas del ciclo de desarrollo de WS no son diferentes a las etapas de desarrollo de software tradicional (ver figura 6.6), se debe enfatizar que la manera de abordar las etapas de diseño (3) y, codificación y pruebas. En la práctica, los WS pasan a formar parte de las funciones que trabajan sobre los parámetros que reciben. El nuevo paradigma de sistemas como lo es IoT ha tenido

gran apoyo en los servicios web para el intercambio de información entre software y hardware heterogéneos.

6.1. Introducción

El proceso de desarrollo de software cada vez es requerido en el menor tiempo posible, haciendo que se piense en códigos de software que sean utilizados en más de una aplicación. Una de las soluciones ha sido pensar en un diseño modular, sin embargo, con la demanda actual de aplicaciones no es suficiente. Por lo tanto, se han desarrollados las arquitecturas orientadas a servicios. El desarrollo orientado a servicio, apoya el desarrollo de aplicaciones escalables y por supuesto al la reutilización de software. Además de poder integrar aplicaciones heterogéneas. Las aplicaciones actuales se componen de una serie de componentes o servicios (web) reutilizables, que pueden encontrarse distribuidos a lo largo de una serie de máquinas conectadas en red.

Los Servicios Web nos permitirán distribuir nuestra aplicación a través de Internet, pudiendo una aplicación utilizar los servicios ofrecidos por cualquier servidor conectado a Internet. [190].

Web Services

Son software construido en base a un conjunto de protocolos y estándares, y que se ejecuta en un servidor de aplicaciones. Su objetivo primordial es en intercambio de datos entre aplicaciones sin importar el lenguaje de programación con el que fueron desarrolladas.

En este capítulo, se presenta un enfoque de trabajo de las arquitecturas orientadas a servicios con el desarrollo de WS mediante la implementación de RESTful y SOAP, dando a conocer sus nuevos avances en el mundo actual y evidenciando las facilidades de uso a lo largo del tiempo, y mejorando significativamente a muchas aplicaciones de empresas y organizaciones reconocidas, tales como: Yahoo!, eBay, Twitter, Amazon, Google, Flickr, Banco, Aerolíneas, etc.

6.2. Fundamentos de los Web Services

En la actualidad las empresas han convertido a los WS en un mecanismo importante para dar a conocer los servicios en líneas. Por lo general, los WS se pueden encontrar a disposición como un componente accesible por medio del Protocolo de transferencia de hipertexto con sus abreviaturas HTTP (*Hypertext Transfer Protocol*), y el Protocolo de Acceso a Objetos Simples más conocido con sus iniciales como SOAP (*Simple Object Access Protocol*) [191].

Existe un sinnúmero de herramientas que facilitan el uso de los WS, es por eso que muchas aplicaciones están optando por las comunicaciones a través de la web, siendo

un recurso que ha trascendido en el tiempo. Algunas de las aplicaciones realizan la comunicación mediante interfazs pragmáticas aceptando diferentes tipos de solicitudes, donde podemos encontrar tanto procesos simples como complejos, también existen aplicaciones que tienen la capacidad y garantía de publicar información en la web, por medio de los WS pueden contar con las siguientes ventajas [192]:

- **Seguridad:** Cuenta con *Web Services Security (WS-Security)* (Seguridad de los WS) es un protocolo de comunicación que permite configurar la seguridad de los Web Services para contrarrestar ataques externos. Este protocolo es flexible para acomodar nuevos mecanismos de seguridad y ofrecer mecanismos alternos
- **Despliegue:** Transferencia de las diferentes partes que componen el servicio.
- **Facilidad de uso:** Puede utilizar una herramienta particular o cualquier otro objeto para lograr el objetivo particular.
- **Extensibilidad:** Tiene la capacidad de ampliar un sistema y aumentar el esfuerzo de la aplicación.
- **Estandarización:** Ajuste de las normas para que coincidan con la aplicación.

Características de los Web Services

Cuenta con su protocolo de seguridad, su despliegue lo hace mediante la transferencia de sus diferentes partes, se puede usar cualquier objeto para lograr el objetivo particular, facilita la extensión de las aplicaciones, ajustándose a normas y estándares.

6.3. Procesos de un Servicio Web

Los WS tienen diferentes características y procesos que le ayudan a complementar la comunicación sobre la web, en la figura 6.1, se muestra el flujo de los procesos en un WS, que hacen más útil al momento de interactuar con el usuario [190].

6.4. Arquitectura Orientada a Servicios o *Service-Oriented Architecture* (SOA)

La arquitectura orientada a servicios o SOA (Service Oriented Architecture), se la considera una estructura que tiene la gran capacidad de regular y manejar las capacidades distribuidas en diferentes dominios. Por lo general los dominios se encuentran bajo el control de las diferentes organizaciones, estas tienen la capacidad de proporcionar un medio uniforme para interactuar y utilizar los mecanismos óptimos para obtener los resultados esperados [193].

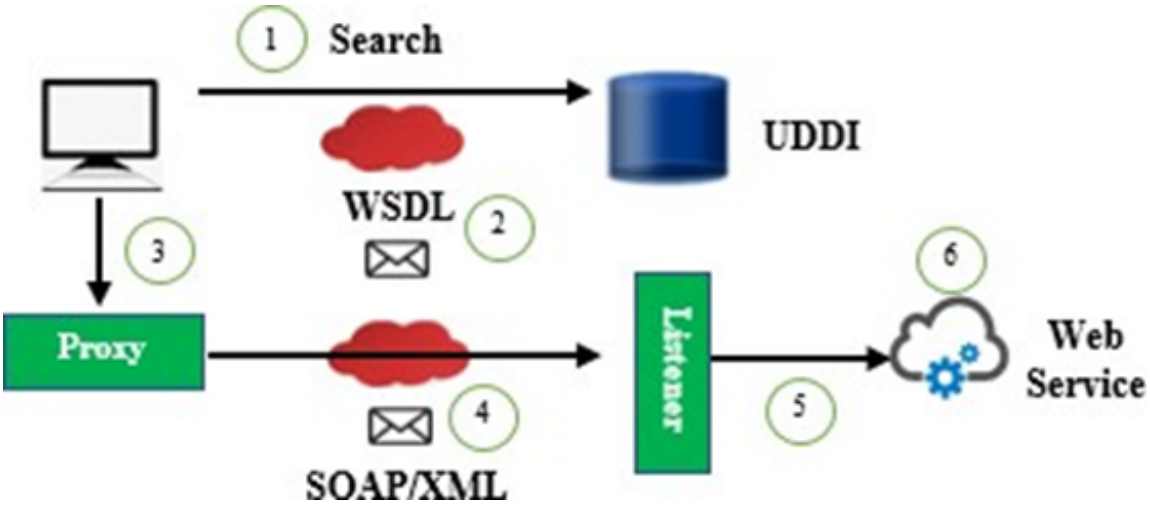


Figura 6.1: Flujo de procesos de un servicio web [192].

SOA es considerada como una arquitectura de software que permite desarrollar, publicar, descubrir y conocer la utilización de servicios para proveer soluciones a los requerimientos que proporcionan los negocios. Además, se conoce a SOA por ser una arquitectura flexible contribuye a la implementación de arquitecturas muy complejas que poseen las diferentes organizaciones. También por medio de la invocación de servicios proporciona un mapa que concede la interacción entre distintos sistemas y servicios externos [194].

De igual manera se conoce que los Web Service fundamentan una forma adecuada de poder realizar la implementación de interfazs de aplicaciones que se basan en servicios, admitiendo que algunas aplicaciones que son realizadas con otras tecnologías sean ejecutadas en algunos entornos que permitan la comunicación e integración, aplicando los estándares o especificaciones vinculados a los Web Services [193, 194]:

- **WSDL o Lenguaje de Descripción de WS (*Web Service Description Language*):** Da a conocer en formato XML las diferentes funcionalidades brindadas por el Web Service. Especifica la ubicación del servicio, las operaciones que posee, los mensajes de intercambio y el formato de invocación de los servicios.

- **SOAP:** Permite el cambio de información estructurada como XML. Protocolo que accede la interacción con WS basado en XML, incorporando una sintaxis para mensajes, reglas y convenciones.

- **UDDI o Descripción, Descubrimiento e Integración Universal (*Universal Description Discovery and Integration*)**: Estándar que ayuda agilizar el registro y descubrimiento de Web Services. Da a conocer un mecanismo donde los proveedores puedan describir sus servicios de forma normal para que los clientes reciban información detallada del servicio que va a requerir.

estándares o especificaciones vinculados a los Web Services

Las aplicaciones que son realizadas con otras tecnologías y que sean ejecutadas en algunos entornos que permitan la comunicación e integración, es gracias a la aplicación de estándares y al ser desarrolladas siguiendo especificaciones: WSDL, SOAP, y UDDI

Para los Web Services es crucial que estos estándares permitan la interacción de servicios instanciando casos particulares de diferentes modelos, tal como se muestra en la figura 6.2 que da a conocer un ejemplo de utilización de los estándares [190]. Al diseñar un sistema distribuidos basándose en el paradigma SOA. También tener como consideración los distintos perfiles tecnológicos que están a disposición para poder encontrar los beneficios que proporciona esta arquitectura [194]:

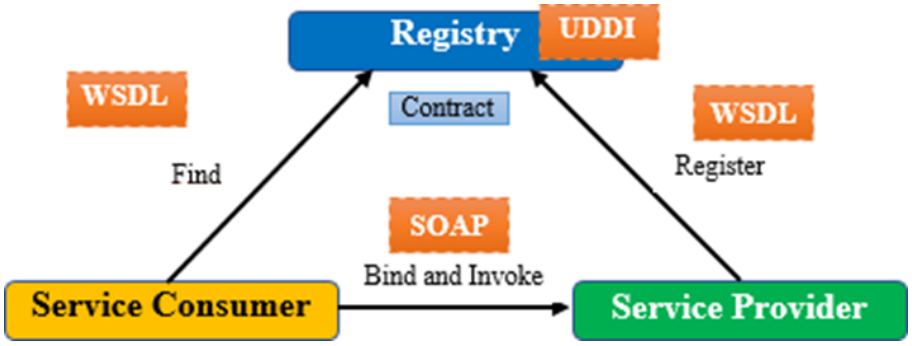


Figura 6.2: Modelo de Interacción de los Web Services [193].

- **Aumentar el tiempo**. Para hacer alteraciones en los procesos.
- **Jerarquía para el desarrollo de lógica de negocios**. Tener la capacidad en base a la subcontratación (*outsourcing*).
- **Modelos de negocio**. Capacidad para coordinar con varias entidades.
- **Aplicación SOA**. Capacidad para cambiar los componentes sin causar trastornos en los procesos de negocio adquiridos.
- **Diversas Tecnologías**. Simplicidad para incorporarlas.

En las arquitecturas orientada a servicios se puede considerar como bloques de construcciones fundamentales a los WS. SOA se visualiza como prototipo arquitectónico y una disciplina para el diseño de software que admite de manera suelta el desarrollo acoplado con el uso de distintos módulos de software [195]. En la figura 6.3 muestra la arquitectura para el consumno de WS por parte del cliente.

A SOA en los últimos tiempos se le ha considerado indispensable para las industrias por tener la disposición para conceder una mayor interoperabilidad entre varios

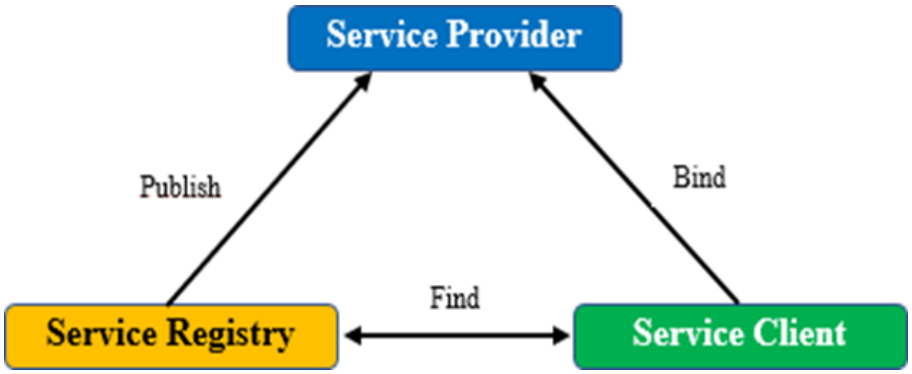


Figura 6.3: Esquema básico de la arquitectura orientada a servicios [195].

sistemas heterogéneos. Además, tiene la capacidad de trabajar y realizar llamadas a procedimientos remotos mejor conocido como RPC (*Remote Procedure Call*) mediante WS SOAP y REST también denominado como Transferencia de Estado Representacional (*Representational State Tranfer*) que es considerado como el estilo de transferencia de estado representacional. También cabe señalar que esta arquitectura respalda la transformación de las diversas operaciones comerciales en un grupo de servicios vinculados que tienen la capacidad de poder entrar a través de Internet [196].

6.4.1. Ciclo de vida de los WS

El ciclo de vida de los WS inicia con la necesidad de satisfacer las necesidades de información de los clientes internos y externos, de llevar a cabo procesos y con la identificación de la solución como el desarrollo de software orientado servicios, e incluso con la necesidad de la administración, y monitorización de estos servicios, como se puede apreciar en la figura 6.4.

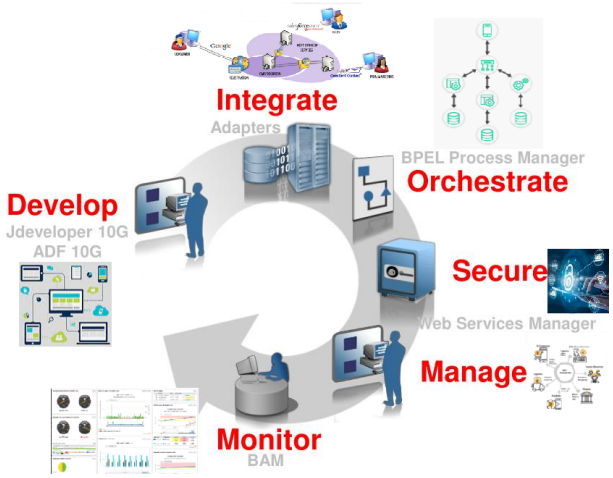


Figura 6.4: Ciclo de vida de los WS.

Se puede simplificar en cuatro etapas: Desarrollo (Diseño, construcción y pruebas),

integración (despliegue/ejecución), gestión y monitorización. La gestión de los WS es una etapa transversal en todo el ciclo de vida de los WS, desde la etapa inicial hasta la fase propia de su gestión, siendo un concepto importante en cada etapa de su ciclo de vida.

Etapas del ciclo de vida de los Web Services

Desarrollo (diseño, construcción y pruebas), integración (despliegue/ejecución), gestión y monitorización.

La Orquestación o composición de WS la llevan a cabo los clientes, al crear sistemas consumidores de WS. Es por eso que algunos autores no la toman como parte del ciclo de vida de los WS. Esta misma filosofía se adopta en este texto, para tratarla como parte de la composición de WS.

Por su parte, la seguridad de los WS, puede ser tratada como parte de la gestión de los WS, por ende debe ser considerada desde la fase de diseño hasta su gestión y monitorización. Uno de los modelos de seguridad de los WS se encuentra el modelo de seguridad que usa seguridad punto a punto como se ilustra en la figura 6.5.

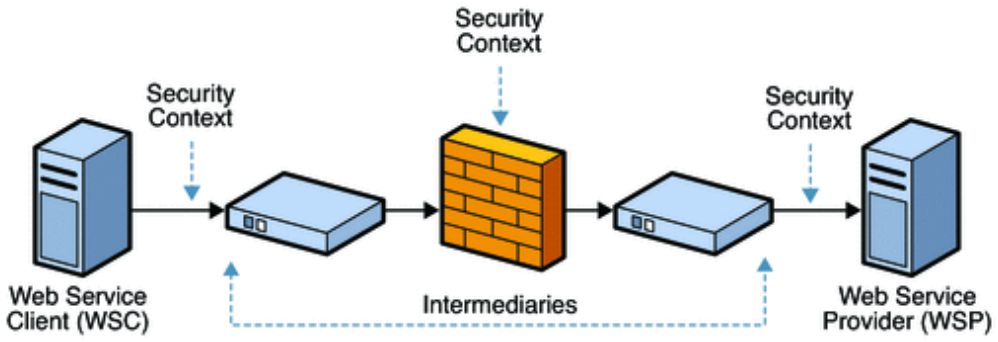


Figura 6.5: Modelo de seguridad punto a punto [197].

La implementación de las seguridades son necesarias tanto del lado del servidor como del lado del cliente. Para abordar el tema de las seguridades en los WS es necesario al menos un capítulo o un libro completo para mayor precisión, es por eso que, en este libro sólo se menciona.

Las **actividades o etapas del ciclo de desarrollo de WS** no son diferentes a las etapas de desarrollo de software tradicional. En la figura 6.6) [198], se amplían (más detalle) las etapas del ciclo de vida de los WS (ver figura 6.4) se debe enfatizar que la manera de abordar las etapas de diseño (3) y, codificación y pruebas (4) son desde el punto de vista de los servicios que un proveedor brinda a sus clientes. En otras palabras, cada servicio debe ser independiente, aunque eso no limita a que, para dar un servicio deba consumir servicios de otros proveedores.

En la práctica, los WS pasan a formar parte de las funciones que trabajan sobre los parámetros que reciben. Sí se ha decidido implementar servicios en en Java, por

Figura 6.6: Ciclo de desarrollo de los WS.

ejemplo. Los WS serían una especie de métodos estáticos. A continuación se describe de manera muy breve cada una de las actividades del proceso de desarrollo de WS hasta su puesta a disposición de los clientes [198]:

Etapas del desarrollo de los Web Services

Construcción, despliegue, depuración y pruebas, publicación.

1. Reúna los **requisitos** del usuario.
2. **Analizar** los componentes comerciales para reutilizarlos o cree un nuevo servicio.
3. **Diseñar** el servicio web o los WS.
4. **Codificación y pruebas** de los WS implementando lógica de negocios con el uso de interfazs y clases. La interfaz es donde la se expondrán los servicio para el consumo, y la clase es la implementación real de los servicios derivados de los componentes de software.
5. **Construir** los WS. Mientras en la codificación y pruebas se da la funcionalidad al WS, en la construcción se lleva a cabo se le da la característica de WS como un componente.
6. **Despliegue** de los WS en el servidor web de destino según el script de implementación (que es específico del servidor).
7. **Pruebas y depuración** de los WS usando el cliente de servicio web (donde el cliente es específico del servidor).

8. **Publicación** de los WS si se requiere la publicación en el registro de servicios.

6.4.2. Seguridad en los WS

Se pueden aplicar algunas medidas para asegurar que la información enviada por el proveedor inicial del servicio y el cliente final del servicio a través de una cadena de intermediarios (proveedores) llegue sin contratiempos respecto a su integridad y privacidad. Las diferentes medidas de seguridad deben considerarse desde el diseño e implementación de los WS. Las medidas de seguridad se pueden aplicar a través de [198]:

- **Seguridad a nivel de transporte (TLS)**. Al aplicar esta medida de seguridad, la seguridad está en el transporte. Sólo el servidor (proveedor) que inicia el servicio web, y el cliente que inicia la solicitud del servicio están protegidos. Los nodos intermedios o servidores (peticionarios intermedios) no están protegidos. Los dos medios más comunes son la capa de conexión segura (SSL) o HTTPS.

- **Seguridad a nivel de mensaje (MLS)**. Aplicando estas medidas, el mensaje se asegura el mensaje en toda su ruta. Es necesario aplicar estándares como XML Encryption0, XML Signature0, XML Key Management0, WS-Security0, SAML, etc. para asegurar el mensaje XML y

- **Seguridad a nivel de infraestructura**. Para aprovechar mejor los servicios que las plataformas de alojamiento de WS. Estas plataformas proporcionan mecanismos de seguridad para que sean aprovechados por quienes utilizan sus servicios.

Niveles de seguridad

Seguridad a niveles de transporte, mensajes, e infraestructura.

6.5. Metodología SOA

La metodología y el *framework* que presenta SOA permite describir transparentemente las lógicas de negocio, entregando una integración y consolidación de las actividades requeridas. Por lo tanto, se debe considerar los aspectos que se muestran en la figura 6.7 [194].

Las fases de la metodología de desarrollo de software basada en SOA, se pueden visualizar en la figura 6.6, incluyendo ahora, en las fases respectivas, el diseño y codificación de la lógica de interacción del usuario final y el sistema. Además, en la fase de construcción la orquestación (composición) de servicios web.

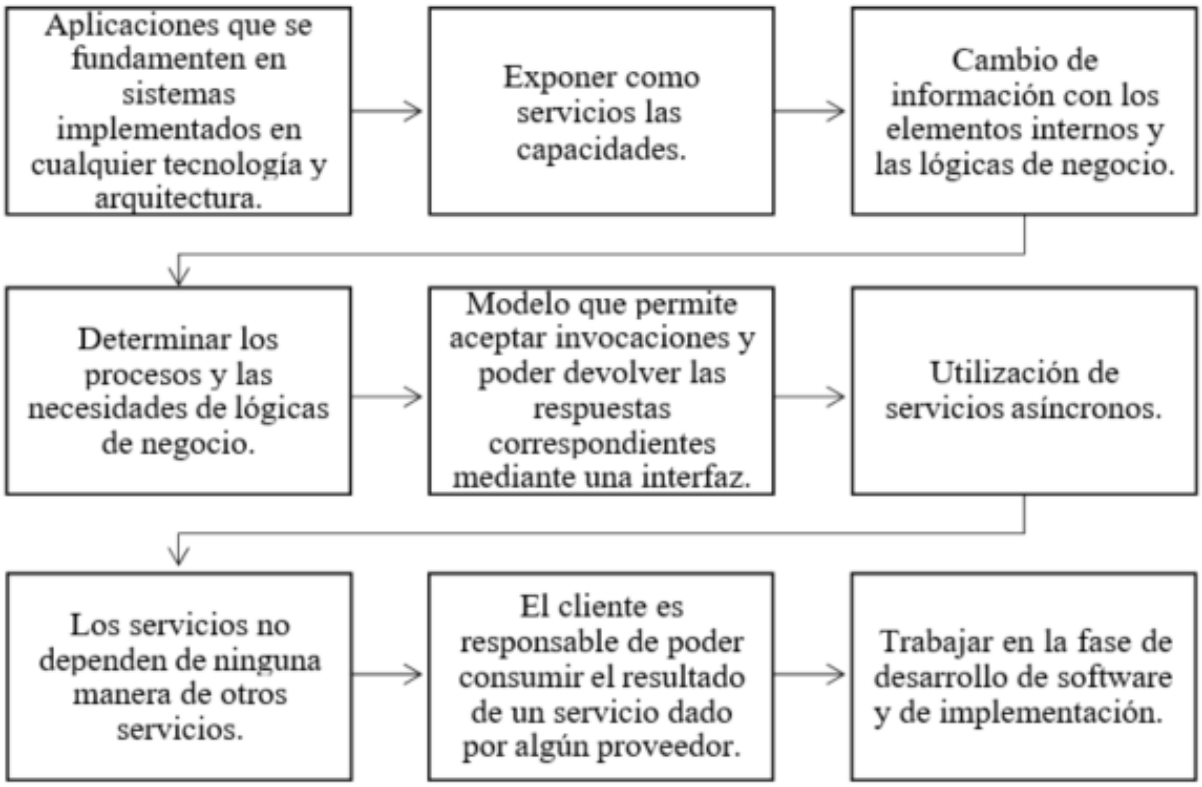


Figura 6.7: Principios y actividades de la metodología SOA

6.5.1. Ciclo de vida de la composición de servicios

En la figura 6.8, el ciclo de vida de la composición de servicios se muestra completo, partiendo desde la creación misma (lógica de negocios de los WS) de los WS (proveedor) hasta el consumo de ellos (solicitante).

Un orquestador de servicios, puede construir los sistemas que cubran las necesidades de procesamiento e intercambio de información, utilizando los WS que estén a su disposición, para ello el constructor de los WS debe describirlos y publicarlos (WSDL y UDDI), el orquestador (cliente) debe seleccionar los WS según sus objetivos, posteriormente debe construir sus sistemas compuestos por los WS. Estos sistema resultados de la composición de WS, deben ser probados, y una vez probados para asegurar que cubren las expectativas de los usuarios finales, pasan a la fase de ejecución y explotación (producción o uso) [199, 200].

6.5.2. Ventajas de SOA

Las ventajas de SOA están muy estrechamente ligadas con las ventajas de los WS. Entre las ventajas principales de SOA se pueden mencionar las siguientes [201]:

- **Adaptabilidad**: Los WS deben ser implementados en entornos dinámicos, donde nuevos WS pueden conectarse en cualquier momento, los servicios existentes

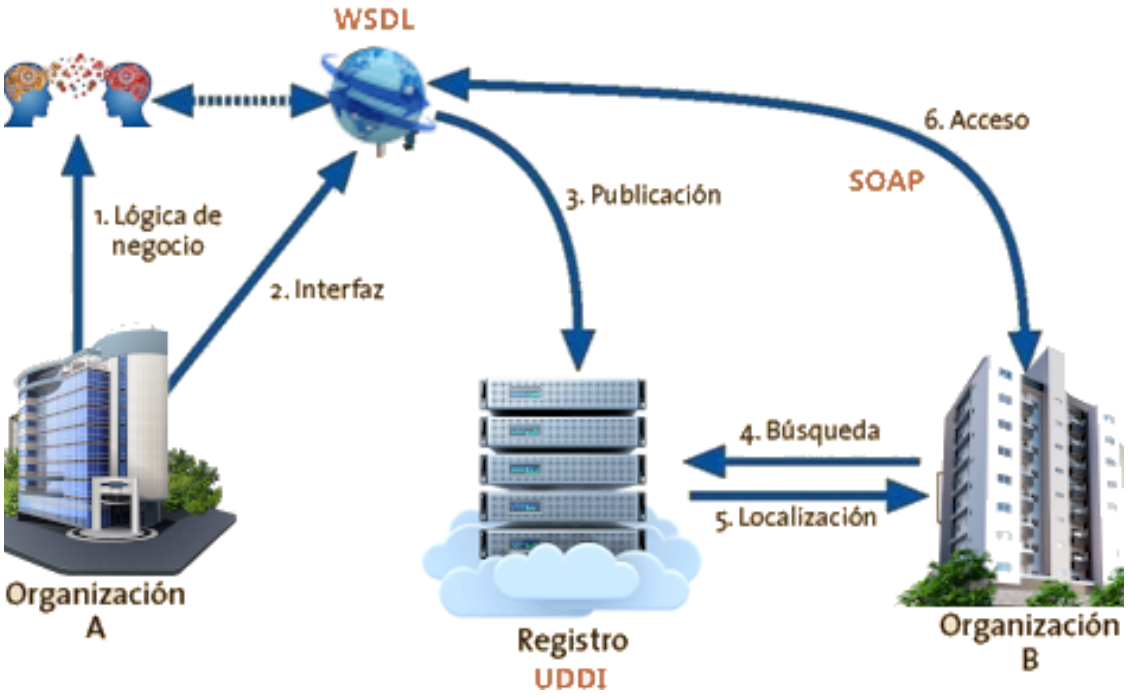


Figura 6.8: Ciclo de vida de la composición de servicios

pueden cambiar su forma de trabajar, o podrían eliminarse o dejar de estar disponibles temporalmente, y el contenido y las capacidades (por ejemplo, atributos de QoS) de los servicios podrían cambiar. En otras palabras, facilita la adaptación al cambio, con la integración con sistemas heredados [201].

- **Escalabilidad**: Fácilmente se pueden añadir nuevos servicios y no necesariamente estarán disponibles en el mismo servidor. Por otro lado no es lo mismo componer dos WS que componer 50 o 100 servicios.
- **Interoperabilidad** : Un sistema puede ser abordado desde el punto de vista de los servicios que va a consumir y cada servicio puede ser implementado en el lenguaje que domine el desarrollador y/o en el lenguaje que sea el más idóneo para la funcionalidad que se va a implementar en el WS. Resumiendo, los servicios web pueden ser implementados en lenguajes de programación diferentes y desplegados en plataformas diferentes. Esto no afecta la composición de los WS, y es transparente al momento de consumirlos.
- **Reutilización de software**: Es sin duda la principal arquitectura que facilita el reuso de software, en vista de que, un buen orquestador de servicios puede crear más de un sistema con los mismos servicios web disponibles.
- **Integración de sistemas externos**: Los sistemas de la organización pueden intercambiar información con sistemas externos (Proveedores, servicios bancarios, por mencionar algunos) de manera rápida y a menor costo.

Ventajas de los Web Services

Adaptabilidad, escalabilidad, interoperabilidad, reutilización de software, e integración de sistemas externos.

6.5.3. Desventajas de SOA

Aunque SOA es muy prominente, las desventajas que se pueden encontrar serían derivadas de las mismas ventajas:

- Para su implementación se requiere un dominio muy alto de los procesos y reglas del negocio.
- Al estar **orientado a estándares**, al momento de implementar sea como cliente o como proveedor, podría ser necesario modificar el código para su correcta implementación.
- El uso de WS es para sistemas que no es alta su **taza de transferencia de datos**.
- También parece importante mencionar que, cuando la organización está frente a este **cambio de paradigma de desarrollo**, puede resultar duro su aceptación.
- La implementación de WS, puede representar un coste adicional de hardware y software, en vista que se requiere un servidor que gestione los procesos del negocios publicados. Adicionalmente a esto, se puede incurrir en costes de mantenimiento y capacitación del persona de las áreas de TI.

Desventajas de los Web Services

Implementación poco compleja, tráfico en la red, cambio de paradigma poco aceptado, costos adicionales de hardware y software, entre otras.
Pueden aparecer más ventajas y desventajas, dependerán de la óptica del lector.

6.6. Ejercicios de aplicación

6.6.1. Gestión de usuarios con Python desde el Back-End

Crear los servicios web en Python y las interfazs para el usuario final en HTML con JavaScript (JS) para la gestión de usuarios (Consultar, agregar, actualizar, y eliminar).

Para resolver el problema planteado, se realizaron las siguientes actividades en el BACK-END (Lógica de negocios o Servidor):

1. Se creó un archivo JSON con los datos que servirán para que el servidor satisfaga las peticiones de los clientes. Su contenido se muestra a continuación:

```
users =
[
  {'id': 1, 'name': 'Arialdo', 'password': '123'},
  {'id': 2, 'name': 'Anbrez', 'password': '456'},
  {'id': 3, 'name': 'Duval', 'password': '789'},
  {'id': 4, 'name': 'Geovanny', 'password': '012'}
]

```

Listado 6.1: Arreglo JSON de usuarios.

2. Utilizando el Framework Flask[1] de Python[2] se crearon las funciones necesarias correspondientes a las peticiones HTTP (Web Services REST) para que sean consumidas (llamadas) desde el cliente. Para no tener inconvenientes tanto para publicar (servidores) como para consumir (clientes) se utilizó. CORS[3]. CORS hace posible que sea indistinto tanto la plataforma servidor donde se publiquen como la plataforma cliente donde se consuman.

```
from flask import Flask
from flask.globals import request
from flask.json import jsonify
from flask_cors import CORS, cross_origin

app = Flask(__name__)
cors = CORS(app)
app.config['CORS_HEADERS'] = 'Content-Type'
app.config['CORS_HEADERS'] = 'Content-Type'
from users import users

@cross_origin()

@app.route('/test')
def test():
    return jsonify({'response': 'exito'})

@app.route('/users', method=['GET'])
def getUsers():
    return jsonify(users)

@app.route('/users/<string:username>', method=['GET'])
def getUser(username):
    retorno = [user for user in users
         if user['username']== username]
    return jsonify(retorno)

@app.route('/users', methods=['POST'])
def addUser():
```

[1]https://flask.palletsprojects.com/en/2.0.x/
[2]https://www.python.org/
[3]https://flask-cors.readthedocs.io/en/latest/

```
    newUser = {
         'userid': request.json['userid'],
         'username': request.json['username'],
         'password': request.json['password']
    }
    users.append(newUser)
    return jsonify(users)

if __name__ == '__main__':
    app.run(debug=True, port=5000)
```

Listado 6.2: Método ver (Servicio Web).

3. Una vez creado el Servicio Web se lanza la aplicación para que el servidor esté listo para escuchar peticiones (en este caso por el puerto 5000). Si todo va bien, debe aparecer una pantalla con el contenido similar al de la figura 6.9.

```
PS C:\Users\Arialdo\Documents\Ingeniería en Sistenas\8vo Semestre\Aplicaciones Distribuidas\Tareas\Expo\NewProyect> & C:/Users/Arialdo
9/python.exe "c:/Users/Arialdo/Documents/Ingeniería en Sistenas/8vo Semestre/Aplicaciones Distribuidas/Tareas/Expo/NewProyect/app.py"
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 143-021-416
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Figura 6.9: Salida al publicar los servicios web

Una vez ya definida la lógica de negocios, que es cómo va a responder el servidor a las peticiones, se procede a realizar las actividades del lado del cliente o la lógica de la interfaz para la interacción con el cliente.

1. Para la interacción con el usuario se crea una página Web en HTML5 y JS para que reciba los datos del Servicio Web y los muestre de una manera simple en un componente h3. El código es el que se muestra en el listado 6.3.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Start Page</title>
        <meta http-equiv="Content-Type" content="text/html; charset=
    UTF-8">
        <!--Script propio del autor -->
        <script src="js/MiJS.js" type="text/javascript">
        </script>
        <!-- Script de la API de JQuery -->
        <script src="js/jquery-3.1.1.js" type="text/javascript">
```

```
11          </script>
12      </head>
13      <!-- Cuerpo de la página -->
14      <body>
15          <h1>Gestión de Usuarios</h1>
16          <input id="txtID" type="input" placeholder="ID"/>
17          <br />
18          <input id="txtName" type="input" placeholder="UserName"/>
19          <br />
20          <input id="txtPass" type="input" placeholder="Password"/>
21           <br />
22          <input type="button" value="Agregar" onclick="agregar()"/>
23          <input type="button" value="Ver" onclick="ver()"/>
24          <input type="button" value="Filtrar" onclick="buscar()"/>
25          <!-- Salida por la solicitud
26               Los resultados erán obtenidos mediante AJAX.
27               AJAX actualizará el cuerpo del encabezado h3 con el id "
    traido"-->
28          <h3 id="traido"></h3>
29      </body>
30  </html>
31
```

Listado 6.3: Código HTML5 (GUI).

2. Lo que queda es crear en un archivo de JS los eventos para realizar los llamados de las funciones del servicio web y se prueba su respuesta. A su vez, para que no se tenga que refrescar toda la página, se utiliza AJAX. El código JacaScript es el siguiente:

```
1  function ver()
2  {
3  $.ajax({
4          method: "GET",
5          dataType: "json",
6          contentType: "application/json;
7          charset=utf-8",
8          url: "http://localhost:5000/users",
9          success: function (data) {
10             document.getElementById("traido").
11             innerHTML=JSON.stringify(data);
12         },
13         error: function (data) {
14             alert("Error");
15         }
16     });
17 }
```

Listado 6.4: Método ver (Servicio Web).

```
1  function buscar()
2  {
```

```
$.ajax({
        method: "GET",
        dataType: "json",
        contentType: "application/json;
        charset=utf-8",
        url:"http://localhost:5000/users/" +
        document.getElementById("txtName").value,
        success: function (data) {
          document.getElementById("traido").
          innerHTML=JSON.stringify(data);
        },
        error: function (data) {
            alert("Error");
        }
    });
}
```

Listado 6.5: Método buscar (Servicio Web).

```

function agregar()
{
    json = {
        "id":
        document.getElementById("txtID").value,
        "name":
        document.getElementById("txtName").value,
        "password":
        document.getElementById("txtPass").value
    };

    $.ajax({
        method: "POST",
        dataType: "json",
        contentType: "application/json;
        charset=utf-8",
        url: "http://localhost:5000/users",
        data: JSON.stringify(json),
        success: function (data) {
          document.getElementById("traido").
          innerHTML=JSON.stringify(data);
        },
        error: function (data) {
            alert("Error");
        }
    });
}

```

Listado 6.6: Método agregar (Servicio Web).

Los resultados de este ejemplo sencillo son los siguientes:

- Para la petición **GET** sin ningún parámetro (método getUsers de Python) se tiene los resultados de todos los registros del archivo como se muestra en la imagen 6.10

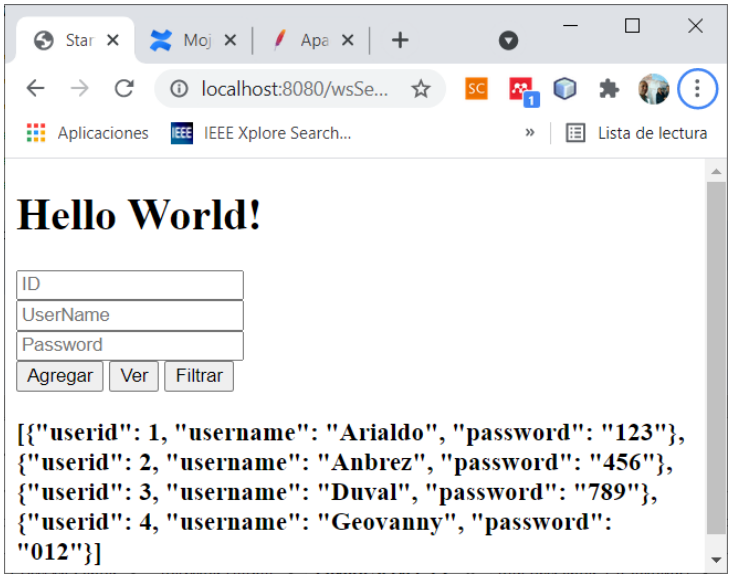


Figura 6.10: Resultado de la petición GET sin Parámetros

- Para la petición **GET** con un parámetro (método getUser de Python) se tiene los resultados de todos los registros del archivo como se muestra en la imagen 6.11

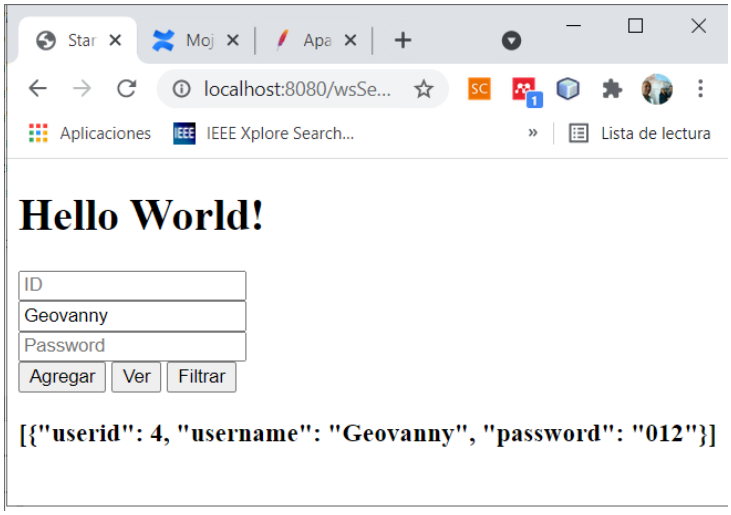


Figura 6.11: Resultado de la petición GET con un Parámetro

- Para la petición **POST** (método addUser de Python), se ha implementado para agregar un recurso (registro) en el servidor. En la figura 6.12 se muestran los resultados de esta petición.

  Las operaciones de actualizar o editar (petición PUT) y de eliminar (DELETE), se deja para la práctica de los lectores.

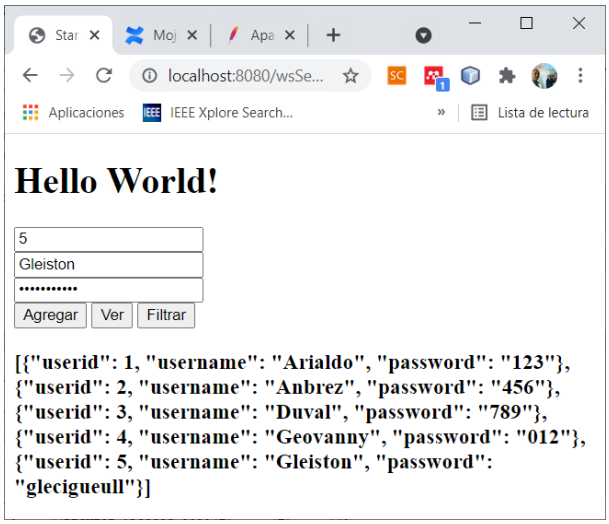


Figura 6.12: Resultado de la petición POST

6.6.2. Servicios web con PHP

En este ejercicio práctico, se va a realizar una aplicación web en el cual, se construya un servicio web en PHP para la gestión de las seguridades en su mínima expresión, de una aplicación web, y el Front-End consuma el servicio web construido.

Para resolver este problema, se utilizará VisualStudio Code (VSC), como se muestra en la figura 6.13

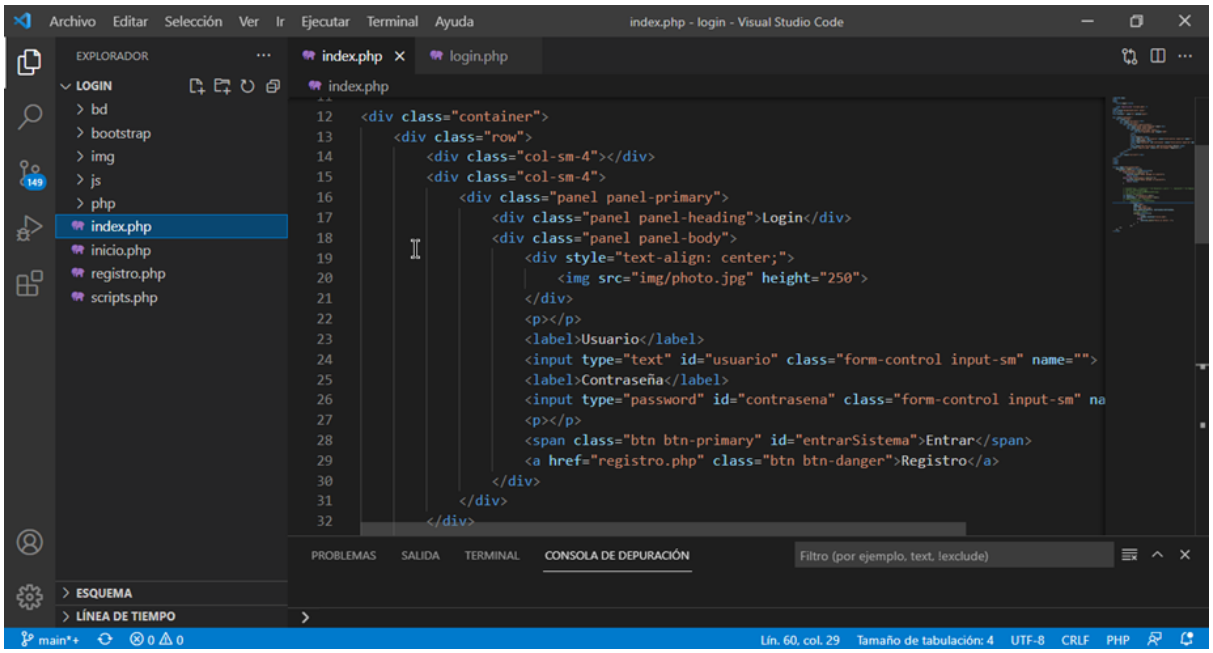


Figura 6.13: Entorno de trabajo de VisualStudio Code

La estructura completa del proyecto se muestra en la figura 6.14.

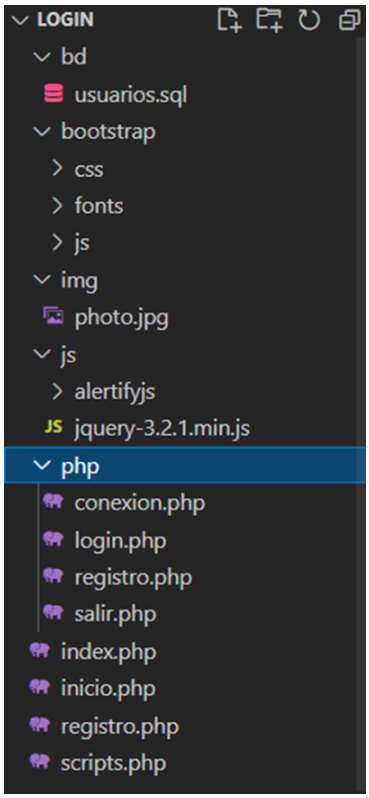


Figura 6.14: Estructura del proyecto en VisualStudio Code

Registro de nuevos usuarios

En el listado de código 6.7 se muestra el código HTML de la interfaz para ingresar los datos de un nuevo usuario para ser registrado en la base de datos del sistema. Y en el listado de código 6.8 la lógica de validación de ingreso de los datos y el llamado a los servicios web que estarán listos para hacer las operaciones respectivas sobre la base de datos.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Registro</title>
5      <?php require_once "scripts.php"; ?>
6  </head>
7  <body style="background-color: gray">
8  <br><br><br>
9  <div class="container">
10   <div class="row">
11     <div class="col-sm-4">
12     </div>
13     <div class="col-sm-4">
14       <div class="panel panel-danger">
15         <div class="panel panel-heading">
16           Registro de usuario
```

```
17          </div>
18          <div class="panel panel-body">
19              <form id="frmRegistro">
20                <label>Nombre</label>
21                <input type="text" class="form-control input-sm" id="nombre"
    name="" />
22                <label>Apellido</label>
23                <input type="text" class="form-control  input-sm" id="
    apellido" name="" />
24                <label>Usuario</label>
25                <input type="text" class="form-control  input-sm" id="usuario
    " name="" />
26                <label>Password</label>
27                <input type="text" class="form-control  input-sm" id="
    password" name="" />
28                <p></p>
29                <span class="btn btn-primary"  id="registrarNuevo"> Registrar
    </span>
30              </form>
31              <div style="text-align: right;">
32                  <a href="index.php" class="btn
33                      btn-default"> Login
34                  </a>
35              </div>
36            </div>
37          </div>
38        </div>
39      <div class="col-sm-4">
40      </div>
41    </div>
42  </div>
43  </body>
44  </html>
45
```

Listado 6.7: GUI para ingreso de datos de nuevos usuarios.

```
1  <script type="text/javascript">
2    $(document).ready(function(){
3      $('#registrarNuevo').click(function(){
4        if($('#nombre').val()==""){
5          alertify.alert("Debes agregar el nombre");
6          return false;
7        }
8        else if($('#apellido').val()==""){
9          alertify.alert("Debes agregar el apellido");
10         return false;
11       }else if($('#usuario').val()==""){
12         alertify.alert("Debes agregar el usuario");
13         return false;
14       }else if($('#password').val()==""){
15         alertify.alert("Debes agregar el password");
16         return false;
```

```
17        }
18        cadena="nombre=" + $('#nombre').val() +
19          "&apellido=" + $('#apellido').val() +
20          "&usuario=" + $('#usuario').val() +
21          "&password=" + $('#password').val();
22        $.ajax({
23          type:"POST",
24          url:"php/registro.php",
25          data:cadena,
26          success:function(r)
27          {
28            if(r==2){
29              alertify.alert("Ese usuario ya existe,
30                 prueba con otro :)");
31            }
32            else if(r==1){
33              $('#frmRegistro')[0].reset();
34              alertify.success("Agregado con exito");
35            }else{
36              alertify.error("Fallo al agregar");
37            }
38          }
39        });
40      });
41    });
42  </script>
```

Listado 6.8: Script parte del archivo intefaz Registro.html (listado 6.7)

La figura 6.15 muestra la respuesta del servidor para el cliente que solicite la página para registrarse como nuevo usuario.

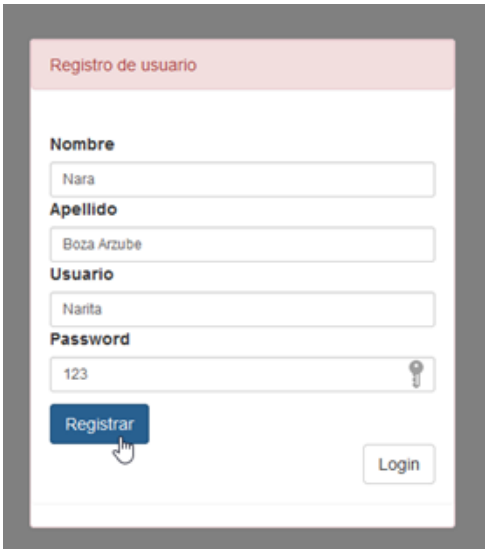


Figura 6.15: Registrando un nuevo usuario.

Una vez llenos los campos del formulario como se ve en la figura 6.15 da clic en

el botón azul con leyenda Registrar, y si los datos son correctos y no existe registros coincidentes con ese usuario, le aparecerá la pantalla con el mensaje **Agregado con éxito** como se muestra en la figura 6.16.

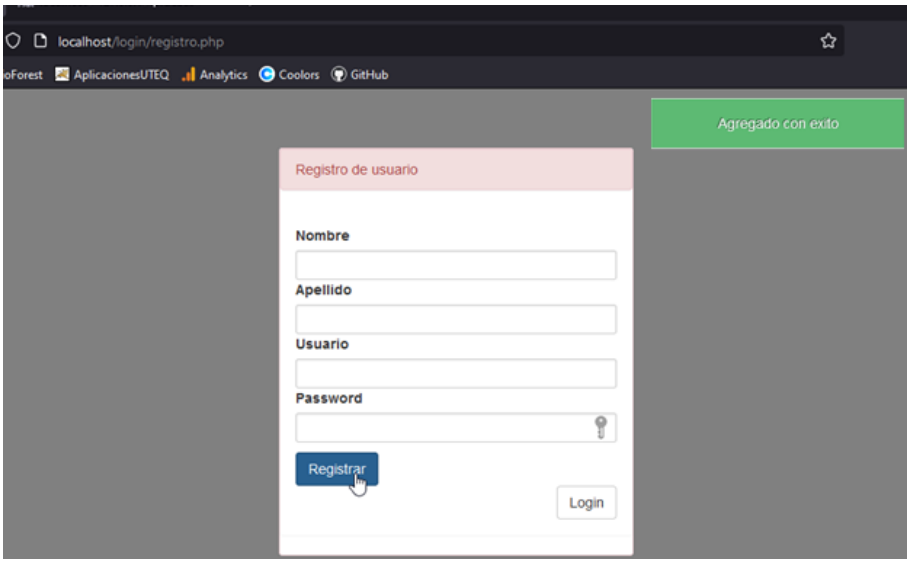


Figura 6.16: Retroalimentación al agregar un nuevo usuario

Para agregar ese nuevo registro a la base de datos se utiliza el servicio web hecho en PHP.

Inicio de sesión

El código HTML, y el código JS (y AJAX) que hace uso de las funciones de PHP para la página de inicio de sesión es el que se muestra en el listado de código 6.9.

```
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
    <?php require_once "scripts.php"; ?>
</head>
<body style="background-color: gray">
<br><br><br>
<form action="" name="f1" method="post">
<div class="container">
  <div class="row">
    <div class="col-sm-4"></div>
      <div class="col-sm-4">
        <div class="panel panel-primary">
          <div class="panel panel-heading">Login</div>
            <div class="panel panel-body">
              <div style="text-align: center;">
                <img src="img/photo.jpg" height="250">
```

```
19                </div>
20                <p></p>
21                <label>Usuario</label>
22                <input type="text" id="usuario"
23                  class="form-control input-sm" name="">
24                <label>Contraseña</label>
25                <input type="password" id="contrasena"
26                  class="form-control input-sm" name="">
27                <p></p>
28                <span class="btn btn-primary"
29                  id="entrarSistema">Entrar</span>
30                <a href="registro.php" class="btn btn-danger">
31                  Registro
32                </a>
33              </div>
34            </div>
35          </div>
36        <div class="col-sm-4"></div>
37      </div>
38  </div>
39  </form>
40  </body>
41  </html>
```

Listado 6.9: GUI para inicio de sesión.

El resultado del diseño de la página de inicio de sesión será algo similar a la captura mostrada en la figura 6.17. Para darle alguna funcionalidad (como la validación de los datos ingresados) a esta página, se debe agregar a su contenido del archivo (código HTML) la programación necesaria, para ello se ha utilizado JQuery (JavaScript listado de código 6.10.)

Figura 6.17: Página de inicio de sesión

```
<script type="text/javascript">
    $(document).ready(function(){
        $('#entrarSistema').click(function(){
            if($('#usuario').val()==""){
                alertify.alert("El usuario es obligatorio");
                return false;
            }
            else if($('#contrasena').val()==""){
                alertify.alert("El password es obligatorio");
                return false;
            }

            var Usuario =  $('#usuario').val();
            var Contrasena =  $('#contrasena').val();
            $.ajax({
                 type:"POST",
                 dataType:'json',
                 url: "php/login.php",
                 data:{Usuario:Usuario, Contrasena:Contrasena
            },
            success:function(r){
            if(r==1){
              window.location="inicio.php";
            }
            else{
              alertify.alert("Fallo al entrar :(");
            }
        }
    });
  });
});
</script>
```

Listado 6.10: Código JavaScript para validar los datos ingresados

La publicación de la página de inicio de sesión en un servidor de aplicaciónes (renderizado del código HTML, con su respectivo CSS y JS), da como resultado la pantalla en la cual iniciará sesión el usuario, tal como se muestra en la figura 6.17.

Si el navegante ingresa un usuario y clave correctos la aplicación le mostrará la página de inicio, indicando que ha ingresado correctamente.

Página de inicio

La página de inicio simplemente es para asegurarse que los datos del usuario que inició sesión fueron almacenados correctamente en la sesión de trabajo del servidor de aplicaciones, para ser reconocido como el usuario que se autenticó en la aplicación, como se muestra en la figura 6.18. El código HTML y PHP de la página de inicio se muestra en el listado 6.11.

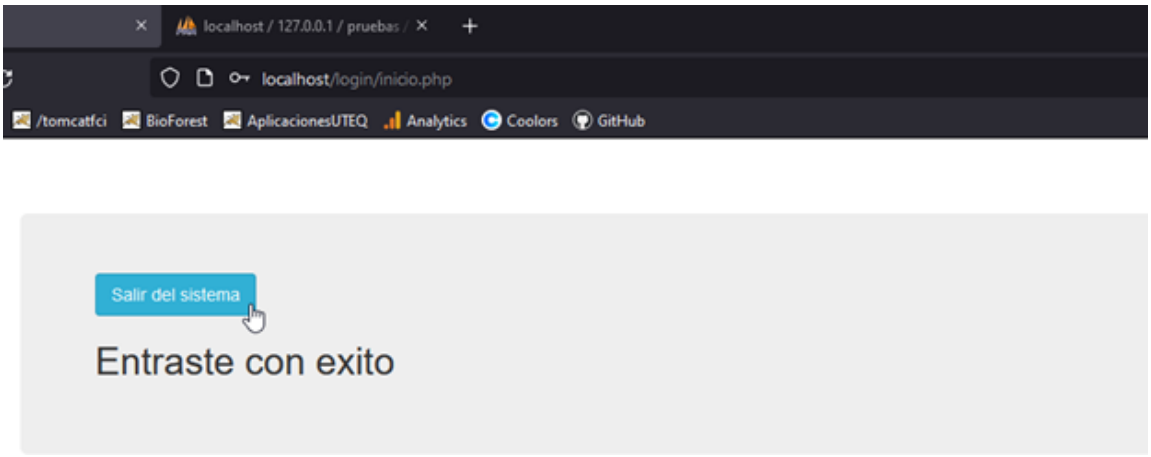


Figura 6.18: Página usuario válido

```php
<?php
  session_start();
  if(isset($_SESSION['user']))
  {
?>
    <!DOCTYPE html>
    <html>
    <head>
      <title>Inicio</title>
      <?php require_once "scripts.php"; ?>
    </head>
    <body>
    <br /><br /><br />
      <div class="container">
        <div class="row">
          <div class="col-sm-12">
            <div class="jumbotron">
              <a href="php/salir.php" class="btn btn-info">
                Salir del sistema
              </a>
              <h2>Entraste con exito</h2>
            </div>
          </div>
        </div>
      </div>
    </body>
    </html>
    <?php
  } else {
   header("location:index.php"); }
?>

```

Listado 6.11: GUI página de inicio.

Si no se encuentra el registro que coincida con los datos ingresados, le presenta el mensaje correspondiente y no podrá acceder a la página de inicio, como se muestra en la figura 6.19.

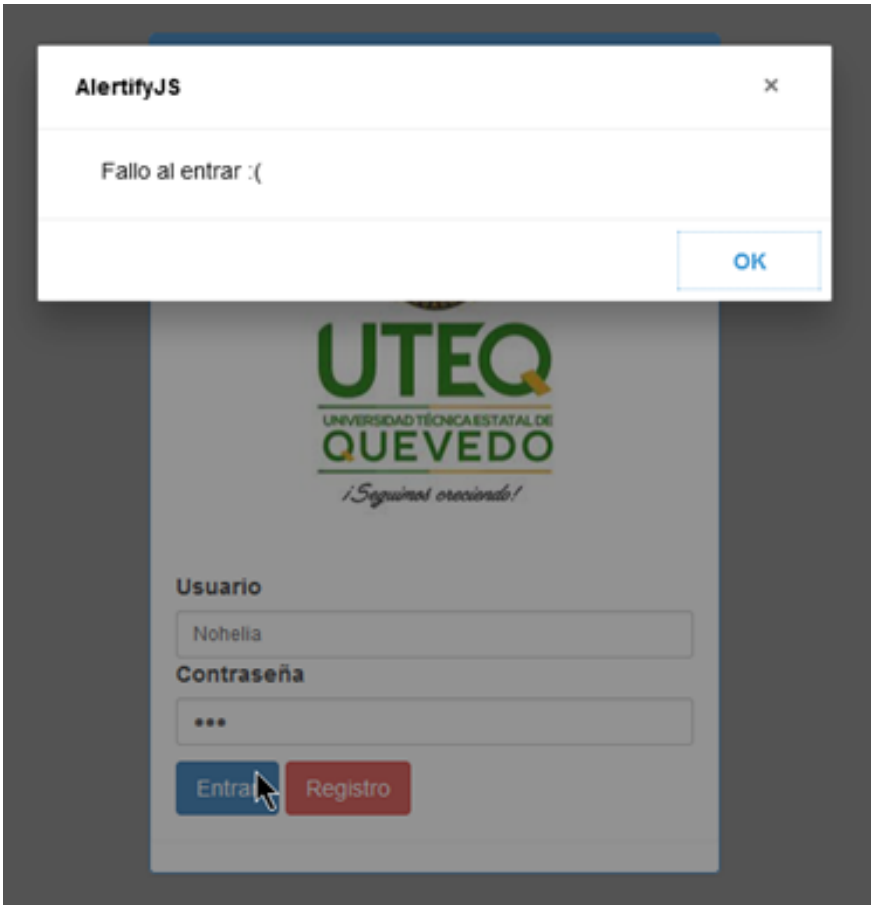


Figura 6.19: Página inicio de sesión con retroalimentación negativa

Archivos JS y CSS

EL código HTML para usar los recursos JavaScript de jQuery y de Alertify, y los CSS de Bootstrap y de Alertify se muestran en el listado de código 6.12.

```
1 <link rel="stylesheet" type="text/css"
2       href="bootstrap/css/bootstrap.css">
3 <link rel="stylesheet" type="text/css"
4       href="js/alertifyjs/css/themes/default.css">
5 <link rel="stylesheet" type="text/css"
6       href="js/alertifyjs/css/alertify.css">
7 <script src="js/jquery-3.2.1.min.js"></script>
8 <script src="js/alertifyjs/alertify.js"></script>
9
```

Listado 6.12: Uso de recursos externos (js y css) en páginas HTML.

BackEnd

Como se mencionó al inicio de la solución de este problema, se utilizó VSC. La estructura del proyecto a lo que respecta el Back-End se muestra en la figura 6.20.

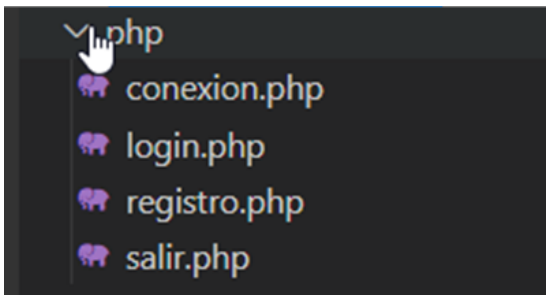


Figura 6.20: Estructura del Back-End del proyecto.

6.6.3. Servicios Web RESTful con Core C# de .Net

En este ejercicio se trabajará con datos de clientes. Se busca crear los WS para la gestión de los clientes.

Front-End

En el entorno de trabajo de Visual Studio .Net, se crea un proyecto, se escoge la plantilla de Aplicación web ASP.Net Core con C# como se muestra en la figura 6.21

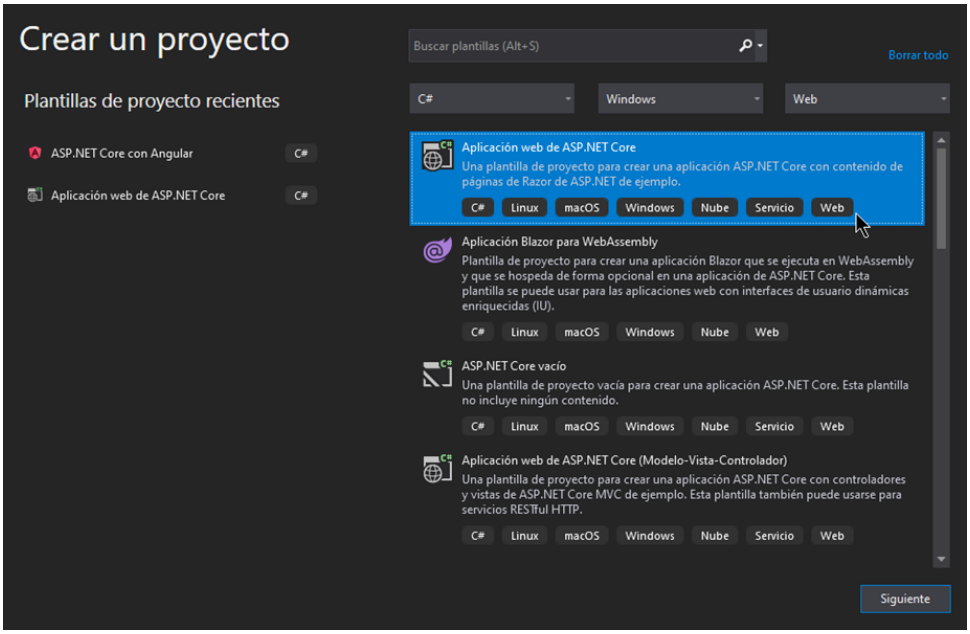


Figura 6.21: Entorno de trabajo de ASP.Net.

Esta aplicación será el Front-End que será quién gestione la vista o interacción con el usuario. En el listado 6.13 se muestra el código de la página **Inicio.html**.

```
<html ng-app="app">
<head>
    <!-- CSS -->
    <link href="lib/bootstrap/dist/css/bootstrap.css"
        rel="stylesheet" />
    <!-- jQuery and JS bundle w/ Popper.js -->
    <script src="lib/jquery/dist/jquery.js"></script>
    <script src="lib/bootstrap/dist/js/bootstrap.js">
    </script>
    <script src="lib/angular.js"></script>
</head>
<body ng-controller="inicioController">
  <div class="container">
    <h2>LISTA DE CLIENTES</h2>
    <div class="row mb-3 mt-3">
      <div class="col-4">
        <button type="button" class="btn btn-primary"
            ng-click="IrFormularioCrear()">
                Crear Nuevo
        </button>
      </div>
    </div>
    <div class="row">
      <div class="col-12">
        <table id="tablesearch" class="table table-striped">
          <thead>
            <tr>
              <!--<th scope="col">#</th>-->
              <th scope="col">Documento Identidad</th>
              <th scope="col">Nombres</th>
              <th scope="col">Telefono</th>
              <th scope="col">Correo</th>
              <th scope="col">Ciudad</th>
              <th>Acciones</th>
            </tr>
          </thead>
          <tbody>
            <tr ng-repeat="datos in result">
              <!--<td>{{datos.IdUsuario}}</td>-->
              <td>{{datos.DocumentoIdentidad}}</td>
              <td>{{datos.Nombres}}</td>
              <td>{{datos.Telefono}}</td>
              <td>{{datos.Correo}}</td>
              <td>{{datos.Ciudad}}</td>
              <td>
                <button id="btnEditar" class="btn btn-success
                btn-sm mr-1 editar" value="{{datos.IdUsuario}}">
                    Editar
                </button>
                <button id="btnEliminar" class="btn btn-danger
```

```
                btn-sm eliminar" value="{{datos.IdUsuario}}">
                    Eliminar
                </button>
            </td>
        <tr>
      </tbody>
    </table>
  </div>
</div>
  <script src="js/site.js"></script>
  <script src="js/Inicio.js"></script>
</body>
</html>

```

Listado 6.13: GUI para mostrar lista de clientes.

Cuando el usuario solicite la página Inicio.html, el servidor le dará como resultado el contenido renderizado como se muestra en la figura 6.22.

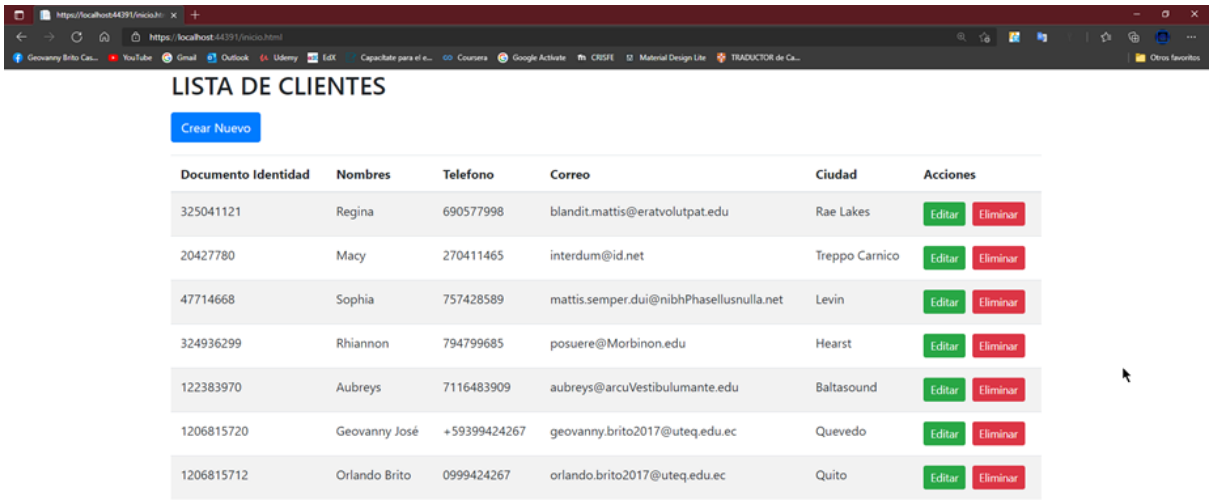


Figura 6.22: Captura de la página Inicio.html.

Uno de los archivos JS que se han creado y el más sencillo pero importante para todos los archivo de interfaz de usuario es el Site.js. Su código es el que se muestra en el listado 6.14.

```
//  Documentación en https://docs.microsoft.com/aspnet/core/
    client-side/bundling-and-minification
// Código personalizado de JS.
// Sólo se pondrá Dirección en donde apunta al Back-End (ahora)
var URL = "http://localhost:58683/api/Usuario/";
```

Listado 6.14: JavaScript de Site.js

El código JS que gobierna el comportamiento de la página Inicio.html (Inicio.js), se muestra en el listado de código 6.15:

```
app = angular.module('app', []);
app.controller('inicioController', function ($scope, $http) {
    $scope.result = [];
    window.onload = function () {
    /* Función que obtendrá los datos de la Web API para mostrarlos en
    nuestra página Web */
        Obtener();
    };

    $scope.IrFormularioCrear = function IrFormularioCrear() {
        window.location = "Registro.html";
    }

    function Obtener() {
        $.get(URL).done(function (response) {
            $scope.$apply(function () {
                $scope.result = response;
                console.log($scope.result);
            });
        });
    }
    $('#tablesearch').on('click', '#btnEditar', function() {
        console.log($(this).val());
        window.location = "Registro.html?id=" + $(this).val();
    });
    $('#tablesearch').on('click', '#btnEliminar', function()
    {
        console.log($(this).val());
        $.ajax({
            method: "DELETE",
            url: URL + $(this).val()
        }).done(function (response) {
            console.log(response);
            if (response) {
                Obtener();
            } else {
                alert("Error al eliminar")
            }
        });

    });
});

```

Listado 6.15: JavaScript de Site.js

Otra de las interfazs para la interacción con el usuario que se ha considerado en este ejercicio, es para el registro de nuevos clientes. El código de la página **Registro.html** es el del listado 6.16:

```
<html>
  <head>
        <!-- CSS -->
```

```
4         <link href="lib/bootstrap/dist/css/bootstrap.css"
5               rel="stylesheet" />
6
7         <!-- jQuery and JS bundle w/ Popper.js -->
8         <script src="lib/jquery/dist/jquery.js"></script>
9         <script src="lib/bootstrap/dist/js/bootstrap.js">
10        </script>
11        <script src="lib/angular.js"></script>
12    </head>
13    <body>
14      <div class="container">
15        <h2>REGISTRO DE CLIENTES</h2>
16        <div class="row mt-3">
17          <div class="col-12">
18            <form>
19              <input type="hidden" id="txtidusuario" />
20              <div class="form-group">
21                <label for="exampleInputEmail1">
22                  Documento Identidad
23                </label>
24                <input type="text" class="form-control"
25                  id="txtdocumento" placeholder="">
26              </div>
27              <div class="form-group">
28                  <label for="exampleInputEmail1">
29                      Nombres
30                  </label>
31                  <input type="text" class="form-control"
32                  id="txtnombres" placeholder="" />
33              </div>
34              <div class="form-group">
35                  <label for="exampleInputEmail1">
36                      Telefono
37                  </label>
38                  <input type="text" class="form-control"
39                      id="txttelefono" placeholder="" />
40              </div>
41              <div class="form-group">
42                  <label for="exampleInputEmail1">
43                      Correo
44                  </label>
45                  <input type="text" class="form-control"
46                      id="txtcorreo" placeholder="" />
47              </div>
48              <div class="form-group">
49                  <label for="exampleInputEmail1">
50                      Ciudad
51                  </label>
52                  <input type="text" class="form-control"
53                      id="txtciudad" placeholder="" />
54              </div>
55              <button type="button" class="btn btn-primary"
56                  onclick="GuardarUsuario()">
```

```
57                 Guardar
58             </button>
59             <button type="button" class="btn btn-warning"
60                 onclick="IrFormularioInicio()">
61                 Volver
62             </button>
63           </form>
64         </div>
65       </div>
66     </div>
67     <script src="js/site.js"></script>
68     <script src="js/registro.js"></script>
69   </body>
70 </html>
71
```

Listado 6.16: JavaScript de Site.js

Cuando el servidor responde a la petición del usuario por la página Registro.html, le devolverá una página como la mostrada en la figura 6.23

Figura 6.23: Captura de la página Registro.html.

El código JS que gobierna el comportamiento de la página Registro.html (Registro.js), básicamente el llamado a los WS es el listado de código JavaSrcipt 6.17. En la figura 6.24 se muestra la captura de la ventana cuando se ha registrado correctamente el nuevo cliente. La retroalimentación de la aplicación debe ser analizada de tal manera que no sea aburrida o tediosa para el usuario. Una de las mejores formas es retroalimentar al usuario cuando una acción se ha realizado correctamente es permitirle continuar con la siguiente tarea a efectuar. En este caso la pantalla muestra el formulario listo para ingresar un nuevo elemento.

```
1 var editar = false;
2 window.onload = function () {
3     var id = $.urlParam('id');
```

```
4       console.log(id);
5       if (id != null) {
6           editar = true;
7           $("#txtidusuario").val(id);
8           PintarUsuario(id);
9       }
10  };
11
12  $.urlParam = function (name) {
13      var results = new RegExp('[\?&]' + name +
14          '=([^&#]*)').exec(window.location.href);
15      if (results == null) {
16          return null;
17      }
18      return decodeURI(results[1]) || 0;
19  }
20
21  function IrFormularioInicio() {
22      window.location = "Inicio.html";
23  }
24
25  function PintarUsuario(idUsuario) {
26    $.get(URL + idUsuario)
27      .done(function (response) {
28          console.log(response);
29          $("#txtdocumento").val(response.DocumentoIdentidad),
30            $("#txtnombres").val(response.Nombres),
31            $("#txttelefono").val(response.Telefono),
32            $("#txtcorreo").val(response.Correo),
33            $("#txtciudad").val(response.Ciudad)
34      });
35  }
36
37  function GuardarUsuario() {
38      if (editar) {
39          var data = {
40              IdUsuario: $("#txtidusuario").val(),
41              DocumentoIdentidad: $("#txtdocumento").val(),
42              Nombres: $("#txtnombres").val(),
43              Telefono: $("#txttelefono").val(),
44              Correo: $("#txtcorreo").val(),
45              Ciudad: $("#txtciudad").val()
46          }
47          // Por medio de AJAX ($) actualisamos con el metodo
48          // PUT de REST
49          $.ajax({
50              method: "PUT",
51              url: URL,
52              contentType: 'application/json',
53              data: JSON.stringify(data), // access in body
54          })
55              .done(function (response) {
56                  console.log(response);
```

```
57                  if (response) {
58                      alert("Se guardaron los cambios");
59                      window.location = "Inicio.html";
60                  } else {
61                      alert("Error al Modificar")
62                  }
63              });
64
65      } else {
66
67          var data = {
68              DocumentoIdentidad: $("#txtdocumento").val(),
69              Nombres: $("#txtnombres").val(),
70              Telefono: $("#txttelefono").val(),
71              Correo: $("#txtcorreo").val(),
72              Ciudad: $("#txtciudad").val()
73          }
74          // Por medio de AJAX ($) guardamos con el metodo
75          // post de REST
76          $.post(URL, data).done(function (response) {
77              console.log(response);
78              if (response) {
79                  alert("Usuario Creado");
80                  window.location = "Inicio.html";
81              } else {
82                  alert("Error al crear");
83              }
84          });
85      }
86  }
87
```

Listado 6.17: JavaScript de Site.js

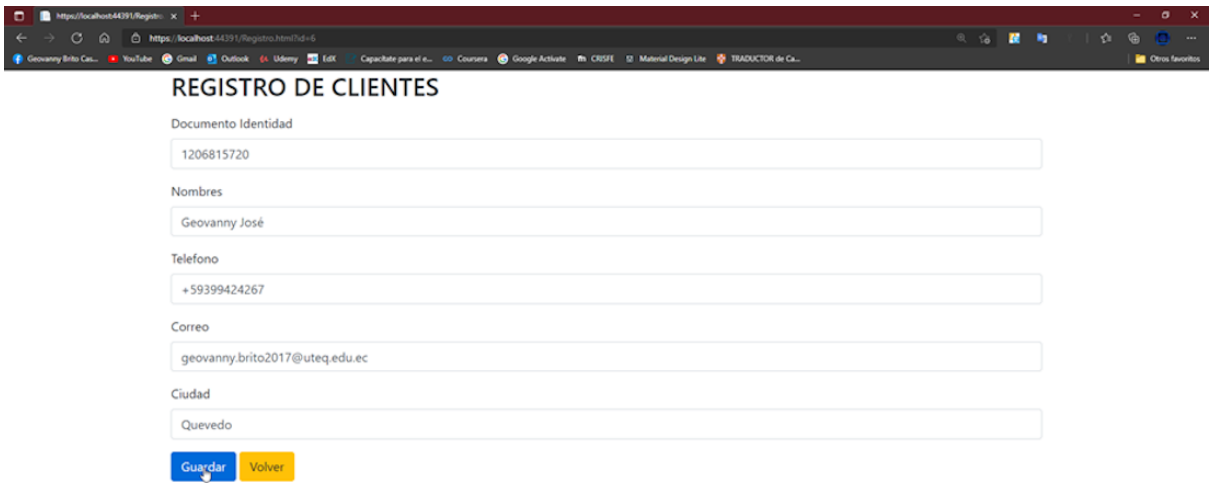


Figura 6.24: Captura Retroalimentación al Registrar un nuevo cliente.

6.7. Conclusiones

Los servicios web brindan la posibilidad de cara a la interoperatividad de aplicaciones. Resultan una solución interesante, ya que, de una manera muy cómoda se puede reutilizar las funcionalidades (operaciones) de un sistema que pudieron costar mucho esfuerzo desarrollar, en otros sistemas o aplicaciones que necesitan se implemente. Un caso actual, es una aplicación web y una móvil, necesitan consultar los mismos datos y realizar con ellos las mismas operaciones. Por otro lado, existen muchos casos en qué, ciertos procesos u operaciones y la consulta de ciertas informaciones son requeridas por terceros, como por ejemplo, la consulta de inventarios por parte de nuestros proveedores para surtir al almacén de los productos agotados. Las agencias de viaje necesitan conocer los itinerarios de las aerolíneas y la disponibilidad de asientos en sus vuelos. Todos estos ejemplos se ven resueltos simplemente con la construcción de servicios web y ponerlos a disposición de terceros para que los consuman, satisfaciendo así sus necesidades.

Los servicios web, por sus bondades de interoperabilidad están siendo una de las tecnologías fundamentales en el desarrollo de los sistemas basados en IoT, siendo estos sistemas altamente heterogéneos, por lo tanto, con requisitos de altas capacidades de interoperabilidad.

Evaluación a los lectores

La evaluación ofrece posibilidades para fortalecer y consolidar los aprendizajes, así como los logros de los objetivos o propósitos en cualquier campo de estudio. El lector para garantizar la consecución de los objetivos planteados debe ser capaz de responder de manera sencilla y clara los siguientes problemas planteados.

1. Dibujar un cuadro sinóptico con los tipos de tecnologías de servicios web, conjuntamente con sus semejanzas, características, ventajas y desventajas.

2. Escriba un caso de estudio donde se requiera utilizar eb services SOAP y otro caso de estudio donde se requiera web services RESTful.

3. Desarrollo una aplicación web en la que se implemente servicios web SOAP para consumir los datos de una base de datos.

4. Desarrollo una aplicación web en la que se implemente servicios web RESTful para consumir los datos de una base de datos.

5. Desarrollo una aplicación web que consuma los servicios web SOAP y RESTful creados como solución a los problemas 3 y 4.

Capítulo 7

WEBSOCKETS

Objetivos

Al finalizar la lectura y la práctica del contenido de está unidad, el lector será capaz de:

- Explicar las diferencias entre el tráfico habitual HTTP y el que existe con los WebSockets.
- Describir los beneficios de los WebSockets respecto al aprovechamiento de la estructura tecnológica existente.
- Desarrollar aplicaciones de comunicación en tiempo real utilizando WebSockets.
- Valorar las características de los WebSockets frente a la comunicación HTTP tradicional.

Resumen

La tecnología de Sockets se refiere a un concepto abstracto mediante el cual dos procesos intercambian un conjunto de datos de forma organizada y confiable. WebSockets es una tecnología de monitorización en tiempo real eficiente para el acceso asíncrono entre un cliente y un servidor de aplicaciones. El modelo de solicitud-respuesta es la base para la más elemental aplicación web. Sin embargo, esto puede resultar bastante limitado, al requerir que el servidor envíe datos al cliente sin que este los solicite. Esto lo hace posible la tecnología de Sockets y WebSockets. Con AJAX y jQuery se pueden construir aplicaciones web en tiempo real y comunicación punto a punto. AJAX tecnología proporciona solicitudes parciales, en lugar de consultar toda la página. jQuery es una biblioteca multiplataforma de JavaScript, desarrollada para simplificar la interacción con los documentos HTML, manipular los DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

7.1. Introducción

Desde el tiempo que surgió las tecnologías de la Web 2.0 (La Web social) [93], las aplicaciones web se han convertido en una de las principales formas para que los usuarios puedan acceder a la información de Internet. Debido al uso masivo de la tecnología Web en diferentes áreas, la aparición constante de los datos y las altas solicitudes de conexión concurrente se necesita una mayor demanda en las aplicaciones web para la transmisión de datos. La Web 2.0 se encuentra reflejada principalmente en dos aspectos: El primero es la gran cantidad de datos aportados por la alta concurrencia. El segundo aspecto es la transferencia de datos en tiempo real [202].

La comunicación en tiempo real o RTC por sus siglas en inglés *Real-Time Communication*, es un modo de comunicación que permite a todos los usuarios intercambiar información al instante o con un pequeño valor de latencia [203]. En este texto se presentan tres tecnologías push basados en servidors existentes que sirven para la monitorización en tiempo real

La tecnología de Sockets se refiere a un concepto abstracto mediante el cual dos procesos intercambian un conjunto de datos de forma organizada y confiable. El monitor de Socket rastrea ciertas condiciones TCP/IP que ocurren en el Socket, y si se detecta una condición específica a la salida del cliente [204]. Por lo general, está diseñado para aplicaciones de escritorio y es adecuado para escenarios de desarrollo de aplicaciones de monitorización en tiempo real. Sin embargo, esta tecnología no es compatible con los navegadors [205]. Para hacer uso de tecnologías equivalentes a Sockets en los navegadors, éstos deben conectarse a servidorses de aplicaciones TCP/IP. Esta tecnología se denomina WebSocket.

Sockets

Es un concepto abstracto mediante el cual dos procesos intercambian un conjunto de datos de forma organizada y confiable. Sockets se puede implementar en aplicaciones de escritorio.

La especificación del WebSocket es un estándar emergente para el desarrollo Web actual. WebSocket es una tecnología Web que proporciona canales de comunicaciones *full-duplex* a través de una única conexión TCP. WebSocket fue desarrollado como parte de HTML5 con interfaz JavaScript. La conexión *full-duplex* puede enviar mensajes entre el cliente y el servidor. Esto es posible ya que los WebSockets proporcionan una alternativa estandarizada para que el servidor envíe contenido al navegador sin que éste lo solicite y permitiendo que los mensajes pasen del cliente al servidor y del servidor al cliente mientras se mantiene la conexión abierta [206]. WebSocket tiene varias diferencias que generan ventajas respecto al protocolo de petición estándar solicitud-respuesta.

WebSockets

Es una es una tecnología estándar emergente para el desarrollo web, que proporciona canales de comunicaciones *full-duplex* a través de una única conexión TCP.

Las principales diferencias de los WebSockets en comparación con el tráfico habitual de la red a través de HTTP es el protocolo que utilizan [207]:

- El protocolo no sigue la convención tradicional del modelo de solicitud-respuesta.
- La interfaz WebSocket permite la comunicación bidireccional continua entre navegadors y servidors.
- Proporciona mejoras dinámicas sobre la conexión convencional full-duplex sobre HTTP.
- Simplifica la complejidad de la comunicación bidireccional de la Web y la administración de la conexión.
- Una vez que un cliente y un servidor han abierto una conexión WebSocket, ambos puntos finales pueden enviarse datos asíncronamente entre sí.

En este texto se describen varias tecnologías del monitorización en tiempo real antes de usar WebSockets, ventajas de su uso, los beneficios que proporcionan. También se da a conocer la eficiencia del protocolo WebSocket sobre el sondeo HTTP. Se muestran un ejemplo de WebSocket como tecnología de monitorización en tiempo real basada la Web (aplicaciones web), demostrando cómo realiza la sincronización con el servidor desde el cliente, entre otros temas.

7.2. Antecedentes

Con el surgimiento de la Web social se aumentó la cantidad de usuarios conectados simultáneamente a una aplicación web. La cantidad de peticiones que genera el cliente por cada usuario que se encuentra en un sitio Web puede generar conflictos. Un ejemplo puede ser la sobrecarga de peticiones en el servidor. De esta manera, produciría que el servidor no responda ya que este no podría generar una respuesta. Para hacer una petición desde el cliente al servidor se ha utilizado tradicionalmente el modelo Solicitud-Respuesta mediante el protocolo HTTP [208, 209].

7.2.1. Modelo solicitud-respuesta HTTP

El modelo de solicitud-respuesta es la base para la más elemental aplicación web. El cliente envía una solicitud y el servidor da una respuesta según los datos enviados. Basándose en este modelo, se puede desarrollar cualquier arquitectura. Sin embargo,

esto puede resultar bastante limitado, al requerir que el servidor envíe datos al cliente sin que este los solicite, porque el servidor no puede enviar datos sin una solicitud del cliente. Con la satisfacción de este requisito, en 2005, con la ayuda de las tecnologías de la Web social, las aplicaciones web evolucionaron, mejorando la usuabilidad y la eficiencia [210]. Estas tecnologías se describen a continuación.

7.2.2. Sondeo HTTP (HTTP polling)

Debido a que el servidor no puede hacer solicitar conexión con el cliente, éste (el cliente) debe sondear repetidamente al servidor para obtener los cambios en la información. Un método ampliamente utilizado para resolver este problema es el sondeo HTTP. El navegador envía una solicitud al servidor en un periodos de tiempo cortos. Esto puede producir, que, si hay un número considerable de usuarios conectados simultáneamente, el tiempo de respuesta puede exceder el límite establecido [211].

Sondeo corto (short polling)

El sondeo corto implica enviar constantemente solicitudes al servidor, preguntando si hay eventos que deben actualizarse en la vista. Este método se considera ineficiente porque consume mucha sobrecarga cada vez que se desconecta la conexión, después de configurar una nueva conexión. Además que sólo se puede recibir una actualización del cliente por intervalo de detección [212]. Una tecnología que aplica el sondeo corto es AJAX (Asynchronous JavaScript And XML[1]).

Solicitud-Respuesta HTTP y Sondeos

Solicitud-Respuesta HTTP: El cliente envía una solicitud y el servidor da una respuesta según los datos enviados. Sondeo HTTP el cliente se encarga de sondear los cambios en la información. Mientras que Sondeos: El cliente esta atento (hace sondeos) a los cambios que se pueden producir en la información.

AJAX

Esta tecnología proporciona solicitudes parciales, en lugar de consultar toda la página, ya que accede a parte de la página, optimizando así la carga de la red. Pero de igual manera, el cliente aún solicita todas las solicitudes a través de HTTP. Esto quiere decir que sin una solicitud previa, el servidor no podrá enviar una respuesta al cliente [213] (ver listado 7.1).

[1] eXtensive Markup Language: Especificación de W3C como lenguaje de marcado de propósito general. Su principal objetivo es compartir datos a través de diferentes sistemas como Internet

jQuery

jQuery es una biblioteca multiplataforma de JavaScript, desarrollada para simplificar la interacción con los documentos HTML, manipular los DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web [214]. jQuery fue lanzada en 2006 en el BarCamp de New York por John Resig. Su lema es "haga más, escriba menos", en vista de que, las tareas que, escritas en JavaScript nativo tomarían muchas líneas de código, usando jQuery serían escritas de un modo más ágil.

```
$.ajax({
    type: "GET", /* Se declara el método a utilizar en este caso GET */
    url: "localhost", /*Se especifica la dirección del servidor*/
    beforeSend: function (xhr) {
        /* Evento que se ejecuta antes de realizar el
         * envío de datos al servidor Se suele utilizar
         * para realizar validaciones y Enviar los
         * datos al servidor cuando se cumpla una
         * condición */
    },
    sucess: function (data){
        /* Evento que define qué se hará
         * Cuando el servidor haya respondido con un
         * estado de correcto
        */
    },
    error: function (objXMLHttpRequest){
        /* Evento que se hará cuando el servidor
         * Haya respondido con un estado de error!
        */
    }
})
```

Listado 7.1: Código AJAX para llamar al servidor usando jQuery

Solicitud-Respuesta HTTP y Sondeos

El sondeo corto se produce de manera independiente de la presencia o no de una actualización de datos, y para una baja latencia, el intervalo de sondeo debe ser bajo, lo que significa un alto uso de recursos y tráfico en la red. Por estas desventajas, se utiliza más el sondeo largo [215].

Sondeo largo (long polling)

Consiste en realizar una única petición al servidor de forma que éste responde diciendo que va a devolver la respuesta en trozos, como se puede observar en la figura 7.1. La petición queda abierta hasta que el servidor responda con todas las porciones de respuesta que solicita el cliente, que no son otra cosa que eventos que se producen en el servidor notificando un cambio de estado en el cliente [216]. `XMLHttpRequest` es una

tecnología basada en el sondeo largo.

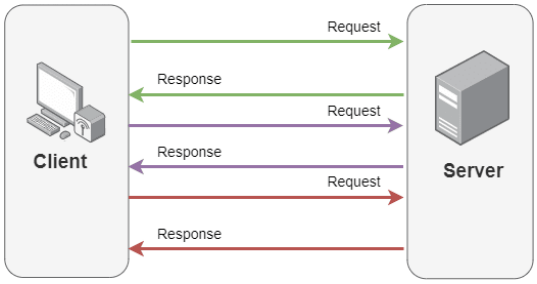


Figura 7.1: Modelo de comunicación del *long Polling* [217]

Transmisión HTTP (Streaming HTTP)

No hay necesidad de enviar solicitudes con frecuencia. Todo el proceso sólo necesita una petición de conexión HTTP. El servidor Web mantiene la conexión abierta y envía periódicamente datos al cliente (ver figura 7.2. Esto puede funcionar en la mayoría de los navegadors excepto en Internet Explorer. La compatibilidad no es muy buena [218].

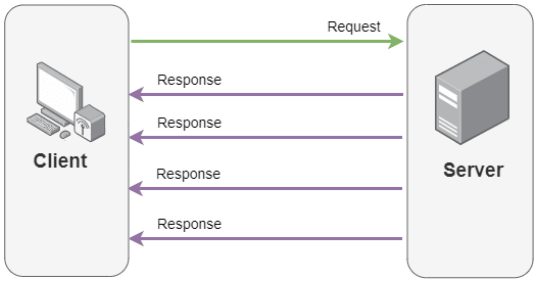


Figura 7.2: Modelo de comunicación *Streaming HTTP* [217]

- XMLHttpRequest

Es una interfaz utilizada para enviar solicitudes HTTP y HTTPS al servidor Web [219]. Su nombre `XMLHttpRequest` (XHR) no hace referencia a que sólo admite datos en XML ya que pueden usar cualquier codificación basada en texto. La interfaz se implementa como una clase desde la cual la aplicación desde el lado del cliente puede generar el número de instancias necesarias para generar solicitudes al servidor. Sin embargo, el uso de XHR conlleva un desperdicio de ancho de banda y demoras, siempre que el navegador siga sondeando los datos regularmente y el servidor continúe respondiendo [220]. Para instanciar y utilizar un objeto XMLHttpRequest se pude seguir el esquema de código de programación:

```
1 var datos;
2 //Instanciar un objeto del tipo XMLHttpRequest
3 var xhr = new XMLHttpRequest ();
```

```
4  /*
5   * Especifica el méotodo, URL y si la solicitud
6   * será asíncrona
7   */
8  xhr.open("GET", "tabla.json", true);
9  archivo.send(datos); //Envía la petición
10
11 //Evento ejecutado cuando xhr cambia de estado
12 xhr.onreadystatechange = function(){
13     //Estado del objeto 4: Completado
14     if (xhr.readyState === 4){
15         //Al ser 200 el estado devuelto es OK
16         if (xhr.status === 200){
17
18       }
19     }
20 }
```

Listado 7.2: Instanciación del obejto XMLHttpRequest

Sondeo largo Streaming y XMLHttpRequest

Sondeo largo: La petición queda abierta hasta que el servidor responda con todas las porciones de respuesta solicitada por el cliente.
Streaming HTTP: El servidor Web mantiene la conexión abierta y envía periódicamente datos al cliente.
XMLHttpRequest: Es una interfaz utilizada para enviar solicitudes HTTP y HTTPS al servidor Web (no sólo datos XML)

Además de `AJAX` y `XMLHttpRequest`, la biblioteca PrimeFaces (a partir de la versión 5.0) del Framework JSF (desde versiones 2.2) proporciona un componente que realiza el sondeo HTTP.

- Ejemplo de p:poll

Obtener la hora del servidor en tiempo real en Java Web utilizando la biblioteca de PrimeFaces. En el listado 7.3 se muestra el código HTML para la interfaz que muestra la hora.

```
1 <h:form>
2    <h:outputText id="lblHora" value="#{mbDate.currentDate}" style="font-
     weight: bold;" />
3    <h:poll interval="1" listener="#{mbDate.getDate()}" update="
     lblCurrentTime" />
4 </h:form>
```

Listado 7.3: Código HTML para mostrar el reloj.

En el bloque de código anterior se muestra el código `XHTML` necesario para hacer uso del componente `p:poll`, y además un `h:outputText` para visualizar el texto resultado

de la hora actual. En este caso se usaron ciertos atributos (ver tabla 7.1) para llevar a cabo la solicitud de datos al servidor.

Tabla 7.1: Atributos utilizados del component p:poll de PrimeFaces

Atributo	Descripción
interval	Periodo en segundos en que enviará nuevamente la petición
listener	Función del *Manage Bean* que se ejecutará cuando se dispare el evento
update	Se especifica el componente que se debe actualizar cuando termine el evento

```
1     public void getDate(){
2         currentDate = SimpleDateFormat.format (date);
3     }
```

Listado 7.4: Funciónj java para obtener la fecha

Se utilizó un objeto de la clase *Date* y el método `format` de la clase *SimpleDateFormat*. El objeto currentDate contiene la fecha actual del servidor y el método format de la clase `SimpleDateFormat` retorna la fecha en el formato que se desea obtener.

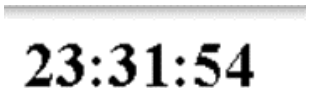


Figura 7.3: La hora como salida en el cliente

Èn la 7.3 muestra la salida en el cliente según los datos enviados desde el servidor. Éste se actualiza cada segundo dado que se estableció que se actualice en ese periodo en el atributo interval.

El consumo de recursos es una preocupación para los desarrolladores, cuando se analiza el rendimiento de la aplicación. En la figura 7.4 muestra los recursos consumidos por el servidor, consumo de recursos que se producen en el servidor debido a las constantes peticiones que realiza el cliente. Por ser una aplicación sencilla ha sido desarrollada en NetBeans 8.2 con Java jdk[2] 8.

[2]jdk: *Java Development Kit*

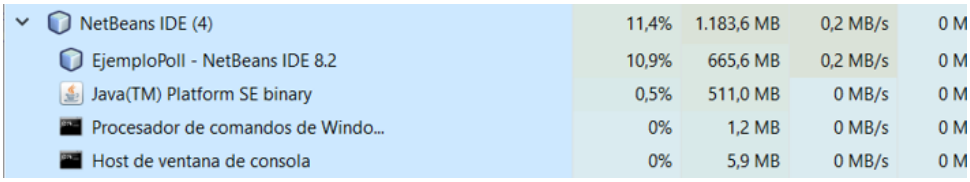


Figura 7.4: La hora como salida en el cliente

El consumo de recursos en el cliente por un lapso de 12 segundos, y la cantidad de solicitudes que ha realizado el cliente al servidor se muestra en la 7.5.

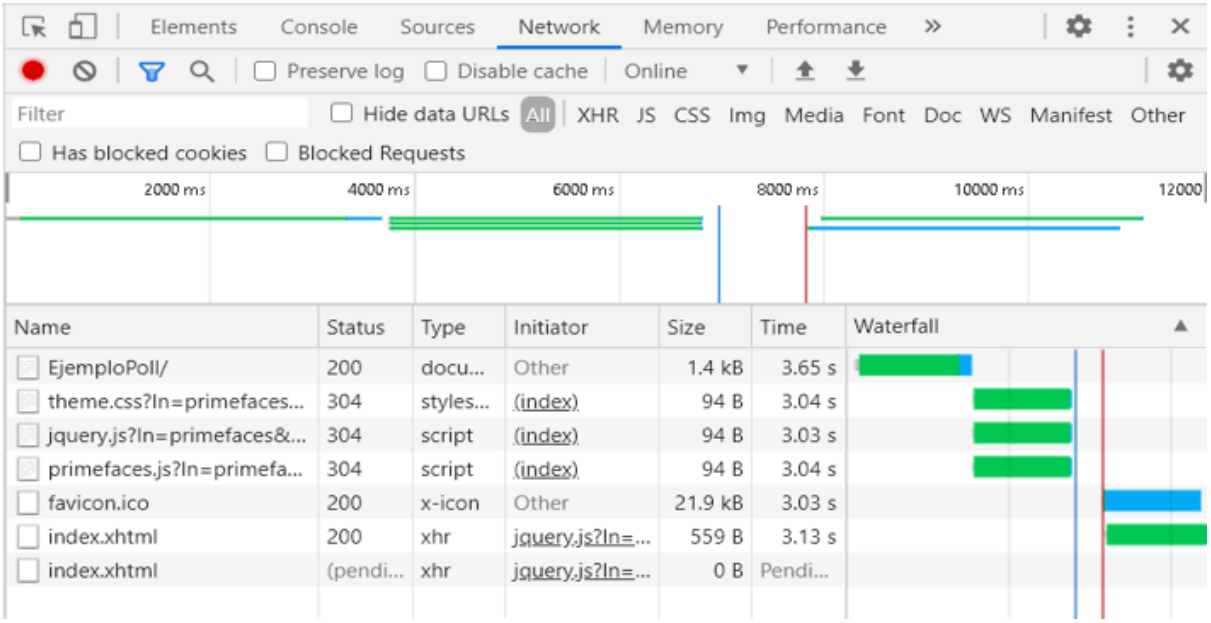


Figura 7.5: Recursos consumidos por el cliente en 12 segundos

Además se muestra que para realizar la primera petición el servidor tarda en devolver la respuesta [221].

Se realizó una prueba 70 segundos después. Por cada petición que el cliente solicita al servidor, el cliente transmite 559B de información hacia el servidor. 70 peticiones por la cantidad de bytes enviados de cada envío da un total de 38.21KB transmitidos hacia el servidor en ese lapso de tiempo, como se ve en la figura 7.6.

7.3. Trabajando con WebSockets

Las tecnologías como los sondeos y la transmisión HTTP, pueden producir un problema de sobrecarga cuando el cliente genera muchas solicitudes HTTP. En caso de que haya alta concurrencia, el tráfico de red aumentará rápidamente. Además de la alta latencia entre generar un nuevo evento y la notificación al cliente. Para aplicaciones que requieren notificaciones en tiempo real, estas soluciones no son las mejores [222]. Como

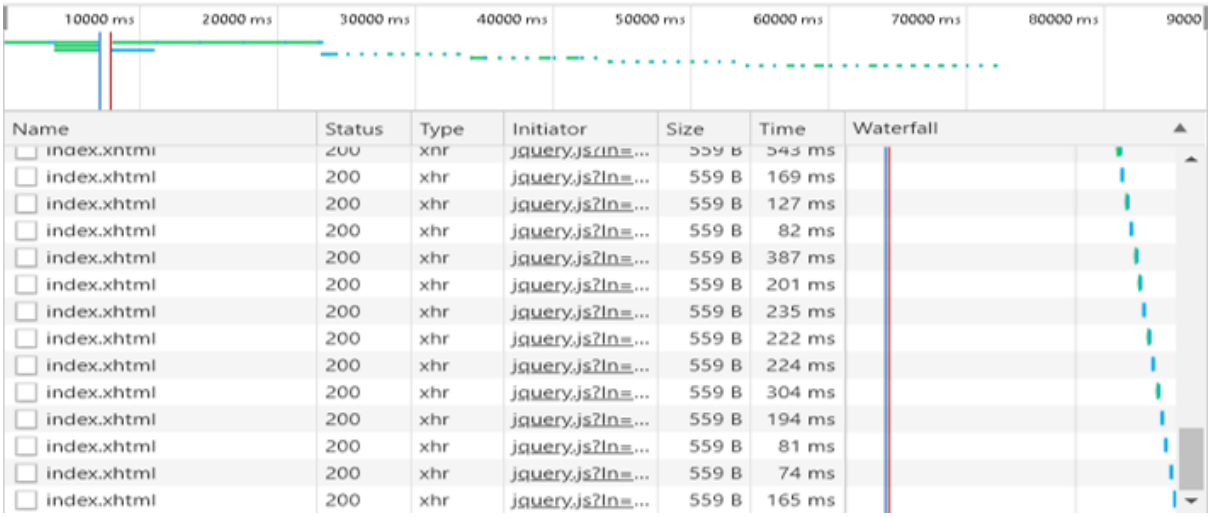


Figura 7.6: Recursos consumidos en el cliente en 70 segundos

solución, surge el concepto de Sockets. Ésta rastrea ciertas condiciones TCP/IP que ocurren en el Socket. En caso que cumpla con una condición específica se visualizará en la salida del cliente.

El protocolo WebSocket, estandarizado por -RFC 6455-[3] en el año 2011 [223], dio a los desarrolladores de JavaScript acceso a una interfaz de red bidireccional, similar a un Socket, en la que el JavaScript del lado del cliente puede abrir una conexión WebSocket a un servidor. Con este protocolo los desarrolladores pueden crear aplicaciones web que reciben información o mensajes "empujados" desde el lado del servidor en tiempo real. Este proceso se realiza sin desperdiciar bytes ni incurrir en latencia por la constante construcción de nuevas conexións TCP [224].

WebSocket es una tecnología que ofrece un canal de comunicación bidireccional, que hace posible la comunicación interactiva entre el navegador y el servidor en tiempo real [225]. WebSocket se desarrolla en JavaScript como protocolo de comunicación bajo las especificaciones de World Wide Web(W3C), además, el protocolo de WebSocket se ejecuta utilizando TCP [219].

WebSockets

El cliente recibe información del servidor sin necesidad de que el cliente le haya solicitado, contrario a las peticiones HTTP.

La tecnología WebSocket fue desarrollada como parte de la interfaz HTML5 con JavaScript. Refleja una conexión *full-duplex* de un solo Socket sobre el cual se pueden enviar mensajes entre el cliente y el servidor. También simplifica mucha la complejidad alrededor de la comunicación bidireccional de la Web y la administración de la conexión [226]. Web Sockets de HTML5 no es sólo otra mejora incremental de las

[3]Especificaciones para la implementación del protocolo WebSocket

comunicaciones HTTP convencionales, sino que también representa un avance enorme, especialmente para las aplicaciones web en tiempo real, impulsadas por eventos [227]. Proporciona mejoras dinámicas sobre la conexión convencional full-duplex sobre HTTP. La principal diferencia de los WebSockets en comparación con el tráfico de red habitual a través de HTTP es el protocolo.

El protocolo WebSocket no sigue la convención tradicional de respuesta a las solicitudes. Este protocolo permite a los desarrolladores crear aplicaciones web que reciben información o mensajes "enviados" en tiempo real desde el lado del servidor, sin perder bytes ni incurrir en latencia debido a la construcción constante de nuevas conexións TCP [228]. La conexión permanece abierta y activa mientras el cliente o el servidor no cierre la conexión.

7.3.1. Ventajas

El protocolo WebSocket es autónomo respecto HTTP, pero es totalmente compatible con sus características, además, WebSocket cuenta con todos los beneficios de la estructura existente como [229]:

- Soporte nativo en navegador, incluido la seguridad establecida en el origen.
- Transversal de proxy y firewall.
- Permiten un número potencialmente ilimitado de servicios que se ejecutan en un solo puerto TCP.
- Eliminación de los límites de longitud de TCP simple.

7.3.2. Características

Resumiendo las características de los WebSockets se presentan las más importante como [223, 229–231]:

- No sigue los parámetros tradicionales de request-response. Los WebSockets se basan totalmente en eventos, por lo que se convierten en una tecnología ligera para los navegadors.
- Hace más eficiente la comunicación en tiempo real, ya que siempre se puede utilizar las solicitudes constantes (Sondeo) a través de HTTP para recibir notificaciones o mensajes.
- WebSocket les ahorra muchos recursos al servidor y al cliente tales como: ancho de banda, potencia de CPU y espacio de memoria RAM.
- Una vez realizada la conexión entre el cliente y el servidor, ambos pueden enviarse mensajes en tiempo real, este tipo de comunicación se denomina *full-duplex*.

- La comunicación entre el cliente y servidor es más sencilla, a diferencia de los métodos anteriores como HTTP y AJAX.
- WebSocket es compatible con otros protocoloss de nivel superior, lo que le agrega modularidad y fomenta el desarrollo de componentes reutilizables y más seguros.
- WebSocket proporciona redes de estilo TCP para aplicaciones HTML5, sin destruir o afectar la seguridad del navegador.

Características de los WebSockets

Se basan totalmente en eventos, haciendo más eficiente la comunicación en tiempo real. Ahorran muchos recursos tanto al servidor como al cliente, haciendo una comunicación sencilla, proporcionando redes de estilo TCP para aplicación HTML5, además, es compatible con otros protocolos de nivel superior.

7.4. WebSocket API

WebSocket es un protocolo, pero también existe una API, que permite la administración del protocolo con la ejecución y el uso de métodos, atributos y eventos. La API fue desarrollada por el W3C, aplicando todos los estándares de las herramientas de autor, mientras que el protocolo fue desarrollado por Internet Engineering Task Force (IETF), la cual es una organización que forma estándares para las aplicaciones web. [229, 230].

La API de WebSocket es compatible con todos los navegadors y cuenta con todas las características necesarias para utilizar una conexión bidireccional completa. Permite realizar y ejecutar acciones necesarias, como abrir o cerrar conexión, enviar o recibir mensajes y visualizar evento activados por el servidor [229]

7.4.1. Concepto y política de conexión

La API de WebSocket es la que implementa una tecnología avanzada que hace posible abrir una sesión de comunicación interactiva entre el navegador del usuario y un servidor. Esta API, se puede utilizar cuando es requisito una comunicación de doble vía e independiente. Al requerir enviar mensajes a un servidor y recibir respuestas controladas por eventos sin tener que consultar al servidor para una respuesta [232].

La API de WebSocket permite crear una conexión y enviar datos a cualquier servidor, al contrario de HTTP. WebSocket utiliza una verificación de origen, lo que quiere decir que el encargado de permitir la conexión es el servidor (ver figura 7.7). El marco de WebSocket no incluyen encabezado HTTP, por lo que el host de origen se envía al servidor en una solicitud HTTP normal [230, 233, 234].

El proveedor del servicio es el responsable de verificar el host y permitir o no la conexión, tomando en cuenta que la política de origen verificado, no impide que nadie

se conecte al servicio. Sin embargo, la política protege al cliente contra ataques de falsificación de solicitud de servicio entre los sitios [230].

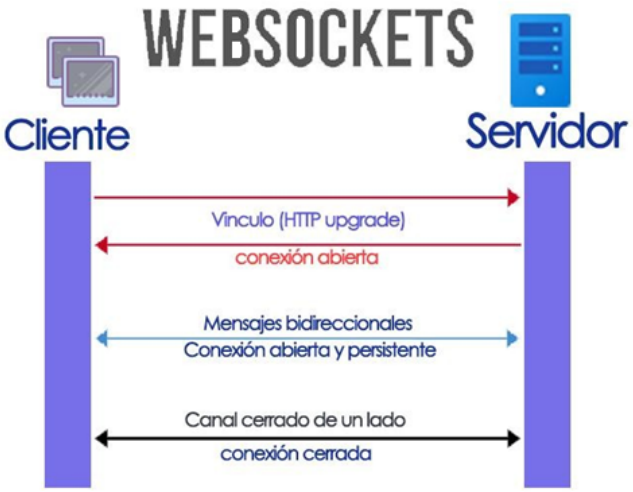


Figura 7.7: Comunicación WebSocket

7.4.2. El constructor de WebSocket

Para poder establecer la conexión de WebSocket a un servidor, se crea la instancia de un objeto WebSocket, enviando como argumento la URL al que se desea conectar (ver figura 7.8). Para el tráfico de datos el protocolo tiene dos esquemas **ws(no cifrado)** que es similar al esquema HTTP y **wss(cifrado)** que representa una conexión segura a través de TLS (Transport Layer Security), el cual utiliza la misma seguridad que HTTPS [229, 235].

Figura 7.8: URL WebSocket

```
1 //Crear un nuevo objeto WebSocket
2 var ws = new WebSocket("ws://www.websocket.org");
```

Listado 7.5: Creación de un objeto WebSocket.

También se puede enviar un argumento opcional que es el nombre o la matriz de protocolos que el servidor debe incluir en su respuesta para establecer la conexión. En el listado 7.6 se presenta el código para crear un objeto WebSocket.

```
1 //Crear un nuevo objeto WebSocket
2 var ws;
```

```
3 ws = new WebSocket("ws://www.websocket.org", "MyProtocolo");
```

Listado 7.6: Creación de objeto WebSocket especificando el protocolo.

7.4.3. Eventos de WebSocket

La API y el protocolo de WebSocket se basan en eventos, el código WebSocket utiliza estos eventos para administrar los datos y el estado de la conexión. El modelo de programación de WebSocket es asíncrono, lo que significa que la aplicación ejecuta los eventos mientras la conexión se encuentre abierta. WebSocket cuenta con cuatros eventos diferentes: *0* (abrir), *message* (mensaje), *error* (Error) y *close* (cerrar) [229].

Evento *open* (onopen)

Cuando se logra crear una conexión con el WebSocket, se ejecuta el evento open y se establece la conexión. La devolución de llamada correspondiente a este evento se denomina onopen [223, 229]. En el listado 7.7 se muestra el código para programar el evento onopen del WebScoket.

```
1 ws.onopen = function (e) {
2    console.log("Connection open...");
3 };
```

Listado 7.7: Código de evento *open*.

Evento *message* (onmessage)

Los mensajes de WebSocket contienen los datos del servidor, además de texto, los mensajes de WebSocket pueden manejar datos binarios y vectores. El evento del mensaje se activa cuando se recibe un mensaje. La devolución de llamada al evento de mensaje se denomina onmessage [223, 229]. En el listado 7.8 se muestra la programación para el evento onmessage del WebSocket.

```
1 ws.onmessage = function(e) {
2  if (typeof e.data === "string") {
3   console.log("String message received", e, e.data);
4    } else {
5       console.log("Other message received", e, e.data); }
6 };
```

Listado 7.8: Evento *message*.

Evento *error* (onerror)

El código de la función que se ejecutará cuando se dé el evento onerror del WebSocket se muestra en el listado 7.9. El evento de onerror se ejecuta en respuesta de alguna falla

o error en los demás eventos, estos errores también pueden provocar el cierre de las conexións del WebSocket. En el área de este evento se puede programar la lógica para reconectarse al servidor y manejar las excepciones [223, 229].

```
ws.onerror function(e) {
    console.log("WebSocket Error: ", e);
   handleErrors (e);
};
```

Listado 7.9: Evento de *error*.

Evento *close* (onclose)

El evento cerrar se ejecuta cuando se cierra la conexión de WebSocket, la devolución al llamar al evento se denomina `onclose`. El código de ejemplo que se ejecutará se muestra en el listado 7.10. Una vez que se cierra la conexión, el cliente y el servidor ya no pueden recibir ni enviar mensajes [223, 229].

```
   ws.onclose = function(e) {
       console.log("Connection closed", e);
   };
```

Listado 7.10: Evento *close*.

Eventos de los WebSockets

Los eventos de los WebSockets son: *open* (`onopen`), *message* (`onmessage`), *error* (`onerror`) y *close* (`onclose`).

7.4.4. Métodos de la clase WebSocket

Los objetos WebSocket tienen dos métodos: `send` y `close`.

Método send

Una vez abierta la conexión con el servidor, se puede ejecutar el método `send`. Se utiliza el método `send` para enviar mensajes desde el cliente al servidor. En el código del listado 7.11 se muestra el uso del método `send`, enviando un texto y un arreglo. Después de realizar él envió del mensaje se puede dejar la conexión abierta o cerrarla con el método close [223].

```
// enviar texto simple
ws.send("hola mundo");
// enviar un Blob
var blob = new Blob("blob contents");
ws.send(blob);
// enviar un ArrayBuffer
```

```
7 var a = new Uint8Array([8,6,7,5,3,0,91);
8 ws.send(a.buffer);
```

Listado 7.11: Ejemplo de método send con texto - blob y Array.

7.4.5. Método `close`

Para cerrar la conexión de WebSocket se debe utilizar el método `close`. Después de ejecutar este método no se podrá enviar más datos al WebSocket. También se puede pasar dos argumentos al método `close()`: código (un código de estado numérico) y razón (una cadena de texto), estos argumentos transmite información al servidor sobre por qué el cliente cerró la conexión [223, 229]. En el listado 7.12 se muestra el uso del método `close` con el código de cierre normal y un mensaje de acuerdo a la acción ejecutada.

```
1 // Cierre la conexion de WebSocket
2 ws.close(1000, "Closing normally");
3
```

Listado 7.12: Método cerrar conexión.

7.4.6. Atributos del objeto WebSocket

WebSocket es la clase (objeto WebSocket) proporcionada por la API para crear y gestionar conexións WebSocket con un servidor, así como para enviar y recibir datos por medio de la conexión. Esta clase proporciona algunos atributos que permiten conocer el estado de la conexión como también especificar y obtener datos sobre los elementos que intervienen en ella. La descripción de cada uno de los atributos se da a continuación [236]:

- `readyState`

El atributo readyState ayuda a identificar el estado de la conexión del WebSocket (Ver tabla 7.2), la conexión puede tener cuatro estados como [223, 229]:

Tabla 7.2: Valores del atrinuto de estado `readyState`

Atributo	Valor	Estado
WebSocket.CONNECTING	0	La conexión está en proceso.
WebSocket.OPEN	1	La conexión está establecida.
WebSocket.CLOSING	2	La conexión está pasando por el cierre.
WebSocket.CLOSING	3	La conexión está cerrada.

- bufferedAmount

Este atributo sirve para saber la cantidad de datos o el número de bytes que se van a enviar al servidor, con esto se puede validar la cantidad de bytes que se requiere enviar [223, 229, 236]. En el listado 7.13

```
1 if (ws.bufferedAmount < THRESHOLD) {
2  ws.send (getApplicationState () );
3 }
```

Listado 7.13: Validación de tamaño de envío.

- protocol

Este atributo protocolo proporciona información sobre el protocolo que se utilizando entre el cliente y el servidor, es decir indica el nombre del protocolo y que protocolo se usara en el WebSocket [223].

- URL

URL como atributo de los WebSockets especifica o contiene la URL absoluta del recurso WebSocket. La URL del recurso WebSocket utiliza su propio esquema personalizado: ws para la comunicación en texto plano (por ejemplo, ws://ejemplo.com/socket), y wss cuando se requiere un canal cifrado (TCP+TLS).

7.4.7. Navegación segura con WebSocket

Para tener una navegación segura en una página, es necesario que los navegadors web implementen medidas de seguridad que le garantice la integridad cliente y del servidor. A continuación, se mencionarán las más importantes que ayudan a controlar [230]:

- El número de conexións de WebSocket.
- Los intentos de conexión del WebSocket a las direcciones IP o números de puertos.
- Los intentos de conexión a destinos como localhost o red interna.
- Tamaños de mensajes o de marco de WebSocket poco comunes.
- Existencia de host que tienen como origen servicios que están en listas negras o listas blancas.

7.5. Comparación de TCP, HTTP y WebSocket

En la tabla 7.3 se presenta una comparativa entre los protocolos TCP, HTTP y WebSockets.

Tabla 7.3: Comparativa entre TCP, HTTP y WebSocket [235]

Características	TCP	HTTP	WebSocket
Direccionamiento	Dirección IP y puerto	URL	URL
Transmisión simultanea	Duplex completo	Media dúplex	Media dúplex
Contenido	Contenido	Mensajes MIME	Texto y mensajes binarios
Límites del mensaje	No	Yes	Yes
Orientado a la conexion	Yes	No	Yes

7.6. Desarrollo de una aplicación websocket con node.js

Antes de comenzar con el ejemplo se describirá un concepto de la tecnología base para desarrollar la aplicación con WebSocket.

7.6.1. Node.js

Fue creado con la idea de un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node.js está diseñado para crear aplicaciones de red escalables [237] Este entorno de desarrollo permite a los programadores escribir y ejecutar todo el código en el lado del servidor. Node.js usa una estructura de entrada y salida sin bloque. Además de ser controlado a través de eventos que le permiten ser siendo liviano y eficiente. El punto más fuerte de Node.js es la creación de aplicaciones web realmente rápidas, debido a que este es capaz de soportar grandes cantidades de conexións simultáneas, dando como resultado una alta escalabilidad en las aplicaciones web [238, 239].

7.6.2. Ejemplo:

Se debe crear una carpeta del proyecto y mediante un terminal o la ventana de comandos de Windows (o cmd) del equipo se instala un paquete NodeJS, para ellos se utiliza el comando `npm init -y`. Generando así un package.json.

```
{
    "name": "ejemplo",
    "version": "1.0.0",
    "description": 3,
    "main": "index.js",
    "scripts": {
       "start": "nodemon server/main.js"
   }
```

```
9 }
```

Listado 7.14: Packege.json.

También se descarga las librerías `express`[4] y `socket.io`, para ellos se utiliza los comandos del listado 7.15 en un terminal o en la ventana de comandos de Windows (`cmd`) como requisito se debe estar en la carpeta del proyecto.

```
1 1$ npm install -save express
2 2$ npm install -save socket.io
```

Listado 7.15: Comando para instalar `express` y socket.io.

En la carpeta del proyecto se crea un archivo JavaScript con el nombre main.js, en este archivo se definen con *express* las variables como están en el listado 7.16 y se lo vincula con un servidor HTTP, luego se vinculan todas esas variables al servidor WebSocket creado con socket.io.

```
1 var express = require('express');
2 var app = express();
3 var server = require('http').Server (app);
4 var io = require('socket.io') (server);
```

Listado 7.16: Creación del servidor y del WebSocket.

Se Vincula o se podrá escuchar al servidor creado en localhost por el puerto 8080 (ver listado 7.17.

```
1 server.listen (8080, function() {
2    console.log("Servidor corriendo en http://localhost:8080");
3 });
```

Listado 7.17: El servidor escuchando en el puerto 8080.

Una vez que se ha vinculado al servidor, luego se va a crear los elementos que se enviarán, en este caso un arreglo que contenga el nombre del autor y el mensaje (ver listado 7.18). Este arreglo servirá para probar el funcionamiento del chat al momento que el cliente se conecte.

```
1 var messages = [{
2     id: 1,
3     text: "Hola empleado",
4     author: "Pepe"
5 }];
```

Listado 7.18: Mensaje arreglo.

El el listado 7.19 se muestra el código para enviar un mensaje por medio del método emit propio de la Socket.IO. El objeto io almacena el servidor de WebSocket, entonces cuando un cliente realice una conexión al servidor, se enviará el `array messages` con el evento 'messages' a través de `io.on()`.

[4]Express.js, conocido como Express simplemente, es un marco de aplicación web para el back-end exclusivo para Node.js. Está diseñado para crear aplicaciones web y APIs.

```
io.on('connection', function(socket) {
    console.log('Un cliente se ha conectado');
   socket.emit('messages', messages);
});
```

Listado 7.19: Evento para enviar mensajes.

Luego se crea el un archivo HTML que contenga un formulario, pero no se usa el atributo `action` para que no se recargue la página, este formulario contendrá una etiqueta `div` con un `id="mensajes"`, para insertar los mensajes enviados. También tendrá dos entradas de tipo texto, uno para los mensajes y el otro para el nombre del autor. El listado 7.20 muestra el código HTML para la interfaz de usuario para un *chat* empresarial básico.

```
<!DOCTYPE html>
<html lang="en">
    <head>
       <meta charset="UTF-8">
      <title>My Aplicacion con Sockets</title>
      <script src="/socket.io/socket.io.js"></script>
      <script src="main.js"></script>
      <link href="estilos.css" rel="stylesheet" type="text/css" />
   </head>
   <body onload="nombreAzar();">
       <form onsubmit="return addMessage(this)">
          <div class="menu">
             <div class="back"><i class="fa fa-chevron-left"></i>
                <img src="https://img.icons8.com/fluent/48/000000/company.
   png" draggable="false"/>
             </div>
             <div class="name">Chat grupal de la empresa</div>
             <div class="last">Seguir adelante</div>
          </div>
          <ol class="chat" id="messages">
          </ol>
          <input class="emojis" type="submit" value="">
          <input class="textarea" type="text" id="texto" placeholder="
   Cuentanos algo..."/>
          <input id="username" class="textarea2" type="text" id="texto" />
          <br/>
      </form>
   </body>
</html>
```

Listado 7.20: Archivo HTML.

Para recibir el mensaje se crea un archivo JavaScript (en la carpeta public), en este archivo se crea una variable para la conexión con el servidor y la llamada al evento mensajes.

```
var socket io.connect('http://localhost:8080', {
                        'forceNew':true
```

```
                    });
socket.on('messages', function(data) {
    console.log(data);
    render(data);
})
```

Listado 7.21: Conexión del lado del cliente.

Después se indica cuales en el archivo main.js del lado del servidor cuáles serán los ficheros estáticos, en este caso se requiere que todos los archivos JavaScript de la capeta `public` sean archivos del lado del servidor.

```
app.use(express.static('public'));
```

Listado 7.22: Código para poner los ficheros en público.

En el cliente se crea una función JavaScript (render) para manipular el código HTML para mostrar los mensajes enviados. Esta función recibirá el arreglo de mensajes. Éste es concatenado y se envía al div que tiene el id=”messages”. El código se muestra en el listado 7.23.

```
function render (data) {
    var html data.map (function(elem, index) {
        return('<li class="self"'+
              '<div class="msg"></div>'+
                '<p><strong>${elem.author}: </strong></p>'+
                '<p>${elem.text}</p>'+
                '</li>');
         }).join("");
    document.getElementById('messages').innerHTML == html;
}
```

Listado 7.23: Método para mostrar los mensajes en el HTML.

Al método render se lo llama en el evento “on” de la variable socket que se creó en el archivo main del cliente. El código respectivo se muestra en el listado 7.24

```
socket.on('messages', function(data) {
    console.log(data);
    render(data);
});
```

Listado 7.24: Evento “on” para recibir el array y enviarlo al método render.

Ahora se requiere tomar los datos que se van a enviar en el mensaje, es decir, el nombre del emisor (usuario emisor) y el texto del mensaje. Para esto, se crea una función llamada `addMessage` (ver listado 7.25), la cual se ejecutará cada vez que se presione el botón enviar del formulario HTML. Para lograrlo, suficiente programar el evento `onsubmit` del formulario HTML (atributo `<form onsubmit=“addMessage”>`). Dicho método extraerá el mensaje y el nombre de usuario que se encuentra en los controles input del formulario (entradas del formulario).

```
<form onsubmit="return addMessage(this)">
```

Listado 7.25: Atributo del formulario.

```
function addMessage(e) {
    var message = {
                    author: document.getElementById("username").value,
                    text: document.getElementById('texto').value,
                  };
    socket.emit('new-message', message); return false;
}
```

Listado 7.26: Método para extraer y enviar el mensaje.

```
socket.on('new-message', function(data) { messages.push(data);
    io.sockets.emit('messages', messages);
});
```

Listado 7.27: Evento que añade los nuevos mensajes y los envía al socket.

Después de realizar todo el código, se puede verificar el funcionamiento de la aplicación (ver figura 7.9, en este caso todo funcionó correctamente y los mensajes se envió a todos los usuarios conectados al servidor.

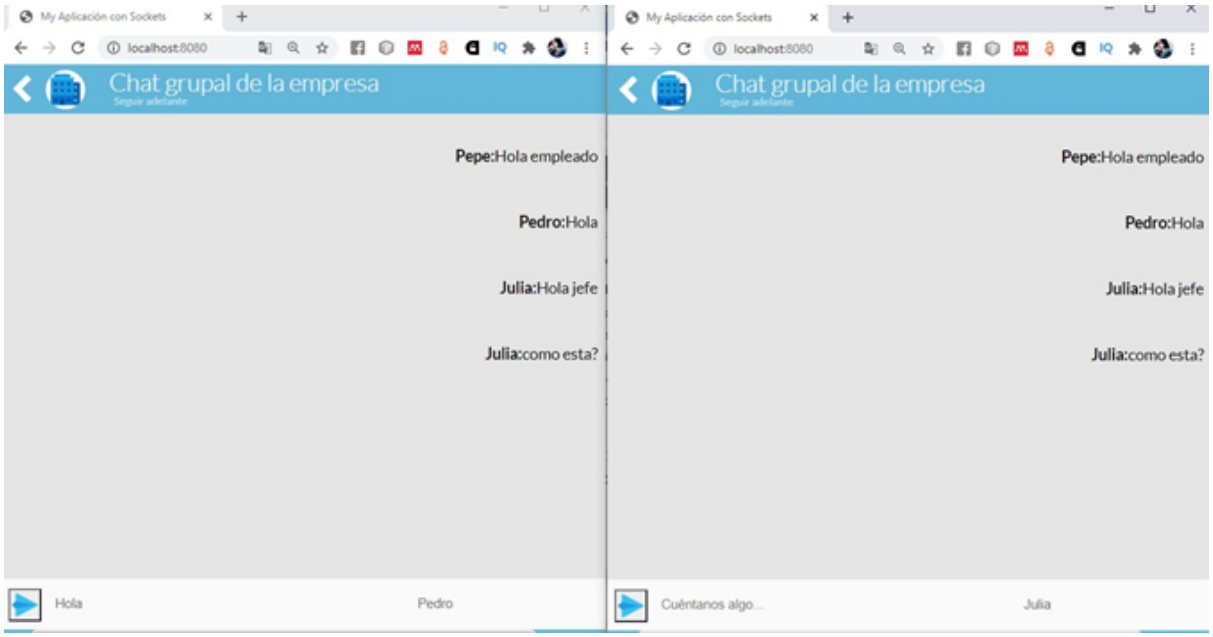


Figura 7.9: Interfaz gráfica

7.6.3. Ejercicio 2 - Chat con JSF 2.3

Chat sencillo utilizando algunas novedades JSF 2.3 + CDI 2.0 + Bean Validation 2.0. Para el despliegue de esta aplicación debe:

- Iniciar GassFish: ’./asadmin start-domain domain1’
- La URL: ’https://localhost:8080/jsf-chat’ o con el servidor y nombre de la aplicación donde sea desplegado.

Características

- JSF 2.2 - Marcado amigable de HTML5
- JSF 2.3 - Selectores
- JSF 2.3 - Código JS desde el bean gestionado
- JSF 2.3 - Inyección de artefactos
- JSF 2.3 - f:webSocket
- WebSocket 1.1 - Proporcionado por Tomcat

Archivo pom.xml

En los proyectos Maven es imprescindible el arcvhivo `pom.xml`, en vista de que, en este archivo se colocan todas las dependencias (referencias a las bibliotecas y/o API que se usan en el proyecto). En este caso, el contenido de este archivo es el que se muestra en el listado 7.28.

```
1  <project xmlns="http://maven.apache.org/POM/4.0.0"
2     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.
       apache.org/xsd/maven-4.0.0.xsd">
4     <modelVersion>4.0.0</modelVersion>
5     <groupId>com.github.wesleyegberto.adoptajsr</groupId>
6     <artifactId>jsf-chat</artifactId>
7     <version>0.0.1-SNAPSHOT</version>
8     <packaging>war</packaging>
9
10    <properties>
11       <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
12       <maven.compiler.source>1.8</maven.compiler.source>
13       <maven.compiler.target>1.8</maven.compiler.target>
14    </properties>
15
16    <dependencies>
17       <dependency>
18          <groupId>jakarta.platform</groupId>
19          <artifactId>jakarta.jakartaee-api</artifactId>
20          <version>9.1.0</version>
21          <scope>provided</scope>
22       </dependency>
23    </dependencies>
```

```
24
25    <build>
26       <finalName>jsf-chat</finalName>
27       <plugins>
28          <plugin>
29             <groupId>org.apache.maven.plugins</groupId>
30             <artifactId>maven-compiler-plugin</artifactId>
31             <version>3.6.1</version>
32             <configuration>
33                <source>1.8</source>
34                <target>1.8</target>
35             </configuration>
36          </plugin>
37          <plugin>
38             <groupId>org.apache.maven.plugins</groupId>
39             <artifactId>maven-war-plugin</artifactId>
40             <version>2.3</version>
41             <configuration>
42                <failOnMissingWebXml>false</failOnMissingWebXml>
43             </configuration>
44          </plugin>
45       </plugins>
46    </build>
47 </project>
```

Listado 7.28: Contenido del archivo `pom.xml`.

Clases Auxiliares

Estas clases no están directamente relacionadas con la solución del problema. Es decir, pueden ser reemplazadas con algunas otras o se pueden construir de diferente manera. Es este ejemplo, aunque puede ser reemplazadas, son necesarias para el funcionamiento correcto del chat que se pretende implementar.

Clase `UsernameNotUniqueException`

```
1  package gcgu.jsfchat;
2
3  public class UsernameNotUniqueException extends RuntimeException {
4     private static final long serialVersionUID = 1L;
5
6     public UsernameNotUniqueException(String message) {
7        super(message);
8     }
9
10    public UsernameNotUniqueException(String message, Throwable cause) {
11       super(message, cause);
12    }
13
14 }
```

Listado 7.29: Clase excepción usuario existe

Clase LoggerProducer

```
1  package gcgu.jsfchat;
2
3  import java.util.logging.Logger;
4
5  import jakarta.enterprise.inject.Produces;
6  import jakarta.enterprise.inject.spi.InjectionPoint;
7
8  public class LoggerProducer {
9     @Produces
10    public Logger produceLogger(InjectionPoint injectionPoint) {
11       Class<?> clazs = injectionPoint.getMember().getDeclaringClass();
12       return Logger.getLogger(clazs.getName());
13    }
14 }
```

Listado 7.30: Clase necesaria para implementar el inicio de sesión

Clase CookieHelper

```
1  package gcgu.jsfchat;
2
3  import java.io.UnsupportedEncodingException;
4  import java.util.Base64;
5  import java.util.LinkedHashMap;
6  import java.util.Map;
7
8  import jakarta.enterprise.context.RequestScoped;
9  import jakarta.faces.annotation.RequestCookieMap;
10 import jakarta.faces.context.ExternalContext;
11 import jakarta.inject.Inject;
12 import jakarta.servlet.http.Cookie;
13
14 @RequestScoped
15 public class CookieHelper {
16
17    @Inject
18    @RequestCookieMap
19    private Map<String, Object> cookies;
20
21    @Inject
22    private ExternalContext externalContext;
23
24    public CookieHelper() {
25    }
```

```
26
27     public boolean containsCookie(String cookieName) {
28         return cookies.containsKey(cookieName);
29     }
30
31     public String getCookieValue(String cookieName) {
32         Object cookieValue = cookies.get(cookieName);
33         if (cookieValue != null && cookieValue instanceof Cookie) {
34             Cookie cookie = (Cookie) cookieValue;
35             try {
36                 return new String(Base64.getDecoder().decode(cookie.getValue())
    , "utf-8");
37             } catch (UnsupportedEncodingException e) {
38                 return null;
39             }
40         }
41         return null;
42     }
43
44     public void setCookie(String cookieName, String cookieValue) {
45         String encodedValue = null;
46         try {
47             encodedValue = Base64.getEncoder().encodeToString(cookieValue.
    getBytes("utf-8"));
48         } catch (UnsupportedEncodingException e) {
49             throw new RuntimeException(e);
50         }
51         Map<String, Object> props = new LinkedHashMap<>();
52         props.put("httpOnly", Boolean.TRUE);
53         externalContext.addResponseCookie(cookieName, encodedValue, props);
54     }
55 }
```

Listado 7.31: Gestión de datos del lado del cliente (Cookies)

Clase JSFConfigurator

```
1 package gcgu.jsfchat;
2
3 import jakarta.faces.annotation.FacesConfig;
4 import jakarta.faces.annotation.FacesConfig.Version;
5
6 // Para habilitar las características de JSF 2.3, principalmente inyecciónin
      de los artefactos EL y CDI
7 @FacesConfig(version = Version.JSF_2_3)
8 public class JSFConfigurator {
9 }
```

Listado 7.32: Clase Java JSFConfigurator

Clase User (Clase Entity)

La clase User es una clase Bean por su forma, únicamente contiene los atributos de un Usuario con los métodos get y set además de su constructor. El código de esta clase se muestra en el listado 7.33

```
1  package gcgu.jsfchat.model;
2
3  import java.io.Serializable;
4
5  /**
6   * Chat User.
7   */
8  public class User implements Serializable {
9     private static final long serialVersionUID = 1L;
10
11    private String nickname;
12    private String name;
13
14    public User() {
15    }
16
17    public User(String nickname, String name) {
18       this.nickname = nickname;
19       this.name = name;
20    }
21
22    public String getNickname() {
23       return nickname;
24    }
25
26    public void setNickname(String nickname) {
27       this.nickname = nickname;
28    }
29
30    public String getName() {
31       return name;
32    }
33
34    public void setName(String name) {
35       this.name = name;
36    }
37
38    @Override
39    public String toString() {
40       return "User [nickname=" + nickname + ", name=" + name + "]";
41    }
42 }
```

Listado 7.33: Código de la clase User

Clase UserConverter

Esta clase es importante para poder trabajar con los datos que el usuario ingresa para trabajar con la clase `User`. Como muestra el listado de código 7.34, trabaja con los objetos que se usan en la GUI.

```
1  package com.github.wesleyegberto.adoptajsr.jsfchat.converter;
2
3  import jakarta.faces.component.UIComponent;
4  import jakarta.faces.context.FacesContext;
5  import jakarta.faces.convert.Converter;
6  import jakarta.faces.convert.FacesConverter;
7  import jakarta.inject.Inject;
8
9  import gcgu.jsfchat.db.UserDAO;
10 import gcgu.jsfchat.model.User;
11
12 @FacesConverter(forClass = User.class, managed = true)
13 public class UserConverter implements Converter<User> {
14    @Inject
15    private UserDAO userDao;
16
17    @Override
18    public User getAsObject(FacesContext context, UIComponent component,
      String value) {
19       return userDao.getByUsername(value);
20    }
21
22    @Override
23    public String getAsString(FacesContext context, UIComponent component,
      User value) {
24       if (value == null || value.getNickname() == null)
25          return "No user";
26       return value.getNickname();
27    }
28 }
```

Listado 7.34: Clase Java UserConverter

Clase UserDAO

El nombrar a las clases de acceso a datos como DAO (*Data Access Object*) es tradicional. Es por eso que la clase que accede a la base de datos de los usuarios se le ha llamado UserDAO, su código ejemplo se muestra en el listado 7.35.

```
1  package gcgu.jsfchat.db;
2
3  import java.util.Collections;
4  import java.util.Map;
5  import java.util.logging.Logger;
6
7  import jakarta.annotation.PostConstruct;
```

```
8  import jakarta.enterprise.context.RequestScoped;
9  import jakarta.faces.annotation.ApplicationMap;
10 import jakarta.inject.Inject;
11
12 import gcgu.jsfchat.UsernameNotUniqueException;
13 import gcgu.jsfchat.model.User;
14
15 @RequestScoped
16 public class UserDAO {
17
18    @Inject
19    private Logger LOG;
20
21    private final Map<String, Object> USER_DB;
22
23    public UserDAO() {
24       USER_DB = null;
25    }
26
27    @Inject
28    public UserDAO(@ApplicationMap Map<String, Object> applicationMap) {
29       this.USER_DB = Collections.synchronizedMap(applicationMap);
30    }
31
32    @PostConstruct
33    public void postConstruct() {
34    }
35
36    public boolean isNicknameUsed(String nickname) {
37       return USER_DB.containsKey(nickname);
38    }
39
40    public void createUser(User user) {
41       Object prevUser = USER_DB.putIfAbsent(user.getNickname(), user);
42       if (prevUser != null && prevUser != user) {
43          throw new UsernameNotUniqueException("Nickname is already in use!"
   );
44       }
45       LOG.info("User registered: " + user);
46    }
47
48    public User getByUsername(String username) {
49       return (User) USER_DB.getOrDefault(username, null);
50    }
51 }
```

Listado 7.35: Clase Java UserDAO

Clase UserBean

Esta clase (ver listado de código 7.36) permite tener acceso a los datos del usuario (clase User) desde la interfaz (ver listado de código 7.36).

```
1  package gcgu.jsfchat.controller;
2
3  import java.io.Serializable;
4
5  import jakarta.enterprise.context.SessionScoped;
6  import jakarta.inject.Named;
7
8  import gcgu.jsfchat.model.User;
9
10 @Named
11 @SessionScoped
12 public class UserBean implements Serializable {
13    private static final long serialVersionUID = 1L;
14    private User user;
15
16    public void startSessionFor(User user) {
17       this.user = user;
18    }
19
20    public void finishSession() {
21       this.user = null;
22    }
23
24    public User getUser() {
25       return user;
26    }
27
28    public String getName() {
29       if (user == null)
30          return "Anonymous";
31       return user.getName();
32    }
33
34    public String getNickname() {
35       if (user == null)
36          return "Anonymous";
37       return user.getNickname();
38    }
39
40    public boolean isLogged() {
41       return user != null;
42    }
43
44 }
```

Listado 7.36: ManageBean para la clase User

Clase `FakeEndpoint`

Una clase *EndPoint* de *WebSocket* representa un elemento en *Integration Server* que maneja las conexións de WebSocket. Cada *endpoint* de WebSocket tiene una dirección asociada, que se utiliza para ubicar e identificar el *endpoint*. Esta dirección consta principalmente de un identificador uniforme de recursos (URI), que especifica la ubicación del *endpoint*.

El *endpoint* de WebSocket se registra como un *endpoint* de servidor o como un *endpoint* de cliente. En términos simples, un *endpoint* del servidor WebSocket es una dirección web (URL) en la que los clientes de un servicio específico pueden acceder a él. Al hacer referencia a esa URL, los clientes de WebSocket pueden acceder a las operaciones proporcionadas por ese servicio. El *endpoint* de WebSocket contiene servicios de devolución de llamada tanto para el servidor como para el cliente. Estos servicios activan la notificación, lo que permite que las aplicaciones realicen alguna acción como parte del procesamiento del evento. Hay una serie de servicios de devolución de llamada manejados por la aplicación WebSocket e incluyen los mostrados en la tabla 7.4 [240]:

Tabla 7.4: Atributos utilizados del component p:poll de PrimeFaces

Servicio	Descripción
_onConnect	Se invoca cuando la conexión *WebSocket* se establece correctamente.
_onText	Se invoca cuando el *endpoint* de *WebSocket* recibe un mensaje de texto.
_onBinary	Se invoca cuando el *endpoint* de *WebSocket* recibe un mensaje binario.
_onClose	Se invoca cuando se cierra una conexión *WebSocket*.
_onError	Se invoca cuando un *endpoint* de *WebSocket* encuentra un error.

```
1  \begin{lstlisting}
2   package gcgu.jsfchat.endpoint;
3
4  import jakarta.websocket.Endpoint;
5  import jakarta.websocket.EndpointConfig;
6  import jakarta.websocket.Session;
7
8  public class FakeEndpoint extends Endpoint {
9     @Override
10    public void onOpen(Session session, EndpointConfig config) {
11
12    }
```

```
13 }
```

Listado 7.37: ManageBean para la clase User

Clase ChatConnection

Una de las clases muy importantes para esta Aplicación sin duda es la clase en la que se establece la conexión para iniciar a enviar y recibir los mensajes en tiempo real. Como se presenta el código de la clase ChatConnection en el listado 7.38, esta clase tiene todos los métodos necesarios para el intercambio de mensajes por medio de WebScokets.

```
1  package gcgu.jsfchat.service;
2
3  import java.io.IOException;
4  import java.util.Set;
5  import java.util.concurrent.CopyOnWriteArraySet;
6  import java.util.logging.Logger;
7
8  import jakarta.websocket.OnClose;
9  import jakarta.websocket.OnError;
10 import jakarta.websocket.OnMessage;
11 import jakarta.websocket.OnOpen;
12 import jakarta.websocket.Session;
13
14 public class ChatConnection {
15
16     private static final Logger LOG = Logger.getLogger("ChatConnection");
17     private static final Set<ChatConnection> connections = new
       CopyOnWriteArraySet<>();
18     private String nickname;
19     private Session session;
20
21     public ChatConnection() {
22     }
23
24     @OnOpen
25     public void start(Session session) {
26         LOG.info("Incoming connection");
27         this.session = session;
28         connections.add(this);
29         String message = String.format("%s %s", nickname, "se ha unido.");
30         broadcast(message);
31     }
32
33     @OnClose
34     public void end() {
35         LOG.info("Ended connection");
36         connections.remove(this);
37         String message = String.format("%s %s", nickname, "se ha
       desconectado.");
38         broadcast(message);
```

```
39      }
40
41      @OnMessage
42      public void incoming(String message) {
43          LOG.info("Mensaje entrante");
44          // No se debe confiar en el cliente
45          String filteredMessage = String.format("%s: %s", nickname, message.
    toString());
46          broadcast(filteredMessage);
47      }
48
49      @OnError
50      public void onError(Throwable t) throws Throwable {
51          LOG.info("Chat error: " + t.getMessage());
52      }
53
54      private static void broadcast(String msg) {
55          for (ChatConnection client : connections) {
56              try {
57                  //Por el acceso simultáneo
58                  synchronized (client) {
59                      client.session.getBasicRemote().sendText(msg);
60                  }
61              } catch (IOException e) {
62                  LOG.info("Failed to send message to client: " + e.
    getMessage());
63                  connections.remove(client);
64                  try {
65                      client.session.close();
66                  } catch (IOException e1) {
67                  }
68                  String message = String.format("%s %s", client.nickname, "
    Se ha desconectado.");
69                  broadcast(message);
70              }
71          }
72      }
73  }
```

Listado 7.38: Clase Conection

Clase ChatRoom

El objetivo que cumple la clase `ChatRoom` es el de gestionar la información acerca de la comunicación que se está dando. Es decir, nuevos usuarios que se unen, los mensajes enviados (histórico de mensajes), entre otras funciones (ver lisatdo de código 7.39

```
1  package gcgu.jsfchat.service;
2
3  import java.time.LocalDateTime;
4  import java.time.format.DateTimeFormatter;
5  import java.util.logging.Logger;
```

```
6
7  import jakarta.annotation.PostConstruct;
8  import jakarta.enterprise.context.ApplicationScoped;
9  import jakarta.faces.push.Push;
10 import jakarta.faces.push.PushContext;
11 import jakarta.inject.Inject;
12
13 import gcgu.jsfchat.model.User;
14
15 @ApplicationScoped
16 public class ChatRoom {
17
18     private static final DateTimeFormatter DATE_FORMAT =
19             DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");
20
21     @Inject
22     private Logger LOG;
23
24     // Sobre la marcha crea un endpoint
25     @Inject
26     @Push(channel = "chatChannel")
27     private PushContext chatChannel;
28
29     @Inject
30     @Push(channel = "clock")
31     private PushContext clockChannel;
32
33     private String chatHistory = "";
34
35     public ChatRoom() {
36     }
37
38     @PostConstruct
39     public void postConstruct() {
40         if (chatChannel != null) {
41             LOG.info("Channel: " + chatChannel);
42         } else {
43             LOG.info("El canal es nulo");
44         }
45     }
46
47     public void notifyNewUser(User user) {
48         LOG.info("Un nuevo uusuario en la sala: " + user.getNickname());
49     }
50
51     public String getChatHistory() {
52         return chatHistory;
53     }
54
55     public void broadcastMessage(String user, String message) {
56         String fullMessage = String.format("%s: %s", user, message);
57         LOG.info("Broadcasting: " + fullMessage);
58         chatChannel.send(fullMessage);
```

```
59          chatHistory = chatHistory + "\n" + fullMessage;
60      }
61
62      public void broadcastClock() {
63          clockChannel.send(LocalDateTime.now().format(DATE_FORMAT));
64      }
65  }
```

Listado 7.39: Clase ChatRoom

Clase ChatRoomOberver

```
1   package gcgu.jsfchat.service;
2
3   import java.util.logging.Logger;
4
5   import jakarta.enterprise.context.ApplicationScoped;
6   import jakarta.enterprise.event.Observes;
7   import jakarta.faces.event.WebsocketEvent;
8   import jakarta.faces.event.WebsocketEvent.Closed;
9   import jakarta.faces.event.WebsocketEvent.Opened;
10  import jakarta.inject.Inject;
11
12  //Por ser visible y actualizable por todos los usuarios
13  @ApplicationScoped
14  public class ChatRoomObserver {
15
16      @Inject
17      private Logger LOG;
18
19      public void onOpen(@Observes @Opened WebsocketEvent event) {
20          LOG.info("Conexión abierta " + event.getChannel() + " from " +
21                   event.getUser());
22      }
23
24      public void onClose(@Observes @Closed WebsocketEvent event) {
25          String channel = event.getChannel();
26          LOG.info("Canal " + channel + " ha sido cerrado correctamente!");
27          LOG.info("Conesión cerrada " + event.getChannel() + " from " +
28                   event.getUser() + " con el código " + event.getCloseCode()
    );
29      }
30  }
```

Listado 7.40: Clase ChatRoomOberver

Controlador de la página Index

Para el controlador se debe crear una clase con la notación `jakarta.enterprise.inject.Model`. El código de esta clase se muestra en el listado 7.41. Encargada de la

validación de datos de usuarios, así como de su gestión (gestión de datos de cookies).

```
1  package gcgu.jsfchat.controller;
2
3  import java.util.logging.Logger;
4
5  import jakarta.enterprise.inject.Model;
6  import jakarta.faces.application.FacesMessage;
7  import jakarta.faces.context.FacesContext;
8  import jakarta.inject.Inject;
9
10 import gcgu.jsfchat.CookieHelper;
11 import gcgu.jsfchat.UsernameNotUniqueException;
12 import gcgu.jsfchat.db.UserDAO;
13 import gcgu.jsfchat.model.User;
14
15 @Model
16 public class IndexController {
17    @Inject
18    private Logger LOG;
19    private static final String COOKIE_REMEMBERME = "jsf-chat-rme";
20
21    @Inject
22    private FacesContext facesContext;
23    @Inject
24    private CookieHelper cookieHelper;
25    @Inject
26    private UserDAO userDao;
27    @Inject
28    private UserBean userBean;
29
30    private User user;
31    private boolean rememberMe;
32    private boolean userAlreadySignedUp;
33
34    public User getUser() {
35       if (user == null) {
36          if (cookieHelper.containsCookie(COOKIE_REMEMBERME)) {
37             user = startSessionFromCookie();
38          }
39          if (user == null) {
40             user = new User();
41          }
42       }
43       return user;
44    }
45
46    public String getMessage() {
47       return "Hello JSF 2.3!";
48    }
49
50    public boolean isRememberMe() {
51       return rememberMe;
```

```
52      }
53
54      public void setRememberMe(boolean rememberMe) {
55          this.rememberMe = rememberMe;
56      }
57
58      public boolean isUserAlreadySignedUp() {
59          return userAlreadySignedUp;
60      }
61
62      public void verifyNicknameUniqueness() {
63          String displayAttribute = "";
64          if (userDao.isNicknameUsed(user.getNickname())) {
65              displayAttribute = "block";
66          } else {
67              displayAttribute = "none";
68          }
69          // just to show how to execute a JS
70          facesContext.getPartialViewContext().getEvalScripts()
71                  .add("document.getElementById('nickname-error').style.display='
    " +
72                          displayAttribute + "'");
73      }
74
75      public String signIn() {
76          try {
77              userDao.createUser(user);
78              if (this.rememberMe) {
79                  createCookieToUser();
80              }
81              userBean.startSessionFor(user);
82              facesContext.addMessage(null, new FacesMessage(FacesMessage.
    SEVERITY_INFO,
83                      "All set", "User registered successfully."));
84              return goToChatRoom();
85          } catch (UsernameNotUniqueException ex) {
86              facesContext.addMessage(null, new FacesMessage(FacesMessage.
    SEVERITY_ERROR,
87                      "Nickname is already in use", "Please, insert another nickname
    ."));
88          }
89          return null;
90      }
91
92      public String goToChatRoom() {
93          return "chat-room?faces-redirect=true";
94      }
95
96      private void createCookieToUser() {
97          String textValue = String.format("%s;%s", user.getNickname(), user.
    getName());
98          cookieHelper.setCookie(COOKIE_REMEMBERME, textValue);
99      }
```

```
100
101    private User startSessionFromCookie() {
102       String cookieValue = cookieHelper.getCookieValue(COOKIE_REMEMBERME);
103       if (cookieValue != null && cookieValue.contains(";")) {
104          LOG.info("Cookie: " + cookieValue);
105          User user = createUserFromCookie(cookieValue);
106          // just a workaround to in-memory DB
107          if (!userDao.isNicknameUsed(user.getNickname())) {
108             userDao.createUser(user);
109          }
110          userBean.startSessionFor(user);
111          userAlreadySignedUp = rememberMe = true;
112          return user;
113       }
114       return null;
115    }
116
117    private User createUserFromCookie(String cookieValue) {
118       String[] tokens = cookieValue.split(";");
119       return new User(tokens[0], tokens[1]);
120    }
121 }
```

Listado 7.41: Controlador de la página Index.xhtml

Página `index.xhtml`

El contenido de la interfaz inicial o el index de la aplicación se muestra en 7.42. GUI de bienvenida e identificación del usuario.

```
1  <html xmlns="http://www.w3.org/1999/xhtml"
2        xmlns:h="http://xmlns.jcp.org/jsf/html"
3        xmlns:f="http://xmlns.jcp.org/jsf/core"
4        xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
5        xmlns:jsf="http://xmlns.jcp.org/jsf">
6      <head jsf:id="head">
7          <meta charset="utf-8" />
8          <meta name="viewport" content="width=device-width,
9                initial-scale=1" />
10         <title>Welcome to JSF 2.3 Chat!</title>
11         <h:outputStylesheet library="css" name="bootstrap.min.css" />
12     </head>
13     <body jsf:id="body">
14         <div class="container">
15             <div class="jumbotron">
16                 <h1>Welcome to JSF 2.3 Chat</h1>
17                 <p>#{indexController.message}</p>
18             </div>
19             <div class="row">
20                 <div class="col-sm-12 center-block"><h2>Enter your
    identification to begin:</h2></div>
21             </div>
```

```
22            <form class="form-horizontal" jsf:id="form"
23                  jsf:prependId="false">
24                <div class="form-group">
25                    <label for="nickname">Nickname</label>
26                    <input type="text" class="form-control"
27                           jsf:id="nickname" jsf:name="nickname"
28                           jsf:value="#{indexController.user.nickname}"
29                           jsf:required="true">
30                        <f:ajax event="blur" execute="@this" listener=
31                            "#{indexController.verifyNicknameUniqueness()}"
     />
32                    </input>
33                    <h:message errorClass="form-control alert-danger"
34                               for="nickname" />
35                    <span id="nickname-error" class="form-control
36                            alert-danger" style="display:none">
37                        Nombre de usuario ya está en uso.</span>
38                </div>
39                <div class="form-group">
40                    <label for="name">Nombre</label>
41                    <input type="text" class="form-control"
42                           jsf:id="name" jsf:name="name" jsf:value=
43                           "#{indexController.user.name}"
44                           jsf:required="true" />
45                    <h:message errorClass="form-control alert-danger"
46                               for="name" />
47                </div>
48                <div class="form-group">
49                    <label for="rememberme">Remember me?</label>
50                    <input type="checkbox" jsf:id="rememberme"
51                           jsf:name="rememberme" jsf:value="#{
    indexController.rememberMe}" />
52                </div>
53                <div class="form-group">
54                    <input type="submit" class="btn btn-primary"
55                           value="Signin" jsf:rendered="#{not
    indexController.userAlreadySignedUp}" jsf:action="#{indexController.
    signIn()}">
56                    </input>
57                    <h:link styleClass="btn btn-success" rendered=
58                            "#{indexController.userAlreadySignedUp}"
59                            outcome="chat-room?faces-redirect=true">
60                        Ir a la sala de Chat
61                    </h:link>
62                </div>
63                <div class="form-group">
64                    <h:messages globalOnly="true" errorClass=
65                        "alert alert-danger" warnClass="alert
66                        alert-warning" infoClass="alert alert-success"
67                        style="padding-left:0px;list-style:none;" />
68                </div>
69            </form>
70        </div>
```

```
71      </body>
72  </html>
```

Listado 7.42: Código XHTML de la página Index

Página chat-room.xhtml

La página chat-room.xhtml se muestra su código xhtml en el listado 7.43 y su código JavaScript en el listado 7.44. Está página es en la que el usuario identificado o anónimo va a poder enviar los mensajes a toda la sala.

Primera parte: Código XHTML de la página chat-room.xhtml

```
1  <html xmlns="http://www.w3.org/1999/xhtml"
2        xmlns:h="http://xmlns.jcp.org/jsf/html"
3        xmlns:f="http://xmlns.jcp.org/jsf/core"
4        xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
5        xmlns:jsf="http://xmlns.jcp.org/jsf">
6      <head jsf:id="head">
7          <meta charset="utf-8" />
8          <meta name="viewport"
9                content="width=device-width, initial-scale=1" />
10         <title>Bienvenido al chat Versión 2.0!</title>
11         <h:outputStylesheet library="css" name="bootstrap.min.css" />
12     </head>
13     <body jsf:id="body">
14         <!-- El Socket JSF es sólo unidireccional: del Servidor al Cliente
       (Spec 10.4.1.7). Necesitaremos hacer una llamada AJAX para emitir un
       mensaje; Debe estar en el cuerpo -->
15         <f:websocket scope="application" channel="chatChannel"
16            onopen="handleOpen" onmessage="handleMessage"
17            onclose="handleClose" />
18
19         <f:websocket scope="application" channel="clock"
20            onopen="handleOpen" onmessage="handleMessage"
21            onclose="handleClose" />
22
23         <div class="container">
24             <div class="jumbotron">
25                 <h1>Bienvenido al Chat versión 2.0!</h1>
26             </div>
27             <form class="form-vertical" jsf:id="form"
28                   jsf:prependId="false">
29                 <div class="row">
30                     <div class="col-sm-12 center-block"><h2>Welcome #{
       userBean.name}, vamos conversando!</h2></div>
31                 </div>
32                 <div class="row">
33                     <div class="col-sm-12">
34                         <textarea id="chat-history"
```

```
35                                  class="form-control"
36                                  rows="10" cols="70" readonly="true"
37                                  style="resize:none;"></textarea>
38                      </div>
39                  </div>
40                  <div class="row">
41                      <div class="col-sm-12">
42                          <div class="input-group">
43                              <input type="text" class="form-control"
44                                  placeholder="Enter your message"
45                                  jsf:id="message" jsf:name="message"
46                                  jsf:value="#{chatController.message}"/>
47                              <span class="input-group-btn">
48                                  <button type="button"
49                                          class="btn btn-primary"
50                                          jsf:id="btn-send">
51                                      Send
52                                      <f:ajax event="click"
53                                          execute="@this message"
54                                          render="@this message chat-history"
55                                          listener=
56                                          "#{chatController.sendMessage()}" /
    >
57                                  </button>
58                                  <button type="button" class="btn btn-
    primary" jsf:id="btn-update">Clock
59                                      <f:ajax event="click" execute="@this"
    render="@this" listener="#{chatController.updateClock()}" />
60                                  </button>
61                              </span>
62                          </div>
63                      </div>
64                  </div>
65
66                  <div class="row">
67                      <div class="col-sm-12">
68                          <h:messages globalOnly="true"
69                              errorClass="alert alert-danger"
70                              warnClass="alert alert-warning"
71                              infoClass="alert alert-success" style="padding-
    left:0px;list-style:none;" />
72                      </div>
73                  </div>
74              </form>
75          </div>
76      </body>
```

Listado 7.43: Código XHTML de la página `chat-Room.xhtml` (primera parte del archivo)

Segunda parte: Código JavaScript de la página `chat-room.xhtml`

```
    <script>
        function handleOpen(event) {
            console.log('Openning ', event);
        }

        function handleChatChannelMessage(message, event) {
            const textarea = document.getElementById('chat-history');
            textarea.append(message);
            textarea.append('\n');
            textarea.scrollTop = textarea.scrollHeight;
        }

        function handleClockChannelMessage(message, event) {
            alert('Time is: ' + message)
        }

        function handleMessage(message, channelName, event) {
            if (channelName === 'chatChannel') {
                handleChatChannelMessage(message, event);
            } else if (channelName === 'clock') {
                handleClockChannelMessage(message, event);
            } else {
                alert('Unhandled channel: ' + channelName);
            }
        }

        function handleClose(code, channel, event) {
            console.log('Handle close: ' + code, channel, event);
            if (code == -1) {
                console.log('Websockets not supported by client')
            } else if (code == 1000) { // Normal close (as result of
   expired session or view)
                console.log('Normal close, will try to reconnect');
            } else {
                // Abnormal close reason (as result of an error)
                alert('Error during the reconnect!')
            }
        }
    </script>
</html>
```

Listado 7.44: Código JavaScript de la página `chat-room.xhtml` (segunda parte del archivo)

Consdideraciones

Se requiere que los desarrolladores conozcan las siguientes consideraciones:

- Especificaciones de JSF 2.3: A partir de esta versión, JSF debe ejecutarse en un contenedor que soporte CDI versión 2.0.
- Cuando se ejecuta en GassFish 5 a veces podemos obtener un `NullPointerException` (NPE) que podría estar relacionado con el porblema de *`AbstractMethodError`*

when <`f:websocket`> *is used #265* (para tratar de solucionar, se puede visitar https://github.com/ocpsoft/rewrite/issues/265)

Migración

Si se requieremigrar a otro framework de los favoritos del desarrollador:

- Cambie de importaciones de javax a jakarta.
- Cambie las referencias de javax a jakarta en los ficheros: web.xml', Context.xml y jsf_messages.properties

Pruebas

Este proyecto fue probado en:

- Java EE 8 (usando refs 'javax')
- Jakarta EE 8 (usando refs 'javax')
- Jakarta EE 9 (usando refs 'jakarta')

Enlaces

Para mayor ilustración de todo lo que involucran los WebSockets, y las demás herramientas que se han utilizado en este ejercicio sencillo, se puede visitar:

- https://www.javacodegeeks.com/2017/03/testing-java-ee-8-specifications.html
- https://dzone.com/articles/javaserver-faces-23-1
- https://javaserverfaces.java.net/nonav/2.3/whatsnew.html
- http://www.omnifaces-fans.org/p/jsf-23-tutorial.html
- http://arjan-tijms.omnifaces.org/p/jsf-23.html
- http://www.omnifaces-fans.org/2015/12/jsf-23-default-producer-example.html
- https://github.com/AnghelLeonard/JSF-2.3

7.7. Conclusiones

HTTP es un protocolo que permite que el cliente se comunique con el servidor y el servidor atienda sus requerimientos. Mientras HTTP satisface estas necesidades, no permite que el servidor se comunique con el cliente sin que éste lo haga primero. Para solucionar esta falta de comunicación (servidor-cliente) surgieron los WebSocket.

Los WebSockets son una tecnología eficiente para desarrollar aplicaciones en las cuales el cliente y el servidor requieren comunicarse entre sí independientemente. Los WebSockets permiten desarrollar aplicaciones realmente dinámicas, que tengan una comunicación bidireccional en tiempo real.

Las aplicaciones web desarrolladas usando WebSockets, evitan que los usuarios recarguen o se redirijan a otra página para obtener la información o recursos que el servidor envía al cliente o para que el cliente solicite al servidor. También es una tecnología que ahorra los recursos tecnológicos tanto del lado del servidor como del lado del cliente. Los WebScokets son flexibles al aceptar otros protocolos de comunicación que ayudan aumentar su nivel de eficiencia y seguridad.

Evaluación a los lectores

La evaluación es una actividad continua del mismo proceso formativo y de capacitación profesional. El lector debe demostrar el logro de los objetivos planteados para este capítulo dando respuesta a los siguientes problemas planteados:

- Dibuje una tabla para exponer las características del flujo HTTP y las del flujo de WebSockets, además de sus semejanzas, diferencias, ventajas (beneficios) y desventajas.
- Describir un caso de estudio, el cual requiere WebSockets para ser resuelto. Desarrolle una aplicación que resuelva el caso de estudio.

Capítulo 8

SERVICIOS DE MENSAJERÍA DE JAVA

Objetivos

Al finalizar la lectura y la práctica del contenido de está unidad, el lector será capaz de:

- Explicar las características de los servicios de mensajería Java.
- Identificar las ventajas y desventajas de JMS frente a otras tecnologías de comunicación entre aplicaciones.
- Integrar los dominios de mensajería (P2P y Pub/Sub) en el desarrollo de aplicaciones web.
- Usar las tecnologías de apoyo para producir y consumir mensajes síncronos y asíncronos.

Resumen

La API del servicio de mensajería de Java o JMS es un estándar de mensajería que permite a los componentes de la aplicación basados en Java EE crear, enviar, recibir y leer mensajes. Permite la comunicación distribuida que está débilmente acoplada, además, es confiable y asíncrona. En las aplicaciones JMS no existe comunicación directa; en su lugar, el remitente envía el mensaje a su destino, y el destinatario recibe el mensaje desde el depósito destino. Hay dos maneras de enviar y recibir mensajes con JMS. Mensajería punto a punto permite a un cliente enviar un mensaje a otro cliente a través de una abstracción intermedia llamada cola. La mensajería de publicación y suscripción permite a un cliente enviar un mensaje a varios clientes a través de una abstracción intermedia llamada tema.

8.1. Introducción

Los mensajes son solicitudes, informes o eventos asíncronos son consumidos por las aplicación empresarial, no por los humanos. Contienen información vital necesaria para coordinar estos sistemas. Contienen datos con un formato preciso que describen acciones empresariales específicas. A través del intercambio de estos mensajes, cada aplicación puede hacer crecer el negocio. [241, 242].

La API del servicio de mensajería de Java o JMS por sus siglas en inglés *Java Message Service* es un estándar de mensajería que permite a los componentes de la aplicación basados en *Java Platform Enterprise Edition (Java EE)* crear, enviar, recibir y leer mensajes. Permite la comunicación distribuida que está débilmente acoplada, además, es confiable y asincrónica. En versiones anteriores de Java EE, JMS era la única forma de invocar métodos EJB de forma asíncrona [242].

En versiones anteriores de Java EE, la única forma de invocar métodos EJB de forma asíncrona era utilizando beans dirigidos por mensajes (que se trata en la siguiente sección).

8.1.1. Definición

El JMS es una API de Java que permite a las aplicaciones crear, enviar, recibir y leer mensajes. Diseñada por Sun y varias empresas asociadas, la API JMS define un conjunto común de interfazs y una semántica asociada que permite a los programas escritos en Java comunicarse con otras softwares de mensajería [243]. Esta API reduce al mínimo el conjunto de conceptos que un programador debe aprender para utilizar los productos de mensajería, pero proporciona suficientes características sofisticadas para soportar este tipo de productos. También se esfuerza por maximizar la portabilidad de las aplicaciones JMS entre los proveedores JMS [242, 243].

8.1.2. Características

La API JMS permite una comunicación con las siguientes características [244]:

- Débilmente acoplada, lo que significa que todos sus componentes trabajan independientes entre sí.
- Comunicación asíncrona, lo que significa que un servidor JMS entrega un mensaje al cliente, pero el cliente no tiene que leerlo inmediatamente.
- Es fiable, lo que significa que un servidor JMS garantiza que un mensaje se entregue una vez y sólo una vez.

8.1.3. Ventajas y Desventajas

JMS proporciona algunas ventajas esenciales como [244, 245]:

- Persistencia en los mensajes. Los mensajes se encolan hasta que su destinatario los lee.
- Equilibro de la carga y seguridad.
- Comunicación asincrónica. El servidor no tiene que entablar conversación con el cliente para enviarle un mensaje.
- Entrega garantizada de los emisores a los receptores.

JMS es una buena manera que las aplicaciones se comuniquen entre sí, pero tiene algunas **desventajas** [244, 245]:

- Hay que enviar la cabecera y otra información con el contenido del mensaje, por lo que la cantidad total de información es mayor que el propio contenido del mensaje, lo que puede aumentar el tráfico de la red.
- Cada mensaje llega a los receptores a través de un servidor, lo que hace que la comunicación más lenta que una conexión directa.

8.2. Arquitectura

La API JMS permite que las aplicaciones Java EE se envíen mensajes entre sí. con Java EE 7 se introdujo JMS 2.0, una nueva versión de JMS que simplifica mucho el desarrollo de aplicaciones que involucran mensajería. En las aplicaciones JMS no existe comunicación directa; en su lugar, el remitente envía el mensaje a su destino, y el destinatario recibe el mensaje desde el depósito destino. El depósito destino del mensaje puede ser: una cola de mensajes cuando se utiliza el dominio de mensajería Punto a Punto (PTP), y un tema de mensajes cuando se utiliza el dominio de mensajería Publicar/Suscribir (pub/sub) [242].

Una aplicación JMS se compone de las siguientes partes [243]:

- Un proveedor JMS es un sistema de mensajería que implementa las interfazs JMS y proporciona características administrativas y de control.
- Los clientes JMS son los programas o componentes, escritos en el lenguaje de programación Java, que producen y consumen mensajes.
- Los mensajes son los objetos que comunican información entre los clientes JMS.
- Los objetos administrados son objetos JMS pre configurados creados por un administrador para el uso de los clientes. Los dos tipos de objetos administrados son los destinos y las fábricas de conexión.

- Los clientes nativos son programas que utilizan la API de cliente nativa de un producto de mensajería en lugar de la API JMS. Una aplicación creada por primera vez antes de que la API JMS estuviera disponible y modificada posteriormente es probable que incluya tanto clientes JMS como nativos.

En la figura 8.1 ilustra la forma en que interactúan estas partes. Las herramientas administrativas permiten enlazar los destinos y las fábricas de conexión en un espacio de nombres de la API *Java Naming and Directory Interface (JNDI)*. Un cliente JMS puede buscar los objetos administrados en el espacio de nombres y luego establecer una conexión lógica con los mismos objetos a través del proveedor JMS [242, 243].

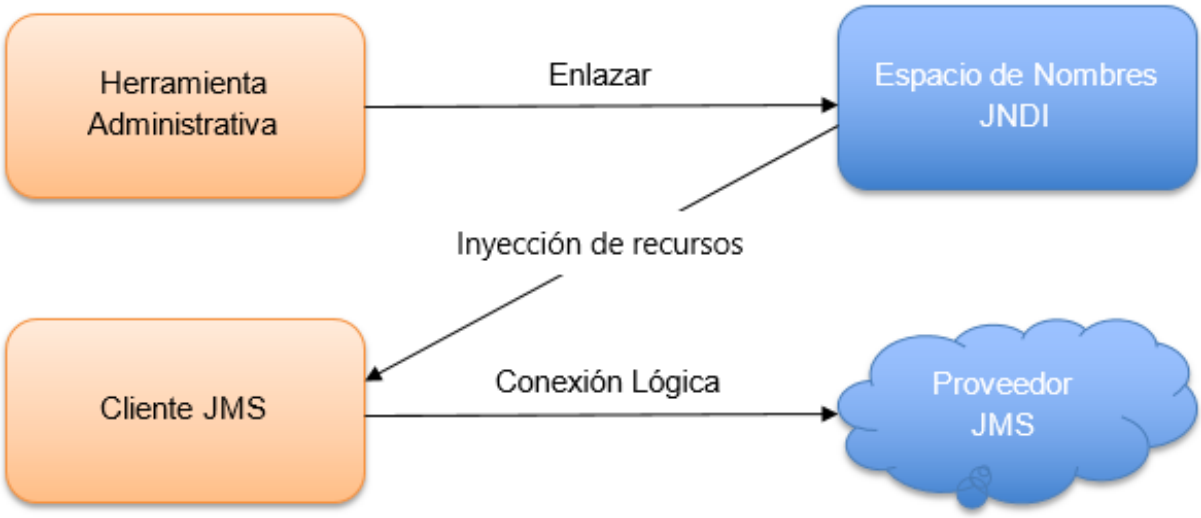


Figura 8.1: Interactión de una aplicación JMS

8.3. Dominios de mensajería

JMS admite dos tipos de modelos de mensajería: punto a punto y publicar-y-suscribir. Estos modelos de mensajería se denominan a veces dominios de mensajería. La mensajería punto a punto y la mensajería de publicación y suscripción se suelen abreviar p2p y pub/sub, respectivamente [246].

JMS soporta los dos principales estilos de mensajería proporcionados por los productos de mensajería empresarial [241]. A continuación se detalla cada uno.

8.3.1. Mensajería punto a punto (P2P)

Permite a un cliente enviar un mensaje a otro cliente a través de una abstracción intermedia llamada cola. El cliente que envía el mensaje lo envía a una cola específica. El cliente que recibe el mensaje lo extrae de esa cola como se muestra en la figura 8.2.

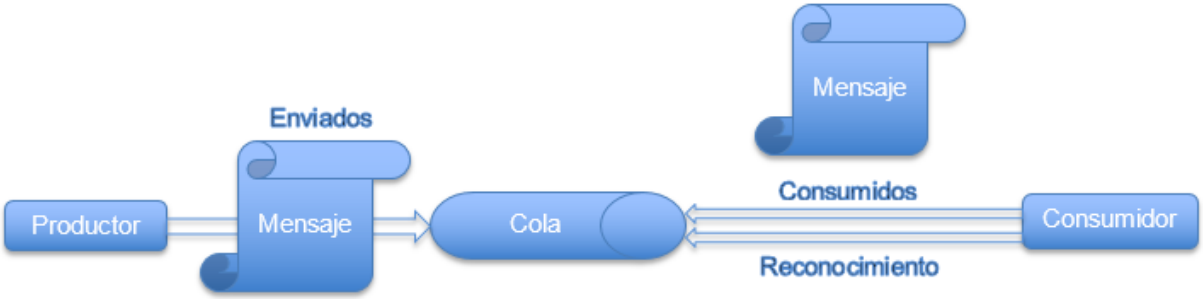


Figura 8.2: Interacción de una aplicación JMS Cola [247]

8.3.2. La mensajería de publicación y suscripción (pub/sub)

Permite a un cliente enviar un mensaje a varios clientes a través de una abstracción intermedia llamada tema. El cliente que envía el mensaje lo publica en un tema específico. El mensaje se entrega entonces a todos los clientes suscritos a ese tema, como se muestra en la figura 8.3.

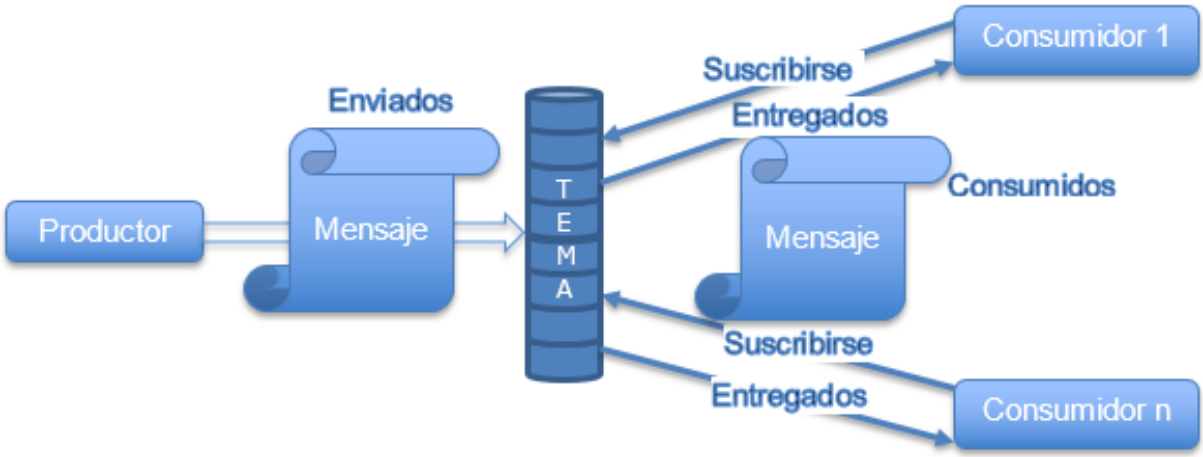


Figura 8.3: Interactión de una aplicación JMS Tema [247].

Al implementar la mensajería con el dominio de pub/sub (tema o *topic*), el mismo mensaje puede ser enviado a todos los suscriptores de un tema. Sin embargo, el envío de mensajes a estos dos tipos de recipientes es muy similar. En ambos se inyecta los recursos necesarios y realiza las llamadas correspondientes a la API de JMS [242, 248].

8.4. Consumo de mensajes

Una cuestión importante de JMS es que, el consumo de mensajes se define por las propiedades de temporización del sistema. Consumir un mensaje significa recibir el mensaje, leer el mensaje o tomar el mensaje del destino (cola o tema). Un mensaje es producido por un cliente de mensajería(emisor o productor), pero el cliente productor

no está interesado en cómo se consumirá el mensaje. Este proceso forma parte de la parte receptora, que es el objetivo del mensaje [244, 248].

La especificación JMS define dos formas de consumir un mensaje [243, 249]:

- **De forma síncrona**: Un consumidor o receptor obtiene explícitamente el mensaje del destino llamando al método de recepción. El método de recepción puede bloquearse hasta que llegue un mensaje o puede expirar si el mensaje no llega dentro de un límite de tiempo especificado [250].

  Los recursos JMS son una conexión y una sesión, normalmente combinados en un objeto `JMSContext`. En general, es importante liberar los recursos JMS cuando ya no se utilizan. Aquí hay algunas prácticas útiles a seguir.

  - Si se requiere mantener un recurso JMS sólo dentro de un método de negocio, se debe utilizar una sentencia try-with-resources para crear el `JMSContext` de forma que se cierre automáticamente al final del bloque `try`.
  - Para mantener un recurso JMS durante toda una transacción o solicitud, se debe inyectar un `JMSContext`. Esto también hará que el recurso sea liberado cuando ya no sea necesario.
  - Cuando se requiere mantener un recurso JMS durante toda la vida de una instancia de un *enterprise bean*, se puede utilizar un método de callback `@PostConstruct` para crear el recurso y un método de callback `@PreDestroy` para cerrar el recurso. Sin embargo, normalmente no es necesario hacer esto, ya que los servidores de aplicaciones suelen mantener un pool de conexiones. Si se utiliza un `bean` de sesión con estado y se desea mantener el recurso JMS en un estado de caché, se debe cerrar el recurso en un método de callback `@PrePassivate` y establecer su valor en `null`, y se debe crear de nuevo en un método de callback `@PostActivate`.

- **De forma asíncrona**: Un suscriptor puede registrar un escucha de mensajes con un consumidor, es similar a un escucha de eventos (*listener*). Cada vez que un mensaje llega a destino, el proveedor JMS entrega el mensaje llamando al método onMessage, que actúa sobre el contenido del mensaje [242, 250].

8.5. Uso de la API de JMS en aplicaciones Java EE

El desarrollo de aplicación empresarial utilizando Java se ha vuelto muy eficiente, gracias a la utilización de software, mediante la implementación de APIs. Una de estas APIs es la API de JMS. El uso de la API de JMS en aplicaciones de *Enterprise Bean* y aplicaciones web, así como las diferentes formas de uso según las aplicaciones cliente, se mostrarán a continuación.

8.5.1. Especificaciones generales

Una regla general que se aplica a los contenedores web y EJB es que, por cada conexión deben tener un solo objeto de sesión activo o abierto. Esta regla no se aplica a las aplicaciones cliente. El contenedor de las cliente de aplicación admite la creación de múltiples sesiones para cada conexión.

8.5.2. Creación de recursos JMS con anotaciones `@Resource` en aplicaciones Java EE

Las fabricas de conexiones y los destinos de la aplicación para componentes web o EJB pueden ser creados utilizando anotaciones. Este tipo de recursos (los creados con anotaciones) sólo son visibles para la aplicación para la que son creados.

Para los componentes de una aplicación cliente que utilicen la anotación `@Resource` los recursos JMS normalmente deben declararse como estáticos (ver listado 8.1).

```
1 @Resource(lookup = "jms/ConnectionFactory")
2 private static ConnectionFactory connectionFactory;
3
4 @Resource(lookup = "jms/Queue")
5 private static Queue queue;
```

Listado 8.1: Declaración de recursos JMS

Sin embargo, cuando se utilice esta anotación en un *Sesion Bean*, un *message-driven bean* (MDB) (*bean* controlado por mensajes) o un componente web, el recurso debe declararse como elemento dinámico de la clase (ver listado 8.2):

```
1 @Resource(lookup = "jms/ConnectionFactory")
2 private ConnectionFactory connectionFactory;
3
4 @Resource(lookup = "jms/Topic")
5 private Topic topic;
```

Listado 8.2: Declaración de recursos JMS

Con este mismo fin, también pueden utilizarse elementos de descriptor de despliegue para crear estos recursos. Los elementos especificados en el descriptor de implementación tienen prioridad sobre las anotaciones. Debe utilizar un descriptor de despliegue para crear recursos específicos para los clientes de la aplicación.

Java EE 7 y JMS 2.0 desarrollaron la anotación `@JMSDestinationDefinition` que permite crear de manera automática recursos JMS en el momento del despliegue, como se muestra en el listado 8.3:

```
1 import javax.ejb.Singleton;
2 import javax.ejb.Startup;
3 import javax.jms.JMSDestinationDefinition;
4 import javax.jms.JMSDestinationDefinitions;
5
6 @JMSDestinationDefinitions(
```

```
7     value = {
8             @JMSDestinationDefinition(
9                 name = "java:/queue/glegue-queue",
10                interfaceName = "javax.jms.Queue",
11                destinationName = "duke"),
12            @JMSDestinationDefinition(
13                name ="java:/topic/glegue-topic",
14                interfaceName = "javax.jms.Topic",
15                destinationName = "glegue-topic",
16                description = "La cola es usada para el
17                               propósito más simple")
18            }
19 )
20 @Startup
21 @Singleton
22 public class Inicializador {
23
24     @Resource(lookup = "java:/queue/glegue-queue")
25     private Queue queue;
26
27     @Produces
28     public Queue expose() {
29         return this.queue;
30     }
31 }
```

Listado 8.3: Clase `Inicializador`. Definición de destinos JMS

`@JMSDestinationDefinitions` permite crear los recursos JMS necesarios que serán posteriormente expuestos a la inyección de dependencia con la anotación CDI @Produces. Estos recursos se definen como el atributo *value* de `@JMSDestinationDefinitions` con la anotación `@JMSDestinationDefinition` [249].

Los atributos de `@JMSDestinationDefinition` `name`, `interfaceName` y `destinationName` son elementos requeridos, mientras que, opcionalmente se puede especificar una descripción con el atributo `description`.

Debe especificar la anotación sólo una vez para una aplicación determinada, en cualquiera de los componentes.

Nota:

Si coloca la anotación en un componente de envío, como un cliente en la aplicación, debe especificar el elemento `mappedName` para buscar el Topic, en lugar de utilizar la propiedad `destinationLookup` en la especificación de configuración de activación.

Cuando se inyecta el recurso en un componente, se usa el valor `name` del elemento en la anotación de definición como el valor del atributo `lookup` en la anotación `@Resource` como se muestra en el listado 8.4:

```
1 @Resource(lookup = "java:app/jms/gleguTopic")
2 private Topic topic;
```

Listado 8.4: Inyección de recursos JMS Topic

Los espacios de nombres JNDI portátiles disponibles son: global, app, module, comp, los cuales pueden usarse dependiendo de cómo esté empaquetada la aplicación. A continuación se describe cada uno [249, 251]:

- **java:global:** Hace que el recurso esté disponible para todas las aplicaciones implementadas
- **java:app:** Hace que el recurso esté disponible para todos los componentes en todos los módulos en una sola aplicación
- **java:module:** Hace que el recurso esté disponible para todos los componentes dentro de un módulo determinado (por ejemplo, todos los beans empresariales dentro de un módulo EJB)
- **java:comp:** Hace que el recurso esté disponible para un solo componente (excepto en una aplicación web, donde es equivalente a java:module)

Todos los ejemplos de Envío y recepción de mensajes mediante una aplicación web sencilla, Envío de mensajes desde un bean de sesión a un MDB y Uso de una entidad para unir mensajes de dos gestores de bases de datos o MDB por sus siglas en inglés (Management Data Base) utilizan la anotación `@JMSDestinationDefinition`. Los otros ejemplos de JMS no utilizan estas anotaciones. Los ejemplos que constan únicamente de clientes de aplicaciones no se implementan en el servidor de aplicaciones y, por lo tanto, deben comunicarse entre sí mediante recursos creados administrativamente que existen fuera de las aplicaciones individuales [248].

8.5.3. Uso de la inyección de recursos en *Enterprise Bean* o componentes web

Para la implementación de JMS se puede utilizar la inyección de recursos para inyectar tanto objetos administrados como objetos `JMSContext` en aplicaciones Java EE [249].

Inyección de una `ConnectionFactory`

Para la Inyección de una `ConnectionFactory`, `Queue` (cola) o `Topic` (tema) se utiliza la anotación `@Resource` en una aplicación Java EE. Estos objetos deben crearse administrativamente (desde el servidor de aplicaciones) antes de implementar su aplicación. Es posible que desee utilizar la fábrica de conexiones predeterminada, cuyo nombre JNDI es `java:comp/DefaultJMSConnectionFactory` [252].

Al utilizar la inyección de recursos en un componente de una aplicación cliente, se debe declarar el recurso JMS estático como se muestra en el listado 8.5:

```
1 @Resource(lookup = "java:comp/DefaultJMSConnectionFactory")
2 private static ConnectionFactory connectionFactory;
3
4 @Resource(lookup = "jms/MiCola")
5 private static Queue unaCola;
```

Listado 8.5: Inyección de recursos JMS Topic

En el listado 8.6, se muestra cuando se utiliza esta anotación en un *bean* de sesión. Este es un caso opuesto, un *bean* controlado por mensajes o un componente web, este debe ser dinámico, es decir, se declara el recurso como miembro de la clase al cuál se tiene acceso cuando se ha instanciado la clase (objeto en memoria).:

```
1 @Resource(lookup = "java:comp/DefaultJMSConnectionFactory")
2 private ConnectionFactory fabricaConexion;
3
4 @Resource(lookup = "jms/MiTema")
5 private Topic unTema;
```

Listado 8.6: Inyección de recursos JMS Topic

El error en la declaración de estos objetos, se producirían en tiempo de ejecución.

Inyectar un objeto JMSContext

Para acceder a un `JMSContext` en un *Enterprise bean* o componente web, en lugar de inyectar el recurso `ConnectionFactory` y luego crear un `JMSContext`, puede usar las anotaciones `@Inject` y `@JMSConnectionFactory`, de esta manera estará inyectando un `JMSContext`. Para usar la fábrica predeterminada de conexiones, use un código del listado 8.7:

```
1 @Inject
2 private JMSContext contextDesdeFabricaDefecto;
```

Listado 8.7: Inyección de recursos JMS Topic

Para usar una fábrica de conexiones personalizada, el desarrollador debe utilizar un código como el siguiente, cambiando los nombres de la ConnectionFactoryPropia y de la variable declarada como `JMSContext` (`contextDesdeFabricaPropia`. El código básico se muestra en el listado 8.8:

```
1 @Inject
2 @JMSConnectionFactory("jms/ConnectionFactoryPropia")
3 private JMSContext contextDesdeFabricaPropia;
```

Listado 8.8: Inyección de recursos JMS Topic

8.5.4. Uso de componentes Java EE para producir y recibir mensajes de forma sincrónica

Las aplicaciones que generan los mensajes o que los reciben de forma síncrona para realizar estas operaciones pueden utilizar componentes EJB o componentes web Java EE, como *beans* gestionados, un *servlet* o un *bean* de sesión. Por ejemplo, para enviar mensajes desde un *bean* de sesión a un MDB se utiliza un *bean* de sesión sin estado para enviar mensajes a un tema. Por otro lado, el Envío y la recepción de mensajes mediante una aplicación web simple utiliza *beans* administrados para producir y consumir mensajes [247, 249].

La recepción síncrona sin que se le especifique un tiempo de espera no suele ser el mejor diseño de aplicación para un componente web o EJB, debido a que inmoviliza los recursos del servidor. Por tal razón, al implementar una recepción síncrona se debe especificar un valor de tiempo de espera o utilizar un *bean* controlado por mensajes para recibir mensajes de forma asincrónica [249].

La implementación de la API de JMS en un componente Java EE es en muchos aspectos similar a su uso en una aplicación cliente. Las principales diferencias son los temas de gestión de recursos y transacciones.

Gestión de recursos JMS en componentes web y EJB

Los recursos JMS son una conexión y una sesión, normalmente combinadas en un objeto `JMSContext`. En general, es importante liberar recursos JMS cuando ya no se utilicen. A continuación, se muestran algunas prácticas útiles a seguir [246, 253]:

- Al requerir mantener un recurso JMS sólo durante la vida útil de un método de negocios, use una instrucción `try -with-resources` para crear el de modo JMSContext que se cierre automáticamente al final del bloque `try`.

- Si se necesita mantener un recurso JMS por todo el tiempo que dura la transacción o solicitud, se debe inyectar como `JMSContext`. Esto también hace que el recurso sea liberado cuando ya no sea necesario.

- Al requerir mantener un recurso JMS durante el tiempo de vida útil de una instancia de un *Enterprise Bean*, se debe utilizar un método `@PostConstruct`[1] para crear el recurso y un método `@PreDestroy` para cerrar el recurso. Sin embargo, normalmente no es necesario hacer esto, ya que los servidores de aplicaciones suelen mantener un conjunto de conexiones. Si usa un *bean* de sesión con estado y desea mantener el recurso JMS en un estado en caché, debe cerrar el recurso en un método `@PrePassivate` de devolución de llamada y establecer su valor en `null`, y debe crearlo nuevamente en un método `@PostActivate` [247, 253].

[1] Pre, post: métodos de devolución de llamada

Gestión de transacciones en beans de sesión

Las transacciones JTA[2] se pueden utilizar en lugar transacciones locales. Pueden ser transacciones gestionadas por contenedor o transacciones gestionadas por `bean`.Generalmente, se utilizan transacciones administradas por contenedor para métodos de `beans` que realizan envíos o recepciones, permitiendo que el contenedor EJB maneje la delimitación de transacciones. Las transacciones predeterminadas son las administradas por contenedor, por lo tanto, no es necesario que se las especifique. [246].

El uso de transacciones administradas por beans y los métodos de delimitaciones de transacciones de la interfaz `javax.transaction.UserTransaction`, se recomiendan restringir su uso para los desarrolladores con gran experiencia en su manejo, y hacerlo sólo si su aplicación lo requiere. Por lo general, las transacciones administradas por contenedores brindan el comportamiento más eficiente y correcto [248, 249].

8.5.5. Beans controlados por mensajes y la recepción asíncrona de mensajes

Primero se considera necesario definir lo que es un bean controlado por mensajes y cómo trabaja la API JMS con la plataforma Java EE.

Beans controlados por mensajes *(MDB: Message-Driven Beans)*

Un MDB es un escucha de mensajes quien puede recibir mensajes desde cualquiera de los orígenes, es decir, desde una cola o un tema. Los mensajes pueden ser enviados por cualquier componente Java EE (desde una aplicación cliente, otro *Enterprise bean* o un componente web) o desde una aplicación o un sistema de tecnología diferente a Java EE.

Ciclo de vida de un MDB

El contenedor EJB puede crear un conjunto de instancias del MDB. Para cada instancia, el contenedor EJB realiza las siguientes tareas [254]:

- Si un *bean* controlado por mensajes utiliza inyección de dependencias, el contenedor inyecta estas referencias antes de crear la instancia.
- El contenedor llama al método anotado `@PostConstruct`, si lo hay.

Al igual que un *bean* de sesión sin estado, un MDB no cambia al estado pasivo, es decir, sólo tiene dos estados (Ver figura 8.4):

1. inexistente y,
2. listo para recibir mensajes.

Figura 8.4: Ciclo de vida de un MDB

El método anotado `@PreDestroy` (si no hay), es llamado por el contenedor al final del ciclo de vida. En este momento, la instancia del *bean* está lista para ser recolectada por el recolector de basura [254].

La API JMS y la plataforma Java EE

La creación de la API JMS, tuvo como el objetivo más importante permitir a las aplicaciones Java acceder a los sistemas de *middleware* orientados a la mensajería (MOM) existentes. La versión 2.0 de JMS puede proporcionar una capacidad de mensajería completa para una empresa.

Entre los componentes de la plataforma Java EE está la API de JMS, la misma que los desarrolladores de aplicaciones pueden utilizar la mensajería con los componentes de Java EE. En la versión Java EE 8 viene incorporada la versión 2.0 de JMS.

Entre las principales características de la API de JMS son [255]:

- Las aplicaciones cliente, los componentes EJB y los componentes web pueden enviar o recibir mensajes JMS de manera sincrónica. Además, las aplicaciones cliente pueden establecer un escucha (listener) de mensajes permitiéndole recibir los mensajes JMS de forma asíncrona, esto hará que reciba una notificación cuando haya algún mensaje disponible.

- Los MDB, siendo un tipo de EJB, permiten el consumo asíncrono de mensajes en el contenedor EJB. los servidores de aplicaciones suelen agrupar a los MDB para implementar el procesamiento concurrente de mensajes.

- Al ser implementadas las operaciones de envío y recepción de mensajes como transacciones de la JTA, permiten que, en una sola transacción se den las operaciones JMS y los accesos a la base de datos.

[2]API de Transacciones de Java o *Java Transaction API*

Uso de MDB para recibir mensajes de manera asíncrona

Una vez descrito tanto, lo que es un MDB y la relación de la plataforma Java EE con JMS, se puede tener claro como la plataforma Java EE da soporte a un tipo especial de *Enterprise bean*, el MDB, que permite que las aplicaciones Java EE procesen mensajes JMS de forma asíncrona. Como se describe en subsección anterior, otros componentes Java EE web y EJB le permiten enviar y recibir mensajes de manera sincrónica pero no asincrónica [253].

Una clase de bean controlado por mensajes tiene los siguientes requisitos [249]:

- Debe ser decorada con la anotación `@MessageDriven` si no utiliza un descriptor de implementación.
- Debe definirse como `public`, pero no como `abstract` or `final`.
- Debe contener un constructor público sin argumentos.

Es recomendable (no obligatorio), que una clase de MDB implemente la interfaz de escucha de mensajes para el tipo de mensaje que admite. Un bean que admite la API JMS implementa la interfaz `javax.jms.MessageListener`. Esto significa que debe proporcionar un método `onMessage` con la siguiente interfaz:

```
1     void onMessage(Message inMessage)
```

Un mensaje es atendido por el *bean*, gracias a que el contenedor del *bean* llama al método `onMessage` cuando dicho mensaje a llegado. La lógica de negocios que con la cual se requiere procesar el mensaje debe ser implementada en este método. Por lo tanto, el MDB tiene como responsabilidad analizar el mensaje y ejecutar la lógica de negocios necesaria.

Un MDB se diferencia del escucha de mensajes de una aplicación cliente en los siguientes aspectos [253]:

- En un cliente de aplicación, debe crear un `JMSContext`, luego crear un `JMSConsumer`, luego llamar al método `setMessageListener` para activar el oyente. Para un bean controlado por mensajes, sólo necesita definir la clase y anotarla, y el contenedor EJB la creará automáticamente.
- La clase *bean* utiliza la anotación `@MessageDriven`, que normalmente contiene un elemento `activationConfig` que es decorada con anotaciones `@ActivationConfigProperty` para especificar las propiedades utilizadas por el bean o la fábrica de conexiones. Estas propiedades pueden incluir: fábrica de conexiones, tipo de destino, suscripción duradera, selector de mensajes o modo de reconocimiento.
- El contenedor del cliente de la aplicación tiene sólo una instancia de un `MessageListener`, que es llamado en un sólo subproceso a la vez. Sin embargo, si es

necesario que varios subprocesos llamen al MDB, entonces éste debe tener varias instancias, configuradas por el contenedor. Por lo tanto, los MDB pueden permitir un procesamiento de mensajes mucho más rápido que los escuchas de mensajes.

- No es necesario que se especifique un modo de reconocimiento de mensajes a menos que utilice transacciones administradas por *beans*. El mensaje es consumido en la transacción en la cual el método `onMessages` es invocado. En la tabla 8.1 se muestran las propiedades principales para la configuración de los MDB.

Tabla 8.1: Configuración de `@ActivationConfigProperty` para los MDB [249]

Propiedad	Definición
`acknowledgeMode`	Modo de acuse de recibo, utilizado sólo para las transacciones gestionadas por los *beans*; el valor por defecto es `Auto-acknowledge` (también se permite `Dups-ok-acknowledge`)
`destinationLookup`	Nombre de la cola o tema a buscar, del cual el *bean* recibirá mensajes.
`destinationType`	Uno entre `javax.jms.Queue` y `javax.jms.Topic`.
`subscriptionDurability`	Para las suscripciones duraderas, establezca el valor en `Durable`
`clientId`	En el caso de las suscripciones duraderas, el ID del cliente para la conexión (opcional)
`subscriptionName`	Para subscripciones duraderas, el nombre de la subscripción
`messageSelector`	La Cadena para el filtro de mensajes
`connectionFactoryLookup`	Nombre de la fábrica de conexiones para conectar con el proveedor JMS de los mensajes a recibir.

En el listado 8.9, se presenta una alternativa de código del MDB que se puede utilizar para recibir mensajes de forma asíncrona mediante un MDB:

```
1  @MessageDriven(activationConfig = {
2    @ActivationConfigProperty(propertyName = "destinationLookup",
3          propertyValue = "jms/MiCola"),
4    @ActivationConfigProperty(propertyName = "destinationType",
5          propertyValue = "javax.jms.Queue")
6  })
7  public class SimpleMessageBean implements MessageListener {
8      @Resource
9      private MessageDrivenContext mdc;
10     static final Logger logger =
11                         Logger.getLogger("SimpleMessageBean");
12
```

```
13      public SimpleMessageBean() {
14      }
15
16      @Override
17      public void onMessage(Message mensajeEntrada) {
18          try {
19              if (mensajeEntrada instanceof TextMessage) {
20                  logger.log(Level.INFO,
21                          "MESSAGE BEAN: Message received: {0}",
22                          mensajeEntrada.getBody(String.class));
23              } else {
24                  logger.log(Level.WARNING,
25                          "Message of wrong type: {0}",
26                          mensajeEntrada.getClass().getName());
27              }
28          }
29          catch (JMSException e)
30          {
31              logger.log(Level.SEVERE,
32              "SimpleMessageBean.onMessage: JMSException: {0}",
33                  e.toString());
34              mdc.setRollbackOnly();
35          }
36      }
37  }
```

Listado 8.9: Inyección de recursos JMS Topic

Al integrar a JMS con el servidor de aplicaciones mediante un adaptador de recursos, el adaptador de recursos JMS es el encargado de manejar estas tareas en lugar del contenedor EJB.

Comúnmente se inyecta un recurso `MessageDrivenContext` a la clase *bean*, para proporcionarle algunos métodos adicionales que puede utilizar para la gestión de transacciones, por ejemplo el método `setRollbackOnly`. El código 8.10 muestra como hacerlo [249, 253]

```
1       @Resource
2       private MessageDrivenContext mdc;
```

Listado 8.10: Inyección de recursos JMS Topic

Un MDB nunca tiene una interfaz local o remota. En cambio, sólo tiene una clase de *bean*.

Un MDB es similar a un *bean* de sesión sin estado en algunos aspectos: Las instancias de un MDB tienen un tiempo de vida relativamente corto y no retienen estado alguno para un cliente específico. La estructura de la instancia del MDB pueden contener algún estado en el manejo de los mensajes del cliente: por ejemplo, una conexión de base de datos abierta o una referencia de objeto a un objeto de *enterprise bean*.

Así como un *bean* de sesión sin estado, un MDB puede tener muchas instancias intercambiables ejecutándose al mismo tiempo. El contenedor debe agrupar estas instancias con el fin de permitir que los flujos de mensajes se procesen al mismo tiempo.

Además, intenta entregar mensajes en orden cronológico sin perjudicar la concurrencia del procesamiento de mensajes, sin embargo, no garantiza el orden exacto en el que se entregan los mensajes a las instancias de la clase MDB. Çuando el orden de la entrega de los mensajes es un requisito para una aplicación, se debe configurar al servidor de aplicaciones para que use sólo una instancia del MDB [249].

8.5.6. Gestión de transacciones JTA

Las aplicaciones cliente Java EE y los clientes Java SE (Edición estándar o *standard edition*) utilizan transacciones locales JMS. Éstas permiten la agrupación de envíos y recepciones dentro de determinada sesión JMS. Las aplicaciones Java EE que se ejecutan en el contenedor web o EJB pueden utilizar transacciones JTA para garantizar la integridad de los accesos a recursos externos. Cabe recalcar que, la diferencia principal entre ellas es que, una transacción JTA está controlada por los administradores de transacciones del servidor de aplicaciones. Las transacciones JTA se pueden distribuir, lo que significa que pueden abarcar varios recursos en la misma transacción, como un proveedor JMS y una base de datos [255,256].

En una aplicación Java EE que utiliza la API JMS, puede utilizar transacciones para combinar envíos o recepciones de mensajes con actualizaciones de bases de datos y otras operaciones del gestor de recursos. dentro de una sola transacción se puede acceder a los recursos de varios componentes de la aplicación. Por ejemplo, un servlet puede iniciar una transacción, acceder a varias bases de datos, invocar un *enterprise bean* que envía un mensaje JMS, invocar otro *enterprise bean* que modifica un sistema EIS (*Executive Information System* o Sistema Ejecutivo de Información) utilizando la arquitectura del conector y finalmente confirmar la transacción. Sin embargo, esta aplicación no puede enviar y recibibr mensaje JMS dentro de la misma transacción [249,253].

Las transacciones JTA dentro de los contenedores EJB y web pueden ser de dos tipos. estos tipos se muestran en la tabla 8.2 [249,257].

El uso del atributo `Required` se muestra en el código 8.11:

```
1  @TransactionAttribute(NOT_SUPPORTED)
2  @Stateful
3  public class TransactionBean implements Transaction {
4  ...
5      @TransactionAttribute(REQUIRES_NEW)
6      public void firstMethod() {...}
7
8      @TransactionAttribute(REQUIRED)
9      public void secondMethod() {...}
10
11     public void thirdMethod() {...}
12
13     public void fourthMethod() {...}
14 }
```

Listado 8.11: Inyección de recursos JMS Topic

Tabla 8.2: Tipos de transacciones JTA dentro de los contenedores EJB [249, 257]

Transacción JTA	Descripción
Transacciones gestionadas por contenedor	El contenedor EJB controla la integridad de sus transacciones sin la necesidad de llamar `commit` o `rollback`. Este tipo de transacciones (administradas por contenedores) son más fáciles de usar que las transacciones administradas por *beans*. Permite especificar los atributos adecuados de la transacción para sus métodos de *Enterprise Bean*. Se debe utilizar el atributo por defecto `Required` de la transacción con la finalidad de especificar de que un método siempre es parte de una transacción. Si hay una transacción en curso cuando se llama al método, el método será parte de esa transacción; de lo contrario, se iniciará una nueva transacción antes de que se llame al método y se confirmará cuando el método regrese.
Transacciones gestionadas por bean	Este tipo de transacciones deberían ser utilizadas con la interfaz `javax.transaction.UserTransaction`, ya que esta tiene los métodos propios `commity rollback` que pueden ser utilizados para establecer los límites de las transacciones. Las transacciones administradas por *beans*, como se mencionó antes, se recomiendan sólo para aquellos que tienen experiencia en el uso de transacciones dentro de la programación de la lógica de las aplicaciones.

Transacciones gestionadas por beans

En las transacciones gestionadas por beans, los límites de la transacción son especificados explícitamente por en la programación de la sesión o del MDB. Los beans con transacciones gestionadas por contenedor requieren menos codificación, sin embargo, tienen una limitación: Cuando un método se ejecuta, puede estar asociado a una sola transacción o a ninguna. Si esta limitación dificulta la codificación de un *bean*, se debe usar las transacciones gestionadas por el *bean*.

El siguiente pseudocódigo ilustra el tipo de control de **grano fino** que se puede obtener con las transacciones gestionadas por la aplicación. Comprobando varias condiciones, el pseudocódigo decide si iniciar o detener ciertas transacciones dentro del método de negocio:

En el pseudocódigo 8.12 se pretende mostrar como se pueden programar las transacciones gestionadas por la aplicación. Según se den las condiciones el pseudocódigo permite iniciar, ejecutar, detener o deshacer determinadas transacciones dentro del segmento de código de un método de negocios.

```
begin transaction
    ...
    update table-a
    ...
    if (condition-x)
        commit transaction
    else if (condition-y)
        update table-b
        commit transaction
    else
        rollback transaction
        begin transaction
            update table-c
        commit transaction
```

Listado 8.12: Inyección de recursos JMS Topic

Ambos tipos de transacciones, es decir, las transacciones gestionadas por contenedor o transacciones gestionadas por *beans* pueden utilizarse con *beans* controlados por mensajes. Al implementar transacciones administradas por contenedor con `Required` como atributo de la transacción en el método `onMessage` se asegura de que todos los mensajes son recibos y manejados dentro del contexto de una transacción.

El uso de transacciones administradas por contenedor, permite llamar a los siguientes métodos de `MessageDrivenContext` [249]:

- `setRollbackOnly`: Este método se utiliza para el manejo de errores. `setRollbackOnly` marca la transacción en el instante que ocurre una excepción, haciendo que el resultado sea una reversión de la transacción.
- `getRollbackOnly`: Este método sirve para monitorizar el estado de la transacción actual respecto al marcado o no para la reversión.

8.5.7. Diferencias entre las transacciones gestionadas por contenedor y gestionadas por *beans*

Aunque antes ya se han descrito algunas de las diferencias entre las transacciones gestionadas por contenedor y gestionadas por *beans*, aquí se pretende presentar de manera resumida esas diferencias.

Transacciones JTA gestionadas por contenedor	**Transacciones JTA gestionadas por *beans***
El contenedor EJB controla la integridad de sus transacciones sin la necesidad de llamar `commit` o `rollback`.	El programador controla las transacciones invocando los métodos: `UserTransaction.begin` dentro del método `onMessage` y finaliza cuando se llama `UserTransaction.commit` o `UserTransaction.rollback`.
El mensaje es recibido por el método `onMessage` del MDB dentro de la misma transacción.	La entrega de un mensaje al método `onMessage` lo hace fuera del contexto de la transacción JTA.
EL mensaje es procesado en el método `onMessage` dentro de la misma transacción.	Le permite procesar el mensaje utilizando más de una transacción o hacer que algunas partes del procesamiento del mensaje se realicen fuera de un contexto de transacción.
No se especifica la propiedad de configuración de activación `acknowledgeMode` cuando se crea un MDB. El contenedor acusa recibo del mensaje automáticamente cuando consigna la transacción.	Se puede establecer la propiedad de configuración de activación `acknowledgeMode` en `Auto-acknowledge` o `Dups-ok-acknowledge` para especificar cómo se requiere que sea reconocido el mensaje recibido por el MDB.

Tabla 8.3: Diferencias entre los Tipos de transacciones JTA [249, 257]

8.5.8. Métodos no permitidos en transacciones gestionadas por contenedores

Por las propiedades que se han expuesto en una de la subsecciones anteriores, los siguientes métodos se pueden interferir con los límites de transacción establecidos por el contenedor, por lo tanto, quedan prohibidos ser invocados por ninguno de los siguientes métodos:

- Los métodos `commit`, `setAutoCommity` y `rollback` y los de la clase `java.sql.Connection`
- El método `getUserTransaction` de `javax.ejb.EJBContext`
- Cualquier método de `javax.transaction.UserTransaction`

Sin embargo, se pueden utilizar estos métodos para establecer límites en las transacciones administradas por la aplicación.

8.6. Ejercicios

Una aplicación JMS será tan simple o compleja como sean sus requisitos de negocio. Igual que con conector a base de datos para Java o JDBC por sus siglas en ingles (Java Database Connectivity), es común aislar el código JMS mediante componentes o en su propia capa. Para la construcción de una aplicación que implemente JMS, se deben seguir básicamente los siguientes pasos: [258].

1. Adquirir una factoría de conexión.
2. Crear una conexión mediante la factoría de conexión.
3. Comenzar la conexión.
4. Crear una sesión a partir de la conexión.
5. Adquirir un destino.
6. Dependiendo de si enviamos o recibimos.
   - Crear un productor.
     a) Crear un productor.
     b) Crear un mensaje y adjuntarlo a su destino.
   - Crear un consumidor.
     a) Crear un consumidor.
     b) Opcionalmente registrar un *listener* (escucha) de mensajes.
7. Enviar/Recibir el/los mensaje/s.
8. Cerrar los objetos (consumidor, productor, sesión, conexión).

8.6.1. Inyección de objetos

La inyección de objetos se da al decorar a las variables `connectionFactory` y `chatQueue` con la notación `@Resource`. A partir de `connectionFactory` se crean el objeto `Connection` JMS y el session JMS. El código se muestra en el listado 8.13.

```
1 package webtec.gcgu.jms.inject;
2
3 import javax.annotation.Resource;
4 import javax.ejb.Stateless;
5 import javax.jms.Connection;
```

```
6  import javax.jms.ConnectionFactory;
7  import javax.jms.DeliveryMode;
8  import javax.jms.JMSException;
9  import javax.jms.MessageConsumer;
10 import javax.jms.MessageProducer;
11 import javax.jms.Queue;
12 import javax.jms.Session;
13 import javax.jms.TextMessage;
14
15 //No necesita guardar datos de conexiones anteriores
16 @Stateless
17 public class Messages {
18     //Inyección de recursos
19     @Resource
20     private ConnectionFactory connectionFactory;
21     @Resource
22     private Queue chatQueue;
23
24     public void sendMessage(String text) throws JMSException {
25         Connection jmsConnection = null;
26         Session jmsSession = null;
27
28         try {
29             jmsConnection = connectionFactory.
30                             createConnection();
31             jmsConnection.start();
32
33             // Contruye el objeto session
34             jmsSession = jmsConnection.createSession(false,
35                       Session.AUTO_ACKNOWLEDGE);
36
37             /* Construye el objeto Productor de mensajes desde
38              * la Sesión al Topic o Queue
39              */
40             MessageProducer messageProducer =
41                           jmsSession.createProducer(chatQueue);
42             messageProducer.
43                 setDeliveryMode(DeliveryMode.NON_PERSISTENT);
44
45             // Construye un mensaje desde la sesión
46             TextMessage textMessage =
47                         jmsSession.createTextMessage(text);
48
49             //El productor envía el mensaje
50             messageProducer.send(textMessage);
51         } finally {
52             // Limpia o cierra los objetos
53             if (jmsSession != null) {
54                 jmsSession.close();
55             }
56             if (jmsConnection != null) {
57                 jmsConnection.close();
58             }
```

```
59          }
60      }
61
62      /* Método que recibe los mensajes. Muchas tareas son comunes
63       * entre enviar y recibir mensajes.
64       */
65      public String receiveMessage() throws JMSException {
66         Connection jmsConnection = null;
67         Session jmsSession = null;
68         MessageConsumer messageConsumer = null;
69         try {
70             jmsConnection = connectionFactory.createConnection();
71             jmsConnection.start();
72
73             //Crea una sesión
74             jmsSession = jmsConnection.createSession(false,
75                                     Session.AUTO_ACKNOWLEDGE);
76
77             /* Construye un consumidor de mensajes desde la
78              * sesión al Topic o Queue
79              */
80             messageConsumer = jmsSession.createConsumer(chatQueue);
81
82             // Espera por el mensaje
83             TextMessage textMessage =
84                         (TextMessage) consumer.receive(1000);
85
86             return textMessage.getText();
87         } finally {
88             if (messageConsumer != null) {
89                 messageConsumer.close();
90             }
91             if (jmsSession != null) {
92                 jmsSession.close();
93             }
94             if (jmsConnection != null) {
95                 jmsConnection.close();
96             }
97         }
98      }
99  }
```

Listado 8.13: Clase con inyección de recursos

Implementación de JMS usando JNDI

Según los requisitos de la aplicación, se puede conectar a cualquier servidor JMS utilizando JNDI para localizar un origen (fábrica) de conexiones JMS existente. La URL de conexión puede comenzar con la cadena de búsqueda cuando no se conoce directamente la vinculación de la fábrica de conexiones o caso contrario puede comenzar con la cadena JNDI. Los archivos `jar` adecuados del proveedor JNDI deben estar en la

ruta de las clases de tiempo de ejecución. Con el servidor de aplicaciones **GassFish**, por lo tanto, se deben copiar esos archivos en el directorio `lib`. Además, los archivos `jar` de clientes de proveedores JMS es necesario que se encuentren en la ruta de las clases de tiempo de ejecución [259].

La API de JMS ha sido implementada por algunas compañías o autores. Estos autores lo que hacen es disminuir un poco más la complejidad para la implementación final. En este caso se pretende presentar una implementación utilizando la mensajería Solace[3] que proporciona un servicio JNDI para facilitar la integración con las aplicaciones personalizadas. Para más detalle de la implementación de mesagería solace para obtener objetos JMS se puede encontrar en el tutorial[4] de solace usando JNDI. Este y otros ejemplos se pueden revisar y descargar para probar desde github[5].

En el listado 8.14 se presenta una posible implementación de la clase productora de los mensajes que alimenta a una cola (QUEUE).

```
1
2  package package webtec.gcgu.jms.jndi
3
4  import javax.jms.Connection;
5  import javax.jms.ConnectionFactory;
6  import javax.jms.DeliveryMode;
7  import javax.jms.Message;
8  import javax.jms.MessageProducer;
9  import javax.jms.Queue;
10 import javax.jms.Session;
11 import javax.jms.TextMessage;
12
13 import javax.naming.Context;
14 import javax.naming.InitialContext;
15
16 import java.util.Hashtable;
17 /**
18  * Envía un mensaje persistente a una cola utilizando la imple-
19  * mentación de la API JMS de Solace.
20  *
21  * La cola utilizada para los mensajes debe existir en el bro-
22  * ker de mensajes.
23  */
24 public class QueueProducerJNDI {
25
26     final String QUEUE_NAME = "Q/tutorial";
27     final String QUEUE_JNDI_NAME = "/JNDI/" + QUEUE_NAME;
28     final String CONNECTION_FACTORY_JNDI_NAME =
29                                     "/JNDI/CF/GettingStarted";
30
31     public void run(String [] args) throws Exception {
```

[3]https://www.solace.dev/

[4]https://tutorials.solace.dev/jms/using-jndi/

[5]https://github.com/SolaceSamples/solace-samples-jms/blob/master/src/main/java/com/solace/-samples/

```
32
33          String[] split = args[1].split("@");
34
35          String host = args[0];
36          String vpnName = split[1];
37          String username = split[0];
38          String password = args[2];
39
40          System.out.printf("QueueProducerJNDI se conecta a la" +
41                            "mensajería Solace en %s...%n", host);
42
43          /* configurar las variables de entorno para la creación"
44           * del contexto inicial
45           */
46          Hashtable<String, Object> hash = new Hashtable<String,
47                                                         Object>();
48          // Uso de la fábrica de contexto inicial de Solace JNDI
49          hash.put(InitialContext.INITIAL_CONTEXT_FACTORY,
50          "com.solacesystems.jndi.SolJNDIInitialContextFactory");
51
52          // Parámetros de conexión de la mensajería de Solace
53          hash.put(InitialContext.PROVIDER_URL, host);
54
55          // Usuario con el formato: user@message-vpn
56          hash.put(Context.SECURITY_PRINCIPAL, username +
57                                               '@' + vpnName);
58          hash.put(Context.SECURITY_CREDENTIALS, password);
59
60          /* Crea el contexto inicial que se utilizará para buscar
61           * los objetos administrados por JMS.
62           */
63          InitialContext initialContext = new InitialContext(hash);
64          // Búsqueda de la fabrica de conexiones
65          ConnectionFactory connectionFactory = (ConnectionFactory)
66          initialContext.lookup(CONNECTION_FACTORY_JNDI_NAME);
67
68           // Conexión a la mensajería Solace
69           Connection connection =
70                       connectionFactory.createConnection();
71
72           // Crear una sesión no transaccionada y con auto ACK.
73           Session session = connection.createSession(false,
74                               Session.AUTO_ACKNOWLEDGE);
75
76           System.out.printf("Conectado a la VPN de mensajería " +
77               " de '%s' con nombre de usario cliente '%s'.%n",
78               vpnName, username);
79
80           // Búsqueda de la cola.
81           Queue queue = (Queue) initialContext.
82                                        lookup(QUEUE_JNDI_NAME);
83
84           // Productor de mensajes para la cola
```

```
85          MessageProducer messageProducer =
86                                  session.createProducer(queue);
87
88          // Mensaje de texto.
89          TextMessage message =
90              session.createTextMessage("Hello world Colas!");
91
92          System.out.printf("Enviando mensaje '%s' a la cola " +
93              "'%s'...%n", message.getText(), queue.toString());
94
95          // Enviar el mensaje
96          /* NOTE: El bus de mensajes Solace no admite la prio-
97           * ridad de los mensajes JMS
98           */
99          messageProducer.send(queue, message,
100                               DeliveryMode.PERSISTENT,
101                               Message.DEFAULT_PRIORITY,
102                               Message.DEFAULT_TIME_TO_LIVE);
103
104         System.out.println("Sent successfully. Exiting...");
105
106         /* Cerrar todo en el orden inverso al de apertura
107          * NOTA: como las interfaces de abajo extienden Auto-
108          * Closeable, con ellas es posible utilizar la sen-
109          * tencia Java "try-with-resources
110          */
111         messageProducer.close();
112         session.close();
113         connection.close();
114         /* El contexto inicial tiene que estar cerrado; no
115          * extiende AutoCloseable
116          */
117         initialContext.close();
118     }
119
120     public static void main(String [] args) throws Exception {
121         if (args.length != 3 || args[1].split("@").length != 2)
122         {
123             System.out.println("Forma: QueueProducerJNDI " +
124                 "<host:port> <client-username@message-vpn> " +
125                 "<client-password>");
126             System.out.println();
127             System.exit(-1);
128         }
129         if (args[1].split("@")[0].isEmpty()) {
130             System.out.println("Username no ingresado");
131             System.out.println();
132             System.exit(-1);
133         }
134         if (args[1].split("@")[1].isEmpty()) {
135             System.out.println("Message-vpn no ingresada");
136             System.out.println();
137             System.exit(-1);
```

```
138         }
139         new QueueProducerJNDI().run(args);
140     }
141 }
```

Listado 8.14: Clase con inyección de recursos

La cliente QUEUE, consumidora de los mensajes se muestra en el listado 8.15.

```
1  package package webtec.gcgu.jms.jndi;
2
3  import javax.jms.Connection;
4  import javax.jms.ConnectionFactory;
5  import javax.jms.JMSException;
6  import javax.jms.Message;
7  import javax.jms.MessageConsumer;
8  import javax.jms.MessageListener;
9  import javax.jms.Queue;
10 import javax.jms.Session;
11 import javax.jms.TextMessage;
12
13 import javax.naming.Context;
14 import javax.naming.InitialContext;
15
16 import com.solacesystems.jms.SolJmsUtility;
17 import com.solacesystems.jms.SupportedProperty;
18
19 import java.util.Hashtable;
20 import java.util.concurrent.CountDownLatch;
21 /**
22  * Recibe un mensaje persistente de una cola utilizando la
23  * implementación de la API JMS de Solace. La conexión con
24  * el enrutador de mensajes de Solace se configura mediante
25  * JNDI.
26  *
27  * La cola utilizada para los mensajes debe existir en el
28  * broker de mensajes (servidor Glassfish por ejemplo).
29  */
30 public class QueueConsumerJNDI {
31
32     final String QUEUE_NAME = "Q/tutorial";
33     final String QUEUE_JNDI_NAME = "/JNDI/" + QUEUE_NAME;
34     final String CONNECTION_FACTORY_JNDI_NAME =
35                                     "/JNDI/CF/GettingStarted";
36
37     // Latch utilizado para la sincronización entre hilos
38     final CountDownLatch latch = new CountDownLatch(1);
39
40     public void run(String [] args) throws Exception {
41
42        String[] split = args[1].split("@");
43
44        String host = args[0];
45        String vpnName = split[1];
```

```
46          String username = split[0];
47          String password = args[2];
48
49          System.out.printf("QueueConsumerJNDI conectándose a
50                          Mensajería Solace de %s...%n", host);
51
52          /* Configurar las variables de entorno para la
53           * creación del contexto inicial
54           */
55          Hashtable<String, Object> hastTable =
56                              new Hashtable<String, Object>();
57          // Usar la fábrica de contexto inicial de Solace JNDI
58          hastTable.put(InitialContext.INITIAL_CONTEXT_FACTORY,
59          "com.solacesystems.jndi.SolJNDIInitialContextFactory");
60
61          /* Asignar los parámetros de conexión del enrutador de
62           * mensajes Solace
63           */
64          hastTable.put(InitialContext.PROVIDER_URL, host);
65          // Con formato user@message-vpn
66          hastTable.put(Context.SECURITY_PRINCIPAL, username +
67                      '@' + vpnName);
68          hastTable.put(Context.SECURITY_CREDENTIALS, password);
69
70          /* Crea el contexto inicial que se usará para
71           * buscar los objetos administrados por JMS.
72           */
73          InitialContext initialContext = new
74                                          InitialContext(hastTable);
75          // Busca la fabrica de conexiones
76          ConnectionFactory connectionFactory =
77                          (ConnectionFactory) initialContext.
78                          lookup(CONNECTION_FACTORY_JNDI_NAME);
79
80          // Crear conexión para el router Solace
81          Connection connection =
82                          connectionFactory.createConnection();
83
84          // Crear una sesión ACK de cliente no transada.
85          Session session = connection.createSession(false,
86                      SupportedProperty.SOL_CLIENT_ACKNOWLEDGE);
87
88          System.out.printf("Conectado a la VPN de mensajes de" +
89             " Solace '%s' como cliente con el nombre de " +
90             "usuario '%s'.%n", vpnName,  username);
91
92          // Búsqueda de la cola.
93          Queue queue =
94                (Queue) initialContext.lookup(QUEUE_JNDI_NAME);
95
96          // Desde la sesión, Crear un destino consumidor.
97          MessageConsumer messageConsumer =
98                          session.createConsumer(queue);
```

```
99
100         /* Uso de la clase interna anónima para recibir
101          * mensajes de forma asíncrona
102          */
103         messageConsumer.setMessageListener(new MessageListener()
104         {
105             @Override
106             public void onMessage(Message message) {
107                 try {
108                     if (message instanceof TextMessage) {
109                         System.out.printf("Mensjae recibido:
110                         '%s'%n" , ((TextMessage)message).
111                                                 getText());
112                     } else {
113                         System.out.println("Mensaje recibido.");
114                     }
115                     System.out.printf("Contenido del mensaje:
116                     %n%s%n", SolJmsUtility.dumpMessage(message));
117
118                     /* ACK del mensaje recibido manualmente
119                      * debido a la SupportedProperty.
120                      * SOL_CLIENT_ACKNOWLEDGE establecido
121                      * anteriormente
122                     message.acknowledge();
123                     // Desbloqueo del hilo principal
124                     latch.countDown();
125                 } catch (JMSException ex) {
126                     System.out.println("Error al procesos
127                                         mensaje entrante.");
128                     ex.printStackTrace();
129                 }
130             }
131         });
132
133         // Comienza a recibir los mensajes
134         connection.start();
135         System.out.println("Esperando por mensajes...");
136         /* Bloqueo del hilo principal en la siguiente
137          * sentencia hasta que se reciba un mensaje
138          */
139         latch.await();
140
141         connection.stop();
142         /* Cerrar todo en el orden inverso al de apertura
143          * NOTA: como las interfaces de abajo extienden
144          * AutoCloseable, con ellas es posible utilizar la
145          * sentencia Java "try-with-resources
146          *
147          * Revise la documentación en https://docs.oracle.com/
148          * javase/tutorial/essential/exceptions/tryResource
149          * Close.html
150          */
151         messageConsumer.close();
```

```
152         session.close();
153         connection.close();
154         /* The initial context needs to be close; it does not
155         * extend AutoCloseable
156         */
157         initialContext.close();
158     }
159 
160     /* Función pricipal, debe ser ejecutado con 3 argumentos:
161      * 1. Nombre del servidor
162      * 2. usuario@servidor
163      * 3. password
164      */
165     public static void main(String[] args) throws Exception {
166         if (args.length != 3 || args[1].split("@").length != 2)
167         {
168             System.out.println("Usage: QueueConsumerJNDI " +
169                 "<host:port> <client-username@message-vpn>" +
170                 " <client-password>");
171             System.out.println();
172             System.exit(-1);
173         }
174         if (args[1].split("@")[0].isEmpty()) {
175             System.out.println("No client-username entered");
176             System.out.println();
177             System.exit(-1);
178         }
179         if (args[1].split("@")[1].isEmpty()) {
180             System.out.println("No message-vpn entered");
181             System.out.println();
182             System.exit(-1);
183         }
184         new QueueConsumerJNDI().run(args);
185     }
186 }
```

Listado 8.15: Clase con inyección de recursos

Ya se ha presentado las clases tanto productoras como consumidoras de una cola de mensajes. Los comentarios guiarán al lector para comprender mejor la programación de las clases. A continuación con la misma temática se presentan las clases productoras y consumidoras de mensajes desde un Topic o publish/subscribe. En este caso, para el desarrollo de este tema, se ha utilizado *Servlets*. En el listado 8.16 se muestra el posible código de su implementación.

```
1 package package webtec.gcgu.jms.jndi;
2 
3 import java.io.IOException;
4 import java.io.PrintWriter;
5 
6 import javax.jms.TextMessage;
7 import javax.jms.Topic;
8 import javax.jms.TopicConnection;
```

```
9  import javax.jms.TopicConnectionFactory;
10 import javax.jms.TopicPublisher;
11 import javax.jms.TopicSession;
12 import javax.jms.TopicSubscriber;
13 import javax.naming.InitialContext;
14 import javax.servlet.ServletException;
15 import javax.servlet.annotation.WebServlet;
16 import javax.servlet.http.HttpServlet;
17 import javax.servlet.http.HttpServletRequest;
18 import javax.servlet.http.HttpServletResponse;
19
20 /**
21  * Servlet implementation class JMSSamplePubSub
22  */
23
24 /**
25 * escenario: Construir un flujo pub/sub no duradero</br>
26 * Conectarse al ME utilizando la connection factory jmsTCF </br>
27 * Crear u suscriptor no duradero para el topic jmsTopic </br>
28 * Publicar un mensaje sencillo en el topic jmsTopic </br>
29 * El suscriptor recibe el mensaje de topic jmsTopic
30 * y el mensaje que sea impreso en la consola </br>
31 **/
32
33 @WebServlet("/JMSSamplePubSub")
34 public class JMSSamplePubSub extends HttpServlet {
35     private static final long serialVersionUID = 1L;
36
37     /**
38      * @Ver HttpServlet#HttpServlet()
39      */
40     public JMSSamplePubSub() {
41         super();
42         // TODO Auto-generated constructor stub
43     }
44
45     /**
46      * @Ver HttpServlet#doGet(HttpServletRequest request,
47      * HttpServletResponse response)
48      */
49     protected void doGet(HttpServletRequest request,
50                          HttpServletResponse response)
51                          throws ServletException, IOException {
52
53         // TODO Auto-generated method stub
54         String strAction = request.getParameter("ACTION");
55         PrintWriter out = response.getWriter();
56         try{
57             if(strAction == null)
58             {
59                 out.println("especifique una acción");
60                 out.println("ejemplo http://<host>:<port>/" +
61                                 "JMSApp/JMSSamplePubSub?" +
```

```
                                    "ACTION=nonDurableSubscriber");
            }
            else if(strAction.
                    equalsIgnoreCase("nonDurableSubscriber"))
            {
                /* Crear un suscriptor no duradero y publicar
                * y recibir el mensaje del tema
                */
                nonDurableSubscriber(request, response);
            }
            else if(strAction.
                        equalsIgnoreCase("durableSubscriber"))
            {
                /*
                 * Crear un cliente (abonado) duradero y
                 * publicar y recibir el mensaje del tema
                */
                durableSubscriber(request, response);
            }
            else if(strAction.
                        equalsIgnoreCase("publishMessages"))
            {
                // Publicar cinco mensajes en el Topic
                publishMessages(request, response);
            }
            else if(strAction.
            equalsIgnoreCase("unsubscribeDurableSubscriber"))
            {
                // Dar de baja al abonado duradero registrado
                unsubscribeDurableSubscriber(
                                            request,
                                            response
                                            );
            }
            else
            {
                out.println("Incorrecta acción especificada. " +
                "La acción debe ser una de las siguientes: ");
                out.println("ACTION=nonDurableSubscriber");
                out.println("ACTION=durableSubscriber");
                out.println("ACTION=publishMessages");
                out.println("ACTION=" +
                                "unsubscribeDurableSubscriber");
            }
        }
        catch(Exception e)
        {
            out.println("Ha ocurrido algo inesperado, " +
              "compruebe los registros o reinicie el servidor");
            e.printStackTrace();
        }
    }
    /*
```

```
115     * @param request HttpRequest
116     * @param response HttpResponse
117     * @throws Exception si ocurre un error.
118     */
119     public void nonDurableSubscriber(HttpServletRequest request,
120         HttpServletResponse response) throws Exception
121     {
122         PrintWriter out = response.getWriter();
123         out.println("NonDurableSubscriber Started");
124 
125         // Crear una fabrica de conexión al Topic
126         TopicConnectionFactory cf1 = (TopicConnectionFactory)
127           new InitialContext().lookup("java:comp/env/jmsTCF");
128 
129         // Crear una conexion al Topic
130         TopicConnection con = cf1.createTopicConnection();
131         con.start();
132 
133         //Crear una sesión del Topic
134         TopicSession session = con.createTopicSession(false,
135                         javax.jms.Session.AUTO_ACKNOWLEDGE);
136         // Búsqueda por JNDI del Topic
137         Topic topic = (Topic) new
138             InitialContext().lookup("java:comp/env/jmsTopic");
139 
140         // Crear un subscriptor no durable
141         TopicSubscriber sub = session.createSubscriber(topic);
142 
143         // Crear un Topic publicador
144         TopicPublisher publisher =
145                           session.createPublisher(topic);
146 
147         // Publicar un mensaje en el Topic
148         publisher.publish(session.
149                 createTextMessage("Liberty PubSub Message"));
150         TextMessage msg = (TextMessage) sub.receive(2000);
151         if (null == msg) {
152             throw new Exception("No message received");
153         }
154         else {
155             out.println("Received message for non-durable " +
156                                         "subscriber " + msg);
157         }
158         if (sub != null)
159             sub.close();
160         if (con != null)
161             con.close();
162 
163         out.println("NonDurableSubscriber Completed");
164     } // Método para Suscriptor no durable
165 
166     /**
167     * Escenario de prueba: Crea un flujko Pub/Sub duradero</br>
```

```
168     * Conectate usando la fábrica de conexiones jmsTCF </br>
169     * crea un suscriptor duradero (Nombre DURATEST) para el
170     * Topic jmsTopic </br>
171     * Publica un mensaje simple en el Topic jmsTopic </br>
172     * Suscriptor recibe el mensaje del Topic jmsTopic </br>
173     *
174     * @param request HttpRequest
175     * @param response HttpResponse
176     * @throws Exception if an error occurs.
177     */
178    public void durableSubscriber(HttpServletRequest request,
179                HttpServletResponse response) throws Exception {
180        PrintWriter out = response.getWriter();
181        out.println("DurableSubscriber Started");
182
183        //Crear la fabrica de conexion para el Topic
184        TopicConnectionFactory cf1 = (TopicConnectionFactory) new
185                  InitialContext().lookup("java:comp/env/jmsTCF");
186        // Búsqueda del Topic del JNDI
187        Topic topic = (Topic) new
188              InitialContext().lookup("java:comp/env/jmsTopic");
189
190        // Crear la conexión al Topic
191        TopicConnection con = cf1.createTopicConnection();
192        con.start();
193        TopicSession session = con.createTopicSession(false,
194                          javax.jms.Session.AUTO_ACKNOWLEDGE);
195
196        // Crear Suscriptor duradero
197        TopicSubscriber sub =
198                session.createDurableSubscriber(topic, "DURATEST");
199
200        // create a publisher
201        TopicPublisher publisher = session.createPublisher(topic);
202
203        // publish the message
204        publisher.publish(session.createTextMessage("Liberty " +
205                                            "PubSub Message"));
206
207        TextMessage msg = null;
208        do {
209            msg = (TextMessage) sub.receive(2000);
210            if(msg!=null)
211                out.println("Received  messages " + msg);
212        } while (msg != null);
213
214        if (sub != null)
215            sub.close();
216        if (con != null)
217            con.close();
218
219        out.println("DurableSubscriber Completed");
220    }// end of DurableSubscriber
```

```
221
222     /**
223      * Prueba: Publicar mensajes en el Topic</br>
224      * Conectarse usando la connection factory jmsTCF </br>
225      * Publicar 5 mensajes en el Topic jmsTopic </br>
226      *
227      * @param request HttpRequest
228      * @param response HttpResponse
229      * @throws Exception si ocurre algún error.
230      */
231     public void publishMessages(HttpServletRequest request,
232             HttpServletResponse response) throws Exception {
233         PrintWriter out = response.getWriter();
234         out.println("PublishMessage Started");
235
236         TopicConnectionFactory cf1 = (TopicConnectionFactory)
237             new InitialContext().lookup("java:comp/env/jmsTCF");
238         TopicConnection con = cf1.createTopicConnection();
239         int msgs = 5;
240
241         TopicSession session = con.createTopicSession(false,
242                             javax.jms.Session.AUTO_ACKNOWLEDGE);
243
244         Topic topic = (Topic) new
245             InitialContext().lookup("java:comp/env/jmsTopic");
246
247         TopicPublisher publisher =
248                                 session.createPublisher(topic);
249         // Enviar los cinco mensajes
250         for (int i = 0; i < msgs; i++) {
251             publisher.publish(session.createTextMessage(
252                             "Mensaje de Liberty PubSub : " + i));
253         }
254         if (con != null)
255             con.close();
256         out.println(msgs + " Mensajes publicados");
257         out.println("Publicación de mensajes hecho.");
258     }//Fin de publishMessages
259
260
261     /**
262      * Test scenario: Unsubscribe the durable subscriber</br>
263      * Connects to ME using connection factory jmsTCF </br>
264      * Creates/Opens durable subscriber (named DURATEST)
265      * for topic jmsTopic </br>
266      * Consumes all messages to the topic jmsTopic </br>
267      * Subscriber unsubscribes from topic jmsTopic </br>
268      *
269      * @param request HttpRequest
270      * @param response HttpResponse
271      * @throws Exception si ocurre algún error.
272      */
273     public void unsubscribeDurableSubscriber(
```

```
                                HttpServletRequest request,
                                HttpServletResponse response)
                                throws Exception
    {
        PrintWriter out = response.getWriter();
        out.println("Comienza el UnsubscribeDurableSubscriber");

        TopicConnectionFactory cf1 = (TopicConnectionFactory)
        new InitialContext().lookup("java:comp/env/jmsTCF");
        TopicConnection con = cf1.createTopicConnection();

        con.start();
        TopicSession session = con.createTopicSession(false,
                        javax.jms.Session.AUTO_ACKNOWLEDGE);

        Topic topic = (Topic) new
            InitialContext().lookup("java:comp/env/jmsTopic");

        TopicSubscriber sub = session.createDurableSubscriber(
                                topic, "DURATEST");
        /* Todos los mensajes existentes son consumidos por el
        * suscriptor duradero DURATEST
        */
        TextMessage msg = null;
        do {
            msg = (TextMessage) sub.receive(2000);
            if (msg != null)
                out.println("Mensajes recibidos : " + msg);
        } while (msg != null);

        if (sub != null)
            sub.close();

        // Le da de baja al suscriptor duradero
        session.unsubscribe("DURATEST");

        if (con != null)
            con.close();

        out.println("UnsubscribeDurableSubscriber hecho!");
    }//Fin del método unsubscribeDurableSubscriber
}
```

Listado 8.16: Clase con inyección de recursos

Una descripción paso a paso de la implementación de JMS con WebSphere Application Server se puede revisar en [260].

8.6.2. Ejemplos paso a paso usando NetBeans 12

Para este ejercicio se estima que se tiene creado los recursos tanto para las fabricas de conexiones como para consumir los mensajes o suscriptores (JNDI QUEUE y TOPIC),

como se ve en la figura 8.5. Primero se presentará la creación de un emisor y de un

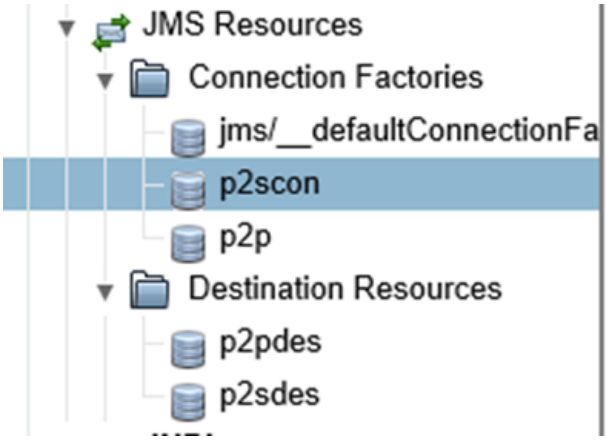


Figura 8.5: JNDI para JMS

receptor cola de mensajes.

Se debe crear una aplicación Enterprise Application como se muestra en la figura 8.6 y en la figura 8.7.

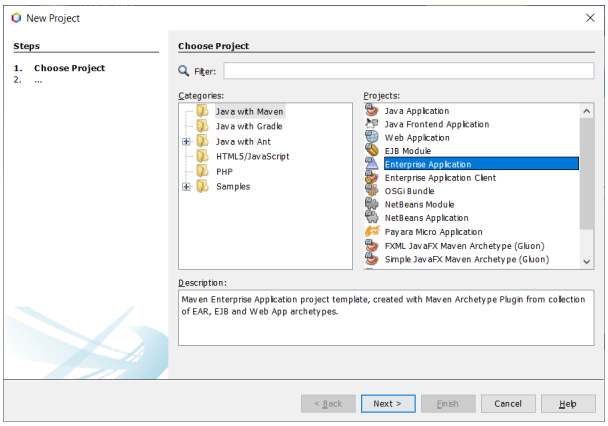


Figura 8.6: Aplicación Enterprise

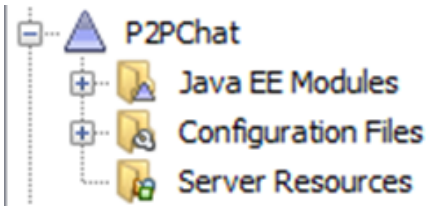


Figura 8.7: Aplicación Enterprise

Paso 1: Adquirir (obtener) una fabrica de conexiones

```
    Context context = InitialContext();
    QueueConnectionFactory queueCF = (QueueConnectionFactory)
                                        context.lookup("p2p");
    Queue queue = (Queue) context.lookup("p2pdes");
```

Paso 2: Crear una conexión mediante la fábrica de conexiones:

```
    QueueConnection qConnection = queueCF.createQueueConnection();
```

Paso 3: Iniciar la conexión

```
    qConnection.start();
```

Paso 4: Crear una sesión a partir de la conexión

```
    QueueSession qSession = qConnection.createQueueSession(
                              false, Session.AUTO_ACKNOWLEDGE);
```

Paso 5: Crear un emisor de mensajes para la cola

```
    QueueSender qSender = qSession.createSender(queue);
```

Paso 6: Al requerir enviar mensajes, se crea un productor. Para ello se ha creado un formulario con una caja de texto para escribir el mensaje y tres botones: Conectar, enviar y cerrar, como se muestra en la figura 8.8.

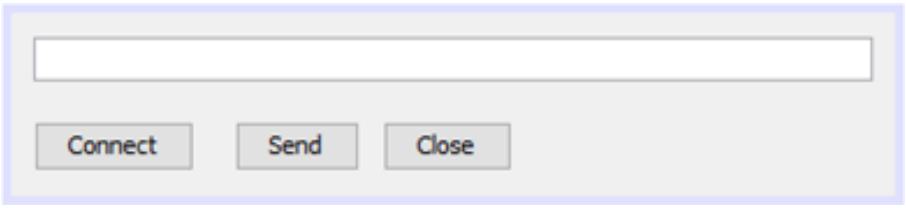


Figura 8.8: JFrame para productor de mensajes

Paso 7. Crear un mensaje y ponerlo a disposición del destino

```
    TextMessage textMessageSender = qSession.createTextMessage(
                                  this.jtxtMensaje.getText());
```

Paso 8: Al requerir recibir mensajes, se crea un consumidor. Para ello se ha creado un formulario con una area de texto para ir adjuntando los mensajes, y dos botones: Conectar, y cerrar, como se muestra en la figura 8.9

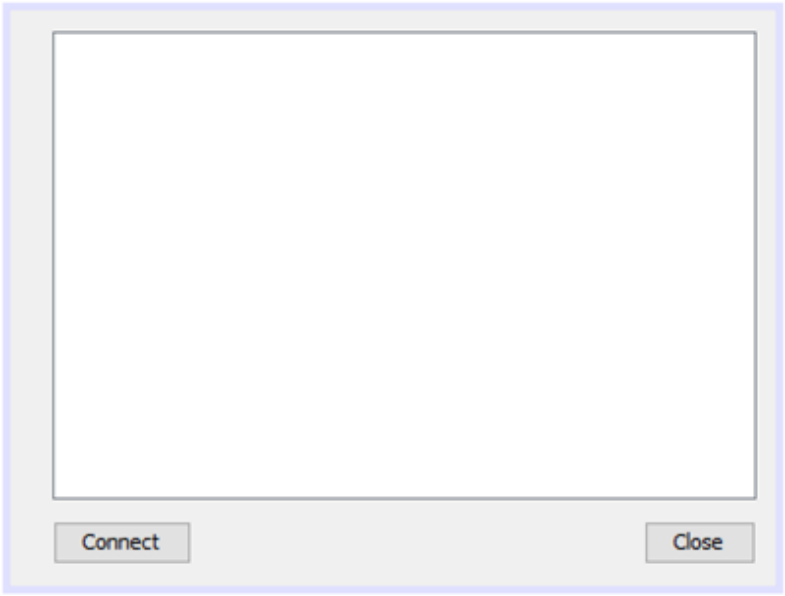


Figura 8.9: JFrame para consumidor (destino) de mensajes

Paso 9. Opcionalmente se crea un escuchante (listener) de mensajes. El cuerpo del escuchante sería similar al siguiente código:

```
1     QueueReceiver qReceiver;
2     String content = "";
3     TextMessage textMessageReceive = (TextMessage)
4                                         qReceiver.receive();
5     if(textMessage != null){
6         content += this.jtxtMessage.getText() +
7                             textMessageReceive.getText() + "\n";
8     }
```

Paso 10. Enviar o recibir el/los mensaje(s)

```
1     //Para enviar
2     qSender.send (textMessageSender);
3
4     //Para recibir
5     qReceiver.receive();
```

Paso 11. Cerrar los objetos: Los objetos se cierran en orden inverso de su apertura: Consumidor, Productor, Sessión, Conexión. Podría ser suficiente con cerrar la conexión, sin embargo, se considera que los otros objetos se cerrarían abruptamente.

```
1     qReceiver.close();
2     qSender.close();
3     qSession.close();
4     qConnection.close();
```

Una vez presentada paso a paso la implementación de JMS QUEUE tanto un productor como también un consumido de los mensajes, ahora se presenta paso a paso como crear una aplicación que genere los mensajes para un Topic, y la clase para los suscriptores.

Se da por hecho la creación de los recursos JMS, tanto la fábrica de conexiones como los destinatarios JMS Topic, como se ve en la figura 8.10.

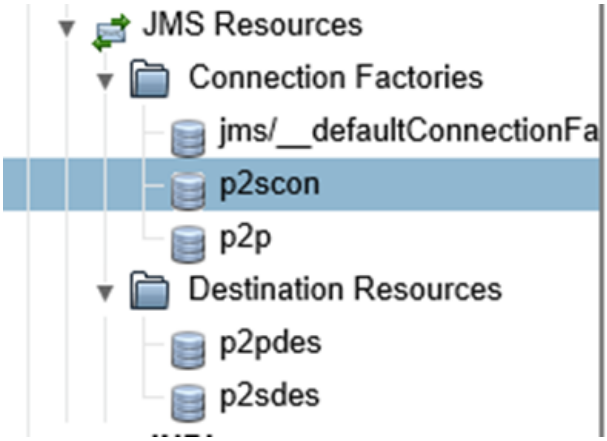


Figura 8.10: Recursos JNDI para JMS Topic

Así mismo, como se creó la aplicación Enterprise usando el IDE Netbeans en el ejemplo de QUEUE, debemos crear una aplicación para Topic. La Aplicación creada se verá como la figura 8.11.

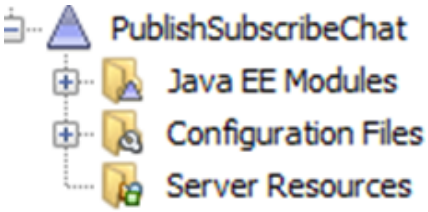


Figura 8.11: JFrame para consumidor (destino) de mensajes

Paso 1: Localizar una fabrica de conexiones para Topic

```
1        Context context = new InitialContext();
2        TopicConnectionFactory tConnectionFactory = new
3                                  (TopicConnectionFactory)
4                                  context.lookup("p2scon");
5
```

Paso 2: Crear una conexión usando la fábrica de conexiones

```
1        TopicConeccion tConnection  =
2                tConnectionFactory.createTopicConnection();
3        //Identificamos al cliente con el Nombre ingresado como ID
4        tConnection.setClientID(this.jTxtName.getText());
5
```

Paso 3: Iniciamos la conexión

```
1        Connection.start();
2
```

Paso 4: Crear una sesión a partir de la conexión

```
1        TopicSession tSession = tConnection.createTopicSession(
2                                false,
3                                TopicSession.AUTO_ACKNOLOWLEDGE);
4
```

Paso 5: Crear un Publicador/Suscriptor con la interfaz

```
1        Topic topic = (Topic) context.lookup("p2sdes");
2        TopicPublisher tPublisher =
3                                tSession.createPublisher(topic);
4        TopicSubscriber tSubscriber =
5                            tSession.createDurableSubscriber(
6                               topic, this.jTxtName.getText());
7
```

Figura 8.12: JFrame para enviar/recibir los mensajes

Paso 6 Enviar/recibir el/los mensajes

```
        \\ENVIAR MENSAJES
        TextMessage textMessageS = tSession.createTextMessage ();
        textMessageS.setText(jTxtName.getText() + ":" +
                             jTxtMessage.getText();
        tPublisher.send(textMessageS);

        \\RECIBIR MENSAJES
        TextMessage textMessageR = (TextMessage)
                                   tSubscriber.receive ();
        if( textMessageR != null){
            this.jTxtAreaContent.setText(
                              this.jTxtAreaContent.getText() +
                              "\n" + textMessageR.getText());
        }

```

Paso 7: Cerrar la conexión Finalmente nos toca cerrar los elementos que intervienen en la comunicación, siempre en orden inverso al que se abrieron.

```
        tSubscriber.close();
        tPublisher.close();
        tSession.close();
        tConnection.close();

```

8.7. Conclusiones

Los objetos de aplicaciones que normalmente se ejecutan en máquinas distintas se comunican accediendo a los servicios del middleware orientado a mensajes (MOM) mediante un conjunto estándar de interfazs que JMS define. De esta manera se da la comunicación entre procesos o máquinas. JMS es el sistema de mensajería de Java, proporciona un estándar entre plataformas para la comunicación síncrona y asíncrona de sucesos y de datos en toda la organización.

JMS es una tecnología de interfaz para acceder a servidores de mensajes desde su aplicación Java o J2EE. Durante los últimos años, JMS ha ganado una enorme popularidad en el negocio de EAI (*Enterprise Application Integration* o integración de aplicación empresarial) debido a su capacidad para proporcionar la llamada mensajería fiable. Este consejo técnico examina en detalle lo que significa la mensajería "fiablez cómo conseguirla. Dependiendo todo de cómo configure y utilice la API JMS.

Evaluación a los lectores

Resulta obvio que en un proceso de enseñanza/aprendizaje hay que valorar el grado de consecución de los objetivos de dicho proceso. Esto es, hay que verificar si el alumno ha adquirido o no lo que estaba previsto que adquiriese al término del proceso y en qué grado lo ha adquirido. Esa es la función de la evaluación [50]. Es por ello que el lector debe responder y resolver los siguientes problemas:

- Dibujar un cuadro sinóptico en el que se presenten las características, aplicaciones (tipo de problemas), ventajas y desventajas de JMS.
- Dibujar un cuadro comparativo entre JMS y otras tecnologías de comunicación entre aplicaciones.
- Desarrollar una aplicación web en la que se requiera el uso de JMS P2P y Pub/Sub síncronos.
- Desarrollar una aplicación web en la que se requiera el uso de JMS P2P y Pub/Sub asíncronos.

APÉNDICE - MATERIAL COMPLEMENTARIO

El material complementario puede ser descargado de forma gratuita desde la fuente de su preferencia.

- Sitio web personal del autor principal (http://www.gleiston.site)
- Drive de la UTEQ:
  - Texto en pdf: https://cutt.ly/MOM2KCs
  - Código fuente: https://cutt.ly/EOM9eIo
  - Videos: https://cutt.ly/LOM9WlB
- Desde el gestor de versiones GitHub:
  - Introducción a las aplicaciones web: https://cutt.ly/mOM3Dwn
  - Evolución Web: https://cutt.ly/WOM3HYG
  - Accesibilidad Web: https://cutt.ly/EOM3Zzz
  - IDE NetBeans: https://cutt.ly/POM3VHN
  - EJB: https://cutt.ly/COM3Mb6
  - WebServicies: https://cutt.ly/OOM30h1
  - WebSocket: https://cutt.ly/4OM33Ot
- Videos explicativos desde la plataforma de Youtube:
  - Introducción a las aplicaciones web: https://cutt.ly/gOM4xkO
  - Evolución Web: https://cutt.ly/uOM4EtR
  - Accesibilidad Web: https://cutt.ly/mOM4P5G

- IDE NetBeans (Parte 1): https://cutt.ly/MOM4VVK
- IDE NetBeans (Parte 2): https://cutt.ly/mOM44l1
- EJB: https://cutt.ly/WOM41Cs
- WebServicies: https://cutt.ly/sO1q3b2
- WebSocket: https://cutt.ly/pOM7qAm

Glosario

acoplada Femenino de acoplado. Ver acoplar, y acoplado. 241, 242

acoplar Unir o encajar entre sí dos piezas o cuerpos de manera que ajusten perfectamente.. 113

API Una API o interfaz de programación de aplicaciones es un conjunto de definiciones y protocolos que se usa para diseñar e integrar el software de las aplicaciones.. 10, 111, 114, 208, 210, 212, 215, 241–246, 251–254, 257, 264, 283

aplicaciones web Ver aplicación web.. 1–8, 12, 15–22, 28, 31, 34, 38, 39, 42, 43, 89, 97, 100, 107, 108, 161, 197–200, 206–208, 214, 215, 240, 241, 246

aplicación empresarial Una aplicación empresarial es una colección de componentes que proporciona una funcionalidad a nivel de empresa que se puede utilizar internamente, externamente o con otras aplicaciones empresariales. Puede crear aplicaciones empresariales de componentes individuales, que están relacionados entre sí.. 160

aplicación empresarial Ver aplicación empresarial.. 111–114, 120, 159, 242, 246, 283

aplicación web En la ingeniería de software se define a las aplicaciones web como aquellas herramientas que para utilizarse se debe acceder a un servidor web a través de internet o de una intranet, mediante un navegador. Resumiendo, es un programa que se codifica en un lenguaje interpretable por los navegadores web y a quién se le confía su ejecución.. 7, 8, 21, 65, 197, 199, 215, 249, 251, 283

asíncrona Femenino de asíncrono. Ver asíncrono.. 114, 241, 242, 246, 253–255, 283

asíncrono Dos señales son asíncronas o no están sincronizadas, cuando sus correspondientes instantes significativos no coinciden. 197, 241, 242, 253, 283

browser Es el término en inglés que identifica a un navegador web (Internet). Es un software, programa o aplicación, que ofrece al usuario el acceso a la web.. 4, 18

componente En el área del desarrollo de software, un componente es una unidad modular de un software con interfaces y dependencias bien definidas que permiten ofertar o solicitar un conjunto de servicios o funciones.. 111–121, 125, 153, 159, 160

conexión Según la RAE (Real Academía de la Lengua Española) Conexión es unión que se establece entre dos o más cosas (aparatos, sistemas, lugares, etc.) o personas para que entre ellas haya una relación o una comunicación.. 198–200, 202, 203, 206–216, 227, 228

consistencia (Consistency): es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Esto significa, por ejemplo, que se ejecutan aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.. 159

CSS Cascade Style Sheet, en español: Hojas de estilo en cascada. Es muy usado para el diseño de páginas web e interfaces de usuario escritas en HTML o XHTML. Entre los lenguajes con los que se puede complementar CSS como tecnologías para ser usadas en el diseño tanto de páginas web como de aplicaciones para móviles están HTML y JavaScript.. 4

desacoplar Acción que separa lo que estaba acoplado.. 114

disponibilidad Se trata de la capacidad de un servicio, de unos datos o de un sistema, a ser accesible y utilizable por los usuarios (o procesos) autorizados cuando estos lo requieran.. 159

dominio Un dominio de internet o dominio web llamado comúnmente, es un nombre único que identifica a una subárea de internet (o de la web).. 6–8

escalabilidad Se entiende por escalabilidad a la capacidad de adaptación y respuesta de un sistema con respecto al rendimiento del mismo a medida que aumentan de forma significativa el número de usuarios del mismo.. 159, 171, 172

GassFish Es la implementación de referencia de Java EE de código abierto. GlassFish es un servidod e aplicaciones para Java de código abierto . 99, 111, 113, 116, 140, 143, 219, 238, 264

hardware Es un término en inglés adoptado en el idioma español para nombrar a la parte física o tangible de un sistema informático. Por lo tanto, hardware comprende los componentes eléctricos, electrónicos, electromecánicos y mecánicos, desde el mismo computador hasta los cables de red, pasando por pantallas, circuitos, placas, memorias, discos duros, dispositivos periféricos y cualquier otro material en estado físico que sea necesario para desarrollar y utilizar a un sistema informático.. 16, 119, 162, 172

HTML *HyperText Markup Language* en español: Lenguaje de marcado de Hipertexto. Lenguaje utilizado para la creación de páginas web.. 3, 4, 6, 7, 18, 24–27, 34, 36, 134, 172, 179, 182–184, 186, 197, 201, 203, 216, 217

HTML5 Es la última versión estable de HTML. HTML5 convierte a HTML de un **simple formato de marcado** de elementos de comunicación para estructurar documentos en una **plataforma completa** de desarrollo de aplicaciones. Entre otras características, HTML5 incluye nuevos elementos y API de JavaScript para mejorar el almacenamiento, la multimedia y el acceso al hardware.. 198, 206, 208, 219

HTTP *HyperText Transfer Protocol* en español: Protocolo de Transferencia de Hipertexto. Conocido también como el protocolo de Internet.. 1, 8–14, 24, 27, 197, 199, 200, 202, 203, 205–209, 213–215, 240

HTTPS *HyperText Transfer Protocol Secure* en español: Protocolo de Transferencia de Hipertexto Seguro. Usado para transacciones o intercambio de información sensible. 12, 14, 202, 203, 209

interfaz Este concepto se emplea para referirse a la dinámica física y lógica de interconexión (interacción) entre dos equipos o sistemas independientes, así mismo, entre un sistema informático y la persona (usuario).. 15, 89, 90, 95, 99, 106, 107, 118, 119, 121, 125, 126, 128–130, 134, 135, 153, 163, 164, 168, 172, 174, 179, 189, 190, 198, 199, 202, 203, 206, 216, 218, 226, 234, 242, 243, 252, 254, 256, 258, 281, 283

Internet Red de redes de computadoras.. 1, 4, 5, 8, 11, 13, 15, 16, 23, 24, 26–30, 34, 36–38, 41, 43, 198, 200, 202, 208

interoperabilidad Concepto asignado para medir la capacidad de los sistemas de información y de los procedimientos a los que éstos dan soporte, de compartir datos y posibilitar el intercambio de información y conocimiento entre ellos.. 111, 119, 120, 172, 195

lenguaje scripting Son una popular familia de lenguajes de programación que se pueden utilizar para satisfacer rápidamente las exigencias más comunes. Los más comunes son los que se utilizan en el desarrollo web del lado del cliente o navegador. Sin embargo, también existen lenguajes scripting del lado del servidor. 4, 9, 18

lógica de negocios El término lógica de negocio es la **lógica de programación** de un sistema y corresponde a la codificación de las reglas de negocios del mundo real que determinan cómo la información puede ser creada, almacenada y actualizada, eliminada, e inclusive compartida.. 99, 111–113, 116, 118–121, 159, 165, 254

lógica de presentación Es la lógica de programación que está diseñada específicamente para las pantallas de las aplicaciones informáticas. Esta lógica gestiona la experiencia de navegación del usuario con herramientas de navegación especiales cuya función es comunica la información y capturar los datos. También conocida como Interfaz Gráfica de Usuario o en inglés GUI (*Graphic User Interface*).. 111, 119, 120

lógica de programación Es la organización y planificación de instrucciones en un algoritmo escritas generalmente en un lenguaje de programación ejecutables por una computadora, con el objetivo de

resolver un problema. La lógica de la programación es la organización coherente de las instrucciones del programa para que su objetivo sea alcanzado. También llamada **lógica de negocios**. 18

middleware Es el programa o programas de software que brinda(n) servicios y funciones comunes a las aplicaciones (del usuario), además de lo que ofrece el sistema operativo. Entre las más comunes, son las encargadas de la gestión de los datos, los servicios de aplicaciones, la mensajería, la autenticación y la gestión de las API.. 253, 283

monitorización Acción de monitorizar. Monitorizar: Observar mediante aparatos especiales el curso de uno o varios parámetros fisiológicos o de otra naturaleza para detectar posibles anomalías. 1, 197–199

navegador Ver browser. 4, 18, 140, 198–200, 202, 206–208, 213

plugin Son miniprogramas (programas) son complementos para añadir funcionalidades extras o mejoras a los programas.. 9

portal web Espacio donde una red informática ofrece servicios y recursos de una manera sencilla e integrada, que son accedidos por medio de un navegador.. 3–7, 21

protocolo En informática y telecomunicación, se denomina protocolo al conjunto de reglas y estándares que controlan la secuencia de mensajes que se envían y reciben durante una comunicación entre equipos que forman una red (teléfono, computadores, routers, por mencionar algunos).. 198, 199, 206–210, 213, 240

reutilización En el campo de los sistemas basados en computadoras, la reutilización es el uso de elementos software (partes de sistemas) existentes, durante la construcción de nuevos sistemas de software. El enfoque de la reutilización no es sólo el código fuente, sino cualquier producto intermedio generado en el proceso de desarrollo. 116, 120, 172

seguridad También llamada ciberseguridad. Se refiere a los procesos que se deben aplicar para la protección de la información y, especialmente,

tecnología Conjunto de teorías y de técnicas que permiten el aprovechamiento práctico del conocimiento científico. Conjunto de los instrumentos y procedimientos industriales de un determinado sector o producto.. 1–3, 9, 11, 16, 18, 23, 24, 27–31, 34, 36, 38, 39, 41, 43, 44, 90, 104, 107, 111–113, 116, 117, 120, 160, 164, 165, 195, 197–200, 202, 205–208, 214, 240, 241, 252, 283

transaccional Perteneciente o relativo a la transacción. Se dice que algo es trasactional cuando trabaja bajo o soporta transacciones.. 114, 116, 160

transacción Una transacción es una interacción con una estructura de datos compleja, compuesta por varios procesos que se han de aplicar uno a continuación del otro. Una transacción puede contener una o más operaciones o subtransacciones que deben realizarse en su totalidad de una sola vez y sin que la estructura afectada sea accedida para manipulación por el resto del sistema.. 112–118

web Es el diminutivo de World Wide Web o WWW.. 1, 2, 4, 23–31, 34, 36–43

WWW World Wide Web, es un sistema de documentos de hipertexto al que se accede mediante Internet. A través de cualquier dispositivo que sea capaz de ejecutar alguno de los programas de software conocidos como navegadores.. 1, 24, 28, 38, 43

XHTML *Extensible HyperText Transfer Protocol Secure* en español: Lenguaje de marcado de Hipertexto extensible. Es básicamente HTML expresado como XML. Es mucho más estricto que HTML, esto permite corregir errores de las páginas web de manera más fácil.. 34, 153

XML *Extensible Markup Language* en español: Lenguaje de marcado extensible. Es un lenguaje de un nivel abstracto (metalenguaje) que permite definir lenguajes de marcas desarrollado por el World Wide Web Consortium utilizado para almacenar datos en forma legible. Lenguaje usado para el intercambio de información transparente a los lenguajes de programación que se usen.. 4, 32, 34–36, 38

Bibliografía

[1] L. Chung and B. A. Nixon, "Dealing with Non-Functional Requirements," in *1995 17th International Conference on Software Engineering*, (Seattle, WA, USA), pp. 25–37, Association for Computing Machinery (ACM), apr 1995.

[2] W. Horton, L. Taylor, A. Ignacio, and N. L. Hoft, *The Web Page Design Cookbook: All the Ingredients You Need to Create 5-Star Web Pages*, vol. 1995. New York, NY, United States: John Wiley & Sons, Inc., 1995.

[3] MDN Contributors, "HTML: Lenguaje de etiquetas de hipertexto," 2022.

[4] MDN Contributors, "CSS | MDN," 2022.

[5] M. Spiliopoulou, "Web Usage Mining for Web Site Evaluation," *Communications of the ACM*, vol. 43, no. 8, pp. 127–134, 2000.

[6] J. Peña, B. V. Hanrahan, M. B. Rosson, and C. Cole, "After-Hours Learning: Workshops for Professional Women to Learn Web Development," *ACM Transactions on Computing Education*, vol. 21, pp. 1–31, mar 2021.

[7] D. C. Rajapakse and S. Jarzabek, "An investigation of cloning in web applications," tech. rep., National University of Singapore, 2005.

[8] J. Conallen, "Modeling Web Application Architectures with UML," *Communications of the ACM*, vol. 42, no. 10, pp. 63–70, 1999.

[9] I. Wikimedia Foundation, "Website - Wikipedia," feb 2021.

[10] A. Udiyono, C. Lim, and Lukas, "Botnet Detection Using DNS and HTTP Traffic Analysis," in *ACM International Conference Proceeding Series*, (New York, NY, USA), pp. 1–6, Association for Computing Machinery, sep 2020.

[11] A. Delignat-Lavaud and K. Bhargavan, "Network-Based Origin Confusion Attacks against HTTPS Virtual Hosting," in *WWW 2015 - Proceedings of the 24th International Conference on World Wide Web*, (Republic and Canton of Geneva, Switzerland), pp. 227–237, Association for Computing Machinery, Inc, may 2015.

[12] D. Priefer, "Model-driven development of content management systems based on Joomla," in *ASE 2014 - Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, (New York, NY, USA), pp. 911–914, Association for Computing Machinery, Inc, sep 2014.

[13] S. Koller, "¿Qué Es un Subdominio, para qué Sirve y cómo Gestionarlo?," nov 2020.

[14] M. Almishari and X. Yang, "Ads-portal domains: Identification and measurements," *ACM Transactions on the Web*, vol. 4, pp. 1–34, apr 2010.

[15] S. D. Kanrar and N. K. Mandal, "Session Based Storage Finding in Video on Demand System," in *ACM International Conference Proceeding Series*, vol. 10-13-Augu, pp. 131–135, 2015.

[16] M. Sayagh, N. Kerzazi, B. Adams, and F. Petrillo, "Software Configuration Engineering in Practice Interviews, Survey, and Systematic Literature Review," *IEEE Transactions on Software Engineering*, vol. 46, pp. 646–673, jun 2020.

[17] I. Grigorik, "Making the Web Faster with HTTP 2.0," *Queue*, vol. 11, pp. 40–53, oct 2013.

[18] A. Stokes, *TCP/IP*, pp. 1745–1747. GBR: John Wiley and Sons Ltd., 2003.

[19] R. Pelizzi and R. Sekar, "A server- and browser-transparent csrf defense for web 2.0 applications," in *Proceedings of the 27th Annual Computer Security Applications Conference*, ACSAC '11, (New York, NY, USA), p. 257–266, Association for Computing Machinery, 2011.

[20] P. Hoffman, "Putting it together: designs on Internet mail," *netWorker*, vol. 2, pp. 19–23, mar 1998.

[21] S. L. Garfinkel, D. Margrave, J. I. Schiller, E. Nordlander, and R. C. Miller, "How to make secure email easier to use," in *CHI 2005: Technology, Safety, Community: Conference Proceedings - Conference on Human Factors in Computing Systems*, (New York, New York, USA), pp. 701–710, Association for Computing Machinery, 2005.

[22] J. C. Mogul, "Clarifying the Fundamentals of HTTP," in *Proceedings of the 11th International Conference on World Wide Web, WWW '02*, (New York, New York, USA), pp. 25–36, ACM Press, 2002.

[23] D. Gourley, B. Totty, and M. Sayer, *HTTP: The Definitive Guide*. Sebastopol, California, United States: y O'Reilly & Associates, Inc., 2002.

[24] E. R. Harold, *Java Network Programming*. Sebastopol, California, United States: O'Reilly Media, Inc., 4 ed., 2014.

[25] L. T. Walczowski, D. Nalbantis, W. A. Waller, and K. Shi, "Analogue Layout Generation by World Wide Web Server-Based Agents," tech. rep., University of Kent, 1997.

[26] M. Vigil-Hayes, E. Belding, and E. Zegura, "FiDO: A Community-based Web Browsing Agent and CDN for Challenged Network Environments," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, pp. 1–25, sep 2017.

[27] T. Berners-Lee, "Information Management: A Proposal," tech. rep., CERN, 1989.

[28] M. Ginsburg and A. Kambil, "The EDGAR Internet Project: Web Application Development Considerations," in *Association for Information Systems Proceedings of the First Americas Conference on*

Information Systems (H. Ahuja, MKK; Galletta, DF; Watson, ed.), (Pittsburgh), pp. 457–459, 1995.

[29] K. Nath, S. Dhar, and S. Basishtha, "Web 1.0 to Web 3.0 - Evolution of the Web and its various challenges," in *ICROIT 2014 - Proceedings of the 2014 International Conference on Reliability, Optimization and Information Technology*, pp. 86–89, IEEE Computer Society, 2014.

[30] Z. A. King, A. Dräger, A. Ebrahim, N. Sonnenschein, N. E. Lewis, and B. O. Palsson, "Escher: A Web Application for Building, Sharing, and Embedding Data-Rich Visualizations of Biological Pathways," *PLoS Computational Biology*, vol. 11, p. 1004321, aug 2015.

[31] H.-J. Kung and H.-L. Tung, "Web Application Classification," in *Handbook of Research on Public Information Technology*, pp. 520–530, IGI Global, jan 2011.

[32] M. Jazayeri, "Some Trends in Web Application Development," in *FoSE 2007: Future of Software Engineering*, (Minneapolis, MN, USA), pp. 199–213, jun 2007.

[33] H. B. Prajapati and V. K. Dabhi, "High Quality Web-Application Development on Java BE Platform," in *2009 IEEE International Advance Computing Conference, IACC 2009*, pp. 1664–1669, 2009.

[34] C. Y. Lin, "Integrating the Web and Enterprise Systems," *Handbook of Enterprise Integration*, pp. 373–387, 2009.

[35] W3C, "Understanding Success Criterion 2.4.9: Link Purpose (Link Only)," 2018.

[36] C. I. Llanos and M. N. Muñoz, "Design Guidelines for Web Applications Based on Local Patterns," in *Euro American Conference on Telematics and Information Systems - Proceedings of the 2007 Euro American Conference on Telematics and Information Systems, EATIS 2007*, (New York, New York, USA), ACM Press, 2007.

[37] K. Pranathi, S. Kranthi, A. Srisaila, and P. Madhavilatha, "Attacks on Web Application Caused by Cross Site Scripting," in *Proceedings of the 2nd International Conference on Electronics, Communication*

and Aerospace Technology, ICECA 2018, pp. 1754–1759, Institute of Electrical and Electronics Engineers Inc., sep 2018.

[38] T. Orehovački, G. Bubaš, and A. Kovačić, "Taxonomy of Web 2.0 Applications with Educational Potential," *Transformation in Teaching: Social Media Strategies in Higher Education*, pp. 43–72, 2012.

[39] M. Faheem, "Intelligent Crawling of Web Applications for Web Archiving," in *WWW'12 - Proceedings of the 21st Annual Conference on World Wide Web Companion*, pp. 127–131, 2012.

[40] S. A. Adams, "Revisiting the Online Health Information Reliability Debate in the Wake of "Web 2.0": An Inter-Disciplinary Literature and Website Review," *International Journal of Medical Informatics*, vol. 79, no. 6, pp. 391–400, 2010.

[41] M.-H. Yang, Y.-S. Jian, and H.-L. Chen, "Constructing the Evaluation Model for Business-To-Consumers Electronic Commerce from Consumer's Perception," *International Journal of Electronic Business Management*, vol. 4, no. 1, pp. 38–47, 2006.

[42] R. Chopra, *Web Engineering*. Eastern Economy Edition, Prentice Hall India Learning Pvt., Limited, 2016.

[43] M. Hesse and N. Pohlmann, "Internet situation awareness," in *eCrime Researchers Summit, eCrime 2008*, 2008.

[44] T. Levä, H. Hämmäinen, and K. Kilkki, "Scenario analysis on future internet," in *1st International Conference on Evolving Internet, INTERNET 2009*, pp. 52–59, 2009.

[45] T. Berners-Lee, "WWW: Past, present, and future," oct 1996.

[46] M. W. Schranz, "Engineering flexible world wide web services," tech. rep., Technical University of Vienna, 1998.

[47] N. Savage, "Weaving the web," *Communications of the ACM*, vol. 60, pp. 20–22, may 2017.

[48] F. J. Martínez-López, R. Anaya-Sánchez, R. Aguilar-Illescas, and S. Molinillo, "Evolution of the Web," in *Online Brand Communities.*

Using the Social Web for Branding and Marketing, pp. 5–15, Springer, Cham, 2016.

[49] K. Toyoda, S. Tamoto, and D. H. Crocker, "Internet facsimile as an internet office appliance," *IEEE Communications Magazine*, vol. 39, pp. 60–66, oct 2001.

[50] U. Ultes-Nitsche, "Web 3.0 — wohin geht es mit dem World Wide Web?," Tech. Rep. 1, 2010.

[51] Ramos Martin Alicia and Ramos Martin Martin Jesus, *Aplicaciones Web*. Madrid, España: Ediciones Paraninfo, 2 ed., 2014.

[52] Latorre M, "Historia De Las Web, 1.0, 2.0, 3.0 y 4.0," tech. rep., Universidad Marcelino Champagnat, 2018.

[53] J. M. Silva, A. S. M. Mahfujur Rahman, and A. El Saddik, "Web 3.0: A vision for bridging the gap between real and virtual," in *Proceedings of the 1st ACM International Workshop on Communicability Design and Evaluation in Cultural and Ecological Multimedia System*, CommunicabilityMS '08, (New York, NY, USA), p. 9–14, Association for Computing Machinery, 2008.

[54] B. Piñeiro Torres and A. García González, "Evolution of the semantic web towards the intelligent web: From conceptualization to personalization of contents," in *Advances in Intelligent Systems and Computing*, vol. 503, pp. 419–427, Springer Verlag, nov 2017.

[55] S. Luján Mora, *Programación de aplicaciones web: historia, principios básicos y clientes web*. Alicante, España: Editorial Club Universitario, oct 2002.

[56] M. Zajicek, "Web 2.0: Hype or happiness?," in *Proceedings of the 2007 International Cross-Disciplinary Conference on Web Accessibility (W4A)*, W4A '07, (New York, NY, USA), p. 35–39, Association for Computing Machinery, 2007.

[57] H. Delgado, "Web 2.0 Historia, Evolución y Características," oct 2019.

[58] B. P. Balaji, M. S. Vinay, B. G. Shalini, and J. S. Mohan Raju, "Web 2.0 Use in Academic Libraries of Top Ranked Asian Universities," *Electronic Library*, vol. 37, pp. 528–549, jun 2019.

[59] M. Moro, "La Web 2.0 Cambia Nuestra Forma de Navegar en Internet," apr 2010.

[60] X. Li, J. Li, S. Dai, and C. Shen, "Electronic Commerce and Supervision Platform in Agriculture Based on Web 3.0," in *IET Conference Publications*, vol. 2012, 2012.

[61] G. G. Witte, "Content generation and social network interaction within academic library facebook pages," *Journal of Electronic Resources Librarianship*, vol. 26, no. 2, pp. 89–100, 2014.

[62] A. Gustafson, "The Web Standards Project: Mission," mar 1998.

[63] D. Sloan, "Two cultures? The Disconnect between the Web Standards Movement and Research-Based Web Design Guidelines for Older People," Tech. Rep. 2, University of Dundee, Dundee, Scotland, United Kindom, jul 2006.

[64] J. Urbano, "Tipos de Mapas de Navegación Multimedia," 2013.

[65] G. Lucas, "Unidad 7 - Maquetación web - Creación de Páginas Web," jan 2010.

[66] M. Paquette, "Web 2.0: Una Nueva Ola de Amenazas," aug 2007.

[67] W. Maes, T. Heyman, L. Desmet, and W. Joosen, "Browser protection against cross-site request forgery," in *Proceedings of the First ACM Workshop on Secure Execution of Untrusted Code*, SecuCode '09, (New York, NY, USA), p. 3–10, Association for Computing Machinery, 2009.

[68] A. A. Noureddine and M. Damodaran, "Security in web 2.0 application development," in *Proceedings of the 10th International Conference on Information Integration and Web-Based Applications & Services*, iiWAS '08, (New York, NY, USA), p. 681–685, Association for Computing Machinery, 2008.

[69] S. Gupta and B. B. Gupta, "Php-sensor: A prototype method to discover workflow violation and xss vulnerabilities in php web applications," in *Proceedings of the 12th ACM International Conference on Computing Frontiers*, CF '15, (New York, NY, USA), Association for Computing Machinery, 2015.

[70] L. enciclopedia libre Wikipedia, "Inyección SQL - Wikipedia, la enciclopedia libre," *Wikipedia, La enciclopedia libre*, jun 2020.

[71] I. Tariq, M. A. Sindhu, R. A. Abbasi, A. S. Khattak, O. Maqbool, and G. F. Siddiqui, "Resolving cross-site scripting attacks through genetic algorithm and reinforcement learning," *Expert Systems with Applications*, vol. 168, p. 114386, 2021.

[72] J. Hu, W. Zhao, and Y. Cui, "A survey on sql injection attacks, detection and prevention," in *Proceedings of the 2020 12th International Conference on Machine Learning and Computing*, ICMLC 2020, (New York, NY, USA), p. 483–488, Association for Computing Machinery, 2020.

[73] S. J. Andriole, "Business Impact of Web 2.0 Technologies," *Communications of the ACM*, vol. 53, no. 12, pp. 67–79, 2010.

[74] H. Ajjan and R. Hartshorne, "Investigating faculty decisions to adopt web 2.0 technologies: Theory and empirical tests," *The Internet and Higher Education*, vol. 11, no. 2, pp. 71–80, 2008.

[75] S. Murugesan, "Understanding Web 2.0," *IT Professional*, vol. 9, pp. 34–41, aug 2007.

[76] J. Hailpern, L. Guarino-Reid, R. Boardman, and S. Annam, "Web 2.0: Blind to an accessible new world," in *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, (New York, NY, USA), p. 821–830, Association for Computing Machinery, 2009.

[77] P. Thiessen and C. Chen, "Ajax live regions: Reefchat using the fire vox screen reader as a case example," in *Proceedings of the 2007 International Cross-Disciplinary Conference on Web Accessibility (W4A)*, W4A '07, (New York, NY, USA), p. 136–137, Association for Computing Machinery, 2007.

[78] P. Thiessen and C. Chen, "Ajax live regions: ReefChat using the fire vox screen reader as a case example," in *ACM International Conference Proceeding Series*, vol. 225, (New York, New York, USA), pp. 136–137, ACM Press, may 2007.

[79] M. K. K. Brown, B. Huettner, and C. James-Tanny, "Choosing the Right Tools for your Virtual Team: Evaluating Wikis, Blogs, and Other Collaborative Tools," in *IEEE International Professional Communication Conference*, 2007.

[80] Y. Zhang, H. Yang, and L. Kuang, "A Web API Recommendation Method with Composition Relationship Based on GCN," in *2020 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pp. 601–608, 2020.

[81] J. Musser and T. O'Reilly, "Web 2.0 Principles and Best Practices," tech. rep., O'Reilly Media, Inc., Sebastopol, California, United States, nov 2006.

[82] ISOC, "Consolidación en la economía de internet. Informe Global de Internet de Internet Society," tech. rep., Internet Society, 2019.

[83] Grupo Banco Mundial, "Personas que Usan Internet (% de la Población)," 2021.

[84] A. J. Paniagua, "¿Qué es la web 3.0? Eric Schmidt La Define," aug 2007.

[85] K. L. Kroeker, "Engineering the web's third decade," *Commun. ACM*, vol. 53, p. 16–18, Mar. 2010.

[86] R. D. Morris, "Web 3.0: Implications for Online Learning," *TechTrends*, vol. 55, pp. 42–46, feb 2011.

[87] J. Pattnayak, S. Pattnaik, and P. Dash, "Knowledge Management in E-Learning A Critical Analysis," *International Journal Of Engineering And Computer Science*, vol. 6, pp. 2319–7242, 2017.

[88] M. Lal, "Web 3.0 in Education & Research," *BVICAM's International Journal of Information Technology*, vol. 3, no. 2, pp. 16–22, 2011.

[89] B. Aragona, "New Data Science: The Sociological Point of View," in *Studies in Classification, Data Analysis, and Knowledge Organization*, vol. 2, pp. 17–24, Springer Berlin Heidelberg, 2017.

[90] W. Ahmed, "Third Generation of the Web: Libraries, Librarians and Web 3.0," *Library Hi Tech News*, vol. 32, pp. 6–8, jun 2015.

[91] T. Guarda, M. Leon, M. F. Augusto, L. Haz, M. De La Cruz, W. Orozco, and J. Alvarez, "Internet of Things Challenges," in *Iberian Conference on Information Systems and Technologies, CISTI*, IEEE Computer Society, jul 2017.

[92] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, pp. 2787–2805, oct 2010.

[93] M. N. Fesharaki, A. H. Fetanat, and D. F. Shooshtari, "A conceptual model for Socio-Pragmatic Web based on activity theory," *Cogent Education*, vol. 7, pp. 1–25, jul 2020.

[94] D. Matus, "¿Cuál es mejor? Comparamos Cortana, Siri, Google Assistant y Alexa | Digital Trends Español," feb 2021.

[95] K. D. S. Brito, A. A. De Lima, S. E. Ferreira, V. De Arruda Buregio, V. C. Garcia, and S. R. De Lemos Meira, "Evolution of the Web of Social Machines: A Systematic Review and Research Challenges," *IEEE Transactions on Computational Social Systems*, vol. 7, pp. 373–388, apr 2020.

[96] W3C, "About W3C," 2017.

[97] Y. Yesilada, G. Brajnik, M. Vigo, and S. Harper, "Understanding web accessibility and its drivers," in *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility*, W4A '12, (New York, NY, USA), Association for Computing Machinery, 2012.

[98] S. Wee and S. Sanmargaraja, "Accessible Information required by the Independent Disabled Tourists: A Mini Review," *Australian Journal of Basic and Applied Sciences*, vol. 10, no. 1, pp. 65–70, 2016.

[99] S. L. Henry, S. Abou-Zahra, and J. Brewer, "The Role of Accessibility in a Universal Web," in *Proceedings of the 11th Web for All Conference*, W4A '14, (New York, NY, USA), Association for Computing Machinery, 2014.

[100] Web Accessibility Initiative, "Introduction to Web Accessibility | Web Accessibility Initiative (WAI) | W3C," 2005.

[101] W. Jeanne Spellman, "User Agent Accessibility Guidelines (UAAG) 2.0," 2018.

[102] S. E. Park, "Epidemiology, virology, and clinical features of severe acute respiratory syndrome-coronavirus-2 (SARS-COV-2; Coronavirus Disease-19)," 2020.

[103] S. Tejedor, L. Cervi, F. Tusa, and A. Parola, "Education in times of pandemic: reflections of students and teachers on virtual university education in spain, italy, and ecuador," *Revista Latina de Comunicación Social*, no. 78, pp. 1–21, 2020. Copyright - Copyright Laboratorio de Technologías de la Informaciò y Nuevos Análisis, LATINA, de la Universidad de La Laguna 2020; Última actualización - 2020-12-03; SubjectsTermNotLitGenreText - Italy; Spain; Ecuador.

[104] V. Segarra-Faggioni, M. Belen Mora Arciniegas, and G. T. Luna, "Web Accessibility Analysis with Semantich Approach of the Academic Services Web Portal to University Level," in *Iberian Conference on Information Systems and Technologies, CISTI*, vol. 2016-July, IEEE Computer Society, jul 2016.

[105] M. V. R. Leite, L. P. Scatalon, A. P. Freire, and M. M. Eler, "Accessibility in the mobile development industry in brazil: Awareness, knowledge, adoption, motivations and barriers," *Journal of Systems and Software*, p. 110942, 2021.

[106] W3C, "Easy Checks – A First Review of Web Accessibility | Web Accessibility Initiative (WAI) | W3C."

[107] B. Marino and K. F. Mason, "Exploring accessibility in doaj: A case study," *Serials Review*, vol. 46, no. 2, pp. 82–90, 2020.

[108] T. Acosta, S. Luján-Mora, and P. Acosta-Vargas, "Method for Accessibility Assessment of Heading in Online Editors," in *Proceedings of the 2017 9th International Conference on Education Technology and Computers - ICETC 2017*, (New York), pp. 243–247, ACM Press, 2017.

[109] B. A. Shawar, "Evaluating web accessibility of educational websites," *International Journal of Emerging Technologies in Learning*, vol. 10, no. 4, pp. 4–10, 2015.

[110] P. Aycinena, "Access for all," *Commun. ACM*, vol. 51, p. 12–14, Aug. 2008.

[111] J. Carter and M. Markel, "Web accessibility for people with disabilities: An introduction for web developers," *IEEE Transactions on Professional Communication*, vol. 44, pp. 225–233, dec 2001.

[112] R. Burton, D. P. Crabb, N. D. Smith, F. C. Glen, and D. F. Garway-Heath, "Glaucoma and Reading," *Optometry and Vision Science*, vol. 89, pp. 1282–1287, sep 2012.

[113] L. Moreno, P. Martínez, and B. Ruiz-Mezcua, "Disability Standards for Multimedia on the Web," *IEEE Multimedia*, vol. 15, pp. 52–54, oct 2008.

[114] M. Cooper, "Web accessibility guidelines for the 2020s," in *Proceedings of the 13th Web for All Conference on - W4A '16*, (New York), pp. 1–4, ACM Press, 2016.

[115] W3C, "Principios de accesibilidad | Web Accessibility Initiative (WAI) | W3C," 2018.

[116] M. C. A. K. Alastair Campbell, "Web Content Accessibility Guidelines (WCAG) 2.2," 2018.

[117] W3C, "Understanding Conformance | Understanding WCAG 2.0," 2016.

[118] Shawn Lawton Henry., "Web Content Accessibility Guidelines (WCAG) Overview | Web Accessibility Initiative (WAI) | W3C," 2018.

[119] W3C, "How to Meet WCAG (Quickref Reference)," 2018.

[120] Michael Cooper, "Web Content Accessibility Guidelines (WCAG) 2.1," 2018.

[121] W3C, "Understanding Success Criterion 1.1.1: Non-text Content," 2018.

[122] S. Gibbs, C. Breiteneder, and D. Tsichritzis, "Data modeling of time-based media," in *Proceedings of the 1994 ACM SIGMOD international conference on Management of data - SIGMOD '94*, (New York), pp. 91–102, ACM Press, 1994.

[123] S. Shirali-Shahreza and M. H. Shirali-Shahreza, "Accessibility of CAPTCHA methods," in *Proceedings of the 4th ACM workshop on Security and artificial intelligence - AISec '11*, (New York), p. 109, ACM Press, 2011.

[124] W3C, "G158: Providing an alternative for time-based media for audio-only content," 2018.

[125] W3C, "Understanding Success Criterion 1.2.6: Sign Language (Pre-recorded)," 2018.

[126] W3C, "G151: Providing a link to a text transcript of a prepared statement or script if the script is followed," 2018.

[127] W3C, "Understanding Success Criterion 1.3.3: Sensory Characteristics," 2018.

[128] W3C, "Understanding Success Criterion 1.3.4: Orientation," 2018.

[129] W3C, "Understanding Success Criterion 1.4.1: Use of Color," 2018.

[130] W3C, "Understanding Success Criterion 1.4.7: Low or No Background Audio," 2018.

[131] W3C, "Understanding Success Criterion 1.4.8: Visual Presentation," 2018.

[132] E. McLeish, "A study of the effect of letter spacing on the reading speed of young readers with low vision," *British Journal of Visual Impairment*, vol. 25, pp. 133–143, may 2007.

[133] L. Rello and R. Baeza-Yates, "How to present more readable text for people with dyslexia," *Universal Access in the Information Society*, vol. 16, pp. 29–49, mar 2017.

[134] A. M. Sjoblom, E. Eaton, and S. D. Stagg, "The effects of letter spacing and coloured overlays on reading speed and accuracy in adult dyslexia," *British Journal of Educational Psychology*, vol. 86, pp. 630–639, dec 2016.

[135] S. D. Silva, F. M. M. Neto, R. M. De Lima, F. T. De Macedo, J. R. S. Santo, and W. L. N. Silva, "Knowledgemon hunter: A serious game with geolocation to support learning of children with autism and learning difficulties," in *Proceedings - 19th Symposium on Virtual and Augmented Reality, SVR 2017*, 2017.

[136] W3C, "Understanding Success Criterion 2.1.1: Keyboard," 2018.

[137] W3C, "Understanding Success Criterion 2.2.2: Pause, Stop, Hide," 2018.

[138] W3C, "Understanding Success Criterion 2.3.1: Three Flashes or Below Threshold," 2018.

[139] Y.-S. Chang, Y.-H. Hsueh, K.-C. Tung, F.-Y. Jhou, and D. P.-C. Lin, "Characteristics of visual fatigue under the effect of 3D animation," *Technology and Health Care*, vol. 24, pp. S231–S235, dec 2015.

[140] W3C, "Understanding Success Criterion 2.4.4: Link Purpose (In Context)," 2020.

[141] W3C, "Understanding Success Criterion 2.4.7: Focus Visible," 2020.

[142] W3C, "Understanding Success Criterion 2.5.2: Pointer Cancellation," 2018.

[143] W3C, "Understanding Success Criterion 2.5.4: Motion Actuation," 2018.

[144] W3C, "Understanding Success Criterion 2.5.6: Concurrent Input Mechanisms," 2018.

[145] W3C, "Understanding Success Criterion 3.1.1: Language of Page," 2018.

[146] W3C, "Understanding Success Criterion 3.1.3: Unusual Words," 2018.

[147] W3C, "Understanding Success Criterion 3.2.1: On Focus," 2018.

[148] W3C, "Understanding Success Criterion 3.2.5: Change on Request," 2018.

[149] W3C, "Understanding Success Criterion 3.3.1: Error Identification," 2018.

[150] W3C, "Understanding Success Criterion 3.3.4: Error Prevention (Legal, Financial, Data)," 2018.

[151] W3C, "Understanding Success Criterion 3.3.5: Help," 2018.

[152] W3C, "Understanding Success Criterion 4.1.2: Name, Role, Value," 2018.

[153] J. Brewer, "Web accessibility highlights and trends," in *Proceedings of the international cross-disciplinary workshop on Web accessibility - W4A*, (New York), p. 51, ACM Press, 2004.

[154] W3C, "Authoring Tool Accessibility Guidelines (ATAG) Overview | Web Accessibility Initiative (WAI) | W3C," 2013.

[155] O. U. Jan Richards, Inclusive Design Institute, W. Jeanne Spellman, and O. U. Jutta Treviranus, Inclusive Design Institute, "Authoring Tool Accessibility Guidelines (ATAG) 2.0," 2015.

[156] R. Calvo, A. Iglesias, and L. Moreno, "Accessibility barriers for users of screen readers in the moodle learning content management system," *Universal Access in the Information Society*, vol. 13, pp. 315–327, 8 2014.

[157] W. Chen, N. C. Sanderson, S. Kessel, and A. Królak, "Heuristic evaluations of the accessibility of learning management systems (LMSs) as authoring tools for teachers," *First Monday*, vol. 20, no. 9, 2015.

[158] W3C, "User Agent Accessibility Guidelines (UAAG) Overview | Web Accessibility Initiative (WAI) | W3C," 2018.

[159] J. Gunderson, "W3C user agent accessibility guidelines 1.0 for graphical Web browsers," *Universal Access in the Information Society*, vol. 3, pp. 38–47, mar 2004.

[160] R. V. Lerma-Blasco, J. A. Murcia Andrés, and E. Mifsud Talón, *Aplicaciones web*. Madrid, España: McGraw-Hill/Interamericana de España, 1 ed., 2013.

[161] J. P. Bigham, "Increasing web accessibility by automatically judging alternative text quality," in *Proceedings of the 12th international conference on Intelligent user interfaces - IUI '07*, (New York), p. 349, ACM Press, 2007.

[162] W. Yu, R. Kuber, E. Murphy, P. Strain, and G. McAllister, "A novel multimodal interface for improving visually impaired people's web accessibility," *Virtual Reality*, vol. 9, pp. 133–148, jan 2006.

[163] M. Rotard, C. Taras, and T. Ertl, "Tactile web browsing for blind people," *Multimedia Tools and Applications*, vol. 37, pp. 53–69, mar 2008.

[164] J. Lu, Y. Zhou, and J. Zhang, "The concept of web design patterns based on the website design process," in *2011 International Conference of Information Technology, Computer Engineering and Management Sciences*, vol. 4, pp. 49–52, 2011.

[165] W. Arasid, A. G. Abdullah, D. Wahyudin, C. U. Abdullah, I. Widiaty, D. Zakaria, N. Amelia, and A. Juhana, "An Analysis of Website Accessibility in Higher Education in Indonesia Based on WCAG 2.0 Guidelines," in *IOP Conference Series: Materials Science and Engineering* (A. G. Abdullah, A. B. D. Nandiyanto, I. Widiaty, and V. Palilingan, eds.), vol. 306, 2018.

[166] T. Domínguez Vila, E. Alén González, and S. Darcy, "Website accessibility in the tourism industry: an analysis of official national tourism organization websites around the world," *Disability and rehabilitation*, vol. 40, no. 24, pp. 2895–2906, 2018.

[167] P. Acosta-Vargas, P. Hidalgo, G. Acosta-Vargas, M. Gonzalez, J. Guaña-Moya, and B. Salvador-Acosta, "Challenges and improvements in website accessibility for health services," in *Advances in Intelligent Systems and Computing* (T. Ahram, W. Karwowski, A. Vergnano, F. Leali, and R. Taiar, eds.), vol. 1131 AISC, pp. 875–881, 2020.

[168] K. Fatima, N. Z. Bawany, M. Bukhari, and IEEE, "Usability and Accessibility Evaluation of Banking Websites," in *ICACSIS 2020: 2020 12TH INTERNATIONAL CONFERENCE ON ADVANCED COMPUTER SCIENCE AND INFORMATION SYSTEMS (ICACSIS)*, no. 12th International Conference on Advanced Computer Science and Information Systems (ICACSIS), pp. 247–255, 2020.

[169] P. Acosta-Vargas, T. Acosta, and S. Lujan-Mora, "Framework for Accessibility Evaluation of Hospital Websites," in *2018 FIFTH INTERNATIONAL CONFERENCE ON EDEMOCRACY & EGOVERNMENT (ICEDEG)* (L. Teran and A. Meier, eds.), no. 5th International Conference on eDemocracy and eGovernment (ICEDEG), pp. 9–15, 2018.

[170] N. A. Karaim and Y. Inal, "Usability and accessibility evaluation of Libyan government websites," *UNIVERSAL ACCESS IN THE INFORMATION SOCIETY*, vol. 18, no. 1, pp. 207–216, 2019.

[171] R. R. Althar and D. Samanta, "Building Intelligent Integrated Development Environment for IoT in the Context of Statistical Modeling for Software Source Code," pp. 95–115, Springer, Singapore, 2021.

[172] F. F. Borelli, G. O. Biondi, and C. A. Kamienski, "BIoTA: A Buildout IoT Application Language," *IEEE Access*, vol. 8, pp. 126443–126459, 2020.

[173] "NetBeans, Eclipse, IntelliJ IDEA - Explorar - Google Trends," jan 2022.

[174] NetBeans, "Code Assistance in the NetBeans IDE Java Editor: A Reference Guide," feb 2019.

[175] E. T. Com, "Atajos de teclado," tech. rep., 2019.

[176] G. Guerrero-Ulloa, M. J. Hornos, and C. Rodríguez-Domínguez, "TDDM4IoTS: A Test-Driven Development Methodology for Internet of Things (IoT)-Based Systems," in *Communications in Computer and Information Science* (M. Botto-Tobar, M. Zambrano Vizuete, P. Torres-Carrión, S. Montes León, G. Pizarro Vásquez, and B. Durakovic, eds.), vol. 1193 CCIS, (Quito, Ecuador), pp. 41–55, Springer, dec 2020.

[177] D. Galán, "Los IDE más usados en programación JAVA," oct 2019.

[178] Google Trends, "NetBeans, Eclipse, IntelliJ IDEA - Explorar - Google Trends."

[179] P. Manickam, S. Sangeetha, and S. V. Subrahmanya, *Component-Oriented Development and Assembly: Paradigm, Principles, and Practice using Java*. Auerbach Publications, 2013.

[180] B. Burke and R. Monson-Haefel, *Enterprise JavaBeans 3.0*. Developing Enterprise Java Components, O'Reilly Media, Incorporated, 2006.

[181] B. McLaughlin, *Building Java Enterprise Applications: Architecture*, vol. 1. .ºReilly Media, Inc.", 2002.

[182] Kevin Boone, *Applied Enterprise JavaBeans Technology - Kevin Boone - Google Books*. California, U.S.A.: Prentice Hall Professional, ilustrada ed., 2003.

[183] R. P. Sriganesh, G. Brose, and M. Silverman, *Mastering Enterprise JavaBeans 3.0*. John Wiley & Sons, 2006.

[184] Oracle, "Java Platform, Enterprise Edition (Java EE) | Oracle Technology Network | Oracle," 2021.

[185] F. Marchioni, *Practical Java EE 7 Development on WildFly: Quickstart guide for developing,deploying and securing Java EE 7 applications on WildFly application server*. ITBuzzPress, apr 2018.

[186] O. and/or its affiliates, "32.7 The Lifecycles of Enterprise Beans - Java Platform, Enterprise Edition: The Java EE Tutorial (Release 7)," 2014.

[187] T. Roman, Ed; Ambler, Scott W.; Jewell, *Dominando Enterprise Java-Beans*. Bookman, 2004.

[188] O. and/or its affiliates, "32 Enterprise Beans (Release 7)," 2014.

[189] O. and/or its affiliates, "32.2 What Is a Session Bean? - Java Platform, Enterprise Edition: The Java EE Tutorial (Release 7)," 2014.

[190] P. Louridas, "Soap and web services," *IEEE Software*, vol. 23, no. 6, pp. 62–67, 2006.

[191] U. Thakar, A. Tiwari, and S. Varma, "On Composition of SOAP Based and RESTful Services," in *Proceedings - 6th International Advanced Computing Conference, IACC 2016*, pp. 500–505, 2016.

[192] A. W. Mohamed and A. M. Zeki, "Web Services SOAP Optimization Techniques," in *4th IEEE International Conference on Engineering Technologies and Applied Sciences, ICETAS 2017*, vol. 2018-Janua, pp. 1–5, 2018.

[193] P. Bazán, *Aplicaciones, Servicios y Procesos Distribuidos. Una Visión para la Construcción de Software*. Editorial de la Universidad Nacional de La Plata (EDULP), oct 2020.

[194] K. B. Laskey and K. Laskey, "Service Oriented Architecture," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 1, no. 1, pp. 101–105, 2009.

[195] A. Masood, "Cyber Security for Service Oriented Architectures in a Web 2.0 World: An Overview of SOA Vulnerabilities in Financial Services," in *2013 IEEE International Conference on Technologies for Homeland Security, HST 2013*, pp. 1–6, 2013.

[196] R. P. Sumithra and R. Sarath, "Towards RESTful Web Service Composition for Healthcare Domain," in *2015 International Conference on Control Instrumentation Communication and Computational Technologies, ICCICCT 2015*, pp. 845–848, 2016.

[197] Oracle Corporation, "Chapter 11 Securing Web Services Using the Security Token Service (WS-* Specifications) (Sun OpenSSO Enterprise 8.0 Deployment Planning Guide)," 2010.

[198] S. P. Lee, L. P. Chan, and E. W. Lee, "Web services implementation methodology for soa application," in *2006 4th IEEE International Conference on Industrial Informatics*, pp. 335–340, 2006.

[199] S. Elfirdoussi and Z. Jarir, "An Integrated Approach Towards Service Composition Life Cycle: A Transportation Process Case Study," *Journal of Industrial Information Integration*, vol. 15, pp. 138–146, sep 2019.

[200] M. Espinoza-Mejía and P. Álvarez, "El ciclo de vida de un servicio web compuesto: virtudes y carencias de las soluciones actuales," 2007.

[201] SEMIC, "¿Qué es SOA o Arquitectura Orientada a Servicios?," apr 2020.

[202] B. Huang, *Comprehensive Geographic Information Systems*. Elsevier Science, 2017.

[203] R. Gotzhein, *Real-time Communication Protocols for Multi-hop Ad-hoc Networks: Wireless Networking in Production and Control Systems*. Computer Communications and Networks, Springer International Publishing, 2020.

[204] R. L. Maata, R. Cordova, B. Sudramurthy, and A. Halibas, "Design and Implementation of Client-Server Based Application Using Socket Programming in a Distributed Computing Environment," in *2017 IEEE International Conference on Computational Intelligence and Computing Research, ICCIC 2017*, Institute of Electrical and Electronics Engineers Inc., nov 2018.

[205] K. Ma and W. Zhang, "Introducing Browser-Based High-Frequency Cloud Monitoring System Using WebSocket Proxy," *International Journal of Grid and Utility Computing*, vol. 6, pp. 21–29, jan 2015.

[206] M. R. Rahman and S. Akhter, "Real Time Bi-Directional Traffic Management Support System with GPS and WebSocket," in *Proceedings - 15th IEEE International Conference on Computer and Information Technology, CIT 2015, 14th IEEE International Conference on Ubiquitous Computing and Communications, IUCC 2015, 13th IEEE International Conference on Dependable, Autonomic and Se*, pp. 959–964, 2015.

[207] T. Wirasingha and N. R. Dissanayake, "A Survey of WebSocket Development Techniques and Technologies," in *9th International Research Conference-KDU*, pp. 158–164, KDU Lybrary, 2016.

[208] B. Hiremath, A. K. I. J. Of, and U. 2016, "An Alteration of the Web 1.0, Web 2.0 and Web 3.0: A Comparative Study," *academia.edu*, vol. 2, no. 4, 2016.

[209] A. Bauman and R. Bachmann, "Online Consumer Trust: Trends in Research," 2017.

[210] A. Dokhanchi, S. Yaghoubi, B. Hoxha, and G. Fainekos, "Vacuity Aware Falsification for MTL Request-Response Specifications," in *IEEE International Conference on Automation Science and Engineering*, vol. 2017-Augus, (Xi'an, China), pp. 1332–1337, IEEE Computer Society, 2017.

[211] D. Cutting, "Evaluation of long-held http polling for php/mysql architecture," 2015.

[212] N. C. Zakas, "Ajax and Comet," in *Professional Javascript® for Web Developers*, pp. 701–729, Indianapolis: Wiley Publishing, Inc., oct 2015.

[213] Anisya and Y. Wandyra, "Rekayasa Perangkat Lunak Pengendalian Inventory Menggunakan Metode SMA (Single Moving Average) Berbasis AJAX (Asynchronous Javascript and XML)," *Jurnal Teknoif*, vol. 4, no. 2, pp. 11–17, 2016.

[214] A. A. Castillo, *Curso de Programación Web: JavaScript, Ajax y jQuery*. IT Campus Academy, 2017.

[215] A. Iglesias, L. Moreno, P. Martínez, and R. Calvo, "Evaluating the Accessibility of Three Open-Source Learning Content Management Systems: A Comparative Study," *Computer Applications in Engineering Education*, vol. 22, no. 2, pp. 320–328, 2014.

[216] J. V. Mansyelt and J. Johansson, "A Comparision between HTTP Long Polling and Websocket from a Battery Perspective," tech. rep., 2018.

[217] J. V. Dos Santos, M. A. Silveira de Souza, and D. F. Anderle, "Controlando Dispositivos em Tempo Real através do WebSocket," in *Tecnologías e Redes de Computadores: Estudios Aplicados*, ch. 9, Intituto Federal Catarinense, 2015.

[218] S. Wei, V. Swaminathan, and M. Xiao, "Power Efficient Mobile Video Streaming Using HTTP/2 Server Push," in *2015 IEEE 17th International Workshop on Multimedia Signal Processing, MMSP 2015* (S. Wei, V. Swaminathan, and M. Xiao, eds.), (Xiamen, China), Institute of Electrical and Electronics Engineers Inc., nov 2015.

[219] W. Słodziak and Z. Nowak, "Performance analysis of web systems based on xmlhttprequest, server-sent events and websocket," in *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology – ISAT 2015 – Part II* (A. Grzech, L. Borzemski, J. Świątek, and Z. Wilimowska, eds.), (Cham), pp. 71–83, Springer International Publishing, 2016.

[220] N. Edan, A. Al-Sherbaz, and S. Turner, "Design and Implement a Hybrid WebRTC Signalling Mechanism for Unidirectional & Bi-Directional Video Conferencing," *International Journal of Electrical and Computer Engineering*, vol. 8, pp. 390–399, feb 2018.

[221] M. contributors, "Navigation and resource timings - Web Performance | MDN," mar 2021.

[222] H. Vivas, H. Muñoz, M. Cambarieri, and M. Petroff, "Arquitectura de Software con Websocket para Aplicaciones Web Multiplataforma," *ResearchGate*, no. October, p. 11, 2014.

[223] Y. Hu and W. Cheng, "Research and Implementation of Campus Information Push System Based on WebSocket," in *Proceedings of the 2017 12th International Conference on Intelligent Systems and Knowledge Engineering, ISKE 2017*, vol. 2018-Janua, pp. 1–6, 2017.

[224] G. M. Oliveira, D. C. Costa, R. J. Cavalcanti, J. P. Oliveira, D. R. Silva, M. B. Nogueira, and M. C. Rodrigues, "Comparison between MQTT and WebSocket Protocols for Iot Applications Using ESP8266," in *2018 Workshop on Metrology for Industry 4.0 and IoT, MetroInd 4.0 and IoT 2018 - Proceedings*, pp. 236–241, 2018.

[225] D. Skvorc, M. Horvat, and S. Srbljic, "Performance Evaluation of Websocket Protocol for Implementation of Full-Duplex Web Streams," in *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2014 - Proceedings*, pp. 1003–1008, IEEE Computer Society, 2014.

[226] L. Zi-Wei, L. Shao-Bin, L. Yan, and L. Hui-Yong, "Remote ATS Simulation System Based on WebSocket Communication Protocol," in *Proceedings - 10th International Conference on Intelligent Computation Technology and Automation, ICICTA 2017*, vol. 2017-Octob, pp. 327–330, Institute of Electrical and Electronics Engineers Inc., oct 2017.

[227] G. Imre, G. Mezei, and R. Sárosi, "Introduction to a Websocket Benchmarking Infrastructure," in *2016 Zooming Innovation in Consumer Electronics International Conference, ZINC 2016*, pp. 84–87, 2016.

[228] M. A. Bashir, S. Arshad, E. Kirda, W. Robertson, and C. Wilson, "How Tracking Companies Circumvented Ad Block-Ers Using WebSockets," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC*, pp. 471–477, ACM, oct 2018.

[229] V. Wang, F. Salim, and P. Moskovits, *The Definitive Guide to HTML5 WebSocket*. Apress, 2013.

[230] V. Wang, F. Salim, and P. Moskovits, "WebSocket Security," in *The Definitive Guide to HTML5 WebSocket*, pp. 129–147, Apress, 2013.

[231] A. Wessels, M. Purvis, J. Jackson, and S. Rahman, "Remote Data Visualization through Websockets," in *Proceedings - 2011 8th International Conference on Information Technology: New Generations, ITNG 2011*, pp. 1050–1051, 2010.

[232] M. contributors, "WebSockets - Referencia de la API Web," oct 2021.

[233] L.-s. Huang, E. Y. Chen, A. Barth, E. Rescorla, and C. Jackson, "Talking to Yourself for Fun and Profit," *Proceedings of W2SP*, pp. 1–11, 2011.

[234] L. Yang and G. Yang, "Real-Time Wireless Signal Testing and Analyzing System Based on Websocket," in *Proceedings - 2016 6th International Conference on Instrumentation and Measurement, Computer, Communication and Control, IMCCC 2016*, pp. 648–652, Institute of Electrical and Electronics Engineers Inc., dec 2016.

[235] V. Wang, F. Salim, and P. Moskovits, "Introduction to HTML5 WebSocket," in *The Definitive Guide to HTML5 WebSocket*, pp. 1–12, Apress, 2013.

[236] MDN Contributors, "WebSocket - Web APIs," sep 2021.

[237] "Acerca | Node.js."

[238] S. Tilkov and S. Vinoski, "Node.js: Using JavaScript to Build High-Performance Network Programs," *IEEE Internet Computing*, vol. 14, no. 6, pp. 80–83, 2010.

[239] B. Dayley, *Node.js, MongoDB, and AngularJS Web Development*. Developer's Library, Michigan, United States: Pearson Education, ilustrada ed., 2014.

[240] Software AG, "webMethods," 2021.

[241] M. Hapner, R. Burridge, R. Sharma, J. Fialli, K. Stout, and N. Deakin, "Java Message Service v2.0," 2013.

[242] D. R. Heffelfinger, *Java EE 7 With Glassfish 4 Application Server. A Practical Guide To Install And Configure The Glassfish 4 Application Server And Develop Java EE 7 Applications To Be Deployed To This Server*. Birmingham, UK: Packt Pub, 2014.

[243] K. Haase, "Java Messaging Service API Tutorial," p. 278, 2001.

[244] L. Erdogan, *Java Message Service (JMS) for J2EE*. No. 1.1, O'Reilly, 1 ed., aug 2002.

[245] L. Grewe, S. Krishnagiri, and J. Cristobal, "Metrics of a System for Disaster Relief," in *VECIMS 2008 - IEEE Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems Proceedings*, pp. 45–50, IEEE, 2008.

[246] M. Ricbards, R. Monson-Haefel, and D. A. Chappell, *Java Message Service*, vol. 2. 2009.

[247] Oracle, "Java EE 6 APIs - The Java EE 6 Tutorial," 2018.

[248] Oracle, "Introduction to Oracle Java Message Service," 2014.

[249] Oracle Corporation, "Using the JMS API in Java EE Applications," 2017.

[250] B. Aumaille, *J2EE: Desarrollo de Aplicaciones Web*. Barcelona: Ediciones ENI, 1 ed., nov 2002.

[251] IBM Corporation, "Java EE Application Resource Declarations," jun 2021.

[252] JAVA, "JMS API Concepts," 2013.

[253] Oracle Corporation, "Using the JMS API in Java EE Applications," 2010.

[254] Oracle, "The Lifecycles of Enterprise Beans," 2017.

[255] Oracle, "Overview of the JMS API," 2017.

[256] Y. Fain, *Java Programming. 24-Hour Trainer*. Indianápolis, Indiana, Canadá: Wiley Publishing, Inc., 2 ed., 2015.

[257] ORACLE, “Publish/Subscribe Messaging Domain,” 2010.

[258] Universidad de Alicante, “Introducción a JMS (Java Message Service),” pp. 1–31, 2014.

[259] Oracle Corporation, “Connecting to the JMS Server by Using JNDI (Using the JMS Binding Component),” 2010.

[260] IBM, “Developing a JMS client - IBM Documentation,” jul 2021.

Desarrolladores del material de apoyo

Ariel Oswaldo Fernández Loor

Estudiante del noveno semestre de la Carrera de Ingeniería en Sistemas de la Universidad Técnica Estatal de Quevedo.
Cuenta con experiencia como desarrollador de aplicaciones informáticas entre las que cuenta: Un sistema para personas con discapacidad visual Nawi (https://aplicaciones.uteq.edu.ec/nawi/), del cual ha sido parte del equipo de investigación, este trabajo fue enviado para su próxima publicación. Como parte de sus prácticas preprofesionales desarrolló una aplicación para la gestión de documentos en formato PDF para el Registro de la Propiedad Municipal del Cantón Quevedo. Actualmente se encuentra trabajando en el proyecto de vinculación con la sociedad dentro de un convenio entre la UTEQ y el GADM El Empalme, en el que están desarrollando una aplicación Web accesible para personas con diferentes discapacidades (http://fyc.uteq.edu.ec:8080/).
Adicional a su formación como Ingeniero en Sistemas, ha realizado cursos externos como Python para ingenieros nivel básico impartido por INTEGCAP, cursos impartidos por CISCO: Introduction to IoT Español, JavaScript Essentials 1 (JSE), Introduction to Cybersecurity y Programación Básica Python.
Su campo de interés es la programación en lenguajes como JAVA y C#, y la implementación y gestión de bases de datos: Cuenta con experiencia en Microsoft SQL Server y PostgreSQL. Además, Es aficionado a los proyectos de Sistemas basados en Internet de las cosas (Internet of Things: IoT), entre estos ha desarrollado un sistema IoT de orientación para personas con discapacidad visual. Por otra parte, desarrolló un dispositivo para el control y mantenimiento de aguas en piscinas. Por último, trabajó en el desarrollo de un sistema basado en IoT para personas con discapacidad auditiva a mejorar la comunicación entre personas a través de conversión del lenguaje de señas a voz o texto y viceversa.

Duval Ricardo Carvajal Suárez

Estudiante de noveno semestre de la Carrera de Ingeniería en Sistemas de la Universidad Técnica Estatal de Quevedo.
Como parte de su experiencia: es miembro del equipo de desarrollo de las aplicaciones BioForest https://bioforest.uteq.edu.ec, Repositorio UTEQ https://aplicaciones.uteq.edu.ec/RepositorioUTEQ/, Test-Driven Methodology for IoT (Internet of Things)-based Systems TddM4IotS 2.0 https://aplicaciones.uteq.edu.ec/tddm4iots/.
Adicional a su formación profesional, cuenta con certificaciones de manejo básico en Adobe Photoshop y Adobe Illustrator impartido por la municipalidad de la ciudad de Machala, Python para ingenieros nivel básico impartido por INTEGCAP, Introducción al desarrollo web I impartido por el Instituto de Economía Internacional, Desarrollo de Apps Móviles impartido por la Universidad Complutense Madrid e Introducción a la ciberseguridad impartido por CISCO.
Su interés en el desarrollo de aplicaciones web predomina teniendo un claro control sobre el Fronted y Backend, desempeño en el diseño de las interfaces de usuario (GUI), creación de iconos y uso de colores apara asegurar la accesibilidad.

Resumen

TECNOLOGÍAS PARA EL DESARROLLO DE APLICACIONES WEB

Los desarrolladores de aplicaciones web se enfrentan a distintas situaciones al momento de plantear el desarrollo de un software, los clientes entregan una serie de requisitos funcionales y no funcionales que resultan complejos de abarcar sin una guía adecuada. En este libro se presentan varias tecnologías que se pueden usar para el desarrollo web por medio del lenguaje de programación Java.

Dependiendo de los objetivos a cumplir, cada aplicación puede necesitar de distintos recursos para poder completar los procesos de negocio requeridos. Frameworks, APIS, librerías jar, librerías JavaScript, librerías CSS entre otros son algunos complementos mencionados en los contenidos de este libro, brindando una documentación básica de los recursos destinados para ser implementados y consumidos por el desarrollador, con la finalidad de construir un proyecto escalable y sencillo de comprender.

En cada capítulo se explican las funcionalidades que tienen las tecnologías mencionadas, detallando en que lenguaje fueron desarrolladas, la compatibilidad con navegadores, ventajas, desventajas, versiones existentes y comparaciones con otras tecnologías. Se ha construido una aplicaciones por cada tecnología revisada, explicando paso a paso como se debe crear cada proyecto, mostrando capturas de pantalla y fragmentos de código especificando su debido funcionamiento. Además, se ha proporcionado un video tutorial didáctico por cada tecnología, para así despejar dudas o reducir errores que se pueden producir al momento de poner a prueba el contenido del texto.

El lector, uniendo todos los complementos necesarios y siguiendo paso a paso como se debe utilizar cada tecnología, será capaz de desarrollar una aplicación web que cubra todas las necesidades del cliente y que, además, se garantice su correcto funcionamiento. Por otro lado, le permitirá a cada persona relacionada con el ambiente web ampliar su conocimiento teórico y práctico sobre las tecnologías de desarrollo, mejorando su desempeño como desarrollador de aplicaciones web.

ISBN: 978-9942-33-543-2

compAs

Grupo de capacitación e investigación pedagógica

@grupocompas.ec
compasacademico@icloud.com