LIBRO DE TEXTO
ISBN Digital 978-628-7693-52-4

PROGRAMACIÓN WEB

DESDE LA INGENIERÍA DE SISTEMAS

PROGRAMACIÓN WEB DESDE LA INGENIERÍA DE SISTEMAS

ISBN Digital 978-628-7693-52-4

Facultad de Ingeniería
Programa Ingeniería de Sistemas
Grupo de Investigación GIECOM-GITUA

ISBN Digital 978-628-7693-52-4

PROGRAMACIÓN WEB DESDE LA INGENIERÍA DE SISTEMAS

DISEÑO DE PORTADA
Por los Autores mediante ChatGpt4.0.

PUBLICADO POR:
Editorial - Universidad de la Amazonia
2026.

Diseño y diagramación:
Los Autores

Esta Obra está enmarcada bajo el plan de mejoramiento del programa de Ingeniería de Sistemas de la Universidad de la Amazonia.

Esta obra deberá ser citada de la siguiente manera:

Vargas-Losada, H. F.; (Coordinador) 2026. Programación Web desde la Ingeniería de Sistemas. (1ra). Editorial Universidad de la Amazonia. pp.216. Tamaño (20 x 26 cm).

Incluye bibliografía.

ISBN Digital 978-628-7693-52-4

Número y año de edición: Primera edición, febrero 2026.

1. Ingeniería 2. Programación 3. Bases de datos 4. Educación, 5. Informática

CDD: 005.1 ed.22

Vicerrectoría de Investigación e Innovación
Editorial Universidad de la Amazonia
Campus Porvenir: Calle 17 Diagonal 17 con Carrera 3F - Barrio Porvenir
Contacto: vrinvestigaciones@udla.edu.co - editorial@uniamazonia.edu.co

Florencia, Caquetá 2026.

Diseño de cubierta: Autores mediante ChatGpt4.0
Diseño y diagramación: Los autores
Tiraje: Online.

Universidad de la Amazonia
VICERRECTORÍA DE INVESTIGACIÓN E INNOVACIÓN
EDITORIAL UNIAMAZONIA

Esta Obra está enmarcada bajo el plan de mejoramiento del Programa de Ingeniería de Sistemas de la Universidad de la Amazonia.

Impreso y hecho en Colombia / Printed and made in Colombia

EQUIPO DE AUTORES

CAPÍTULO 1. Introducción y Arquitectura de la Programación Web

Gustavo Andrés Macías Ramos
Estudiante del programa de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad de la Amazonia, Grupo de investigación GIECOM.
gu.macias@udla.edu.co
ORCID: https://orcid.org/0009-0002-7142-0814

Heriberto Fernando Vargas Losada
Docente tiempo completo Universidad de la Amazonia, Grupo de investigación GIECOM,
heri.vargas@udla.edu.co
ORCID: https://orcid.org/0000-0002-8561-0793

Edwin Eduardo Millán Rojas
Docente tiempo completo Universidad de la Amazonia, Grupo de investigación en Informática, Innovación y Tecnología de la Universidad de la Amazonia GITUA.
e.millan@udla.edu.co
ORCID: https://orcid.org/0000-0002-4258-4601

CAPÍTULO 2. Programación web del lado del Cliente.

Michael Mauricio Orjuela Cepeda
Estudiante del programa de Ingeniería de Sistemas, Facultad de Ingeniería, Grupo de investigación en Informática, Innovación y Tecnología de la Universidad de la Amazonia GITUA, correo: mi.orjuela@udla.edu.co, ORCID: https://orcid.org/0009-0001-0678-9044

Heriberto Fernando Vargas Losada
Docente tiempo completo Universidad de la Amazonia, Grupo de investigación GIECOM
heri.vargas@udla.edu.co
ORCID: https://orcid.org/0000-0002-8561-0793

Fredy Antonio Verástegui González
Docente tiempo completo Universidad de la Amazonia, Grupo de investigación en Informática, Innovación y Tecnología de la Universidad de la Amazonia GITUA
f.verastegui@udla.edu.co
ORCID: https://orcid.org/0000-0002-0525-6879

EQUIPO DE AUTORES

CAPÍTULO 3. Programación Web del lado del Servidor.

Sebastián Ariza González
Estudiante del programa de Ingeniería de Sistemas, Facultad de Ingeniería, Grupo de investigación en Informática, Innovación y Tecnología de la Universidad de la Amazonia GITUA.
je.rivas@udla.edu.co
ORCID: https://orcid.org/0009-0007-4242-9736

Edwin Eduardo Millán Rojas
Docente tiempo completo Universidad de la Amazonia, Grupo de investigación en Informática, Innovación y Tecnología de la Universidad de la Amazonia GITUA
e.millan@udla.edu.co
ORCID: https://orcid.org/0000-0002-4258-4601

Denis Lorena Álvarez Guayara
Docente tiempo completo Universidad de la Amazonia, Grupo de investigación en Informática, Innovación y Tecnología de la Universidad de la Amazonia GITUA
d.alvarez@udla.edu.co
ORCID: https://orcid.org/0000-0001-8939-8139

CAPÍTULO 4. Persistencia y Servicios Web.

Juan Sebastián Mejía Suaza
Estudiante del programa de Ingeniería de Sistemas, Facultad de Ingeniería, Grupo de investigación en Informática, Innovación y Tecnología de la Universidad de la Amazonia GITUA
juans.mejia@udla.edu.co
ORCID: https://orcid.org/0009-0001-2355-8520

Fredy Antonio Verástegui González
Docente tiempo completo Universidad de la Amazonia, Grupo de investigación en Informática, Innovación y Tecnología de la Universidad de la Amazonia GITUA.
f.verastegui@udla.edu.co
ORCID: https://orcid.org/0000-0002-0525-6879

Denis Lorena Álvarez Guayara
Docente tiempo completo Universidad de la Amazonia, Grupo de investigación en Informática, Innovación y Tecnología de la Universidad de la Amazonia GITUA
d.alvarez@udla.edu.co
ORCID: https://orcid.org/0000-0001-8939-8139

TABLA DE CONTENIDO

PRESENTACIÓN

El libro "*Programación Web desde la Ingeniería de Sistemas*" está diseñado para servir como una guía integral sobre el desarrollo de aplicaciones web, abordando tanto la programación del lado del cliente como la del servidor, y proporcionando una base sólida para construir aplicaciones modernas, escalables y seguras. A través de sus capítulos, el lector obtendrá un entendimiento profundo de los conceptos fundamentales, tecnologías clave y mejores prácticas en el desarrollo web.

En el Capítulo 1: *Introducción y Arquitectura de la Programación Web*, se ofrece una mirada retrospectiva a la evolución de la programación web, desde sus inicios hasta la actualidad, destacando los cambios significativos y los avances tecnológicos que han dado forma a las aplicaciones web modernas. Este capítulo cubre los fundamentos del protocolo HTTP, esencial para la comunicación en la web, y explora la importancia de CSS para el diseño visual y de JavaScript para la interactividad en las páginas web. Además, se discuten las arquitecturas de las aplicaciones web, explicando cómo interactúan los diferentes componentes como el cliente, el servidor y la base de datos, y se presentan metodologías de desarrollo como Agile, que son vitales para gestionar proyectos web de manera efectiva.

El Capítulo 2: *Programación Web del Lado del Cliente*, se enfoca en las tecnologías y herramientas que permiten a los desarrolladores crear interfaces de usuario interactivas y dinámicas. Se examinan los lenguajes del lado del cliente, principalmente JavaScript, y sus frameworks populares como React, Angular y Vue.js, que facilitan la creación de aplicaciones complejas con una estructura de código más organizada y reutilizable. El capítulo también aborda el Modelo de Objetos del Documento (DOM), los eventos y la validación de entradas de datos, elementos cruciales para mejorar la experiencia del usuario y la seguridad de la aplicación. Además, se incluyen consideraciones específicas sobre el rendimiento en el navegador y la compatibilidad entre diferentes navegadores.

El Capítulo 3: *Programación Web del Lado del Servidor,* abarca la lógica del negocio y la gestión de datos detrás de una aplicación web. Este capítulo introduce los lenguajes de programación del lado del servidor, como PHP, Python, Node.js y Java, explicando cómo se utilizan para procesar solicitudes, manejar sesiones y conectar con bases de datos. Se profundiza en el procesamiento de formularios, incluyendo la validación del lado del servidor y la sanitización de datos para evitar vulnerabilidades. También se exploran técnicas de manejo de archivos y consideraciones de seguridad, esenciales para proteger las aplicaciones contra amenazas como la inyección de código y el acceso no autorizado.

En el Capítulo 4: *Persistencia y Servicios Web*, el enfoque se traslada a la integración de datos y la interoperabilidad entre sistemas mediante servicios web. Se explica cómo conectar aplicaciones a bases de datos, utilizando medios de conexión adecuados y asegurando la persistencia de los datos. El capítulo proporciona una visión general de los servicios web, incluyendo RESTful y SOAP, y detalla las tecnologías subyacentes como JSON y XML, que facilitan el intercambio de datos entre aplicaciones. Además, se guía al lector a través del proceso de publicación y consumo de servicios web, destacando cómo estas tecnologías permiten a las aplicaciones web comunicarse y funcionar de manera integrada en un ecosistema digital más amplio.

En su conjunto, "*Programación web*" es un recurso esencial para estudiantes, profesionales y cualquier persona interesada en el desarrollo web. Cada capítulo está diseñado para construir sobre el anterior, ofreciendo una progresión lógica desde los conceptos básicos hasta las técnicas avanzadas. A través de explicaciones detalladas, ejemplos prácticos y ejercicios aplicados, el libro no solo enseña los aspectos técnicos de la programación web, sino que también enfatiza la importancia de la planificación, el diseño y la seguridad en la creación de aplicaciones web robustas y eficientes. Esta obra se convierte así en una herramienta indispensable para aquellos que buscan dominar el arte y la ciencia del desarrollo web.

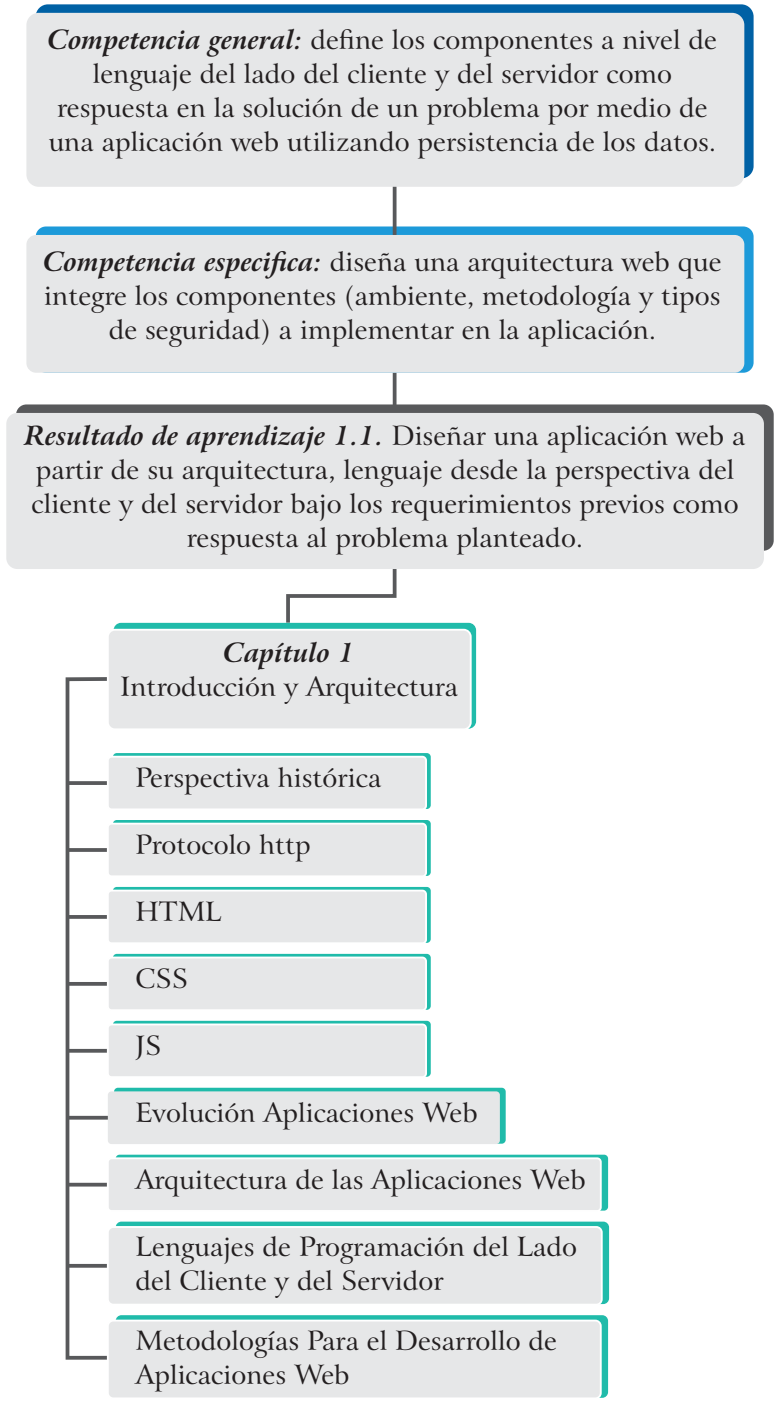
CAPÍTULO 1.

INTRODUCCIÓN Y ARQUITECTURA DE LA PROGRAMACIÓN WEB

Gustavo Andrés Macías Ramos
Heriberto Fernando Vargas Losada
Edwin Eduardo Millán Rojas

[1]*Estudiante del programa de Ingeniería de Sistemas, Facultad de Ingeniería Universidad de la Amazonia, correo: gu.macias@udla.edu.co.*
[2]*Docente de tiempo completo del programa de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad de la Amazonia, correo: heri.vargas@udla.edu.co.*
[3]*Docente de tiempo completo del programa de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad de la Amazonia, correo: e.millan@udla.edu.co.*

ESQUEMA | CAPÍTULO 1.



La competencia general está determinada de la siguiente forma: "Define los componentes a nivel de lenguajes para el cliente y el servidor como respuesta en la solución de un problema por medio de una aplicación web utilizando persistencia de los datos". Asociada a la siguiente competencia específica para la cual se desarrolla en el capítulo uno, definida como: "Diseña una arquitectura web que integre los componentes (ambiente, metodología y tipos de seguridad) a implementar en la aplicación".

El resultado de aprendizaje asociado a la competencia del capítulo uno, se estableció de la siguiente forma R1.1 Diseñar una aplicación web a partir de su arquitectura, lenguaje desde la perspectiva del cliente y del servidor bajo los requerimientos previos como respuesta al problema planteado. Los niveles de desempeño definidos para el resultado de aprendizaje son expuestos en la tabla 1.

Tabla 1.
Niveles de desempeños del resultado de aprendizaje R1.1

Avanzado (5-4,8)	Intermedio (4,7-4)	Básico (3.9 -3)	Bajo (2.9-0)
Diseña una aplicación web mediante la implementación de una arquitectura para sistemas web que incluyen lenguaje desde la perspectiva de lado del Cliente y del servidor en respuesta a un problema previo.	Diseña una aplicación web mediante la implementación de una arquitectura para sistemas web presentando mínimas fallas operativas dando respuesta total al problema previo.	Diseña una aplicación web mediante la implementación de una arquitectura para sistemas web presentando algunas fallas operativas y/o de usabilidad y/o de persistencia que da respuesta parcial a un problema previo.	Diseña una aplicación web mediante la implementación de una arquitectura para sistemas web, pero evidencia fallas operativas, de usabilidad, de persistencia que no da respuesta total a un problema previo.

Nota. *Trabajo realizado a partir del desarrollo de resultados de aprendizaje. Fuente: elaboración propia.*

Las actividades de evaluación determinadas para el resultado de aprendizaje son las siguientes: una evaluación del aspecto teórico y un taller con actividades prácticas las cuales se encontrarán en el apartado de evaluación.

Métodos y Materiales de Aprendizaje

La metodología de autoaprendizaje es un enfoque educativo que pone a la persona que está aprendiendo en el centro del proceso de aprendizaje, haciéndolo responsable de su propia educación y crecimiento. A diferencia de

los métodos de enseñanza tradicionales, en los que el educador desempeña un papel activo en guiar y dirigir el proceso de aprendizaje, en la autoformación, el alumno es quien toma la iniciativa en la fijación de sus metas educativas, en la búsqueda de los recursos necesarios y en la organización de su progreso académico de manera independiente.

En este capítulo, se destaca la importancia de utilizar material en línea como una herramienta efectiva para facilitar el autoaprendizaje del estudiante. Además de los recursos en línea, se complementa con ejercicios prácticos que ayudan a consolidar el entendimiento teórico del estudiante.

Entre los materiales de aprendizaje en línea que se incluyen en esta metodología, se encuentran ejemplos de código específicos para cada tema tratado, ilustraciones, tablas, recomendaciones de sitios web para fortalecer los conocimientos adquiridos, y explicaciones detalladas de cada tema. Estos recursos en línea están diseñados para ser accesibles y fáciles de usar, lo que permite a los estudiantes aprender a su propio ritmo y en su propio tiempo.

La metodología de autoaprendizaje también fomenta la creatividad y la innovación en el proceso de aprendizaje. Al dar a los estudiantes la libertad de establecer sus propias metas educativas y de buscar los recursos que necesitan, se les anima a pensar de manera crítica y a encontrar soluciones creativas a los problemas.

Además, la autoformación también puede ser una herramienta efectiva para aquellos que buscan actualizar sus habilidades y conocimientos en un área específica. Al ofrecer una amplia variedad de recursos en línea y ejercicios prácticos, los estudiantes pueden personalizar su experiencia de aprendizaje y centrarse en las áreas que más les interesan o en las que necesitan mejorar.

Desarrollo Temático

Los contenidos deben estar de acuerdo con los siguientes ítems y desarrollarlos de la siguiente forma:

Perspectiva histórica de la programación web

Para comprender la programación web, es esencial tener claro que es "World Wide Web" (www). Este tejido de información interconectada, accesible a través de navegadores, tuvo sus inicios en 1966. La Web 1.0, que surgió en 1990, se caracterizaba por ofrecer información estática sin interactividad. La evolución hacia la Web 2.0 para el 2004 introdujo funcionalidades como comentarios, apartados donde podían compartir sus opiniones y debatir con otras personas, permitiendo a los usuarios compartir información de manera activa.

En 2010, la Web 3.0, también conocida como web semántica, incorporó lenguajes de red y se centró en la personalización de contenidos, adaptándose

a las preferencias de los usuarios. La más reciente, la Web 4.0, que debutó en 2016, se enfoca en procesos más inteligentes y predictivos, marcando un avance significativo en la interacción web (Latorre, M., 2018).

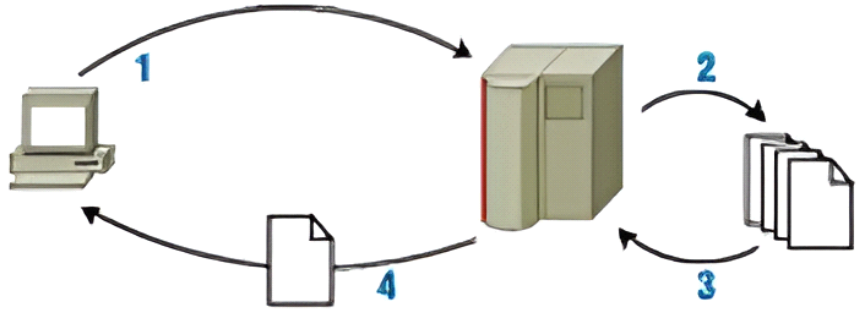
Al hablar de la Web, es imprescindible mencionar a Internet. Lawrence (Larry) Roberts, reconocido como "el padre del internet", lideró el equipo de ingenieros que creó ARPANET, el precursor de la internet actual. En 1972, Robert Kahn, junto a Lawrence Roberts y Vinton Cerf, desarrollaron el modelo de TCP/IP, que permitió la comunicación entre computadoras independientemente de sus sistemas operativos y hardware, sentando las bases para la interconexión de redes diversas.

Además, el lenguaje HTML, el protocolo HTTP y las URL, fundamentales para la creación y funcionamiento de la Web, fueron creados por Tim Berners-Lee, conocido como "el padre de la web" (Luján-Mora, S., 2002). Estas contribuciones han sido fundamentales para la expansión y el funcionamiento de la internet tal como la conocemos hoy en día.

Según Latorre, M. (2018), en sus inicios, la Web 1.0 se limitaba a mostrar información estática. Estas primeras páginas web eran consideradas estáticas, ya que su contenido permanecía inmutable a menos que el programador o administrador modificara manualmente el código HTML que las conformaba.

De acuerdo con Montávez Sánchez, M., & Segura Sánchez, R. (2023), el funcionamiento de una web estática implica que cuando un usuario solicita una página web, esta solicitud es recibida por un servidor. Posteriormente, el servidor busca la página en su repositorio de páginas y, una vez localizada, la recupera para enviarla al navegador del usuario, quien finalmente la visualiza, tal como se detalla en la figura 1.

Figura 1.
Funcionamiento de una web estática

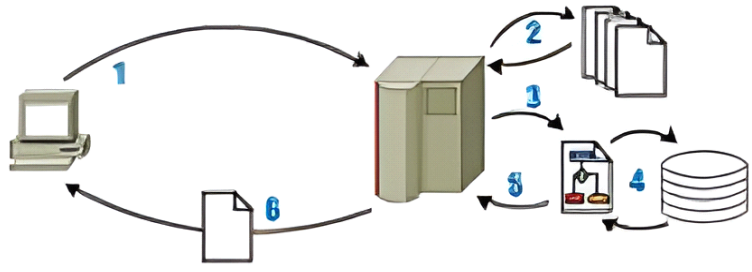


Nota. *Arquitectura de programación web: back-end. Fuente: Montávez Sánchez & Segura Sánchez (2023).*

Según Montávez Sánchez & Segura Sánchez (2023), las limitaciones de las webs estáticas en satisfacer las diversas necesidades de los usuarios fueron el catalizador para el surgimiento de lo que hoy conocemos como una web dinámica. A diferencia de las web estática, la web dinámica tiene la capacidad de adaptar su contenido según las necesidades y solicitudes de los usuarios, ofreciendo servicios y funcionalidades más interactivas y personalizadas.

El proceso de funcionamiento de una web estática inicia con el cliente solicitando una página web. Luego, el servidor receptor de la solicitud busca la página, evaluando si es necesario ejecutar código en el servidor para crearla. En caso afirmativo, se procede con la ejecución, y en ocasiones se accede a una o varias bases de datos para generar la página. Una vez completados estos pasos, la página está lista para que el cliente la visualice en su navegador. La operativa de las web dinámica, como se muestra en la figura 2, destaca por su capacidad de adaptación y personalización del contenido en tiempo real, ofreciendo una experiencia más interactiva y ajustada a las preferencias individuales de los usuarios.

Figura 2.
Funcionamiento de una web dinámica



Nota. *Arquitectura de programación web: back-end. Fuente: Montávez Sánchez & Segura Sánchez (2023).*

A pesar de que la web dinámica destaca por ofrecer una amplia gama de funcionalidades y servicios en comparación con la estática, esta última aún tiene su lugar en la actualidad. Según Montávez Sánchez, M., & Segura Sánchez, R. (2023), la web estática continúa siendo útil para mostrar información estática que se desea mantener inalterada a lo largo del tiempo, sin necesidad de cambios constantes. En los inicios de la web, la mayoría de los sitios eran estáticos. En este modelo, cada página es un documento completo, elaborado y almacenado previamente en el servidor. Cuando un usuario solicita la página, el servidor envía exactamente ese archivo, sin modificar su contenido. Esta simplicidad técnica permite que los sitios estáticos sean rápidos, predecibles y fáciles de mantener cuando el volumen de contenido es limitado. Sin embargo, esta misma rigidez implica que todos los visitantes ven la misma información, sin importar quiénes sean o qué necesidades particulares tengan.

Con el tiempo, a medida que las expectativas de los usuarios crecieron y la complejidad de los servicios digitales aumentó, surgió la necesidad de un enfoque más flexible. Así apareció la web dinámica. En este modelo, las páginas no existen como archivos completos previamente definidos. En cambio, se construyen en tiempo real cada vez que un usuario accede al sitio. El servidor consulta bases de datos, procesa las peticiones y genera el contenido en función de variables como las preferencias del usuario, su historial de navegación o la información más actual disponible.

En definitiva, la diferencia entre web estática y web dinámica no es solo técnica, sino conceptual. Mientras la web estática entrega información fija e idéntica a todos, la web dinámica ofrece una interacción viva, donde el contenido se adapta continuamente al contexto y a cada individuo. Este salto ha sido clave para que internet pase de ser un simple repositorio de documentos a convertirse en un ecosistema activo de servicios, comunicación y personalización.

En los años 2000, con la consolidación del internet en el mercado y el aumento del uso de computadoras, la creciente demanda de diversos servicios y aplicaciones por parte de los usuarios impulsó la creación de una variedad de lenguajes de programación para abordar estas necesidades, como indican Alay, J. I. G., & Sevillano, R. P. C. (2022).

La tabla 2 presenta algunos lenguajes de programación desarrollados en los años 90, cada uno con un propósito específico. Aunque sus objetivos difieren, muchos de ellos estaban orientados al desarrollo web y la creación de aplicaciones dinámicas, en sintonía con la expansión de internet en esa época.

Tabla 2

Evolución de los lenguajes de programación en los 90

Año	Lenguaje	Nivel	Uso
1987	PERL	ALTO	Creado para reportes en sistemas Unix.
			Imágenes generadas por computadora.
			Administración de sistemas.
			Programación de gráficos.
1991	PYTHON	ALTO	Aplicaciones Web.
			Desarrollo de software.
			Seguridad informática.
1993	RUBY	ALTO	Desarrollo de aplicaciones Web, Ruby on Rails
1995	JAVA	ALTO	Programación Web.
			Desarrollo de aplicaciones Web.
			Desarrollo de Software.
1995	JAVASCRIPT	ALTO	Desarrollo de web dinámica.
			Documentos PDF.
			Navegadores Web.
1995	PHP	ABIERTO	Construcción y mantenimiento de páginas web dinámicas.

Nota. *Evolución de los sistemas de lenguaje de programación a lo largo de la historia. Fuente: Alay & Sevillano (2022).*

Por otro lado, la tabla 3 exhibe lenguajes de programación nacidos a partir de los años 2000 y el presente. En contraste con los de la tabla 2, estos lenguajes tienen propósitos más complejos, buscando abarcar una amplia gama de usos. A pesar de su diversidad de enfoques, muchos de estos lenguajes también encuentran aplicación en entornos web, complementando a otros lenguajes y presentando innovadoras herramientas para satisfacer las cambiantes demandas de los clientes en el ámbito digital.

Tabla 3.
Evolución de los leguajes de programación hasta la actualidad

Año	Lenguaje	Nivel	Uso
2001	C#	Orientado a objeto	Funciona para crear sitios y aplicaciones Web.
			Desarrollo de software para sistemas operativos Windows.
			Desarrollo de aplicaciones móviles.
2009	GO	Abierto	Destinado a mejorar el entorno de trabajo de programadores.
			Permite detectar errores en la sintaxis durante la compilación y no durante la aplicación.
			Es muy iterativo, eficiente y rápido.
			Es utilizado para el desarrollo de servicios en la nube.
			Permiten desarrollar aplicaciones, sistemas o servicios de la nube.
			Con gran estabilidad, seguridad y rapidez.
2017	JAVA SE 9	Abierto	Modularización de la JDK,
			Ofrece un completo soporte para http 2.0.
			Interactuar al mismo tiempo al estilo "Read–eval–print loop" o RELP.
2019	JAVA SE 12	Abierto	Mejora el proceso de compilación del JDK,
			Simplifica la estructura de código diaria, expresión tipo switch,
			Optimiza el recolector de basura G1, devolviendo así un conjunto de la memoria Java al sistema operativo cuando este se encuentre inactivo.
2019	PHP	Abierto	Este lenguaje nace en 1994 pero se sigue perfeccionando.
			La aplicación principal del lenguaje PHP, es estructurar sitios web en WordPress.
			Tiene la capacidad de conectar el servidor y la interfaz de usuario, tomando todo el código HTML.

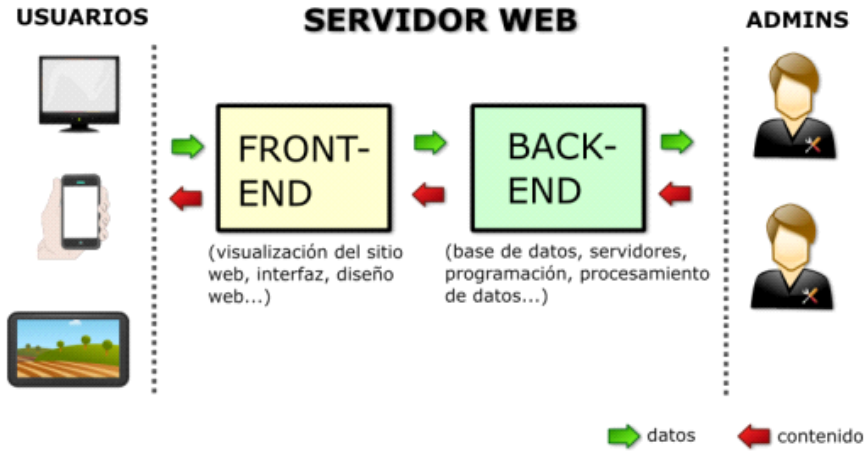
Nota. *Evolución de los sistemas de lenguaje de programación a lo largo de la historia. Fuente: Alay & Sevillano (2022).*

En el contexto de la web dinámica, es esencial mencionar dos conceptos fundamentales que han surgido gracias a ellas y que juegan un papel crucial en el desarrollo web actual: la programación Front-end y Back-end, como destacan Montávez Sánchez & Segura Sánchez (2023).

El Front-end se enfoca en la parte visual de la web, es decir, la interfaz que el cliente ve y con la que interactúa, mientras que el Back-end constituye la parte sólida y funcional del aplicativo web, encargándose principalmente del procesamiento de datos y la interacción con el gestor de datos del servidor.

En la figura 3 se representa de manera gráfica el funcionamiento del Front-end y Back-end, resaltando la importancia de ambos en la creación de webs dinámicas. Esta división de tareas es fundamental para garantizar una experiencia web completa y eficiente, donde la combinación adecuada de Front-end y Back-end es clave para el éxito de un sitio web moderno.

Figura 3.
Esquema programación cliente/servidor.



Nota. *Arquitectura de programación web: back-end. Fuente: Montávez Sánchez & Segura Sánchez (2023).*

Según Marco (2023), la visión general de la historia de la web se puede describir por seis líneas de tiempo:

1. La primera, abarcando 1991-1994, presenta las primeras versiones de HTML, CSS, JavaScript, XML, SVG, IE, Mosaic y Chrome, con una explicación detallada de cada evento.
2. La segunda línea temporal, de 1995 a 1999, se enfoca en la primera guerra de navegadores entre Internet Explorer y Netscape, con una descripción minuciosa.
3. La tercera línea de tiempo, de 2000 a 2003, destaca el reinado de Internet Explorer y los eventos relevantes de ese período.
4. La cuarta línea temporal, de 2004 a 2009, analiza la segunda guerra de navegadores entre Internet Explorer y Firefox, resaltando los eventos clave.
5. La quinta línea temporal se centra en la tercera guerra de navegadores entre Internet Explorer y Google, abarcando 2010-2015, con una explicación detallada de los sucesos.
6. La sexta y última línea de tiempo se enfoca en el dominio de Google desde 2016 hasta la actualidad, detallando los eventos más significativos de cada año.

Según los Colaboradores de Wikipedia (2024), la programación tiene sus raíces en 1801 con el uso de tarjetas perforadas en máquinas de tejer, marcando los inicios de la programación con los primeros códigos de computadoras y los pioneros que sentaron las bases de la programación contemporánea.

Durante las décadas de 1950 a 1980, se establecieron los paradigmas de la programación, surgieron los primeros compiladores y se gestaron los precursores de lenguajes actuales como C. En este período se vieron el desarrollo de lenguajes como C++, Ada, Perl, Objective-C y Tcl.

En la década de 1990, con la llegada del internet, surgieron nuevos lenguajes para el desarrollo orientados a la web, algunos son: HTML, JS, Ruby, PHP, entre otros, marcando un hito en la evolución de la programación. Finalmente, entre 2000 y 2015, se destacan lenguajes como C#, Visual Basic .NET, Swift, entre otros, cada uno con su contribución única al panorama de la programación moderna, resaltando su propósito y relevancia en el desarrollo tecnológico actual.

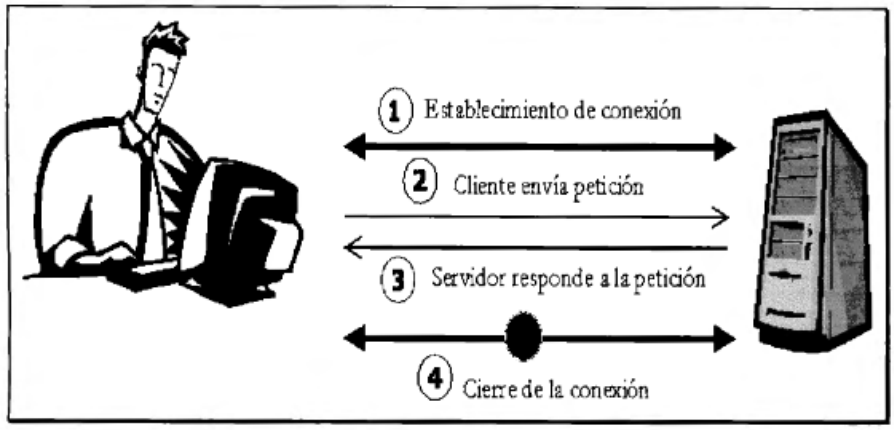
Según Gómez, P. (2023), en su página web se remonta el origen de la programación hasta 1801, destacando la creación del primer sistema de programación y mencionando a sus pioneros clave. Además, realiza un breve recorrido histórico mencionando lenguajes emblemáticos de la época como C y C++. Finalmente, explora la interrelación entre internet y la programación, señalando cómo ha dado lugar al surgimiento de lenguajes como HTML, JavaScript y Python.

Protocolo http

HTTP por sus siglas en inglés (Hypertext Transfer Protocol) se encarga de trasmitir mensajes utilizando el protocolo de transporte TCP en sistemas de información de hipertexto. Habitualmente, el cliente envía una solicitud de recurso al servidor, y este responde con el recurso solicitado (WEB, H., 2013). Desarrollado por el W3C y el IETF, organismos internacionales que buscan facilitar la comunicación entre software web, clientes, servidores y proxies, permitiendo la interacción a través de la arquitectura cliente-servidor. Su funcionamiento se basa en respuestas codificadas de tres dígitos que indican diversos estados, como conexión rechazada, realizada, redirigida, o errores tanto del cliente como del servidor (Bustamante, 2017).

Creado en 1990 por Tim Berners-Lee, Roy Fielding y Henrik Nielsen, el objetivo del protocolo HTTP era posibilitar el intercambio de información sin importar su formato, ya sea texto, hipertexto, imágenes u otros objetos distintos a los documentos HTML (Roberto, G. C., Eduardo, G. G., & Ariel, O. R., 1999). Según la figura 1.4, el funcionamiento del protocolo HTTP inicia con establecer una conexión con el cliente y el servidor. Posteriormente, el cliente envía una petición al servidor, generalmente para obtener un recurso específico. El servidor escucha dicha solicitud, la procesa y, si todo transcurre correctamente, responde con el recurso solicitado al cliente. Finalmente, se cierra la conexión.

Figura 4.
Funcionamiento del protocolo básico del protocolo HTTP



Nota. *Esta figura representa el paso a paso del funcionamiento básico del protocolo HTTP. Fuente: Implementación del Protocolo HTTP Paralelizado en Cliente y Servidor (p. 11), por Roberto Eduardo & Ariel (1999).*

La figura 5 detalla la estructura fundamental de una URI para acceder a recursos a través del protocolo HTTP. La sección del esquema especifica el uso del protocolo HTTP, incluyendo su versión segura en este caso. Luego, la sección de autoridad identifica el nombre o un identificador único asociado al servidor o instancia que aloja el recurso, comúnmente mediante el uso de DNS o una dirección de red IPv4 o IPv6. Por último, la sección de la ruta indica las carpetas o archivos jerárquicamente organizados que se desean acceder.

Figura 5.
Partes de una URI

Nota. *Una URI es una secuencia de caracteres compacta que se encarga de identificar los recursos en HTTP. Fuente: El protocolo HTTP (p. 6), por WEB (2013).*

En la figura 6 se presentan dos ejemplos que siguen la estructura de una URI conforme a la figura 5, con la particularidad de que, después de la sección de la ruta, la URI se extiende aún más. En el primer caso, la URI /view.php?id=91019 indica que se accederá al archivo view.php y se recibirá el valor de una variable llamada id con el valor 91019. En consecuencia, el comportamiento del código PHP en el archivo se determinará en función de ese valor.

Del mismo modo, en la siguiente URI, la carpeta search recibirá los valores de dos variables, q y tbm, con los valores de Madrid e isch respectivamente.

Figura 6.
URIs en HTTP con consultas especificas.

```
https://aulaglobal.uc3m.es/course/view.php?id=91019


https://www.google.com/search?q=madrid&tbm=isch
```

Nota. *Estos ejemplos buscan que existen diversas maneras de acceder a recursos mediante el uso del protocolo HTTP. Fuente: El protocolo HTTP (p. 8), por WEB (2013).*

MDN (2023) explica de manera detallada el funcionamiento básico de internet, destacando la interacción entre los navegadores y los servidores web para mostrar las páginas que visitamos. Al cliquear un enlace, el navegador genera una solicitud al servidor web que alberga la página deseada. Posteriormente, el servidor responde con la información de la página, la cual es interpretada por el navegador y mostrada en la pantalla del usuario.

Además, se aborda el tema de la autenticación, que consiste en verificar la identidad del usuario al acceder a un sitio web protegido. Los sitios web emplean diversos métodos de autenticación, como nombres de usuario y contraseñas, o el uso de cookies HTTP.

El protocolo HTTP facilita el intercambio de datos mediante un sistema de solicitud y respuesta, donde los clientes, como navegadores o webcrawlers, solicitan información a los servidores. Sus inicios van desde 1991 hasta la actual HTTP/2, destacando su relevancia en la comunicación eficaz en Internet. Además, la importancia del uso de las cookies radica en que almacenan datos de visitas anteriores dando versatilidad al protocolo para integrar nuevas funcionalidades. (Equipo editorial, Etecé, 2023).

Según KeepCoding Team (2023), HTTP se destaca como el lenguaje universal que facilita la comunicación entre navegadores web y servidores en Internet. Se define que el protocolo HTTP posibilita la transferencia de diversos tipos de datos, que van desde texto hasta archivos multimedia, entre clientes y servidores. Además, se enfatiza la importancia de comprender los métodos de HTTP para intercambiar información de manera efectiva, así como la relevancia de los códigos de respuesta que resultan fundamentales para interpretar las interacciones en la web.

Introducción al HTML

En la búsqueda de un lenguaje universal para la publicación y distribución global de información comprensible por diversos dispositivos, ya sean

móviles o computadoras, surge HTML (HyperText Markup Language). Este lenguaje, de acuerdo con Casado (2019), se caracteriza por su sintaxis sencilla y clara que permite la descripción de hipertexto, incluyendo gráficos, audios e imágenes. HTML se emplea en la creación y estructuración de páginas web, destacando su capacidad para crear hipervínculos, esenciales para la navegación web, como señala Prescott (2015).

Un archivo HTML, como menciona Prescott (2015), es esencialmente un archivo de texto con la opción de ser creado o alterado mediante un editor de texto. Se basa en utilizar etiquetas jerárquicamente estructuradas que representan los elementos de una página web. Su origen se remonta a 1980, cuando Tim Berners-Lee propone una manera para distribuir documentos, y el primer documento HTML, titulado "HTML Tags", se publicó en 1991, según Luna (2019). A continuación, algunos ejemplos del uso de HTML:

Ejemplo 1, Encabezado y párrafo:
En este primer ejemplo, se muestra un documento HTML básico que contiene un encabezado <h1> con el texto "¡Hola Mundo!" y un párrafo <p> con el texto "Este es un ejemplo sencillo de HTML". El encabezado define el título para la pestañita que se visualiza en el navegador, mientras que el párrafo proporciona texto visible en el cuerpo del HTML.

```
<!DOCTYPE html>
    <html>
        <head>
        <title>Esto hace parte de un título </title>
        </head>
     <body>
        <h1>Esto hace parte del texto que puede ubicar aquí -ejemplo<h1>
        <p>Esto hace parte de un párrafo </p>
     </body>
    </html>
```

Ejemplo 2, Lista desordenada:
En este segundo ejemplo, se presenta una lista desordenada <ul> de frutas que incluye elementos de lista <li> como Manzana, Plátano y Naranja. La etiqueta <h2> se utiliza para el subtítulo "Frutas". Las listas desordenadas son útiles para mostrar elementos sin un orden específico.

```
<!DOCTYPE html>
    <html>
      <head>
        <title>ejercicio 2</title>
      </head>
      <body>
          <h2>Frutas</h2>
```

```
            <ul>
                <li>Manzana</li>
                <li>Plátano</li>
                <li>Naranja</li>
            </ul>
        </body>
    </html>
```

Ejemplo 3. Uso de enlaces:
En este tercer ejemplo, se presenta un enlace <a> que redirige a un sitio web externo cuando se hace clic en él. El texto "este enlace" se muestra como un hipervínculo y al hacer clic en él, el usuario será llevado al sitio web especificado en el atributo href. Los enlaces son fundamentales para la navegación web y la conexión entre diferentes páginas.

```
<!DOCTYPE html>
    <html>
        <head>
            <title>Ejemplo tres</title>
        </head>
        <body>
            <h3>Enlace Externo</h3>
            <p>Visita <a href="https://www.ejemplo.com">clic aquí</a>.</p>
        </body>
    </html>
```

Explicación de las etiquetas HTML usadas para los tres ejemplos:

- <!DOCTYPE html>: Indica que variante de HTML se usara.
- <html>: Indica el origen del HTML.
- <head>: Almacena los metadatos como el título de la página.
- <title>: Es el título que se puede ver en la pestañita del navegador usado.
- <body>: Contiene lo que se puede ver.
- <h1>: Crea un encabezado de nivel 1.
- <p>: Define un párrafo de texto.
- <h2>: Crea un encabezado de nivel 2.
- : Inicia una lista desordenada.
- : Define un elemento de lista dentro de una lista desordenada.
- <h3>: Crea un encabezado de nivel 3.
- <a>: Define un enlace o hipervínculo.
- href: Atributo del enlace que especifica la URL de la redirección.
- Texto entre las etiquetas <a> y : El texto visible que se convierte en el enlace clickeable.

Cabe recalcar que el uso y estructura de las etiquetas para la creación de una página va de acuerdo con la imaginación y necesidades del programador, claramente, respetando la jerarquía de cada una de ellas y aplicando una

adecuada semántica. A continuación, se ofrece un ejemplo más elaborado y con una aplicación práctica significativa: se trata de una página de inicio diseñada para una aplicación dedicada al fútbol. Esta muestra se caracteriza por su simplicidad y facilidad de comprensión, lo que la hace accesible incluso para aquellos que están dando sus primeros pasos en el diseño web. Por el momento, haremos a un lado las referencias a CSS y JavaScript, ya que estos temas, aunque cruciales para el desarrollo web completo, no son el foco principal de nuestra discusión actual. Sin embargo, cabe destacar que profundizaremos en estos aspectos más adelante, ofreciendo una visión completa de cómo estos elementos interactúan para crear experiencias web ricas y dinámicas.

Ejemplo 4. HTML para una página de inicio de futbol.

<!DOCTYPE html>

Esta línea es una declaración y no una etiqueta HTML. Le dice al navegador qué versión de HTML está siendo utilizada. En este caso, <!DOCTYPE html> especifica que está usando HTML5, La versión más reciente. Esta declaración es necesaria para el momento de navegar muestre correctamente la página.

<html lang="es">

Esta etiqueta es el origen de cualquier documento HTML. Toda la estructura va dentro de esta etiqueta.

El atributo lang="es" indica que el idioma principal del contenido de la página es el español. Esto es importante para la accesibilidad y para los motores de búsqueda.

<head>
Dentro de la etiqueta <head> se encuentra la información del documento que no se visualiza en el navegador. Esta sección abarca metadatos, enlaces a estilos, scripts y otras configuraciones iniciales del documento.

<meta charset="UTF-8">
Esta etiqueta especifica que el documento codifica los caracteres mediante UTF-8, el cual hace uso de casi todos los caracteres de todos los idiomas. Esto es importante para que los caracteres especiales se muestren correctamente.

<meta name="viewport" content="width=device-width, initial-scale=1.0">
Esencial para hacer que la página web sea responsiva. Le dice al navegador cómo adaptar el contenido de la página a diferentes tamaños de pantalla, asegurando que se vea bien en dispositivos móviles.

<link rel="stylesheet" href="estilos.css">
Esta línea enlaza una hoja de estilo externa, llamada estilos.css, con el documento HTML. La hoja de estilo contiene reglas CSS que aportan lo bonito a la página.

<title>Página de Inicio de Sesión</title>
Lo que este dentro de esta etiqueta se muestra en la pestaña del navegador y es el título de la página web.

<body>
Todo el contenido el cual se visualizará directamente en el navegador se pondrá en la etiqueta <body>. Esto incluye texto, imágenes, botones, etc.

** **
 es una etiqueta utilizada para agrupar bloques de contenido. El atributo class con el valor "container" indica que esta sección del contenido puede ser referenciada o estilizada específicamente usando CSS.

<img src="imagenes/logo.png" alt="Logo de Fútbol">
Esta etiqueta muestra una imagen en la página. “src” indica el camino hacia la imagen, mientras que el atributo alt ofrece una descripción textual alternativa de la imagen, crucial para la accesibilidad y para situaciones donde la imagen no logre cargarse.

<h1>, <p>, <a>, <button>
<h1> se usa para el encabezado principal de la página. Crucial en la estructura y SEO (mayor rendimiento de los motores de búsqueda).

<p> envuelve el texto de un párrafo.
<a> define un hipervínculo como se mencionaba en ejemplos anteriores, que en este caso envuelve un <button>, haciendo que el botón sea cliqueable y redirija a otra página especificada en el atributo href.

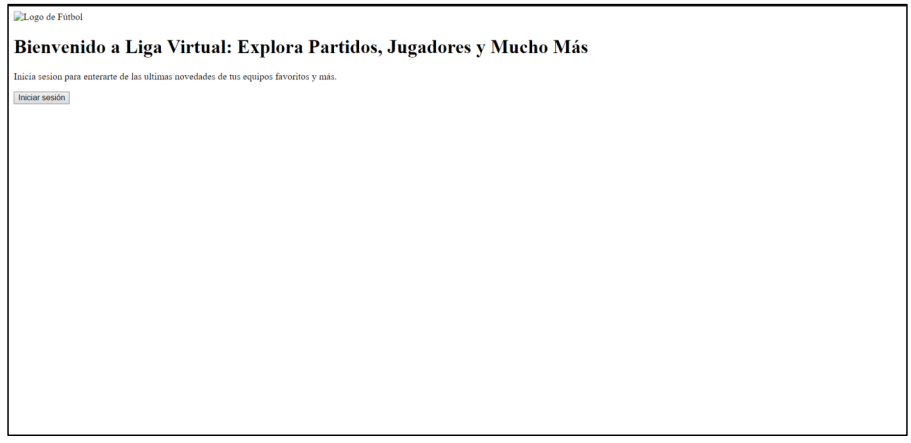
<button id="loginBtn"> define un botón que los usuarios pueden presionar. El atributo id permite identificar de manera única este botón en la página, útil para manipulación con CSS o JavaScript.

Código:

```
<!DOCTYPE html>
        <html lang="es">
                <head>
                        <meta charset="UTF-8">
                         <meta name="viewport" content="width=device-
                        width, initial-scale=1.0">
                         <link rel="stylesheet" href="estilos.css">
                         <title>Página de Inicio de Sesión</title>
                </head>
                <body>
                        <div class="container">
                             <img src="imagenes/logo.png" alt="Logo de
                        Fútbol">
                           <h1>Bienvenido a Liga Virtual: Explora Partidos,
                        Jugadores y Mucho Más</h1>
                             <p>Inicia sesion para enterarte de las últimas
                        novedades de tus equipos favoritos y más.</p>
                              <a href="../Login/indexLogin.html"><button
                        id="loginBtn">Iniciar sesión</button></a>
                 </div>
                </body>
</html>
```

En la Figura 7, se ilustra cómo se visualiza la ejecución del código HTML del ejemplo previo, prescindiendo del archivo CSS que dotaría de estilo a la página. El propósito es emplear este mismo código en etapas subsiguientes para incorporar los conceptos que se abordarán próximamente, observando la evolución de la página a medida que se integran nuevas herramientas. Por ahora, el foco se centra exclusivamente en demostrar la funcionalidad de las etiquetas HTML y cómo estas definen la estructura básica de la página web.

Figura 7.
Ejemplo de ejecución de código HTML.



Nota. *Interfaz generada en código HTML como ejemplo. Fuente: Elaboración propia*

A continuación, se presentan algunos sitios web donde podrá encontrar información de HTML, su funcionamiento e información sobre las etiquetas que usa:

Según MDN (2023), la Mozilla Developer Network ofrece una guía exhaustiva sobre HTML (Lenguaje de Marcado de Hipertexto), cubriendo desde los principios básicos hasta técnicas más sofisticadas. La guía enfatiza la relevancia de la etiqueta <head> para declarar recursos esenciales al comienzo de la carga de la página, lo cual es crucial para optimizar el rendimiento evitando retrasos en el procesamiento inicial. La aplicación práctica de HTML en la estructura de páginas web también se explora detalladamente, abarcando desde la incorporación de contenido multimedia hasta la elaboración de tablas.

El documento profundiza en el uso de HTML para vincular distintas páginas web mediante hipervínculos y para organizar el texto de manera efectiva. Se destacan elementos fundamentales como <html>, <head>, y <meta>, proporcionando además perspectivas sobre la evolución histórica del lenguaje HTML. En esencia, MDN (2023) presenta una guía integral para la comprensión y aplicación de HTML en proyectos de desarrollo web, convirtiéndose en un recurso invaluable tanto para novatos como para desarrolladores con experiencia (MDN, 2023, 24 de julio). Alvarez (2001), en su publicación en DesarrolloWeb.com, ofrece una visión comprensiva del HTML, subrayando su papel crucial para desarrollar páginas web. De acuerdo con Alvarez, el HTML se caracteriza por ser un lenguaje de marcado compuesto por una serie de etiquetas que especifican y organizan lo que va a contener una página web, abarcando desde textos e imágenes hasta listas y videos. Se señala que, a pesar de que HTML fue originalmente concebido para compartir información principalmente textual y gráfica, su uso se ha

expandido significativamente para incluir entretenimiento y funciones de consulta multimedia, mostrando la adaptabilidad y amplitud de aplicaciones del lenguaje.

La evolución de HTML ha llevado a la introducción de varios estándares, incluidos HTML 4.01 y el inminente HTML5, destacando la evolución constante del lenguaje para satisfacer las demandas de una web en constante cambio. A pesar de estos avances, Alvarez enfatiza la simplicidad y accesibilidad del aprendizaje de HTML, afirmando que permite a individuos sin experiencia previa en programación, adentrarse a desarrollar páginas web con relativa facilidad.

Se destaca también que el HTML se escribe usualmente en archivos de texto con extensión .html o .htm, y Alvarez recomienda la práctica de escribir etiquetas en minúsculas para mantener la consistencia y legibilidad del código. Finalmente, se menciona que DesarrolloWeb.com proporciona manuales detallados para aquellos interesados en profundizar su conocimiento de HTML, ofreciendo recursos valiosos para aprendices de todos los niveles (Alvarez, 2001).

Varangouli (2023) ofrece en SEMrush una valiosa compilación de etiquetas HTML cruciales y sus aplicaciones en la organización y diseño del contenido web. Según Varangouli, estas etiquetas juegan un papel esencial no solo en la aplicación de estilos, como por ejemplo el uso de cursivas, sino también en la creación de listas, inserción de contenido multimedia y definición de varios tipos de contenido en la página web. Es más, ellas son las responsables de estructurar y formatear el texto, permitiendo la definición clara de elementos tales como encabezados, párrafos, imágenes, enlaces, y formularios.

Varangouli subraya la importancia de entender cada etiqueta HTML para una implementación efectiva del código y alienta el uso de la guía publicada en SEMrush como un recurso de referencia para aprender acerca de las etiquetas estándar y su correcta aplicación. Resumiendo, la página resalta el rol indispensable de las etiquetas HTML en el desarrollo de sitios web que no solo estén bien estructurados, sino que también sean visualmente atractivos, ofreciendo así una guía esencial para su apropiado uso en el ámbito del desarrollo web (Varangouli, 2023).

CSS

Desarrollar páginas web implica la practica inicial de HTML para estructurar el contenido, definiendo funciones para cada elemento y otorgando significado a los documentos y secciones. Sin embargo, es a través de CSS (Cascade Style Sheet) que se logra modificar los estilos predeterminados de las etiquetas HTML, permitiendo darle presentación al contenido. Esta analogía visualiza al HTML como las vigas y columnas de un edificio, mientras que el CSS representa toda la presentación de este (Tinoco & Solís,

2014). CSS, como lenguaje de estilo, brinda a los diseñadores un control preciso sobre el diseño de los componentes de una página web, separando el contenido de su formato para lograr una presentación coherente y semánticamente significativa (Cabrera, 2013). Con la evolución de CSS y la introducción de CSS3, se han incorporado nuevas propiedades que permiten personalizar el estilo de los elementos de manera más avanzada, simplificando la creación de diseños sofisticados y mejorando la experiencia de diseño web en general (Álvarez, et al., 2017). Algunos ejemplos de CSS se presentan a continuación:

Ejemplo 5: Variar color para el fondo y texto

CSS:

```
body {
   background-color: lightblue;
   color: navy;
}
```

Explicación:

- **background-color: lightblue;** Esta línea alterna el color para el fondo del cuerpo entero usado en la página web a un azul claro (lightblue). El background-color corresponde a una capacidad de CSS la cual especifica el color del fondo de algún componente.
- **color: navy;** Esta línea indica que color tendrá el texto dentro del cuerpo de la página a color azul marino (navy). La propiedad color se utiliza en CSS para definir el color del texto de un elemento.

Ejemplo 6: Ajustar el Tamaño y Estilo de la Fuente de un Párrafo

CSS:

```
p {
   font-size: 16px;
   font-style: italic;
}
```

Explicación:

- **font-size: 16px;** esta propiedad, se redimensiona la medida del texto para todos los párrafos (<p>) a 16 píxeles. La font-size controla el tamaño de la fuente utilizada.
- **font-style: italic;** Esta línea aplica un estilo cursivo al texto de todos los párrafos. La propiedad font-style se usa para definir si el texto se muestra en estilo normal, cursiva o con inclinación oblicua.

Ejemplo 7: Aplicar Bordes y Espaciado Interno a un Elemento Div

CSS:
```
div {
    border: 2px solid black;
    padding: 20px;
}
```

Explicación:
- **border: 2px solid black**; Esta instrucción añade un borde sólido de color negro y 2 píxeles de grosor alrededor de los elementos . La propiedad border es una forma corta de establecer el anchor y color que tiene el contorno del elemento.
- **padding: 20px**; Establece un espaciado interno (padding) de 20 píxeles en todos los lados del contenido dentro del . El padding incrementa la medida entre el bordecito del componente y lo que lleva dentro, haciendo que este último no esté pegado directamente al borde.

Estos ejemplos básicos ilustran cómo CSS permite modificar lo visual de lo que conforma el HTML en una página web, desde ajustes sencillos de color y tipografía hasta la implementación de bordes y espaciados para estructurar y embellecer el diseño. A continuación, aplicaremos los estilos CSS al HTML del **ejemplo 4** visto en el capítulo de HTML:

Ejemplo 8: CSS para HTML de página de fútbol:

HTML: se encuentra el **ejemplo 4**

CSS:
```
body {
    background-image: url('imagenes/fondo.jpg');
    background-size: cover;
    background-repeat: no-repeat;
    background-position: center center;
    font-family: Arial, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 101vh;
  }
```

```
body, h1, p, button {
    margin: 0;
    padding: 0;
  }
    .container {
    text-align: center;
    background-color: transparent;
    backdrop-filter: blur(5px);
    padding: 2rem;
    border-radius: 11px;
    box-shadow: 0 2px 4px rgba(0, 0.1, 0, 0.1);
    border: 3px solid #B0CFF5;
  }

  img {
    max-width: 100px;
    height: auto;
    margin-bottom: 1rem;
  }

  h1 {
    font-size: 1.5rem;
    margin-bottom: 0.5rem;
  }

  p {
    font-size: 1rem;
    color: #2e2222;
    margin-bottom: 1rem;
  }

  button {
    padding: 0.5rem 1rem;
    font-size: 1rem;
    background-color: #007bff;
    color: white;
    border: none;
    border-radius: 6px;
    cursor: pointer;
    transition: background-color 0.4s;
  }

  button:hover {
    background-color: #0056b3;
  }
```

Explicación:
Sección 1: Estilo del body

- **background-image: url('imagenes/fondo.jpg'):** Configura una imagen de fondo que abarque todo el contenido del sitio web. La ruta 'imagenes/fondo.jpg' indica la ubicación de la imagen a usar.
- **background-size: cover:** Ajusta el jpg usado para el fondo para que abarque el área disponible, conservando sus proporciones.
- **background-repeat: no-repeat:** Se encarga que el jgp no muestre varias veces.
- **background-position: center center:** Centra la imagen de fondo tanto horizontal como verticalmente.
- **font-family: Arial, sans-serif:** Define la tipografía para el texto del cuerpo de la página, utilizando Arial como primera opción y, si no está disponible, cualquier fuente sans-serif.
- **display: flex:** Aplica Flexbox al cuerpo, un modelo de diseño que permite a los hijos del contenedor acomodarse de manera más eficiente.
- **justify-content: center:** Alinea los elementos hijos horizontalmente al centro del contenedor body.
- **align-items: center:** Alinea los elementos hijos verticalmente al centro del contenedor body.
- **height: 100vh:** Establece la altura del body al 100% de la altura de la ventana del navegador (viewport height).

Sección 2: Reinicio de Margen y Padding
- Este estilo remueve los márgenes (margin) y el relleno interno (padding) por defecto del navegador para los elementos body, h1, p y button, proporcionando un punto de partida uniforme para el diseño.

Sección 3: Estilo del Contenedor .container
- **text-align: center:** Centra el texto dentro del contenedor.
- **background-color: transparent:** Establece un fondo transparente para el contenedor.
- **backdrop-filter: blur(5px):** Aplica un efecto de desenfoque al área detrás del contenedor.
- **padding: 2rem:** Añade relleno interno alrededor del contenido del contenedor para separarlo de los bordes.
- **border-radius: 10px:** Redondea las esquinas del contenedor.
- **box-shadow: 0 1px 3px rgba(0, 0.1, 0, 0.2):** Añade una sombra ligera alrededor del contenedor para darle profundidad.
- **border: 2px solid #B0CFF5:** Añade un borde sólido con color claro alrededor del contenedor.

Sección 4: Estilos para img, h1, p, y button
- **Imágenes (img):** Se define un ancho máximo y se asegura que la altura se ajuste automáticamente para mantener las proporciones. Además, se añade

un margen inferior.

- Encabezados (h1): Se establece un tamaño de fuente y un margen inferior.
- Párrafos (p): Se especifica un tamaño de fuente, un color y un margen inferior.
- Botones (button): Se personalizan con relleno, tamaño de fuente, color de fondo y de texto, y se elimina el borde. El cursor: pointer; indica que el cursor del ratón se convertirá en un puntero al pasar sobre el botón, y la propiedad transition suaviza el cambio de color de fondo al interactuar con el botón.

Sección 5: Interacción con el Botón

- **button:hover:** Cambia el color de fondo del botón cuando el usuario pasa el cursor sobre él, mejorando la interactividad visual.

Cada uno de estos estilos contribuye a crear una página web atractiva, interactiva y accesible para los usuarios, demostrando cómo CSS puede transformar completamente la apariencia de los elementos HTML.

En la figura 8, podemos visualizar cómo se aplican el estilo CSS para el **ejemplo 4** de HTML, dotando de buena presentación a la página y ambientando al usuario a la temática de esta.

Figura 8.
Ejemplo de CSS para HTML.

Nota. *Elaboración de una interfaz con una hoja de estilo. Fuente: elaboración propia.*

Para finalizar, se presentan algunos sitios web que pueden ayudarlo a fortalecer y reforzar lo visto en esta unidad:

El sitio web CSS | MDN ofrece un compendio exhaustivo sobre CSS (Cascading Style Sheets), es el lenguaje clave que da la parte estética de los sitios web. Este recurso detalla el uso de CSS para modificar aspectos como la tipografía, color, dimensiones y espaciado del contenido, así como para estructurarlo en varias columnas, incorporar animaciones y aplicar diversas propiedades estéticas. CSS desempeña un papel crucial en la gestión visual de documentos HTML o XML, determinando como el contenido se visualiza en diferentes soportes, ya sea en pantalla, impreso, mediante voz u otros medios. Se subraya igualmente que CSS constituye una tecnología fundamental de la Open Web, cuyas especificaciones han sido estandarizadas por el W3C. Desde su primera versión, CSS1, hasta la actualidad con CSS3, el ámbito de aplicación de las especificaciones ha crecido, cubriendo diversos módulos y capacidades. Aunque oficialmente no se ha presentado una "CSS4", el desarrollo constante ha propiciado avances significativos en el diseño y estilo de contenido web. El portal también introduce los fundamentos de CSS, ofreciendo un acercamiento amigable a sus principios básicos, incluyendo cómo funciona, su sintaxis y los primeros pasos para emplearlo en el estilizado de HTML (CSS | MDN, 2024).

El sitio web de HackaBoss presenta un análisis exhaustivo sobre CSS (Cascading Style Sheets) y su papel indispensable en el diseño y desarrollo web. Explica que CSS es el lenguaje encargado de definir la presentación visual de documentos estructurados, principalmente HTML, facilitando la creación de diseños atractivos para sitios web e interfaces de usuario mediante la aplicación de estilos a los elementos web. El término "Cascading Style Sheets" se desglosa de la siguiente manera: "Cascading" hace referencia a la capacidad de los estilos de aplicarse en cascada, permitiendo que ciertas reglas prevalezcan sobre otras; "Style" se refiere a la aplicación de estilos visuales; y "Sheets" señala que estos estilos se organizan en archivos .css, separados del código HTML. Subraya la relevancia de CSS3, señalándolo como un hito significativo en la evolución de este lenguaje de estilos. Además, aclara un punto crucial: ni HTML ni CSS son considerados lenguajes de programación propiamente dichos, destacando la necesidad de JavaScript para implementar funcionalidades más complejas en los sitios web. Se detalla la estructura de CSS, incluyendo la manipulación de propiedades tales como el color, tamaño y margen, así como la capacidad de personalizar aspectos como el ancho, estilo y color de bordes de los elementos (Souto, 2023).

El sitio web de Rock Content se sumerge en la relevancia de CSS (Cascading Style Sheets) para el diseño web, ilustrando la sinergia esencial entre HTML y CSS. Mientras que HTML sienta las bases estructurales de una página web, CSS se encarga de embellecer esta estructura con estilos y elementos visuales. Se subraya la idea de que, aunque técnicamente posible operar sin CSS, su

empleo se torna crucial para diferenciar y realzar un sitio web frente a sus competidores. En esencia, este artículo subraya el papel vital de CSS en la elaboración de páginas web no solo atractivas sino también impactantes visualmente, trabajando de la mano con HTML para cultivar una experiencia estética y gratificante para el usuario (Content, 2021).

JS

Uno de los lenguajes de programación usados para el desarrollo de sitios web interactivos es JavaScript. Permite crear efectos visuales como animaciones, texto que se muestra u oculta, interacciones mediante botones, y alertas para los usuarios. Como lenguaje interpretado, JavaScript no requiere compilación previa, permitiendo que sus programas se ejecuten directamente en cualquier navegador web sin pasos adicionales. JavaScript no tiene una conexión directa con Java. Desde el punto de vista legal, el término "JavaScript" es una marca comercial registrada por Sun Microsystems, información disponible en su sitio web oficial (Pérez, 2019).

Cuando un navegador web carga una página, no trabaja directamente sobre el código fuente tal como lo escribió el desarrollador. En su lugar, transforma ese código en una estructura organizada que le permite manipular, mostrar e interactuar con el contenido de manera flexible. Esa estructura es lo que se conoce como DOM, sigla de Document Object Model, o Modelo de Objetos del Documento. En esencia, el DOM es una representación en forma de árbol de todos los elementos que componen una página web. Cada nodo de ese árbol corresponde a una parte del documento: puede ser una etiqueta HTML, un fragmento de texto, un atributo, o incluso un comentario. Desde el punto de vista del navegador y de los lenguajes de programación, el DOM convierte la página en un conjunto de objetos con propiedades y métodos que pueden ser consultados o modificados.

JavaScript más allá de simplemente añadir animaciones y efectos visuales, es la clave para mejorar la experiencia del usuario en la web, permitiendo desde la validación de formularios hasta la creación de juegos y aplicaciones web sofisticadas. Mientras HTML5 define la estructura y CSS3 el estilo, JavaScript aporta la vitalidad. Este lenguaje versátil posibilita diversas acciones como la manipulación dinámica del DOM (la arquitectura y estructura del HTML), el control de eventos como clics y pulsaciones, la comunicación fluida con servidores sin recargar la página, y el acceso a una amplia gama de APIs web para enriquecer la funcionalidad de las aplicaciones (Gauchat, 2012).

Según Maza (2012), la estructura y flexibilidad de la sintaxis de JavaScript son fundamentales en la construcción aplicativos webs interactivos, permitiendo una amplia gama de funcionalidades mientras se mantiene una codificación intuitiva para los programadores. Este lenguaje de programación permite a los desarrolladores implementar funciones avanzadas y manipular

elementos de la web de manera intuitiva, apoyándose en una serie de reglas claras que rigen la forma en que se deben organizar las instrucciones para que el navegador pueda interpretarlas y ejecutarlas adecuadamente. En la tabla 4, se puede visualizar algunas de las características principales de este lenguaje, desde sus variables hasta funciones y métodos:

Table 4.
Características de JavaScript.

Variables	***Etiquetas que se refieren a un valor cambiante***
Operadores	*Pueden usarse para calcular o comparar valores. Ejemplo: pueden sumarse dos valores, pueden comparase dos valores…*
Expresiones	*Cualquier combinación de variables, operadores y declaraciones que evalúan a algún resultado.* *Ejemplo int Total=100; int Total >100;*
Sentencias	*Una sentencia puede incluir cualquier elemento de la gramática de JavaScript. Las sentencias de JavaScript pueden tomar la forma de condicional, bucle, o manipulaciones del objeto. La forma correcta para separarlas es por punto y coma, esto sólo es obligatorio si las declaraciones múltiples residen en la misma línea. Aunque es recomendable que se acostumbre a terminar cada instrucción con un punto y coma, se ahorrará problemas.*
Objetos	*Estructura "contenedora" de variables, procedimientos y funciones, cada valor refleja una propiedad individual de ese objeto.*
Funciones y Métodos	*Una función es un conjunto de elementos que realizan alguna acción. Puede aceptar los valores entrantes (los parámetros que le asignaremos a la función), y puede devolver un valor saliente. Un método simplemente es una función contenida en un objeto.*

Nota. *Se describen las características del lenguaje JavaScript.*
Fuente: JavaScript (p. 13), por Maza (2012).

Los lenguajes de programación representan una amplia gama de enfoques y necesidades en la comunicación entre humanos y máquinas. Cada lenguaje, como JavaScript, posee características y capacidades únicas que determinan su adecuación para distintos propósitos. A pesar de las diferencias en los objetivos de comunicación entre humanos y computadoras, ambos sistemas se basan en la organización de ideas y la transmisión de instrucciones comprensibles. En el campo de la programación, esta comunicación se orienta hacia la interacción efectiva con sistemas, máquinas y protocolos. Aunque existen similitudes estructurales entre los lenguajes de programación y los idiomas humanos, el foco principal recae en la lógica, la abstracción y la capacidad de crear soluciones lógicas para problemas específicos mediante instrucciones entendibles por las máquinas. En el ámbito informático, la programación implica comprender las capacidades y limitaciones de las máquinas, optimizando el código para asegurar su funcionamiento en diversas plataformas. Este enfoque, familiar para quienes trabajan con HTML y CSS, resalta la importancia de integrar de manera precisa los componentes tecnológicos para lograr una comunicación efectiva y un funcionamiento coherente en entornos variados (Gascón González, 2017).

A continuación, algunos códigos de JavaScript para entender su funcionamiento:

Ejemplo 9: Manipulación del DOM (Document Object Model)

```
<!DOCTYPE html>
<html>
        <head>
                <title>Uso del lenguaje JavaScript</title>
</head>
<body>
                 <h1 id="titulo">¡Titulo!</h1>
                <button onclick="cambiarTexto()">Cambiar texto</button>

 <script>
          function cambiarTexto() {
          var elemento = document.getElementById('titulo');
          elemento.textContent = '¡Nuevo texto!';
  }
 </script>
</body>
</html>
```

Explicación:
En el ejemplo 9, tenemos un botón que al hacer clic ejecuta la función cambiarTexto().

- La función cambiarTexto() obtiene el elemento con el id 'titulo' utilizando document.getElementById('titulo').
- Luego, se cambia el contenido del elemento (el texto dentro del <h1>) utilizando elemento.textContent.

Ejemplo 10: Manejo de eventos

```
<!DOCTYPE html>
<html>
        <head>
                <title>Ejemplo para JS</title>
        </head>
        <body>
                 <button id="boton">cliquea aquí</button>

 <script>
  document.getElementById('boton').addEventListener('click', function() {
    alert('Has cliqueado un botón');
  });
     </script>
```

```
    </body>
</html>
```

Explicación:

- En este caso, usamos addEventListener para escuchar el evento de clic (click) en el botón con el id 'boton'.
- Cuando se hace clic en el botón, se ejecuta la función anónima que muestra una alerta con el mensaje '¡Has hecho clic en el botón!'.

Ejemplo 11: Variables y operaciones aritméticas

```
<!DOCTYPE html>
<html>
        <head>
                <title>Ejemplo de JavaScript</title>
   </head>
   <body>
          <p id="res"></p>
  <script>
     var variable_que_representa _el_n_1 = 10;
     var variable_que_representa _el_n_1 = 5;
     var s = variable_que_representa _el_n_1 + variable_que_representa
_el_n_2;
              var r = variable_que_representa _el_n_1 -
variable_que_representa _el_n_2;
    var m = variable_que_representa _el_n_1 * variable_que_representa
_el_n_2;
     var d = variable_que_representa _el_n_1 / variable_que_representa
_el_n_2;
    document.getElementById('res').textContent = 'Suma: ' + s + '<br>'
+
                                        'Resta: ' + r + '<br>' +
                                        'Multiplicación: ' + m + '<br>' +
                                        'División: ' + d;
   </script>
   </body>
</html>
```

Explicación:

- En este ejemplo, declaramos dos variables (numero1 y numero2) y realizamos operaciones aritméticas básicas con ellas (suma, resta, multiplicación, división).
- Luego, actualizamos el contenido de un párrafo (<p>) con el id 'resultado' para mostrar los resultados de las operaciones utilizando document.getElementById('resultado').textContent. Usamos la

concatenación de cadenas para mostrar los resultados de cada operación.

Evolución de las Aplicaciones Web

En octubre de 1972, ARPANET fue presentado públicamente, sentando las bases de lo que después sería conocido como Internet, tras la transformación de ARPA en DARPA y el uso del término "internetting". Para superar las limitaciones del protocolo NCP, Bob Kahn inició el desarrollo del protocolo TCP en 1972, con la colaboración de Vinton Cerf. Este protocolo, adoptado por el DoD en 1976 para ARPANET, evolucionó hasta dividirse en TCP e IP en 1978, estableciendo las bases del conjunto TCP/IP. El sistema DNS fue implementado en 1984 para facilitar la gestión de nombres de dominio, mientras que Steve Crocker y Jon Postel jugaron roles fundamentales en el desarrollo y gestión de los RFC, documentos clave para la comunicación de datos dentro del ámbito científico.

El termino World Wide Web se le atribuye a Tim Berners-Lee dentro de CERN en 1980, revolucionando como navegar en Internet con el desarrollo de HTTP y HTML a finales de 1990. La Web 1.0 (1991-2003) se caracterizó por páginas estáticas y una comunicación unidireccional, mientras que la Web 2.0, iniciada en 2004, marcó el inicio de una era más interactiva, con el surgimiento de redes sociales, blogs y wikis, favoreciendo la generación de material por parte de los usuarios. La Web 3.0, o Web Semántica, representa un avance hacia una Internet más conectada y accesible, impulsando la interoperabilidad entre sistemas y la utilización de inteligencia artificial para una administración optima de los datos en la red. Este nuevo paradigma busca mejorar la experiencia del usuario, permitiendo un acceso más flexible y personalizado a la información a través de diversas plataformas y dispositivos (R. Hernández Martínez, E. J. López Hernández, A. Hernández Tejeda, M. Osorio de la Cruz, J. O. Ramírez Santiago, y L. Martínez Agustín., 2020).

Arquitectura de las aplicaciones Web

La arquitectura de una aplicación web es la estructura de su construcción y está determinada por la relación entre el servidor y el cliente. Además, Hernández (2023) menciona que emplean lenguajes usados para la programación del el cliente y el servidor, sistemas de gestión que permiten la manipulación de las bases de datos, métodos de almacenamiento en memoria caché y protocolos de comunicación en red. De igual forma, su importancia radica en que muestra la lógica detrás de las respuestas a las consultas de los clientes y cómo los distintos elementos que constituyen una aplicación web se comunican entre sí (Harsh, 2022). Además, también define la fluidez, la capacidad de implementar nuevas actualizaciones, mantener segura la información y mantenibilidad del sistema (Juice Studio, 2022).

Juice Studio (2022) explica que la arquitectura de una aplicación web es su esqueleto, siendo este el modelo por el cual se produce la interacción entre sus componentes, incluyendo bases de datos, sistemas, servidores e interfaces.

De esta forma, en este proceso interviene las siguientes partes:

i) El lado del cliente(frontend) para la interacción con el usuario; ii) el servidor de base de datos, para el almacenamiento de datos utilizados en la aplicación; ii) el lado del servidor(back-end) que almacena la lógica de negocio, procesa las peticiones y envía las respuestas en base a las interacciones del usuario.

En este sentido, siguiendo el planteamiento de Juice Studio (2024) la definición de la arquitectura que se usara para el desarrollo de un producto web es amplia y depende de la forma en la cual se quiere que sea la interacción entre los lados del cliente y del servidor. En otras palabras, los elementos de esta pueden ajustarse o cambiarse de acuerdo con los requerimientos específicos del proyecto en el que se trabaje. Así mismo, Hernández (2023) nos indica que para el desarrollo de una aplicación web se utilizan distintas herramientas, lenguajes, servidores, almacenes de bases de datos, ya que, esta puede requerir inicios de sesión de usuario, seguimiento de la sesión, e incluyen elementos interactivos como formularios, entre otras funciones. Por lo tanto, Hernández (2023) añade que una buena arquitectura incluye los siguientes elementos:

i) Lenguajes de programación orientado al cliente (HTML/CSS, JavaScript;
ii) Lenguajes de programación para el servidor (PHP, ASP.NET, JSP);
ii) Sistemas para la administración de datos (MySQL, Oracle, Microsoft SQL Server);
iv) Procesos para almacenar caché (Redis, Memcached).
v) Protocolos para la red (HTTP, FTP, SMTP, SSH)

De igual forma este agrega que, su correcta definición ayuda a:

1. *Mejorar el rendimiento:* Estas pueden ejecutarse de forma óptima y rápida.
2. *Mayor seguridad:* Garantiza que todos los elementos sean seguros y se puedan defender de malhechores.
3. *Mayor escalabilidad:* Se pueden implementar posibles actualizaciones.
4. *La reutilización:* Otros desarrolladores pueden utilizar su código para otros proyectos.
5. *A su simplicidad:* Ayuda a que sea fácil de entender y utilizar.
6. *A su mantenimiento*: Debe facilitar a los codificadores actualizar el código fácilmente sin afectar el funcionamiento de la aplicación.
7. *A su eficacia:* Permite que los componentes se utilicen de forma eficiente.

Por su parte Harsh (2022) detalla algunos tipos de arquitectura de aplicaciones web:

1. **Arquitectura de una sola página:** En este tipo de arquitectura como lo dice su nombre se trata de una página para todo el aplicativo, en la cual el

usuario pueda tener toda información que necesita, proporcionándole una experiencia rápida y sin fisuras.

2. **Arquitectura de aplicaciones web progresivas:** usa la arquitectura de única página dotando de características offline al producto web.
3. **Arquitectura de renderizado del servidor:** en este, el frontend es renderizado en el servidor back-end luego de haber sido solicitadas.
4. **Arquitectura de Aplicaciones Prerrenderizadas:** este enfoque, el frontend del aplicativo se construye con anterioridad y es almacenado como archivos HTML, CSS y JS en el servidor. Una vez se hace una solicitud, ésta es obtenida directamente y es mostrada.
5. **Arquitectura orientada al servicio:** el aplicativo es dividido en servicios los cuales se pueden interpretar como una funcionalidad del negocio, los cuales se pueden comunicar entre sí.
6. **Arquitectura de microservicios:** En esta arquitectura sus componentes son aún más pequeños y limitados, es decir, todo microservicio tiene su propia codificación y su información tiene pequeños vínculos de otros microservicios.
7. **Arquitectura sin servidor:** se basa en desboronar el aplicativo a funciones que tiene que ejecutar. Dichas funciones son almacenadas en soportes FaaS (Function-as-a-Service), las cuales se ejecutan cuando son solicitadas .

Lenguajes de programación para el cliente

El término "lado del cliente" se refiere a todas las funciones y acciones que tienen lugar en una aplicación web en el navegador del usuario, interpretadas y respondidas por el servidor (Cloudfire, 2021). Esto incluye lo que el usuario ve y cualquier interacción que realice en su navegador (dar clic en un botón, desplazarse por una página, escribir en un campo de entrada o la solicitud de envió de datos). Por su lado, Alegsa (2023) ejemplifica que estas operaciones pueden ser necesarias cuando se requiere acceso a información específica del cliente o cuando el servidor no tiene la capacidad de realizar ciertas tareas para todos los clientes. Finalmente, al llevarse a cabo localmente, sin necesidad de comunicarse constantemente con el servidor, lo que agiliza la experiencia del usuario y reduce la carga en la red (Alegsa, 2023).

En primer lugar, Cloudfire (2021) hace referencia al termino lado del cliente como a todo lo que en una aplicación web se muestra o tiene lugar en el navegador del cliente. Esto incluye lo que ve el usuario, junto con cualquier acción que se lleve a cabo en el navegador del usuario, que luego son interpretadas y respondidas por el servidor. Tales como, dar clic en un botón, desplazarse por una página, escribir en un campo de entrada o la solicitud de envió de datos. Por lo tanto, según Alegsa (2023), se describe a un cliente como un programa en línea que funciona en la computadora del usuario y establece conexión con un servidor. Adicionando que, las operaciones pueden ser realizadas en el lado-cliente ya sea porque requieren acceso a información o funcionalidades que solo están disponibles en el cliente, el usuario necesita

ver o proveer las entradas, y finalmente, el servidor carece de la potencia necesaria para realizar las operaciones para todos los clientes a los que sirve. En este sentido, al realizarse en el lado-cliente las tareas son hechas sin enviar datos mediante la red destinado al servidor. Esto permite en un menor tiempo de respuesta, un uso de ancho de banda más eficiente y una disminución de los riesgos de seguridad. Por otro lado, los lenguajes de programación al lado del cliente de acuerdo con Hostinet(2015) son:

HTML es un lenguaje que usa etiquetas que le dicen al navegador dónde colocar cada elemento, y la forma que tendrán al ser colocados.

JavaScript es quien dota de vitalidad y funcionalidad a una página.

VBScript Este lenguaje se fundamenta en Visual Basic, aunque representa una versión más simplificada de este. Asimismo, está diseñado para programar scripts, pero su compatibilidad se limita exclusivamente a Internet Explorer.

CSS es el encargado de proporcionar la parte grafica de una página, es quien le da los estilos a la parte visual de la página.

Lenguajes de programación del lado del servidor

El lado del servidor en la programación web constituye el conjunto de procesos y operaciones los cuales ejecutara el servidor web para gestionar las solicitudes de los clientes (AppMaster, 2023). Para ello, gestiona lenguajes, marcos y tecnologías de programación tales como Perl, ASP, PHP, CG, entre otros, lo que le posibilita desde interactuar con bases de datos hasta ejecutar la lógica empresarial (La Torre, 2006). Además, de acuerdo con MDN (2021) este enfoque permite almacenar y distribuir información eficientemente, personalizar la experiencia del usuario, controlar el acceso al contenido, gestionar información de sesión, enviar notificaciones y realizar análisis de datos.

Appmaster (2023) explica que el término "del lado del servidor" se refiere “a los procesos, operaciones y funcionalidades que ocurren en el servidor. Aquellos componentes del servidor son los responsables de procesar las solicitudes de los clientes, interactuar con las bases de datos, ejecutar la lógica empresarial y, en última instancia, devolver la respuesta adecuada al lado del cliente”. Por tanto, su núcleo es utilizar lenguajes, marcos y tecnologías de programación del servidor, que permiten a los desarrolladores crear scripts, que son ejecutados en el servidor, con el propósito de presentar, cuando el cliente requiera, diferente información, generalmente extraída de un almacén

de datos. Siguiendo este planteamiento, MDN (2021) nos dice que la programación web del servidor en su mayoría se basa en seleccionar el contenido a retornar al usuario en consideración a sus solicitudes. Adicionalmente, se encarga de realizar tareas como la comprobación, almacenamiento, la recuperación de datos y la distribución de datos, la creación y gestión de API, la autenticación de usuario, él envió de notificaciones, entre otras.

De tal forma, que programación web de lado servidor permite según MDN (2021) lo siguiente:

1. **Almacén y entrega optima de información:** Este tipo de programación permite guardar los datos en una base de datos y entregar distintos tipos de ficheros (texto, PDF, imágenes.). De igual forma, la información puede ser compartida y actualizada.
2. **Experiencia de usuario personalizada:** la información almacenada de cada usuario le permite crear una experiencia personalizada en base a sus hábitos, intereses, y localización. Pudiendo ser utilizada para anticipar respuestas, dar recomendaciones y gestionar notificaciones futuras.
3. **Control de acceso:** La programación del servidor puede dar acceso al contenido dependiendo si el usuario cuenta o no con permisos.
4. **Almacenar información de sesión/estado:** el servidor puede almacenar datos sobre la actividad en el aplicativo que realiza el usuario. Esto permite, que se sepa si el usuario ha ingresado previamente, que información ha diligenciado o su historial de órdenes.
5. **Notificaciones y comunicación:** el servidor puede enviar notificaciones mediante el propio aplicativo o por correo electrónico, SMS, mensajería instantánea, u otros servicios de comunicación.
6. **Análisis de datos:** La programación del servidor se usa para personalizar las respuestas de acuerdo con el análisis de los datos recopilados.

A su vez, La Torre (2006) enlista los siguientes programas de programación al lado del servidor que se detallan a continuación:

CGI Es un sistema que sirve para programar páginas interactivas de servidor.

Perl: Se trata de un lenguaje de programación interpretado, lo que implica que su código de scripts no se compila previamente, sino que se lee y se ejecuta directamente interpretando las instrucciones contenidas en él cada vez que se desea correr.

ASP: Herramienta de Microsoft para el desarrollo de webs interactivas del servidor. Es escrita dentro de la misma página, empleando lenguajes como Visual Basic Script o Jscript.

php: Se trata de un lenguaje de programación para el servidor que es gratuito, funciona en diversas plataformas y es veloz, además de tener una amplia gama de funciones disponibles en su librería.

JSP: Esta tecnología se enfoca en desarrollar aplicaciones web mediante Java, lo que posibilita que los aplicativos webs sean compatibles con diversos servidores web en diferentes plataformas.

Metodologías para desarrollar aplicaciones Web.

Las metodologías son un sistema estructurado de procedimientos lógicos empleados para alcanzar un objetivo que demanda habilidades y conocimientos especializados. Se trata de una etapa específica dentro de un trabajo o proyecto que se sustenta en un fundamento teórico y conlleva a la selección de técnicas o métodos específicos para lograr los objetivos. La metodología abarca el conjunto de métodos utilizados en una actividad particular con el fin de formalizarla y mejorar su eficacia. Define los pasos a seguir y la forma de ejecutarlos para llevar a cabo una tarea de manera exitosa (Maida, 2015).

Las metodologías para el desarrollo de software surgieron a partir de los años 70 con el fin de resolver problemas de creación de software sin control adecuado, lo que resultaba en productos deficientes y clientes insatisfechos. Estas metodologías han mejorado las aplicaciones web al proporcionar guías, pasos y procesos eficientes para tener productos que cumplan con los requerimientos. Actualmente, se utilizan diversas metodologías dependiendo del sistema a desarrollar (escritorio, móvil, web), siendo las de desarrollo web las más destacadas debido a la creciente demanda de presencia en Internet. Las metodologías que más se usan tienen una amplia gama de estándares y componentes, centrándose en aspectos web, como OOHDM, que destaca por cumplir con múltiples criterios para la construcción óptima de aplicaciones web (Ríos et al., 2017).

Para el Instituto Tecnológico de Matehuala (2013), algunas de las metodologías usadas para el desarrollo de aplicaciones web son las siguientes:

La Metodología de Gestión de Relaciones RMM es un procedimiento destinado a la creación de aplicaciones hipermedia, como catálogos en línea o bases de datos. Este enfoque resulta especialmente adecuado para dominios con estructuras bien definidas y relaciones claras entre distintas clases de objetos. El modelo RMM propone un sistema de descripción de estos objetos, sus interconexiones y la forma de navegar a través de ellos en la aplicación.

Algunas características fundamentales de RMM abarcan:

- Enfoque centrado en la información para la concepción de sitios web.
- Utilización de un lenguaje para modelar la lógica de los sitios web,

abarcando información, navegación y presentación.
- Integración en una metodología de desarrollo más amplia.
- Facilitación en la creación de páginas web complejas con múltiples vistas y la reutilización de elementos.
- Mejora en la capacidad de generar enlaces más robustos y mantener la coherencia durante la navegación.

El enfoque UML-Based Web (UWE) Utiliza estrategias basadas en objetos en la descripción de aplicaciones hipermedia, poniendo énfasis en la evaluación de requerimientos, diseño conceptual, navegación y aspectos visuales. Para los elementos hipermedia se presentan mediante representaciones gráficas como diagramas de clases UML, donde los nodos representan clases, los enlaces estereotipados indican asociaciones, y la ayuda para navegar se muestra como clases estereotipadas. Además, se utilizan representaciones de procesos y graficas de estado para aspectos dinámicos, para que UML se emplea para la presentación y navegación. Los conceptos fundamentales de UWE engloban:

- Empleo de la notación estándar (UML).
- Establecimiento de métodos y pasos para el desarrollo de modelos.
- Limitaciones mediante OCL para una mayor precisión.
- El proceso de desarrollo en UWE se desglosa en cuatro etapas:
- Análisis de requisitos para la creación de modelos para casos de uso.
- Diseño abstracto para el modelo de clases de pertenecía.
- Diseño de navegación para los modelos de espacio y estructura de navegación.
- Diseño de presentación para representar las interfaces de usuario a través de modelos estándar de interacción UML.

El Método de Diseño Hipermedia Orientado a Objetos (OOHDM) Presenta un método de desarrollo en cuatro fases para la elaboración de aplicaciones hipermedia, que incluye las siguientes etapas: concepción del diseño, estructuración de la navegación, elaboración de interfaces abstractas y fase de implementación.

- En la fase de diseño abstracto, se comienza capturando los conceptos clave del área de aplicación a través de diagramas que ilustran el comportamiento, la organización y las interconexiones. La Programación Orientada a Objetos simplifica el proceso de pasar del diseño a la fase de desarrollo e implementación.
- El diseño navegacional se fundamenta en objetos conceptuales para generar objetos navegacionales que desempeñan roles como adaptadores y observadores, ampliando la funcionalidad navegacional de los nodos.
- En cuanto al diseño de interfaz abstracta, se enfoca en definir aspectos de la interfaz, separando la navegación del diseño de la interfaz para permitir múltiples interfaces para un mismo modelo. Esto se logra a través del

modelo de interfaz ADVs, que detalla la organización y el comportamiento de la interfaz.
- En la etapa de implementación, se considera el entorno de implementación y se establece la organización de la información, la interfaz y la funcionalidad de la aplicación, especialmente en un entorno web.

Actividades de Evaluación

A continuación, se presentan algunos ejercicios que ayudaran a evaluar los temas vistos en esta unidad:

Aspectos históricos

1. ¿Qué es una aplicación web estática?
   a) Una aplicación que utiliza tecnologías como HTML, CSS y JavaScript para ofrecer contenido dinámico.
   b) Una aplicación que muestra contenido fijo y no interactúa con el usuario.
   c) Una aplicación que requiere una conexión a Internet para funcionar.
   d) Una aplicación que utiliza herramientas para almacenar datos.

2. ¿Cuál de las siguientes tecnologías es esencial para el desarrollo de aplicaciones web dinámicas?
   a) HTML
   b) CSS
   c) JavaScript
   d) Todas las anteriores

3. ¿Qué es AJAX y qué permite hacer en una aplicación web?
   a) Es un lenguaje para programar interfaces.
   b) Es una técnica la cual permite recargar fragmentos de una página sin actualizarla totalmente.
   c) Es un servidor web utilizado para almacenar datos en la nube.
   d) Es un protocolo para proteger los aplicativos webs.

4. ¿Qué es una aplicación web responsiva?
   a) Una aplicación que emplea bases de datos para guardar datos.
   b) Una aplicación que se adapta automáticamente a diferentes dispositivos y tamaños de pantalla.
   c) Una aplicación que requiere una conexión a Internet para funcionar.
   d) Una aplicación que muestra contenido estático sin interacción del usuario.

5. ¿Qué es una PWA (Progressive Web App)?
   a) Una aplicación que utiliza lenguajes de programación modernos como Python o Ruby.
   b) Una aplicación que se instala en el dispositivo del usuario como una app

nativa, pero se ejecuta en el navegador.
c) Una aplicación que no requiere conexión a Internet para funcionar.
d) Una aplicación que solo funciona en dispositivos móviles.

6. ¿Qué es el desarrollo Full Stack en el contexto de las aplicaciones web?
a) Es desarrollar interfaces.
b) El desarrollo del back-end de la aplicación, incluyendo gestores de datos.
c) El desarrollo exclusivo de aplicaciones móviles.
d) Creación de páginas estáticas.

7. ¿Qué ventajas ofrece el uso de frameworks como React, Angular o Vue.js en el desarrollo de aplicaciones web?
a) Mayor control sobre el diseño visual de la aplicación.
b) Mejor rendimiento y optimización del código.
c) Facilidades para trabajar con bases de datos.
d) Menor seguridad en la aplicación.

8. ¿Qué es el concepto de "Single Page Application" (SPA) en el desarrollo web?
a) Una aplicación que consta de una sola página HTML que se actualiza dinámicamente.
b) Una aplicación que solo funciona en dispositivos móviles.
c) Una aplicación que no requiere conexión a Internet para funcionar.
d) Una aplicación que utiliza múltiples páginas HTML para mostrar contenido.

9. ¿Qué es WebAssembly (Wasm) y cuál es su función en el desarrollo web?
a) Es un lenguaje de programación para diseñar interfaces gráficas.
b) Es una tecnología para crear aplicaciones web responsivas.
c) Es un formato binario que permite ejecutar código de alto rendimiento en el navegador.
d) Es un protocolo de comunicación entre el servidor y el cliente en aplicaciones web.

10. ¿Cuál es una tendencia actual en la evolución de las aplicaciones web?
a) Mayor dependencia de tecnologías obsoletas.
b) Desarrollo exclusivo de aplicaciones web estáticas.
c) Mayor énfasis en la seguridad y la experiencia del usuario.
d) Menor importancia de la accesibilidad y la usabilidad.

Este cuestionario tiene como objetivo evaluar el conocimiento del participante en conceptos fundamentales como aplicaciones estáticas y dinámicas, tecnologías modernas como PWAs y frameworks como React, así como su comprensión del desarrollo Full Stack y las tendencias actuales en seguridad, rendimiento y experiencia del usuario. Este cuestionario busca medir la familiaridad y comprensión del participante en los aspectos clave del

desarrollo web, siendo relevante para quienes trabajan en el diseño, desarrollo o gestión de aplicaciones web.

HTML

1. ¿Qué significa HTML?
   a) Hyperlink Texto Markup Languages
   b) Hyper Text Markup Language
   c) Hit-level Texto Markup Languages
   d) Home Texto Markups Languages

2. ¿Cuál es la versión más reciente de HTML a partir de 2022?
   a) HTML5
   b) HTMLX
   c) HTML2022
   d) HTML+

3. ¿Cuál es la etiqueta utilizada para crear un enlace en HTML?
   a) <a>
   b) <link>
   c) <href>
   d) <url>

4. ¿Cuál es la función de la etiqueta <img> en HTML?
   a) Crear un enlace a otra página web
   b) Insertar una imagen en la página
   c) Establecer el título para una página
   d) Definir una tonalidad.

5. ¿Con que creo una lista desordenada en HTML?
   a) <oool>
   b)
   c) <lita>
   d) <list>

6. ¿Cuál es la etiqueta correcta para agregar un salto de línea en HTML?
   a) <nl>
   b)
   c) <lb>
   d) <newline>

7. ¿Qué etiqueta se utiliza para definir el encabezado de una página en HTML?
   a) <header>
   b) <title>
   c) <head>
   d) <h1>

8. ¿Qué función tiene la etiqueta <meta> en HTML?
   a) Insertar metadatos como el autor y la descripción de la página
   b) Definir un menú de navegación
   c) Crear un cuadro de texto
   d) Establecer el tamaño de la fuente

9. ¿Cómo implementar formularios para HTML?
   a) <form>
   b) <input>
   c) <submit>
   d) <button>

10. ¿Cómo establecer el color para el fondo en una página?
   a) Fondo-tonalidad
   b) page-color
   c) color-background
   d) background-color

11. ¿Cuál es la función de la etiqueta en HTML?
   a) Crear una lista de elementos
   b) Establecer un enlace interno en la página
   c) Agrupar y dividir contenido en bloques
   d) Mostrar un mensaje de alerta al usuario

12. ¿Qué etiqueta se utiliza para insertar un video en una página web en HTML5?
   a) <media>
   b) <video>
   c) <movie>
   d) <play>

13. ¿Cuál es la etiqueta que se utiliza para crear un hipervínculo interno en una página web en HTML?
   a) <link>
   b) <a>
   c) <href>
   d) <anchor>

14. ¿Qué función cumple la etiqueta en HTML?
   a) Establecer el idioma de la página
   b) Agrupar y aplicar estilos a elementos en línea
   c) Crear un bloque de texto
   d) Definir el título de la página

15. ¿Con que implemento una tabla en HTML?
   a) <tablón>

b) <trble>
c) <tad>
d) <table>

El objetivo de este cuestionario es evaluar en detalle el conocimiento sobre HTML y sus etiquetas, incluyendo temas como enlaces, imágenes, listas, encabezados, formularios, estructura de la página, entre otros aspectos fundamentales para el desarrollo web con HTML.

CSS

1. ¿Qué significa CSS?
   a) Color Style Sheets
   b) Cat Style Sheets
   c) Centralized Style Sheets
   d) Computerized Style Sheets

2. ¿Para qué sirve CSS en el desarrollo web?
   a) Establecer una estructura y que va a ir dentro del aplicativo web.
   b) Dar el diseño y estilo visual del aplicativo web.
   c) Controlar las funciones interactivas de una página web.
   d) Gestionar la comunicación con el servidor.

3. ¿Cuál es la propiedad que es empleado para modificar el color del fondo para algún elemento en CSS?
   a) colores
   b) background-color
   c) text-color
   d) fonte-color

4. ¿Cómo se selecciona un elemento HTML en CSS utilizando su id?
   a) #nombreId
   b) .nombreId
   c) $nombreId
   d) @nombreId

5. ¿Cuál es la función de la propiedad "display" en CSS?
   a) Cambiar el tipo de fuente de un elemento.
   b) Controlar si un elemento es visible o no en la página.
   c) Establecer el tamaño de un elemento.
   d) Definir la ubicación de un elemento en la página.

6. ¿Qué propiedad se utiliza para centrar un elemento horizontalmente en CSS?
   a) align-center
   b) justify-center
   c) text-align

d) margin

7. ¿Cuál es la forma correcta de aplicar un estilo a un grupo de elementos con la misma clase en CSS?
   a) #nombreClase { }
   b) .nombreClase { }
   c) $nombreClase { }
   d) @nombreClase { }

8. ¿Qué significa el concepto de "cascada" en CSS?
   a) Es la capacidad de aplicar estilos a varios elementos a la vez.
   b) Es la prioridad que tienen las propiedades CSS para determinar su efecto en un elemento.
   c) Es la jerarquía de estilos que se establece entre diferentes hojas de estilo.
   d) Es la estructura en árbol que sigue CSS para aplicar estilos a los elementos.

9. ¿Cuál es la propiedad que se utiliza para establecer el tamaño de un borde en CSS?
   a) border-width
   b) borders-styles
   c) border-colores
   d) border-radius

10. ¿Qué propiedad se utiliza para cambiar el estilo del cursor cuando pasa sobre un elemento en CSS?
   a) cursor-style
   b) hover-cursor
   c) cursor
   d) hover

JavaScript

1. ¿Qué tipo de lenguaje de programación es JavaScript?
   a) Lenguaje de maquina
   b) Lenguaje de programación casas
   c) Lenguaje de marcado
   d) Lenguaje de programación orientado a objetos

2. ¿Cuál es la función de JavaScript en una página web?
   a) Dar la estructura del aplicativo web.
   b) Controlar la parte visual de una página web.
   c) Gestionar la comunicación con el servidor.
   d) Agregar interactividad y dinamismo a una página web.

3. ¿Qué tipo de variable se define sin una palabra clave en JavaScript?
   a) Variable local
   b) Variable global
   c) Variable constante
   d) Variable estática

4. ¿Cuál es la forma correcta de escribir un comentario de una línea en JavaScript?
   a) // Este es un comentario
   b) /* Este es un comentario */
   c) -- Este es un comentario --
   d) <!-- Este es un comentario -->

5. ¿Qué método se utiliza para mostrar un mensaje emergente en JavaScript?
   a) alert()
   b) message()
   c) prompt()
   d) log()

6. ¿Cómo se declara una variable en JavaScript?
   a) var miVariable;
   b) let miVariable;
   c) const miVariable;
   d) Todas las anteriores son correctas

7. ¿Qué operador se utiliza para comparar dos valores en JavaScript?
   a) ==
   b) ===
   c) =
   d) !=

8. ¿Cuál es la función del método "addEventListener" en JavaScript?
   a) Agregar un nuevo elemento al documento HTML.
   b) Cambiar el estilo de un elemento en la página.
   c) Escuchar eventos como clics, teclas presionadas, etc., en un elemento específico.
   d) Crear una función anónima en JavaScript.

9. ¿Qué es el DOM en JavaScript?
   a) Una herramienta para programar.
   b) El Document Object Model, corresponde a la estructura del HTML en memoria.
   c) Un tipo de dato en JavaScript para almacenar texto.
   d) Una librería de JavaScript para crear animaciones.

10. ¿Cómo se accede a un elemento HTML en JavaScript utilizando su id?
    a) getElementById('nombreId')
    b) getElementByClass('nombreId')
    c) getElementByTag('nombreId')
    d) getElement('nombreId')

Estos cuestionarios están diseñados para evaluar el conocimiento y comprensión del participante en CSS y JavaScript, respectivamente, abordando conceptos fundamentales, propiedades clave, sintaxis y funciones esenciales de cada lenguaje para el desarrollo web. Teniendo en cuante el método de aprendizaje, este capítulo solo ofrece algunas bases y conceptos fundamentales con los que el lector podría guiarse, algunas de los temas de las preguntas acá propuestas se encuentran explícitamente en esta unidad, es compromiso por parte del lector asumir su autoaprendizaje e investigar en diferentes recursos (no solo en los recomendados en esta unidad) de ser necesario para poder responder los cuestionarios.

Por último, se plantean algunos ejercicios prácticos con los que podrá practicar y usar en conjunto las herramientas trabajadas para la elaboración de sus primeros bosquejos de páginas web:

- ***Calculadora de Propina:*** Diseña una calculadora de propina que permita al usuario ingresar el total de la cuenta y seleccionar el porcentaje de propina que desea dejar (por ejemplo, 10%, 15%, 20%). Utiliza HTML para crear la estructura del formulario y los elementos de entrada. Utiliza CSS para dar estilo a la calculadora y hacerla visualmente atractiva. Emplea JavaScript para calcular la cantidad de propina basándose en los valores que el usuario ingresa y poder visualizar el resultado.
- ***Lista de Tareas Interactiva:*** Crea una lista de tareas donde los usuarios puedan agregar nuevas tareas, marcarlas como completadas y eliminarlas. Utiliza HTML para la estructura de la lista y los botones de acción. Usa CSS para dar estilo a la lista y hacerla intuitiva. Utiliza JavaScript para manejar las interacciones del usuario, como agregar nuevas tareas, marcarlas como completadas y eliminarlas de la lista.
- ***Galería de Imágenes con Lightbox:*** Diseña una galería de imágenes que muestre miniaturas de fotos y permita al usuario hacer clic en una imagen para verla en tamaño completo en un lightbox (ventana modal). Utiliza HTML para la estructura de la galería y las miniaturas de imágenes. Utiliza CSS para dar estilo a la galería y crear el efecto de lightbox cuando se hace clic en una imagen. Emplea JavaScript para manejar la apertura y cierre del lightbox y la visualización de la imagen en tamaño completo.

Los objetivos de estos ejercicios son variados, enfocándose en evaluar y desarrollar distintas competencias en el diseño y programación web. A continuación, se detallan los objetivos específicos de cada ejercicio:

1. Calculadora de Propina

- *Evaluación de habilidades de diseño de interfaz:* Se busca que el estudiante demuestre su capacidad para crear una interfaz de usuario clara y atractiva mediante HTML y CSS, que sea intuitiva para el usuario final.
- *Comprensión de la interactividad en la web:* El ejercicio pone a prueba la habilidad del estudiante para implementar lógica de programación con JavaScript, específicamente en la captura y manejo de eventos del usuario y en la realización de cálculos basados en la entrada del usuario.
- *Manipulación del DOM:* Se evalúa la capacidad para manipular el Document Object Model (DOM) para actualizar dinámicamente el contenido de la página web y mostrar el resultado del cálculo de la propina.

2. Lista de Tareas Interactiva

- *Gestión de estados en aplicaciones web:* Este ejercicio tiene como objetivo evaluar la habilidad del estudiante para diseñar una aplicación que maneje cambios de estado, como agregar, completar o eliminar tareas, reflejando estos cambios en la interfaz de usuario de manera inmediata.
- *Uso avanzado de eventos y manipulación del DOM:* Se busca profundizar en la comprensión del estudiante sobre cómo capturar eventos del usuario (clicks, entradas de texto) y cómo estos pueden ser utilizados para modificar el DOM, agregando o eliminando elementos de la página.
- *Organización del código y lógica de programación:* Se evalúa la capacidad para estructurar el código de manera lógica y eficiente, implementando buenas prácticas de programación.

3. Galería de Imágenes con Lightbox

- Implementación de efectos visuales con CSS: A través de este ejercicio, se pretende evaluar la capacidad del estudiante para utilizar CSS para crear efectos visuales atractivos y funcionales, como un efecto de lightbox, que mejore la experiencia del usuario.
- Manejo de eventos y manipulación del contenido dinámico: Este proyecto desafía al estudiante a implementar funcionalidades interactivas utilizando JavaScript para manejar eventos como clics en las imágenes, y para controlar la apertura y cierre de un lightbox, además de la carga dinámica de contenido dentro de este.
- Diseño responsive y atención al detalle: Además de los objetivos anteriores, se busca evaluar la habilidad para crear diseños que sean responsivos y se adapten a diferentes tamaños de pantalla, asegurando que la galería sea accesible y atractiva en diversos dispositivos.
- En conjunto, estos ejercicios están diseñados para evaluar y fomentar el desarrollo de competencias clave en programación web, incluyendo el diseño de interfaces, programación interactiva, manejo de estados y eventos, y la creación de contenido dinámico y responsivo.

Para llevar a cabo los ejercicios de la calculadora de propina, la lista de tareas interactiva, y la galería de imágenes con lightbox, el estudiante debe contar

con una serie de recursos tanto de conocimiento como de herramientas técnicas. A continuación, se detallan los recursos necesarios:

Conocimientos Previos:
- HTML: Entender la estructura básica de una página web, incluyendo cómo crear formularios, listas, botones, y manejar imágenes.
- CSS: Saber cómo dar estilo a elementos HTML, incluyendo el posicionamiento, el uso de flexbox o grid para layouts, animaciones, y media queries para diseños responsivos.
- JavaScript (JS): Conocimientos fundamentales de JS para manipular el DOM, manejar eventos, y realizar operaciones lógicas y matemáticas.

Herramientas y Tecnologías:
- Editor de Código: Como Visual Studio Code, Sublime Text, Atom, o cualquier otro editor que permita escribir y organizar el código de manera eficiente.
- Navegador Web Moderno: Chrome, Firefox, Safari, o Edge, para probar y depurar las aplicaciones web. Idealmente, uno que ofrezca buenas herramientas de desarrollo para facilitar la depuración de código.
- Consola de Desarrollo del Navegador: Para depurar el código JavaScript, inspeccionar elementos HTML, y testear estilos CSS.
- Librerías JS (opcional): Aunque no son estrictamente necesarias, las librerías como jQuery pueden simplificar la manipulación del DOM y el manejo de eventos, especialmente para principiantes.

Recursos de Aprendizaje y Referencia:
Los temas vistos en este capítulo sirven para sentar las bases y conocimientos básicos para el desarrollo de los ejercicios y no ofrecen una formación completa, por ello se proponen las siguientes recomendaciones:

1. **Documentación Oficial y Tutoriales:**
   - MDN Web Docs para HTML, CSS, y JavaScript, que ofrece guías completas y referencia de la API.
   - W3Schools para tutoriales básicos y ejemplos.
   - Foros y Comunidades en Línea:
   - Sitios como Stack Overflow para resolver dudas específicas.
   - Grupos en Reddit, GitHub, y otros foros de desarrollo web para consejos y mejores prácticas.

2. **Cursos en Línea y Videos Tutoriales:**
   Plataformas como Coursera, edX, Udemy, y YouTube ofrecen cursos y tutoriales que abarcan desde lo más básico hasta técnicas avanzadas en desarrollo web.

3. **Software de Diseño (Opcional):**
   Para la galería de imágenes y otros elementos visuales, herramientas de

diseño como Adobe Photoshop, Illustrator, o alternativas gratuitas como GIMP y Inkscape pueden ser útiles para preparar y optimizar los recursos gráficos antes de su implementación en la web.

Con estos recursos, el estudiante estará bien equipado para enfrentarse a los desafíos que presentan estos ejercicios, desarrollando no solo habilidades técnicas sino también la capacidad para investigar, solucionar problemas, y aprender de manera autónoma.

REFERENCIAS BIBLIOGRÁFICAS

Latorre, M. (2018). Historia de las web, 1.0, 2.0, 3.0 y 4.0. Universidad Marcelino Champagnat, 1.

Alay, J. I. G., & Sevillano, R. P. C. (2022). Evolución de los sistemas de lenguaje de programación a lo largo de la historia. E-IDEA Journal of Engineering Science, 4(10), 14-26.

Henríquez Bustamante, R. A. (2017). Descripción del protocolo HTTP y análisis de sus transferencias mediante metalenguajes.

Luján-Mora, S. (2002). Programación de aplicaciones web: historia, principios básicos y clientes web. Editorial Club Universitario.

Montávez Sánchez, M., & Segura Sánchez, R. (2023). Arquitectura de programación web: back-end (Tesis de maestría). Universidad de Jaén, Departamento de Informática. http://creativecommons.org/licenses/by-nc-nd/3.0/es/

Roberto, G. C., Eduardo, G. G., & Ariel, O. R. (1999). Implementación del Protocolo HTTP Paralelizado en Cliente y Servidor. http://hdl.handle.net/11285/628362

Cabrera, L. V. (2013). Introducción a CSS. Recuperado de: https://www. cs. us. es/cursos/bd/temas/BD-Tema-10. pdf.

Casado Vara, R. (2019). Introducción a HTML. Universidad de Salamanca. http://hdl.handle.net/10366/139647

Luna, A. C. (2019). Creación de páginas web: HTML 5. ICB, SL (Interconsulting Bureau SL).

Prescott, P. (2015). HTML 5. Babelcube Inc.

Alegsa, L (2023). Definición de Lado-cliente. Alegsa. Recuperado de https://www.alegsa.com.ar/Dic/lado-cliente.php

Alvarez, M. A. (2001). Qué es HTML. DesarrolloWeb.com. Recuperado de https://desarrolloweb.com/articulos/que-es-html.html

Álvarez, M. A., Alvarez, S., Nadie, J., Rousset, D., Vega, J. V., Tresancos, J. P., & Lurita, J. R. C. (2017). Manual de css 3.

Appmaster (2023). Del lado del servidor. Recuperado de: https://appmaster.io/es/glossary/del-lado-del-servidor

Cloudfire (2021). ¿Qué significa lado del cliente y lado del servidor? | Lado del cliente vs. Lado del servidor. Recuperado de: https://www.cloudflare.com/es-es/learning/serverless/glossary/client-side-vs-server-side/

Colaboradores de Wikipedia. (2024). Historia de los lenguajes de programación. Wikipedia, la Enciclopedia Libre. https://es.wikipedia.org/wiki/Historia_de_los_lenguajes_de_programaci%C3%B3n

Equipo editorial, Etecé. (2023). HTTP - Concepto, para qué sirve y cómo funciona. Concepto. https://concepto.de/http/

La Torre, A (2006). Lenguajes del lado servidor o cliente. PHPNuke. Recuperado de: https://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html

Tinoco, E. E. C., & Solís, I. S. (2014). Programación Web con CSS, JavaScript, PHP y AJAX. Iván Soria Solís.

WEB, H. (2013). El protocolo HTTP.

Generalidades del protocolo HTTP - HTTP | MDN. (2023). MDN Web Docs. https://developer.mozilla.org/es/docs/Web/HTTP/Overview

Gómez, P. (2023). Historia de la programación: ¿qué es y cómo ha evolucionado con los años? - DevCamp. DevCamp. https://devcamp.es/historia-de-la-programacion-que-es-y-como-ha-evolucionado-con-los-anos/

Harsh,K (2022). ¿Qué es la Arquitectura de las Aplicaciones Web? Desglosando una Aplicación Web. Kinsta. Recuperado de: https://kinsta.com/es/blog/arquitectura-aplicaciones-web/

Hernandez, Y (2023). ¿Cuál es la arquitectura de una aplicación web? Donguee. Recuperado de: https://www.dongee.com/tutoriales/cual-es-la-arquitectura-de-una-aplicacion-web/

Hostinet(2015). Lenguajes del lado servidor o cliente. Recuperado de: https://www.hostinet.com/formacion/general/lenguajes-del-lado-servidor-o-cliente/

Juice Studio (2022). Arquitectura de aplicaciones web: La guía definitiva. Recuperado de: https://juice-studio.com/arquitectura-de-aplicaciones-web-la-guia-definitiva/

Marco, B. S. (2023). Resumen de la historia de la Web. Páginas web HTML y hojas de estilo CSS. Bartolomé Sintes Marco. www.mclibre.org. https://www.mclibre.org/consultar/htmlcss/otros/historia-resumen.html

MDN (2021). Introducción al lado servidor. Recuperado de: https://developer.mozilla.org/es/docs/Learn/Server-side/First_steps/Introduction

Team, K. (2023). ¿Qué es el protocolo HTTP? | KeepCoding Bootcamps. KeepCoding Bootcamps. https://keepcoding.io/blog/que-es-el-protocolo-http/

R. Hernández Martínez, E. J. López Hernández, A. Hernández Tejeda, M. Osorio de la Cruz, J. O. Ramírez Santiago, y L. Martínez Agustín. (2020). Programación web. https://nagaaralive0012rh.wixsite.com/mi-pagina-web/11-evolucin-de-las-aplicaciones-web

Instituto Tecnológico de Matehuala. (2013). Programacion Web. https://programacionwebisc.wordpress.com/2-5-metodologias-para-el-

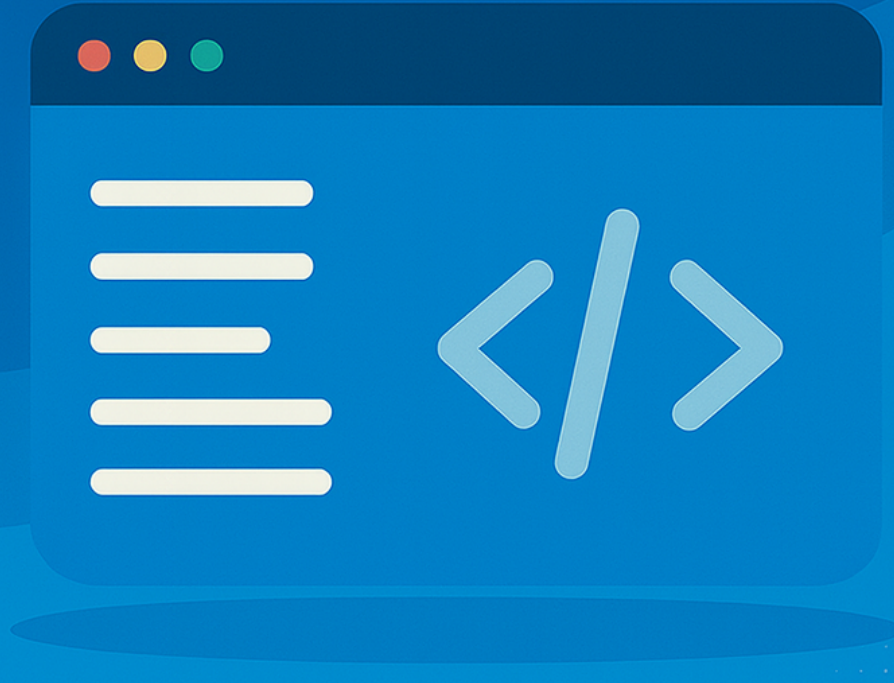
desarrollo-de-aplicaciones-web/
Content, R. R. (2021). ¿Qué es CSS y cuáles son sus funciones en Internet? Rock Content - ES. https://rockcontent.com/es/blog/que-es-css/
CSS | MDN. (2024). MDN Web Docs. https://developer.mozilla.org/es/docs/Web/CSS
Gascón González, U. (2017). JavaScript, ¡Inspírate! http://leanpub.com/javascript-inspirate
Gauchat, J. D. (2012). El gran libro de HTML5, CSS3 y Javascript. Marcombo.
Maida, E. G., & Pacienzia, J. (2015). Metodologías de desarrollo de software.
Maza, M. A. S. (2012). Javascript. Innovación Y Cualificación.
Pérez, J. E. (2019). introduccion a JavaScript.
Ríos, J. R. M., Ordóñez, M. P. Z., Segarra, M. J. C., & Zerda, F. G. G. (2017). Estado del arte: Metodologías de desarrollo en aplicaciones web. 3c Tecnología: glosas de innovación aplicadas a la pyme, 6(3), 54-71.
Souto, V. R. (2023). ¿Qué es CSS3? HACK A BOSS. https://www.hackaboss.com/blog/que-es-css
Varangouli, E. (2023). Lista de tags HTML: hoja de trucos HTML. ¿Qué son y para qué sirven? Semrush Blog. https://es.semrush.com/blog/lista-de-html-tags/

CAPÍTULO 2.

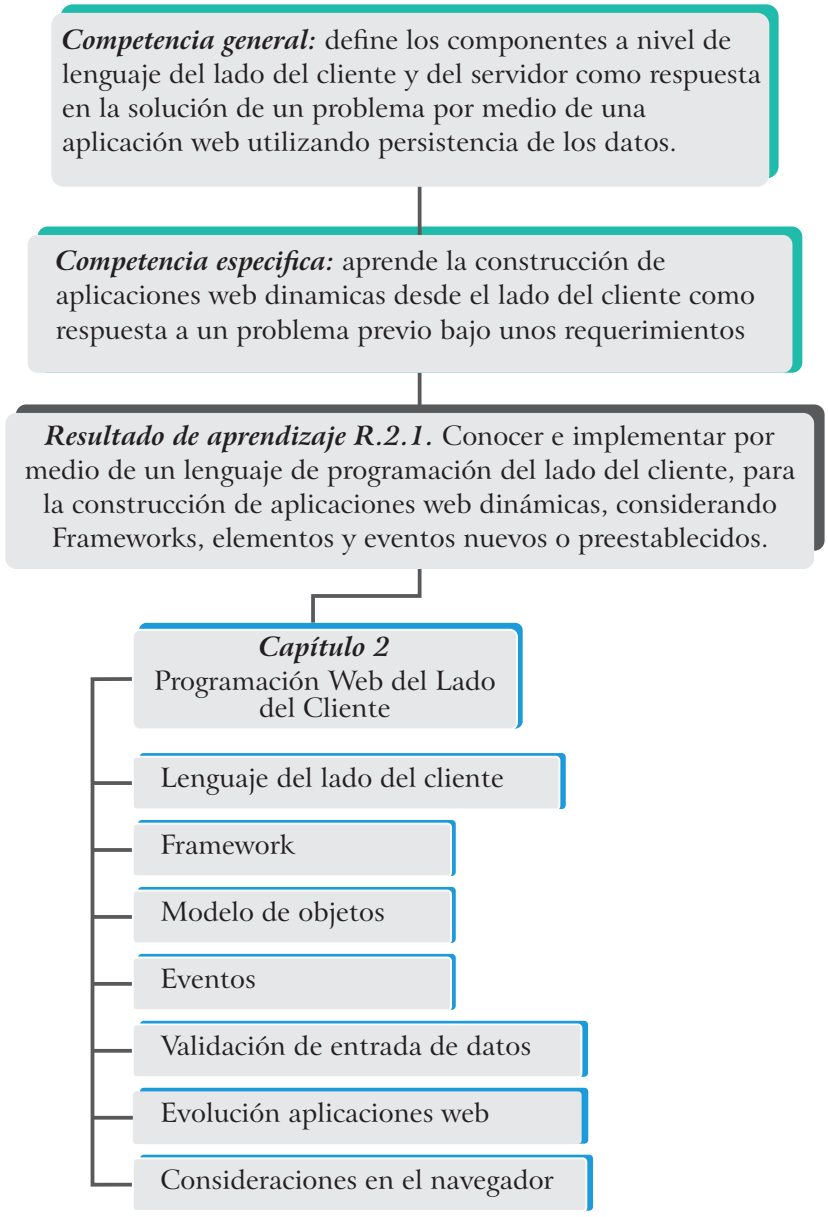
PROGRAMACIÓN WEB DEL LADO DEL CLIENTE

Michael Mauricio Orjuela Cepeda,
Heriberto Fernando Vargas Losada
Fredy Antonio Verástegui González

[1]*Estudiante del programa de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad de la Amazonia, correo: mi.orjuela@udla.edu.co.*
[2]*Docente de tiempo completo del programa de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad de la Amazonia, correo: heri.vargas@udla.edu.co*
[3]*Docente de tiempo completo del programa de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad de la Amazonia, correo: f.verastegui@udla.edu.co*



ESQUEMA | CAPÍTULO 2.



COMPETENCIA Y RESULTADO DE APRENDIZAJE | 2. CAPÍTULO

La competencia general está establecida de la siguiente forma: "Define los componentes a nivel de lenguaje del lado del cliente y del servidor como respuesta en la solución de un problema por medio de una aplicación web utilizando persistencia de los datos.". Asociada a la siguiente competencia específica para la cual se desarrolla en el capítulo uno, definida como: "Aprende la construcción de aplicaciones web dinámicas desde el lado del cliente como respuesta a un problema previo bajo unos requerimientos".

El resultado de aprendizaje asociado a la competencia del capítulo dos, se estableció de la siguiente forma R2.1 Conocer e implementar por medio de un lenguaje de programación del lado del cliente, para la construcción de aplicaciones web dinámicas, considerando Frameworks, elementos y eventos nuevos o preestablecidos. Los niveles de desempeño definidos para el resultado de aprendizaje son expuestos en la tabla 2.

Tabla 2.
Niveles de desempeños del resultado de aprendizaje R2.1

Avanzado (5-4,8)	Intermedio (4,7-4)	Básico (3.9 -3)	Bajo (2.9-0)
Desarrolla la aplicación del lado del cliente con la implementación de frameworks, elementos y componentes ideales que dan respuesta eficiente en su operatividad y usabilidad.	Desarrolla la aplicación del lado del cliente con la implementación de frameworks, elementos y componentes no optimizados que dan respuesta eficiente en su operatividad y usabilidad.	Desarrolla la aplicación del lado del cliente con la implementación de frameworks, elementos y componentes no evidencia el dominio teórico que optimice la operatividad y usabilidad del sistema	Presenta dificultad en el desarrollo de la aplicación del lado del cliente con la implementación de frameworks, elementos y componentes efectivos que den respuesta desde su operatividad y usabilidad.

Nota. *Trabajo realizado a partir del desarrollo de resultados de aprendizaje. Fuente: elaboración propia. Las actividades de evaluación determinadas para el resultado de aprendizaje son las siguientes: Primera entrega de un proyecto relacionado con un sistema de información web utilizando los conceptos teóricos y prácticos de la presente capítulo (Código fuente y sustentación).*

MÉTODOS Y MATERIALES DE APRENDIZAJE

Este libro adopta una metodología de enseñanza híbrida que combina el autoaprendizaje con la orientación estructurada, destinada a encender una comprensión profunda de la programación web del lado del cliente. En este enfoque, se requiere que el estudiante participe activamente en su aprendizaje y complemente la independencia del autoaprendizaje con el apoyo representado por fuentes de aprendizaje cuidadosamente seleccionadas, junto con ejercicios prácticos. Para este fin, se concede acceso a

una gran variedad de recursos en línea, desde documentaciones oficiales hasta tutoriales interactivos y artículos especializados, permitiendo a los estudiantes profundizar en cada materia a su propio ritmo y necesidades. Algunas de las fuentes sugeridas para obtener conocimiento invaluable sobre los aspectos básicos y avanzados del desarrollo web incluyen MDN Web Docs, WjsonSchools y FreeCodeCamp.

El aprendizaje aquí se refuerza a través de una serie de ejercicios prácticos y proyectos que desafían críticamente la capacidad de los aprendices para aplicar los conceptos teóricos de la clase en situaciones del mundo real, fomentando así una mentalidad de resolución creativa de problemas y el desarrollo de aplicaciones web dinámicas. De acuerdo con el enfoque de la educación, insiste en que los estudiantes interactúen y trabajen juntos, a partir de lo cual este proceso hace cumplir que los estudiantes usen los foros para discusión y grupos de estudio. Desde esta dinámica, se promueve un intercambio de ideas y experiencias que enriquecen la experiencia de aprendizaje de todos los miembros. Así, el autoestudio combinado con recursos de calidad ayudará a preparar al estudiante para obtener una habilidad invaluable para el desarrollo profesional futuro, además de habilidades técnicas en programación web del lado del cliente: aprendizaje independiente. Por lo tanto, este enfoque pedagógico busca equipar a los estudiantes no solo con una sólida formación en ingeniería, sino también, ofreciéndoles habilidades blandas como la autogestión, la resolución de problemas y la capacidad continua de aprendizaje, lo cual es muy vital en el frente tecnológico siempre dinámico.

DESARROLLO TEMÁTICO

A lo largo del Desarrollo Temático, se profundizará en aspectos fundamentales de la programación web del lado del cliente, desde los principios básicos de HTML, CSS y JavaScript, hasta implementar frameworks como Angular, React, y Vue.js. Se abordará cómo manipular el DOM, gestionar eventos, validar entradas de usuario y asegurar compatibilidad entre navegadores, todo con el objetivo de poder estar a la vanguardia en la creación de aplicaciones web dinámicas. Además, se proporcionarán recursos y referencias en línea para enriquecer el aprendizaje y facilitar la aplicación práctica de los conocimientos adquiridos.

Lenguaje del lado del cliente

El equilibrio en el desarrollo de aplicaciones web es esencial, y este ha sido influenciado por la aparición de tecnologías web modernas que trasladan algunas funciones tradicionalmente reservadas para el servidor hacia el cliente. Uso Adolfo Rodríguez (2017) destaca este punto al mencionar que "Este conjunto de tecnologías (HTML, CSS, y JS) son las utilizadas para programar sitios web del lado del cliente, es decir, que se ejecutan del lado del navegador del usuario, no del servidor donde se almacena la página web diseñada".

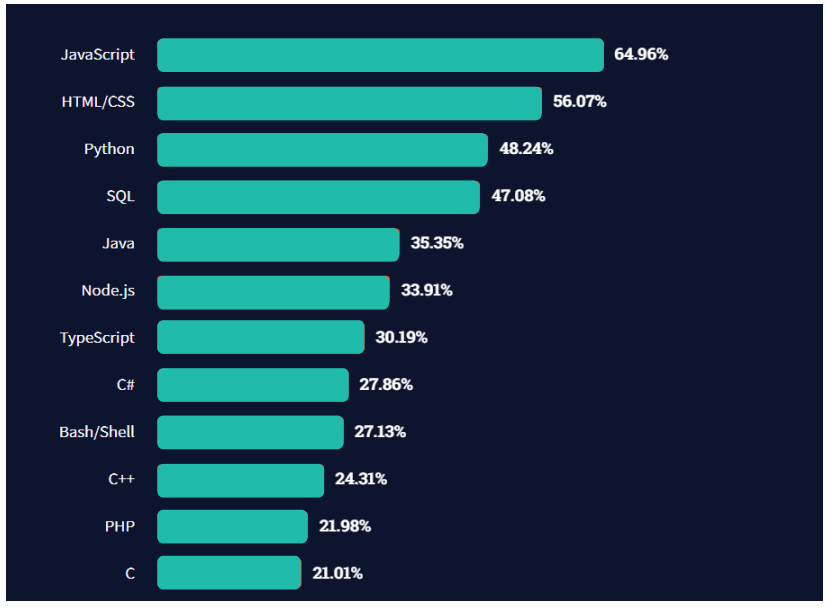
Este hecho resalta cómo el desarrollo del lado del cliente se ha convertido en una parte esencial de la arquitectura web, centrándose en crear experiencias de usuario dinámicas y fluidas mediante la manipulación de la estructura, el estilo y la interactividad de las páginas web. Por ejemplo, esto muestra una clara separación de preocupaciones a través del uso muy efectivo de tecnologías del lado del cliente: estas tecnologías son hoy en día de las más usadas en el entorno de desarrollo web como se puede observar en la Ilustración 1. HTML ha tenido una gran evolución como se puede ver en la Ilustración 2, y su función es estructurar el contenido, mientras que CSS define la presentación, y JavaScript se encarga de la interactividad, cabe recalcar que con la ilustración 3 se puede entender mejor estas funcionalidades. Esta división de roles facilita el desarrollo web y lo hace más eficiente ya que las operaciones se llevan a cabo directamente en el navegador del usuario. Tal división de trabajo simplifica el desarrollo web y mejora la eficiencia, ya que todas las operaciones se realizan en el navegador del usuario.

Además, el entorno de desarrollo del lado del cliente se ha enriquecido significativamente con la adopción de frameworks y librerías que estandarizan y agilizan el proceso de desarrollo. La interacción del usuario con el sitio web que visita está directamente relacionada con el FrontEnd, como señalan Graciela et al. (2021), al diferenciar las responsabilidades del Back-end, que "es la capa de acceso a los datos que no es visible para el usuario final y contiene la lógica de la aplicación que maneja los datos, como bases de datos y servidor". Mientras que el FrontEnd abarca todo lo que el usuario puede ver e interactuar en el sitio web, el Back-end trabaja de manera invisible pero esencial, gestionando la información y las transacciones que ocurren en segundo plano. Además, la tendencia a desplazar más carga computacional al navegador facilita el desarrollo de aplicaciones web no solo visualmente atractivas sino también funcionalmente ricas.

Este enfoque es reforzado por J. D. Walker & Chapra (2014), quienes observan que "Los desarrollos recientes en tecnologías web, incluyendo la evolución de los estándares web, mejoras en el rendimiento de los navegadores y la aparición de bibliotecas de software libre y de código abierto, están impulsando un cambio general de aplicaciones web del lado del servidor al lado del cliente". Siguiendo esta línea Clark (2021), asegura que es en este contexto que el JavaScript ejecutado en el navegador y su naturaleza interpretada empoderan a una página web, añadiendo dinamismo e interactividad para suavizar la experiencia del usuario. A pesar de esta naturaleza tan versátil y su uso popular, la visibilidad del código del lado del cliente plantea desafíos de seguridad complejos y necesita pruebas minuciosas que deben ser consistentes a través de diferentes comportamientos del navegador.

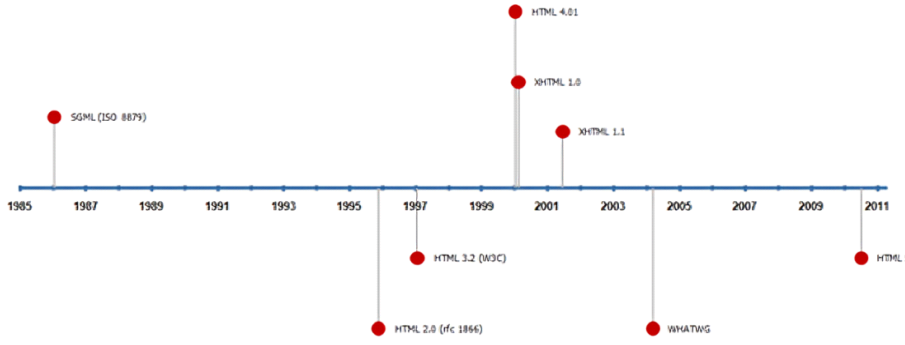
En conclusión, el desarrollo web contemporáneo se caracteriza por una sinergia de funcionalidad y estética, donde el rendimiento del lado del cliente ha emergido como una prioridad. El avance continuo en esta área ha expandido aún más las posibilidades para los desarrolladores de permitir una experiencia inmersiva y eficiente por parte del usuario, quien continuamente redefine sus expectativas y estándares.

Figura 1.
Popularidad de Lenguajes de Programación en Desarrollo Web



Nota. *Esta gráfica muestra la popularidad de lenguajes de programación y tecnologías, con JavaScript liderando seguido de cerca por HTML/CSS, destacando su predominio en el desarrollo web moderno. Fuente: Stack Overflow Developer Survey (2023).*

Figura 2.
Evolución del lenguaje de marcado HTML



Nota. *La imagen muestra una línea de tiempo horizontal que representa la evolución de las versiones de HTML desde 1991 hasta 2011. Fuente: Rodríguez & Botelho (2012).*

Figura 3.
Una analogía entre HTML, CSS y JavaScript

Nota. *La imagen usa analogías humanas para ilustrar el rol de HTML como la estructura de una web (el esqueleto), CSS como el estilo (la piel) y JavaScript como la funcionalidad interactiva (el cerebro), explicando su coordinación en el desarrollo web. Fuente: elaboración propia.*

El estudio de estas tecnologías se puede ampliar con la ayuda de diversas fuentes educativas disponibles en línea. Contiene una enorme colección de guías, tutoriales y referencias detalladas sobre cómo cubrir todo, desde HTML básico —adecuado para el principiante absoluto— hasta temas altamente avanzados y muy especializados. Estos recursos están diseñados principalmente para que los usuarios se familiaricen con las nociones básicas y la sintaxis de HTML, pero al mismo tiempo, también se les proporciona guías sistemáticas sobre cómo aplicar características más intrincadas y sofisticadas del lenguaje: inclusión de multimedia, construcción de tablas, gestión de formularios y solución de problemas frecuentes.

A través de explicaciones detalladas y ejemplos ilustrativos, MDN Web Docs se asegura de que los desarrolladores a cualquier nivel, principiantes o avanzados puedan aplicar HTML efectivamente en sus proyectos web. La plataforma enfatiza fuertemente el HTML como un lenguaje no programable que asume su rol para estructurar el contenido web en lugar de construir comportamientos complejos o estilos visuales, que se logran con la ayuda de CSS y JavaScript. "Esta formulación hace que el usuario exprese claramente la utilidad y el propósito del HTML dentro del entorno de desarrollo web, mientras apela indirecta o directamente a buenas prácticas para marcar el contenido y permitir el acceso al mismo con una accesibilidad y usabilidad óptimas (HTML: Lenguaje de Etiquetas de Hipertexto | MDN, 2023).

W3Schools se destaca como un portal de aprendizaje para tecnologías web básicas como HTML, CSS y JavaScript. El sitio tiene reputación de utilizar

una metodología aplicada que incluye tutoriales interactivos y ejercicios prácticos diseñados específicamente para fomentar un buen aprendizaje independiente. Esto lo convierte en un excelente recurso para cualquier persona que esté comenzando en el mundo del desarrollo web y valioso no solo para establecer una base sólida en conceptos básicos sino también para que los profesionales lo usen como referencia rápida cuando necesiten profundizar o actualizar sus conocimientos. Los contenidos están organizados de tal manera que son intuitivos para permitir al usuario avanzar a través de las lecciones a su propio ritmo y aplicar lo que han aprendido mediante pruebas y proyectos prácticos (W3Schools, 1999). Por otro lado, Codecademy se destaca como una plataforma de aprendizaje en línea interactiva y popular para aprender programación y mejorar las habilidades de desarrollo web mediante el aprendizaje práctico. Desde los bloques de construcción básicos hasta el concepto avanzado, idiomas como Python, JavaScript, incluso el diseño y la estructura web con HTML, CSS, cubre todo. Este enfoque pedagógico promueve un aprendizaje dinámico en el que el estudiante se involucre directamente con el código, facilitando una comprensión más profunda y aplicada de los contenidos (Sims & Bubinski, 2011).

2.4.2 Framework

Los marcos de desarrollo de software son muy importantes, ya que actúan como esqueletos que ofrecen estructuras y conjuntos de herramientas de manera predefinida para construir aplicaciones de manera efectiva. Esto evita tener que empezar cada nuevo proyecto desde cero y optimiza el proceso de desarrollo. Según un análisis de Pandya (2023): "estos marcos de trabajo ofrecen componentes predefinidos y bibliotecas que permiten a los desarrolladores concentrarse en implementar funciones únicas y lógica de negocio específica, en lugar de reinventar soluciones a problemas comunes". Esta eficiencia en el desarrollo asegura que no solo se pueda completar un proyecto en menos tiempo, sino que también el código será más fácil de mantener y más extensible en el futuro. Los frameworks no solo proporcionan una base sólida para el desarrollo de aplicaciones, sino que también promueven un entorno de colaboración continua y aprendizaje entre los desarrolladores, formando comunidades alrededor de estas herramientas para compartir conocimientos y contribuir a mejoras.

Este enfoque es muy valioso, especialmente al desarrollar sitios web, donde es crucial la compatibilidad con diversas plataformas, interfaces y dispositivos. Por ejemplo, React, una biblioteca conocida y popular como se puede observar en la ilustración 4, debido a su adaptabilidad y eficiencia, introduce la idea de usar componentes como bloques de construcción para interfaces de usuario. Estos encapsulan la lógica y la estructura necesarias para proporcionar una experiencia de desarrollo más organizada y mantenible. Un principio clave de React es su enfoque en las declaraciones de la interfaz de

usuario, lo que permite a los desarrolladores describir cómo debería verse la UI en un momento dado sin preocuparse por las transiciones de estado manuales. Utiliza un enfoque que incluye un DOM virtual, una representación en memoria ligera del DOM real, lo que ayuda a optimizar las actualizaciones de la interfaz por parte de React para que sean más ágiles y eficientes.

La capacidad de React para volver a renderizar solo aquellas partes de la aplicación que necesitan ser actualizadas debido a cambios en el estado de los datos, en lugar de todo el árbol del DOM, no solo ofrece un mejor rendimiento, sino que también facilita mucho el desarrollo. Este enfoque declarativo, combinado con un potente sistema de componentes, facilita la construcción de aplicaciones dinámicas complejas manteniendo el código limpio y comprensible (Fedosejev, 2015). Angular se ha convertido en una herramienta indispensable para crear sitios web modernos y adaptables que satisfacen diversas necesidades, desarrollado y mantenido por Google. El uso de TypeScript facilita su desarrollo, proporcionando un código más claro y fácil de manejar, lo cual es esencial en proyectos complejos (Azaustre, 2014). Lo que es especial en Angular es que ayuda a estructurar el código en bloques reutilizables de manera efectiva que proporcionan a los desarrolladores una mejor forma de trabajar manteniendo sus proyectos ordenados.

También es destacable por sus aplicaciones que se cargan sin necesidad de refrescar la página, conocidas como SPA, que facilitan la navegación y mejoran la experiencia de los usuarios. Angular ofrece herramientas con las que es fácil seguir las mejores prácticas de ingeniería de software, resultando en aplicaciones sólidas y confiables. La documentación detallada y una gran comunidad de usuarios ayudarán en el aprendizaje y la resolución de problemas, así que el desarrollo futuro de Angular será tranquilizador frente a los futuros desafíos en el desarrollo web, manteniendo su popularidad y eficacia. Vue.js ha surgido de forma natural como un framework de JavaScript progresivo y adaptable perfecto para construir interfaces de usuario y aplicaciones de una sola página (SPA), conocido por su simplicidad y su enfoque en los datos reactivos y componibles. A diferencia de otros frameworks monolíticos, Vue.js , está diseñado desde el principio para ser incrementable. Cualquier proyecto integrado en otras librerías de JavaScript podría integrarse fácilmente con un mínimo esfuerzo y características poderosas proporcionadas al proyecto Vue.

Ayuda a crear fácilmente componentes de alto rendimiento que son mantenibles y legibles con Vue.js, ya que el sistema reactivo puede rastrear eficazmente las dependencias en la renderización. Como destaca el análisis de Celi (2023), Vue.js se combina con algunas otras tecnologías como Axios para solicitudes HTTP y PHP en el back-end en el desarrollo de aplicaciones web que no son solo dinámicas sino también interactivas con el usuario y fáciles de aprender para una perspectiva de desarrollo rápido. La importancia de cada

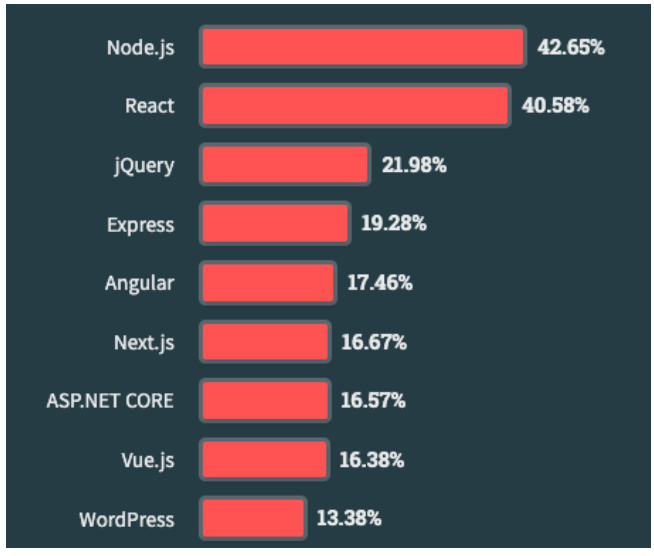
uno tiene sus características, las cuales también pueden inclinar la balanza para un proyecto u otro en el camino de desarrollar aplicaciones web modernas y eficientes. A continuación, se muestra una comparación comprensiva de Vue.js, React y Angular, con ejemplos para cada uno.

Tabla 4.
Comparativa de Características y Aplicaciones de Frameworks Populares

Característica	Vue.js	React	Angular
Lanzamiento	2014	2013	2010
Enfoque	Flexibilidad y simplicidad	Componentes y rendimiento	Complejidad manejable y tipado fuerte
Popularidad	Creciente, con una comunidad activa	Alta, con uso extenso en la industria	Alta, con una base de usuarios leales
Escalabilidad	Buena, con componentes reactivos y reutilizables	Excelente, con optimización del DOM	Muy buena, ideal para aplicaciones empresariales
Facilidad de Uso	Alta, con una curva de aprendizaje suave	Media, con un enfoque único en UI	Media, con una sintaxis estricta pero poderosa

Nota. La información proporcionada se basa en datos actuales y tendencias del mercado, así como en la documentación oficial de cada frameworks. Fuente: elaboración propia.

Figura 4.
Frameworks y librerías de desarrollo web más utilizados.



Nota. La gráfica ilustra la prevalencia en el uso de frameworks y librerías para desarrollo web, resaltando a React con una alta tasa de adopción, solo superada por Node.js, lo que refleja su robustez y preferencia en la comunidad de desarrolladores. Fuente: Atwood & Spolsky (2008).

Ejemplo 1

Creación de un Componente de Contador en React

```
    // Importamos el componente Component de React para poder crear
    nuestra clase.
    import React, { Component } from 'react';
    // Se define una clase 'Contador', que extiende de Component, para tener
    acceso a todas las funcionalidades de un componente React.
    class Contador extends Component {
      // El constructor es un método que se ejecuta al crear una instancia del
    componente.
      // Aquí se inicializa el estado del componente 'Contador' con un contador
    establecido en 0.
      constructor(props) {
        super(props);
        this.state = { contador: 0 };
  }
  // Método 'incrementar' que actualizará el estado, aumentando el contador
en uno cada vez que se invoque.
  incrementar = () => {
    this.setState({ contador: this.state.contador + 1 });
  };
    // El método 'render' devuelve el JSX (sintaxis parecida al HTML) que
queremos mostrar en pantalla.
  // Dentro del método 'render', usamos 'this.state.contador' para mostrar el
valor actual del contador.
    // También incluimos un botón que, al hacer clic, llamará al método
'incrementar'.
  render() {
    return (
      <div>
        <h2>Contador: {this.state.contador}</h2>
        <button onClick={this.incrementar}>Incrementar + 1</button>
      </div>
    );
  }
}
// Exportamos la clase 'Contador' para poder utilizarla en otros archivos.
export default Contador;
```

Este código ilustra los conceptos de estado y eventos en React. Cada vez que el usuario hace clic en el botón, el estado del componente se actualiza, lo que desencadena una nueva renderización del componente con el contador actualizado.

Figura 5.
Vista del componente contador.

Ejemplo 2

Binding de Propiedades en Angular

```
// Importamos los decoradores Component de '@angular/core' para definir
    una nueva clase de componente.
import { Component } from '@angular/core';

// Decoramos la clase con @Component para indicar que es un componente y
    establecemos su selector y plantilla.
@Component({
 selector: 'mi-app',
 template: `
   <h1>J Hola soy {{nombreUsuario}}</h1>
   <input [(ngModel)]="nombreUsuario" type="text">
 `
})
export class AppComponent {
   // Inicializamos la propiedad 'nombreUsuario', que vamos a usar en el
    binding.
 nombreUsuario = 'Usuario';
}
```

Nota. Angular utiliza el binding bidireccional para mantener sincronizados el modelo y la vista. Cualquier cambio en la entrada de texto se reflejará automáticamente en la propiedad nombreUsuario y viceversa, gracias a [(ngModel)]. Fuente: elaboración propia.

Figura 6.
Vista de binding en Angular.

Ejemplo 3.

Renderizado Condicional en Vue.js

```
<template>
 <div>
   <!-- Se utiliza 'v-if' para renderizar este h1 solo si 'usuarioRegistrado' es true
-->
     <h1 v-if="usuarioRegistrado">Hola :), Bienvenid@ {{ nombreUsuario
}}</h1>
     <!-- Se utiliza 'v-else' para definir un bloque alternativo que se muestra
cuando 'usuarioRegistrado' es false -->
   <h1 v-else>Por favor, regístrate.</h1>
 </div>
</template>

<script>
// Se exporta un objeto por defecto que define el componente.
export default {
 // La función 'data' devuelve los datos del componente.
 data() {
   return {
     // 'usuarioRegistrado' controla qué bloque se renderiza.
     usuarioRegistrado: true,
     // 'nombreUsuario' se muestra si 'usuarioRegistrado' es true.
     nombreUsuario: 'Michael Orjuela'
   };
 }
};
</script>
```

Nota. Vue.js permite controlar qué elementos se muestran en la interfaz según las condiciones. Aquí se hace uso de v-if y v-else para mostrar mensajes diferentes dependiendo del estado del usuario. Fuente: elaboración propia.

Figura 7.
Vista de renderizado condicional cuando usuario es True.

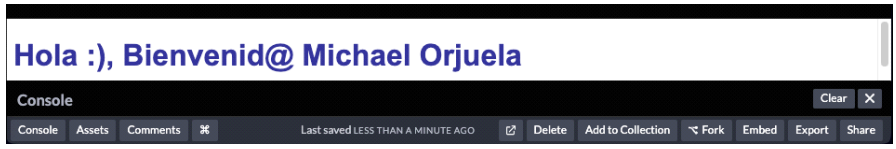
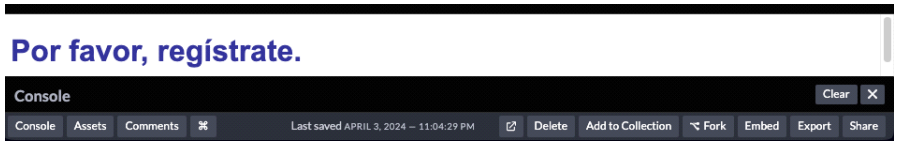


Figura 8.
Vista de renderizado condicional cuando usuario es False.



Nota. En términos de interés en ReactJS, el sitio React.js Wiki debería ser una referencia imprescindible. Fuente: elaboración propia.

Este sitio responde a muchas preguntas comúnmente realizadas y ofrece soluciones sobre problemas que generalmente aparecen al trabajar con React. Ofrece guías paso a paso, resolviendo problemas comunes y ofreciendo consejos de optimización. Es adecuado para todo tipo de desarrolladores que tienen la intención fundamental de sumergirse en este framework y aplicar buenas prácticas en el desarrollo de sus proyectos web (reactjs.wiki, 2023). Si estás interesado en React, el sitio oficial para encontrar todo lo necesario para descargar y comenzar a trabajar con este framework es es.react.dev (Walker, 2016).

El sitio "Angular University" mejor definido es una plataforma de e-learning con muchos cursos y tutoriales sobre Angular. Este recurso es perfecto para cualquier nivel de desarrollador que busque profundizar su comprensión de este framework de JavaScript. Contiene vistas en profundidad sobre RxJS, manejo de estado con NgRx y detalles de optimización de rendimiento (Angular University.io, 2016). Finalmente, FreeCodeCamp ofrece un artículo en español que detalla cómo instalar Angular en Windows. Este recurso útil guiará a los desarrolladores a través de cada paso de la configuración e instalación de Angular, comenzando desde la configuración del entorno y terminando con el trabajo con Angular CLI para tener un comienzo productivo de su desarrollo. (Izquierdo, 2022).

"Lenguaje JS" y su sección sobre Vue.js proporcionan una introducción muy accesible y clara para aquellos que comienzan en Vue.js. Proporciona una guía paso a paso sobre cómo comenzar con la construcción básica de una aplicación utilizando Vue.js; explica a través de ejemplos prácticos y útiles para que incluso un principiante esté en posición de montar su primer proyecto interactivo y reactivo (Roman, 2017). Para descargar Vue.Js se puede visitar su página oficial para descargar el framework y probar a explorar

su potencial. Vue.Js no da la oportunidad de acceder directamente a la aplicación, sino que ofrece varias opciones de descarga, con apoyo simultáneo a través de todos los detalles de la instalación y los primeros pasos en el desarrollo de una aplicación Vue.Js (You, 2022).

2.4.3 Modelo de objetos

El modelo de objetos en programación es una abstracción clave que encapsula datos y comportamiento en estructuras modulares conocidas como objetos, que son fundamentales para la construcción de software complejo. Esta abstracción es la esencia de la programación orientada a objetos (OOP), donde los "objetos" son instancias de "clases" que encapsulan datos y funciones relacionadas, brindando una identidad única, manteniendo un estado y ofreciendo comportamiento a través de métodos, se puede entender mejor este concepto en la Ilustración 5. Salvador & Díaz (n.d.) ilustran cómo estos objetos representan conceptos del mundo real, mejorando la reutilización del código y la eficiencia en la programación. La postura de Morocho (2023) sobre la POO refleja su importancia no solo en la representación de la realidad, sino también en la preservación de la integridad del software a través de la abstracción y la encapsulación de datos. A estas observaciones se suma la visión de Sist Nelly Karina Esparza Cruz (2019), quien subraya los cuatro pilares clave de la POO: abstracción, encapsulación, herencia y polimorfismo. Cada uno de ellos contribuye a un diseño de software más modular, mantenible y escalable. Esto refirma la relevancia de operaciones centradas en las características esenciales, evitando detalles irrelevantes, mientras protege el estado interno de un objeto, algo crucial para la seguridad y robustez del software.

Como complemento a esta perspectiva, González (2021) utiliza la abstracción para simplificar la representación de objetos y facilitar la programación. Además, Mazon (2015) subraya la herencia (Figura 2.6) y el polimorfismo como la base para la reutilización de código y un mecanismo indispensable para la evolución del sistema de software. Esta visión ayuda a estructurar programas desde una perspectiva orientada a objetos y permite el desarrollo de componentes que interactúan de manera coherente y coordinada, resultando en sistemas fuertemente cohesionados y de alta modularidad, fáciles de mantener y evolucionar con el tiempo. La programación orientada a objetos se ajusta bien a las metodologías ágiles, favoreciendo desarrollos de software que pueden crecer y cambiar fácil y sin problemas. Estas características han señalado su valor para estudiantes y profesionales que luchan con tecnologías en constante cambio. A continuación, se llevarán a cabo diferentes ejemplos para entender mejor los conceptos del Modelo de Objetos

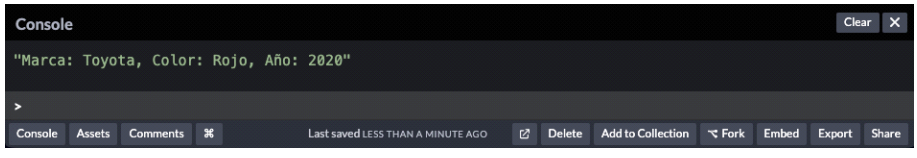
Ejemplo 1

Definición de una Clase y Creación de una Instancia en JavaScript

```
// Se define una clase 'Carro' utilizando la sintaxis de ES6.
class Carro {
   // El constructor inicializa los atributos de la clase.
   constructor(marca, color, año) {
      this.marca = marca;
      this.color = color;
      this.año = año;
   }
   // Método para mostrar información del carro.
   mostrarInformacion() {
            console.log(`Marca: ${this.marca}, Color: ${this.color}, Año:
${this.año}`);
   }
}
// Se crea una instancia de la clase 'Carro'.
const miCarro = new Carro("Toyota", "Rojo", 2020);
// Se llama al método para mostrar la información del carro.
miCarro.mostrarInformacion();
```

Este código muestra cómo se define una clase en JavaScript y cómo se crea una instancia de esa clase. La clase Carro incluye un método para mostrar su información, demostrando encapsulación y abstracción, conceptos clave en OOP.

Figura 9.
Vista de la creación de una instancia.



Ejemplo 2.

Implementación de Herencia en JavaScript

```
// Se define una clase 'Animal' que servirá como clase base.
class Animal {
   constructor(nombre) {
      this.nombre = nombre;
   }

   // Método común a todos los animales.
   comer() {
      console.log(`${this.nombre} está comiendo.`);
```

```
    }
}

// Se define una clase 'Perro' que hereda de 'Animal'.
class Perro extends Animal {
    // Método específico de la clase 'Perro'.
    ladrar() {
        console.log(`${this.nombre} está ladrando.`);
    }
}
// Se crea una instancia de 'Perro'.
const miPerro = new Perro("Toby");

// El perro puede comer y ladrar, demostrando herencia y polimorfismo.
miPerro.comer();
miPerro.ladrar();
```

Nota. En este ejemplo, se utiliza la herencia en JavaScript para extender la funcionalidad de la clase base Animal. La clase Perro hereda el método comer y añade su propio método ladrar. Fuente: elaboración propia.

Figura 10.
Vista de la implementación de herencia.

Ejemplo 3.

```
Implementando Composición en JavaScript
// Se define dos objetos que representan diferentes habilidades.
const puedeCantar = {
    cantar: function() {
        console.log(`${this.nombre} está cantando.`);
    }
};

const puedeBailar = {
    bailar: function() {
        console.log(`${this.nombre} está bailando.`);
    }
};

// Función que asigna habilidades a un objeto.
function crearArtista(nombre) {
    let artista = { nombre: nombre };
```

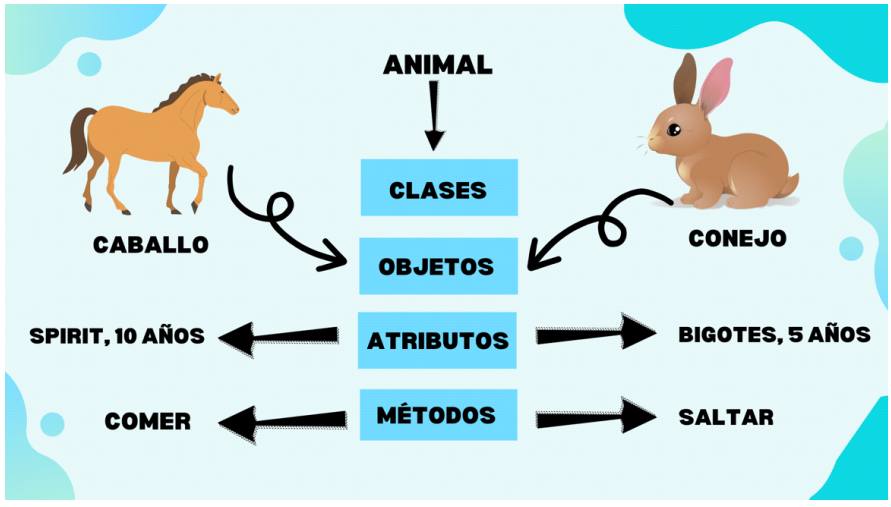
```
    // Se asignan habilidades al artista. Esto es composición.
    Object.assign(artista, puedeCantar, puedeBailar);

    return artista;
}
let artista1 = crearArtista("Elena");
// Ahora Elena puede cantar y bailar, aunque no hay una jerarquía de clases.
artista1.cantar(); // "Elena está cantando."
artista1.bailar(); // "Elena está bailando."
```

Nota. Este ejemplo introduce la composición, una alternativa a la herencia en OOP. En lugar de crear una jerarquía de clases, se construye objetos al combinar múltiples objetos más simples. Esto simplifica la estructura del código y mejora la reutilización de funcionalidades. Fuente: elaboración propia.

Figura 11.
Vista de la implementación de composición.

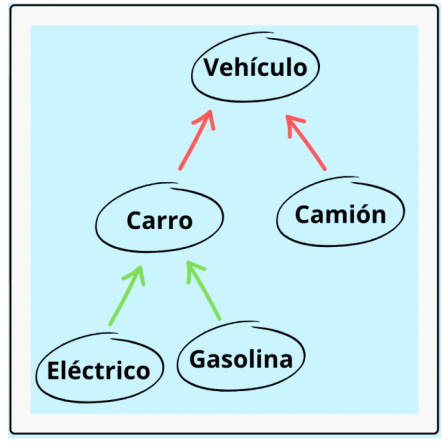
Nota. Vista de resultado por consola. Fuente: elaboración propia.

Figura 12.
Diagrama básico de la programación orientada a objetos.



Nota. Este diagrama ilustra los conceptos fundamentales de la programación orientada a objetos, mostrando cómo se relacionan las clases, objetos, atributos y métodos utilizando el ejemplo de animales, específicamente un perro y un gato. Fuente: Elaboración propia.

Figura 13.
Demostración simple de herencia en la orientación a objetos.



Nota. La imagen muestra un diagrama de herencia en la programación orientada a objetos, donde "Vehículo" es la clase superior y "Carro" y "Camión" son subclases que heredan características de "Vehículo". A su vez, "Carro" se especializa aún más en subclases "Eléctrico" y "Gasolina", demostrando cómo la herencia permite la extensión y especialización en la estructura de clases. Fuente: elaboración propia.

Para aquellos que deseen profundizar en el mundo de la programación orientada a objetos, especialmente en JavaScript y desde el lado del cliente, hay recursos en español que proporcionan conocimientos exhaustivos y aplicables. MDN Web Docs ofrece una documentación extensa que va desde la creación de objetos hasta la complejidad de la herencia y los prototipos en JavaScript, todo explicado con ejemplos prácticos que facilitan la comprensión de cada concepto (MDN, 2023b).

DesarrolloWeb.com es otra fuente invaluable para aprender y profundizar en la programación web, especialmente para aquellos interesados en el modelo de objetos en JavaScript y la programación orientada a objetos. Este portal es rico en contenido educativo que abarca desde lo más básico del lenguaje JavaScript hasta los patrones de diseño más complejos y prácticas avanzadas de desarrollo web. Sus tutoriales detallados, ejemplos de código y artículos técnicos están presentados de forma didáctica y estructurada, permitiendo a los usuarios no solo seguir los pasos, sino también comprender los fundamentos y la lógica detrás del código (Alvarez, 2021a). Además, LibrosWeb.es es un sitio en español que proporciona una perspectiva integral y accesible para aprender JavaScript y sus aplicaciones en la programación orientada a objetos. Este recurso es extremadamente valioso tanto para principiantes que buscan construir una base sólida como para desarrolladores experimentados interesados en técnicas avanzadas. LibrosWeb.es guía a los usuarios a través de conceptos de la POO como clases, herencia y polimorfismo, enfatizando cómo estos principios se aplican en el desarrollo de aplicaciones web modernas (Eguiluz, 2024).

Estos sitios web son cada uno una fuente útil para el desarrollo profesional, vinculando la teoría con la práctica aplicada y asegurando que los programadores puedan mantenerse actualizados y competentes en un entorno de cambios tecnológicos rápidos. Al ofrecer guías paso a paso y profundizar en los detalles técnicos de la POO, estos recursos son fundamentales para cualquier persona que aspire a dominar la programación orientada a objetos y sus aplicaciones prácticas.

Eventos

En el ámbito de la programación, los eventos son fundamentales para las interfaces interactivas y reactivas. Son acciones o sucesos que se producen durante la vida de un programa, como un clic del ratón o una pulsación de tecla, y pueden ser capturados por el software para responder a ellos como se ver en la ilustración 7. Un evento puede ser causado por la interacción directa del usuario con el sistema, o podría ser el resultado de otro tipo de actividad, como una respuesta del servidor o un cambio en los datos. El manejo de eventos es esencial para crear aplicaciones que no solo respondan a las necesidades del usuario, sino que también ofrezcan una experiencia fluida y dinámica. La programación basada en eventos es un modelo en el que la ejecución del programa está controlada por eventos.

Según R. González & Quesada González (2015), el desarrollo tecnológico moderno requiere una sólida comprensión de la informática y la programación, siendo la programación orientada a eventos un enfoque imprescindible en la enseñanza y el aprendizaje de la computación. Este paradigma se refleja en la necesidad de formar ciudadanos competentes con conocimiento de los principios de la informática, donde la gestión de eventos es una habilidad esencial. Guérin (2018), señala que la gestión de eventos es un proceso clave que permite a los controles web interactuar dinámicamente con los usuarios y otros componentes de una página web. Los manejadores de eventos capturan estas interacciones y los programadores pueden escribir lógica personalizada que responda a esos eventos. Manejar eventos es más que simplemente observar lo que hace el usuario; implica pensar y establecer cómo debe actuar el programa a continuación.

A veces, esto puede ser tan simple como cambiar lo que se ve en la pantalla, mientras que otras veces puede implicar coordinar partes más complejas de la aplicación. Según un estudio realizado por Miu et al. (2020) la complejidad y la naturaleza dinámica de las aplicaciones web modernas exigen un manejo sofisticado de eventos del lado del cliente para garantizar una experiencia de usuario fluida y segura. Saber manejar bien los eventos nos prepara para hacer programas que no solo funcionen bien, sino que también sean fáciles y agradables de usar. Por eso, a continuación, se mostrarán 3 ejemplos diferentes en distintos lenguajes de programación para comprender mejor cómo funcionan los eventos.

Ejemplo 1

Manejo del Evento de Clic en un Botón con JavaScript

```
<!-- HTML para capturar clics en un botón -->
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Evento de Clic en Botón</title>
    <script>
        // Función que se ejecutará cuando el botón sea clickeado
        function botonClickeado() {
            alert("¡Gracias por clickear el botón!");
        }
    </script>
</head>
<body>
<!-- Botón que, al ser clickeado, llama a la función botonClickeado -->
<button onclick="botonClickeado()">Haz clic en mí</button>
</body>
</html>
```

Este ejemplo muestra cómo se maneja un evento de clic en JavaScript. Cuando el usuario clickea el botón en la página, se llama a la función botonClickeado() y se muestra una alerta. Es una introducción sencilla al manejo de eventos con JavaScript.

Figura 14.
Vista de validación del botón Clickeado.

Ejemplo 2

Validación de Entrada de Texto con TypeScript

```
// Este script de TypeScript ayudará a comprender cómo se puede validar la
entrada de texto de un usuario.
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Validación de Entrada de Texto con TypeScript</title>
</head>
```

```
<body>
    <!-- Input de texto donde el usuario puede escribir -->
    <input type="text" id="campoTexto" placeholder="Escribe algo aquí">
    <!-- Botón para ejecutar la validación -->
    <button id="botonValidar">Validar Entrada</button>
    <script>
// Se define una función que se llamará cada vez que haya una entrada en el
campo de texto.
function manejarEntradaTexto() {
    // Aquí, 'evento.target' se refiere al elemento HTML que disparó el evento,
que en este caso es el campo de texto.
    // Se obtiene el valor del campo de texto
    const valor = document.getElementById('campoTexto').value;
    // Se realiza una validación simple: si el campo está vacío, se imprime un
mensaje.
    if (valor.trim() === '') {
        console.log('El campo de texto no debe estar vacío.');
    } else {
        console.log(`Entrada válida: ${valor}`);
    }
}
// Escucha el evento click en el botón y llama a manejarEntradaTexto cuando
se hace clic
            document.getElementById('botonValidar').addEventListener('click',
manejarEntradaTexto);
    </script>
</body>
</html>
```

Nota. TypeScript añade tipos a JavaScript, permitiendo una mayor seguridad en el manejo de los eventos. Aquí se valida que la entrada de texto no esté vacía cada vez que el usuario escribe algo. Fuente: elaboración propia.

Figura 15.
Vista de validación de entrada.

Ejemplo 3.

Captura de Eventos de Teclado en un Campo de Entrada con React

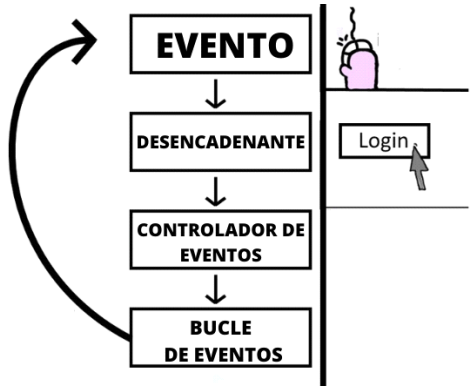
```
// Se define un componente de React llamado 'CampoEntrada'.
class CampoEntrada extends React.Component {
  // Este es un método que se activará cada vez que el usuario presione una tecla.
  manejarKeyPress = (evento) => {
      // El evento contiene una propiedad 'key' que representa la tecla presionada.
    console.log(`Tecla presionada: ${evento.key}`);
  };

  // La función 'render' devuelve el JSX que se mostrará en la página.
  render() {
    return
      // 'onKeyPress' es una prop de React que escucha el evento de teclado.
       // Cuando se presiona una tecla en el campo de entrada, se llama al método 'manejarKeyPress'.
            <input type="text" onKeyPress={this.manejarKeyPress} placeholder="Escribe algo..."/>;
  }
}
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<CampoEntrada/>);
```

Nota. Este código de React muestra cómo se pueden manejar los eventos de teclado en un campo de entrada. Cada vez que el usuario presiona una tecla, el evento se captura y se registra en la consola. Fuente: elaboración propia.

Figura 16.
Vista de eventos del teclado.

Nota. Presenta el ejemplo de cómo se visualiza en el teclado. Fuente: elaboración propia.

Figura 17.
Flujo Básico de la Programación Orientada a Eventos.



Nota. La imagen esquematiza el ciclo de vida de un evento en programación, desde su inicio por un desencadenante, como un clic de usuario, pasando por la gestión a través de un controlador de eventos, hasta su procesamiento continuo dentro de un bucle de eventos. Fuente: García (2014)

A continuación, se destacan tres recursos en línea que ofrecen un conocimiento valioso sobre este tema. FreeCodeCamp puede ayudar a los principiantes o a aquellos que recién comienzan o perfeccionan sus conocimientos en esta área. Este portal está diseñado para todas las ganas de saber algo nuevo en el campo del aprendizaje en línea en desarrollo y programación web, ofreciendo miles de ejercicios y programas de codificación gratuitos. Aunque esta página en sí misma es un poco extensa, el enfoque en la enseñanza práctica con el uso de ejercicios y proyectos la convierte en una fuente ideal para aprender sobre eventos en programación web. A través de los módulos y proyectos propuestos se puede adquirir experiencia real en el trabajo con eventos y otras técnicas de programación web (freeCodeCamp, 2021). Por otro lado, Esdocu viene con tutoriales prácticos sobre cómo manejar eventos usando su navegador JavaScript. Muestra cómo manejar eventos simples, por ejemplo, clics del ratón o toques de botones, como detener la propagación del evento y evitar su comportamiento predeterminado. Solo describe el funcionamiento interno de eventos en JavaScript y su interacción con el DOM (Esdocu, 2023).

DesarrolloWeb.com es una plataforma de referencia para el desarrollador web, donde se presentan numerosos tutoriales, artículos y guías sobre diversos aspectos de la programación web, entre ellos JavaScript y su manipulación de eventos. Una lista exhaustiva de todos los tipos de eventos disponibles para usar en JavaScript, desde los básicos como hacer clic, enfocar y desenfocar hasta tipos más complejos como los que se ocupan del movimiento del mouse y, finalmente, eventos del teclado. Luego, continúa describiendo en detalle cómo se pueden asignar controladores de eventos a elementos HTML mediante el uso de atributos como "onclick" o "onkeyup". Esto es aún más crítico para un desarrollador que debe haber entendido claramente los eventos en JavaScript (Alvarez, 2021).

2.4.6 Validación de entrada de datos

En el desarrollo de aplicaciones web y móviles, la validación de datos surge como un proceso esencial para garantizar la integridad y la seguridad de la información gestionada. Se aplica una confirmación estricta para verificar la corrección, especialmente cuando los usuarios finales insertan datos a través de formularios en línea y aplicaciones móviles, antes de su procesamiento o almacenamiento. Esta es una acción preventiva esencial para evitar las trampas técnicas y proteger el sistema de explotaciones no deseadas que apuntan a las vulnerabilidades. Estudios recientes, como el de Hanna & Jaber (2019) destacan la importancia de automatizar la generación de datos de prueba basada en el análisis de campos de entrada de usuario, proponiendo un enfoque que reduce significativamente el tiempo y esfuerzo requeridos en esta tarea.

En el contexto web, frameworks como Angular y React cambian la forma tradicional de usar la validación del lado del cliente para permitir a los desarrolladores construir de manera eficiente verificaciones complejas en los datos. Estos también tienen como objetivo enriquecer su interacción con el usuario mediante retroalimentación instantánea. Estos frameworks facilitan las implementaciones de validaciones que son seguras y enriquecen su interacción con el usuario a través de retroalimentación instantánea. En este sentido, los métodos de validación en tiempo real funcionan mientras un usuario interactúa con una aplicación; su eficacia ha demostrado integrarse con el estudio de Rivera et al. (2016) , que se utilizó como un ejemplo útil en el contexto del desarrollo de JavaScript del lado del cliente. Este enfoque de prevención debe ser una preocupación de primeros principios para la integridad y seguridad de las aplicaciones que enfrentan a los usuarios. La validación de datos pasa de ser una actividad técnica simple a un requisito legal con la emisión de regulaciones como el GDPR y el CCPA. En tal campo controlado, la falta de validación adecuada puede resultar en consecuencias legales y daño a la reputación de la entidad responsable.

Desde esta perspectiva, se considera: "Es imperativo que el proceso de validación de datos se realice con un entendimiento profundo de las posibles fuentes de entrada y sus variaciones. Este enfoque meticuloso no solo mitiga los riesgos de seguridad, sino que también optimiza la experiencia del usuario al minimizar los errores de entrada" (Candel, 2019). El papel de la validación de datos en la mejora de la seguridad y la suave interacción del usuario es crítico, señalando su importancia en la prevención de la probabilidad de fallas de seguridad y estableciendo una fuente de experiencia de usuario consistente y satisfactoria, sin embargo también se debe tener en cuenta la facilidad con la que el usuario pueda interactuar, por tanto es muy importante que el usuario tenga guías para poder ingresar datos correctos como se observa en la figura 2.8.

La validación de datos no es estática en el ciclo de vida del desarrollo de software; por el contrario, es un proceso dinámico que necesita cuidado continuo y adaptación contra el progreso de las tecnologías y la aparición de nuevas amenazas, como lo representa la investigación de Rustambek (2023) en el ámbito de la programación web resalta cómo la validación de entradas constituye una barrera crítica contra diversas amenazas de seguridad, reforzando la necesidad de prácticas de desarrollo seguras y consideradas. Por lo tanto, las aplicaciones necesitan tener un entendimiento profundo y actualizado de la validación de datos que mantendrá su presencia a lo largo del tiempo, repelerá los ataques y mantendrá la confianza de los usuarios en un ecosistema digital en continuo cambio y crecimiento. A continuación, se integran 3 ejemplos sencillos para entender mejor la validación de datos:

Ejemplo 1

```
Verificación de Formato de Número Telefónico con JavaScript
// Función para validar la estructura de un número telefónico
function validarTelefono(numero) {
    // Se define una expresión regular que representa un número telefónico
de 10 dígitos
   var patronTelefono = /^\d{10}$/;

    // La función 'test' se utiliza para verificar si el número cumple con el
patrón
   if(patronTelefono.test(numero)) {
       console.log("El número de teléfono es válido.");
   } else {
          console.log("El número de teléfono no es válido. Se requieren 10
dígitos numéricos.");
   }
}

// Se aplica la función de validación a un ejemplo de número de teléfono
validarTelefono("3105604749"); // Este número es válido
validarTelefono("31452"); // Este número es inválido
```

Este código usa una expresión regular para verificar si un número telefónico ingresado tiene 10 dígitos, que es un formato común en muchos países. La función 'test' devuelve true si el número es válido o false si no lo es, y se muestra un mensaje correspondiente en la consola.

Figura 18.
Vista de formato de número telefónico.

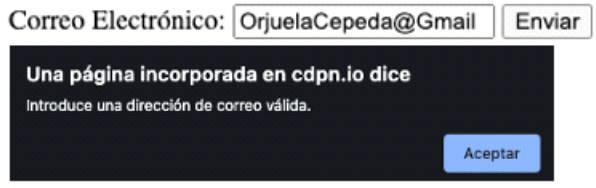
Ejemplo 2.

Validación de Dirección de Email en un Formulario Web

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Validación de Email</title>
    <script>
        // Función para validar el email al enviar el formulario.
        function validarEmail() {
            // Obtenemos el valor ingresado en el campo de email.
            var email = document.getElementById('email').value;

            // Patrón que define la estructura básica de un email.
            var patronEmail = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
            // Comprobamos si el email cumple con la estructura definida.
            if(!patronEmail.test(email)) {
                alert("Introduce una dirección de correo válida.");
                return false; // Evitamos que el formulario se envíe si el email no es
válido.
            }
            return true; // El formulario es válido y se puede enviar.
        }
    </script>
</head>
<body>
    <form onsubmit="return validarEmail();">
        <label for="email">Correo Electrónico:</label>
        <input type="email" id="email" required>
        <input type="submit" value="Enviar">
    </form>
</body>
</html>
```

Nota. El formulario HTML incluye una validación en JavaScript que se activa al enviar. Si el correo electrónico ingresado no se ajusta al patrón definido en la expresión regular, se muestra una alerta y se detiene el envío. Fuente: elaboración propia.

Figura 19.
Vista de validación de Correo Electrónico.



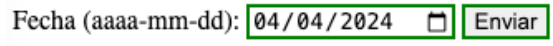
Ejemplo 3

Uso de CSS para Indicar la Validación de Campos de Formulario

```
<!-- HTML con CSS para aplicar estilos a campos de formulario basados en su validación -->
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Validación Visual de Campos</title>
    <style>
            /* Aplicamos estilos a campos válidos e inválidos usando pseudoclases de CSS */
        input:invalid {
            border: 2px solid red; /* Rojo indica campo inválido */
        }
        input:valid {
            border: 2px solid green; /* Verde indica campo válido */
        }
    </style>
</head>
<body>
    <form>
        <label for="fecha">Fecha (aaaa-mm-dd):</label>
        <input type="date" id="fecha" name="fecha" required>
        <input type="submit" value="Enviar">
    </form>
</body>
</html>
```

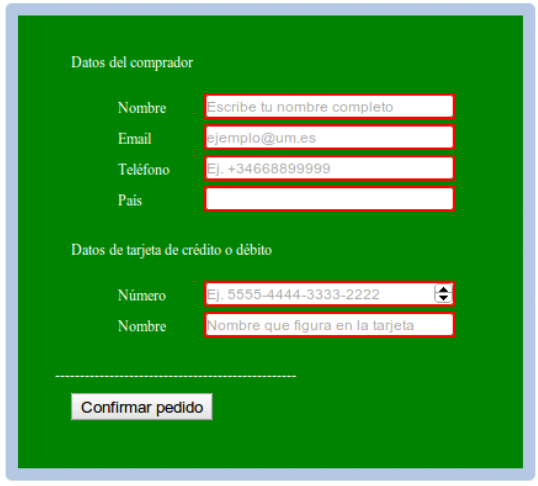
Nota. Este formulario utiliza estilos CSS para brindar una respuesta visual inmediata al estado de validación de los campos de entrada. La pseudoclase :invalid se utiliza para campos que no cumplen con los criterios, mientras que valida se aplica a aquellos que sí lo hacen. Este método no valida los datos por sí mismo, pero mejora la retroalimentación al usuario durante la entrada de datos. Fuente: elaboración propia.

Figura 20.
Vista de la validación de campos.



Nota. Se presenta una validación de los campos de fecha, lo anterior porque el usuario digito mal los valores del campo. Fuente: elaboración propia.

Figura 21.
Validación de entrada de datos.



Nota. La imagen muestra una interfaz de formulario web donde se solicita al usuario ingresar información personal y de pago. Se destacan campos que deben ser validados, como nombre, email, teléfono y detalles de tarjeta de crédito, para asegurar que los datos ingresados sean correctos antes de confirmar el pedido. Fuente: Menéndez & Asensio (2018).

Tras la exposición de ejemplos prácticos sobre la validación de entrada de datos utilizando diferentes tecnologías, es esencial contar con recursos educativos que profundicen en este tema y ofrezcan herramientas adicionales para los desarrolladores, entre los recursos más conocidos están los sitios web como Coder Champ que se presenta como una fuente integral, abarcando desde los fundamentos de los formularios HTML hasta las complejidades de las validaciones del lado del cliente. La página proporciona una guía efectiva en la creación de un formulario, tocando el uso efectivo de tipos de campos de entrada proporcionados en HTML5, y atributos como required y pattern para proporcionar validaciones primarias. Además, subraya la posición de JavaScript en la realización de verificaciones de datos más avanzadas, adaptadas a algunos requisitos de proyectos seleccionados y específicos, con el fin de mejorar la experiencia del usuario mediante la provisión de retroalimentación rápida y relevante (CoderChamp, 2023).

JavaScript.info, reconocido por su pedagogía clara y aplicable, ofrece una cobertura exhaustiva sobre la gestión de formularios y sus validaciones usando JavaScript. Es un recurso ideal para principiantes y también para desarrolladores con más experiencia que buscan fortalecer sus habilidades,

ofreciendo desde ejercicios básicos de validación hasta desafíos que abarcan tipos de datos y formatos más diversificados(Kantor, 2024). Finalmente, 'Tecnologías de la Información' proporciona un extenso material educativo acerca de la validación de datos. Este recurso desglosa el concepto de validación de datos, explorando distintos enfoques y técnicas utilizadas para verificar la integridad y la validez de los datos ingresados en aplicaciones y componentes de software. Además, el sitio instruye sobre métodos específicos para la validación en bases de datos, desde la verificación de rangos hasta la integridad referencial, que es de suma importancia en bases de datos relacionales. La información contenida en este sitio web es crucial para los desarrolladores que buscan comprender y aplicar prácticas rigurosas de validación en su trabajo (Tecnologías Información, 2017).

Estos recursos en línea proporcionan a los desarrolladores una referencia rápida a las habilidades que necesitan para la implementación de estrategias eficientes y seguras de validación de datos enfocadas en preservar la integridad de los datos frente a entradas erróneas o maliciosas.

2.4.7 Consideraciones en el navegador

La programación web del lado del cliente incluye el manejo adecuado y la comprensión de las consideraciones en los navegadores, abordando problemas como la compatibilidad entre diferentes navegadores y ofreciendo una experiencia de usuario consistente y accesible. Superar los problemas con el uso de estándares, como ECMAScript, e interoperabilidad es un factor que debe tenerse en cuenta para superar los problemas de fragmentación y, por lo tanto, tener una web más cohesiva unificada (Aires et al., 2017). Por otro lado, la seguridad también es un asunto de gran preocupación, considerando que los ataques, por ejemplo, inyecciones SQL y XSS, tienden a amenazar la integridad de las aplicaciones web y la privacidad de sus usuarios (Ramallo, 2020). Frente a estos retos, la actualización constante de herramientas y frameworks contemporáneos, como jQuery, Angular y React, ha demostrado ser efectiva, ya que ofrecieron estructuras que permitieron a los desarrolladores abordar, de manera ordenada y sistemática, la complejidad que implica el desarrollo web. Estas mejoras tecnológicas no solo han facilitado el desarrollo de aplicaciones web complejas, sino que también han facilitado la gestión y control de las diversidades en el entorno del cliente y de ejecución, mejorando así la seguridad y la experiencia del usuario (Aires et al., 2017). Estas herramientas han impulsado las mejores prácticas y asegurado patrones de diseño estandarizados para asegurar que las aplicaciones web sean atractivas y accesibles para los usuarios finales. A continuación, se integran 3 ejemplos sencillos para entender mejor las consideraciones en el navegador:

Ejemplo 1

Verificación de soporte de Flexbox en el navegador

```
function verificaSoporteFlexbox() {
 // Crear un elemento div para probar el soporte de Flexbox
 var elemento = document.createElement('div');
 elemento.style.display = 'flex';
 // Comprobar si el estilo 'flex' se aplica correctamente
 if (elemento.style.display === 'flex') {
  // Si el estilo se mantiene, el navegador soporta Flexbox
  console.log("Este navegador soporta Flexbox.");
 } else {
  // Si el estilo no se mantiene, el navegador no soporta Flexbox
  console.log("Flexbox no es soportado.");
 }
}
// Llamada a la función para realizar la verificación
verificaSoporteFlexbox();
```

Este código implementa una técnica básica para determinar si el navegador del usuario soporta Flexbox, creando un elemento div y asignándole display: flex. Si este estilo se mantiene, indica compatibilidad con Flexbox.

Ejemplo 2

Gestión de Cookies con JavaScript

```
// Esta función crea una cookie llamada 'usuario' con un valor de ejemplo y
un tiempo de expiración de 7 días.
document.cookie = "usuario=MichaelOrjuela; max-age=" +
7*24*60*60;
// Aquí dividimos todas las cookies disponibles en un arreglo usando el
punto y coma como separador
var cookies = document.cookie.split(';');
// Luego buscamos una cookie específica por su nombre y mostramos su
valor si es encontrada
var cookieUsuario=cookies.find(cookie=>
cookie.trim().startsWith("usuario="));
if (cookieUsuario) {
      console.log('Cookie de usuario encontrada:',
cookieUsuario.split('=')[1]);
}
```

Este código demuestra cómo almacenar y recuperar datos de las cookies, que es un método común de conservar información del lado del cliente como las preferencias del usuario y datos de sesión. Código tomado de Il. Kantor (2024)

Ejemplo 3

Uso de Características de ES6

```
    // Funciones de flecha de ES6 para una sintaxis más concisa en las
    funciones.
    // Aquí se incrementa cada número en un arreglo por 1.
    [1, 2, 3].map(n => n + 1);

// ES6 introdujo 'let' y 'const' para una mejor gestión del ámbito de las
variables.
// 'let' se puede reasignar, mientras que 'const' es para valores constantes.
let x = 10;
const y = 20;

// Nuevos métodos de cadenas de texto de ES6 como 'startsWith' y 'endsWith'
// para facilitar la búsqueda de patrones en cadenas de texto.
let s = 'Hello World!';
console.log(s.startsWith('Hello')); // Verifica si la cadena comienza con 'Hello'
console.log(s.endsWith('!')); // Verifica si la cadena termina con '!'
```

En este código se ilustra el uso de algunas de las características introducidas en ECMAScript 6 para mejorar la legibilidad y funcionalidad del código JavaScript. Es importante asegurarse de que estas características sean compatibles con el navegador en el que se ejecutará la aplicación. Para practicar y comprender mejor las "*Consideraciones en el navegado*r", es recomendable explorar sitios web que ofrezcan guías y herramientas para mejorar la experiencia de desarrollo y de usuario en la programación web del lado del cliente. El sitio web Splendidus ofrece información sobre el uso de Modernizr para detectar funcionalidades del navegador y adaptar el código según la compatibilidad encontrada, lo que es esencial para asegurar una experiencia de usuario coherente en diferentes navegadores (Torres, 2019).

WebTutor ofrece una plataforma educativa completa, con tutoriales sobre lenguajes y herramientas de programación del lado del cliente como HTML, CSS y JavaScript, además de proporcionar un compilador en línea para que los desarrolladores practiquen y vean los resultados de su código en tiempo real, reforzando así sus habilidades prácticas y teóricas(WebTutor, 2023). Finalmente, web.dev se presenta como un recurso indispensable creado por especialistas y miembros del equipo de Chrome, que ofrece cursos sobre diseño y desarrollo web. Los temas abordados incluyen desde la privacidad y la accesibilidad hasta el diseño y manejo de imágenes responsivas, proporcionando una formación integral necesaria para navegar por el panorama actual de la web (Kinlan & Walton, 2023).

2.5. Actividades de Evaluación

A continuación, se detallan dos ejercicios prácticos: el desarrollo de un sistema de reservación de cine y la creación de un formulario de perfil, poniendo en práctica habilidades en HTML, CSS, y JavaScript.

2.5.1 Ejercicio sobre Programación Orientada a Objetos:

Objetivo: Desarrollar una interfaz web para un sistema de reservación de asientos de cine que permita a los usuarios elegir una película, seleccionar asientos y confirmar su reserva, utilizando HTML, CSS y JavaScript.

Requerimientos:

- **Conocimientos Básicos en:** Uso de HTML para la estructura de las páginas, CSS para definir el estilo y JavaScript para habilitar la interactividad, incluida la lógica de reserva.
- **Herramientas Necesarias:** Un editor de código como Visual Studio Code y un navegador web para probar la funcionalidad.

Funcionalidades Esperadas:

- Diseñar un selector HTML que permita al usuario final elegir entre diferentes películas y horarios disponibles, de forma que cada una de sus elecciones se asocie a una sala y asientos.
- Utilizar JavaScript para crear una representación visual de los asientos disponibles en la sala seleccionada. Debe permitir a los usuarios seleccionar y deseleccionar asientos haciendo clic sobre ellos, y Los asientos ya reservados no deben ser seleccionables.
- Implementar un botón "Confirmar Reserva" que recolecte y muestre un resumen de los asientos seleccionados, la película y el horario elegido, y el precio total de la reserva. Al confirmar, debe actualizarse la disponibilidad de los asientos para reflejar la reserva hecha.
- Aplicar estilos con CSS para mejorar la apariencia del formulario, la representación de los asientos y el resumen de la reserva.

2.5.2 Ejercicio sobre Programación Orientada a Eventos:

Objetivo: Utilizar HTML, CSS y JavaScript para crear un componente interactivo en una página web que permita a los usuarios ingresar y visualizar información de perfil personal, incluyendo nombre, correo electrónico y una breve descripción.

Requerimientos:

- **Conocimientos Básicos en:** HTML para la estructura, CSS para el estilo, y JavaScript para la interacción del usuario.
- **Herramientas Necesarias:** Un editor de código como Visual Studio Code y un navegador web para probar la funcionalidad.

Funcionalidades Esperadas:

- Diseñar un formulario HTML para guardar el nombre del usuario,

correo electrónico y descripción.
- Utilizar JavaScript para recolectar los datos ingresados y mostrarlos dinámicamente en otra sección de la página una vez que el usuario presione un botón de "Guardar Perfil".
- Aplicar estilos básicos con CSS para mejorar la apariencia del formulario y la visualización del perfil.

Referencias Bibliográficas

Aires, B., Bogotá, , México, , Santiago De Chile, D. F. , Gutiérrez González, Á., Luis, J., & Goytia, L. (2017). Desarrollo y programación en entornos web. http://www.alfaomega.com.mx

Alonso, R., & Adolfo, G. (2018). Uso del M-learning para potenciar el aprendizaje de los lenguajes de programación web del lado del cliente. https://repository.unad.edu.co/handle/10596/17692

Alvarez, M. (2021a). Los tipos de eventos en Javascript. https://desarrolloweb.com/articulos/1236.php

Alvarez, M. (2021b). Los tipos de eventos en Javascript. https://desarrolloweb.com/articulos/1236.php

Angular University.io. (2016). Angular University. https://angular-university.io/

Atwood, J., & Spolsky, J. (2008). Stack Overflow Developer Survey 2023. https://survey.stackoverflow.co/2023/#stack-overflow-site-use-newso-sites-learn

Azaustre. (2014). Desarrollo_web_agil.

Candel, J. M. O. (2019). Seguridad en Aplicaciones Web Java. 430. https://www.researchgate.net/publication/338824173_Seguridad_en_aplicaciones_Web_Java

Celi, M. (2023). Vista de Desarrollo de Aplicaciones Web Utilizando Vue.Js, Axios y PHP. https://revista.gnerando.org/revista/index.php/RCMG/article/view/94/86

Clark. (2021). Top 10 Client-Side Development Languages. https://blog.back4app.com/client-side-development-languages/

CoderChamp. (2023). Coder Champ - Web Development, Social Media Tips & Digital Marketing. https://www.coderchamp.com/

Eguiluz, J. (2024). Diseño y programación web (libros, tutoriales y vídeos sobre HTML, CSS, JavaScript, PHP). https://uniwebsidad.com/?from=librosweb

Esdocu. (2023). Eventos del navegador con JavaScript – Esdocu Cursos y Tutoriales. https://esdocu.dev/javascript/eventos-del-navegador

Fedosejev. (2015). React.js Essentials - Artemij Fedosejev - Google Libros. https://books.google.com.co/books?hl=es&lr=&id=Rhl1CgAAQBAJ&oi=fnd&pg=PP1&dq=react+framework&ots=JlsrmzwPTE&sig=fPu-2ZVXVP1SrGtiCLZ0s-

PMPNQ&redir_esc=y#v=onepage&q=react%20framework&f=false
freeCodeCamp. (2021). Aprende a programar gratis: Cursos de programación para gente ocupada. https://www.freecodecamp.org/espanol/
García, M. (2014). Programación orientada a eventos: características, ejemplos, ventajas, aplicaciones. https://www.lifeder.com/programacion-orientada-a-eventos/
González, A. H. (2021). Características más relevantes de la programación orientada a objetos (POO). http://sedici.unlp.edu.ar/handle/10915/119244
González, R., & Quesada González, A. (2015). PROYECTO FIN DE CARRERA.
Graciela, S., Ibarra, P., Quispe, R., Mullicundo, F. F., Lamas, D. A., & Presente, L. (2021). Herramientas y tecnologías para el desarrollo web desde el FrontEnd al Back-end. XXIII Workshop de Investigadores En Ciencias de La Computación (WICC 2021, Chilecito, La Rioja), August 2021, 963–968. http://sedici.unlp.edu.ar/handle/10915/120476
Guérin, B.-Arnaud. (2018). ASP.NET con C# en Visual Studio 2017 : diseño y desarrollo de aplicaciones web.
Hanna, S., & Jaber, H. (2019). An Approach for Web Applications Test Data Generation Based on Analyzing Client Side User Input Fields XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE An Approach for Web Applications Test Data Generation Based on Analyzing Client Side User Input Fields. https://doi.org/10.1109/ICTCS.2019.8923098
Izquierdo, J. (2022). Cómo instalar Angular en Windows: una guía sobre Angular CLI, Node.js y Herramientas de compilación. https://www.freecodecamp.org/espanol/news/como-instalar-angular-en-windows/
Kantor, I. (2024). The Modern JavaScript Tutorial. https://javascript.info/
Kantor, Il. (2024). Cookies, document.cookie. https://javascript.info/cookie
Kinlan, P., & Walton, P. (2023). web.dev. https://web.dev/learn?hl=es-419
Mazon, O. (2015). 46FUNDAMENTOSDEPROGRAMACIONORIENTADASAOBJETOS DEJAVA.
MDN. (2023a). HTML: Lenguaje de etiquetas de hipertexto | MDN. https://developer.mozilla.org/es/docs/Web/HTML
MDN. (2023b). Introducción a los objetos JavaScript - Aprende desarrollo web | MDN. https://developer.mozilla.org/es/docs/Learn/JavaScript/Objects
MDN. (2024a). JavaScript reference - JavaScript | MDN. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference
MDN. (2024b). MDN Web Docs. https://developer.mozilla.org/en-US/
Menéndez, R., & Asensio, B. (2018). Verificación de formularios. DAWEB. Rafael Menéndez-Barzanallana Asensio. Universidad de Murcia (DIS). https://www.um.es/docencia/barzana/DAWEB/Desarrollo-de-aplicaciones-web-teoria-formularios-ejemplo-1.html

Miu, A., Ferreira, F., Yoshida, N., & Zhou, F. (2020). Communication-Safe Web Programming in TypeScript with Routed Multiparty Session Types; Communication-Safe Web Programming in TypeScript with Routed Multiparty Session Types. 27. https://doi.org/10.1145/3446804.3446854

Morocho, D. (2023). Universidad Técnica de Babahoyo Facultad de Administración, Finanzas e Informática proceso de titulación periodo diciembre 2022-mayo 2023 examen complexivo de grado o de fin de carrera prueba práctica previo a la obtención del título de ingeniería en sistemas tema: estudio comparativo entre la programación orientada a objetos y la programación orientada a aspectos.

Pandya, S. (2024). Software Development Frameworks: Overview, Benefits, Key Considerations. https://polyxer.com/blogs/software-development-frameworks/

Ramallo, J. M. (2020). Consideraciones de seguridad para la web en la actualidad. https://www.academia.edu/44701879/Consideraciones_de_seguridad_para_la_web_en_la_actualidad

reactjs.wiki. (2023). React.js Wiki - Preguntas típicas con respuesta y solución. https://www.reactjs.wiki/

Rivera, R. M., Cámara, F. A., Jiménez, D. E., & Díaz, S. H. (2016). Sisdam: aplicación web para el procesamiento de datos según un diseño aumentado modificado sisdam: Web application for processing data according to a Modified Augmented Design. Cultivos Tropicales, 37(3), 153–164. https://doi.org/10.13140/RG.2.1.4550.4243

Rodríguez, F., & Botelho, A. (2012). Historia de HTML | KeepCoding Bootcamps. https://keepcoding.io/blog/historia-de-html/

Roman, J. (2017). Primeros pasos con VueJS - Javascript en español - Lenguaje JS. https://lenguajejs.com/vuejs/introduccion/primeros-pasos/

Rustambek, M. (2023). The Role Of Web Programming In Today's Programming World. In www.wsrjournal.com. www.wsrjournal.com

Salvador, M., & Díaz, G. (2022). Análisis e implicaciones de la implementación del Mapeo Relacional de Objetos en la Programación Orientada a Objetos. https://www.researchgate.net/publication/364331117

Sims, Z., & Bubinski, R. (2011). Learn to Code - for Free | Codecademy. https://www.codecademy.com/

Sist Nelly Karina Esparza Cruz, I. (2019). Programación Orientada a Objetos para principiantes. Portal de Libros Universidad Técnica de Babahoyo.

Tecnologias Información. (2017). Validación de Datos: Definición, elementos y métodos. https://www.tecnologias-informacion.com/validacion.html

Torres, H. (2019). Modernizr - Deteccion de compatibilidad de navegadores | Splendidus. https://htorrespo.github.io/blog/javascript/2019/08/06/web-modernizr-

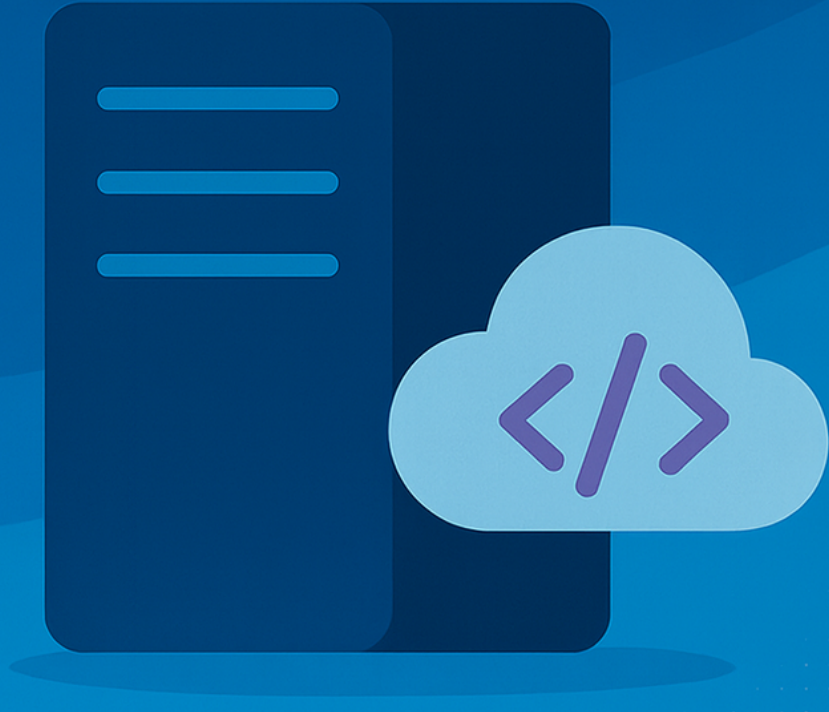
introduction.html
W3Schools. (1999). W3Schools Online Web Tutorials. https://www.w3schools.com/
Walker, J. (2016). React. https://es.react.dev/
Walker, J. D., & Chapra, S. C. (2014). A client-side web application for interactive environmental simulation modeling. Environmental Modelling and Software, 55, 49–60. https://doi.org/10.1016/J.ENVSOFT.2014.01.023
WebTutor. (2023). HTML Tutorials, HTML Introduction, Online HTML Web Tutor. https://webtutor.dev/html/html-introduction
You, E. (2022). Instalación — Vue.js. https://es.vuejs.org/v2/guide/installation

CAPÍTULO 3.

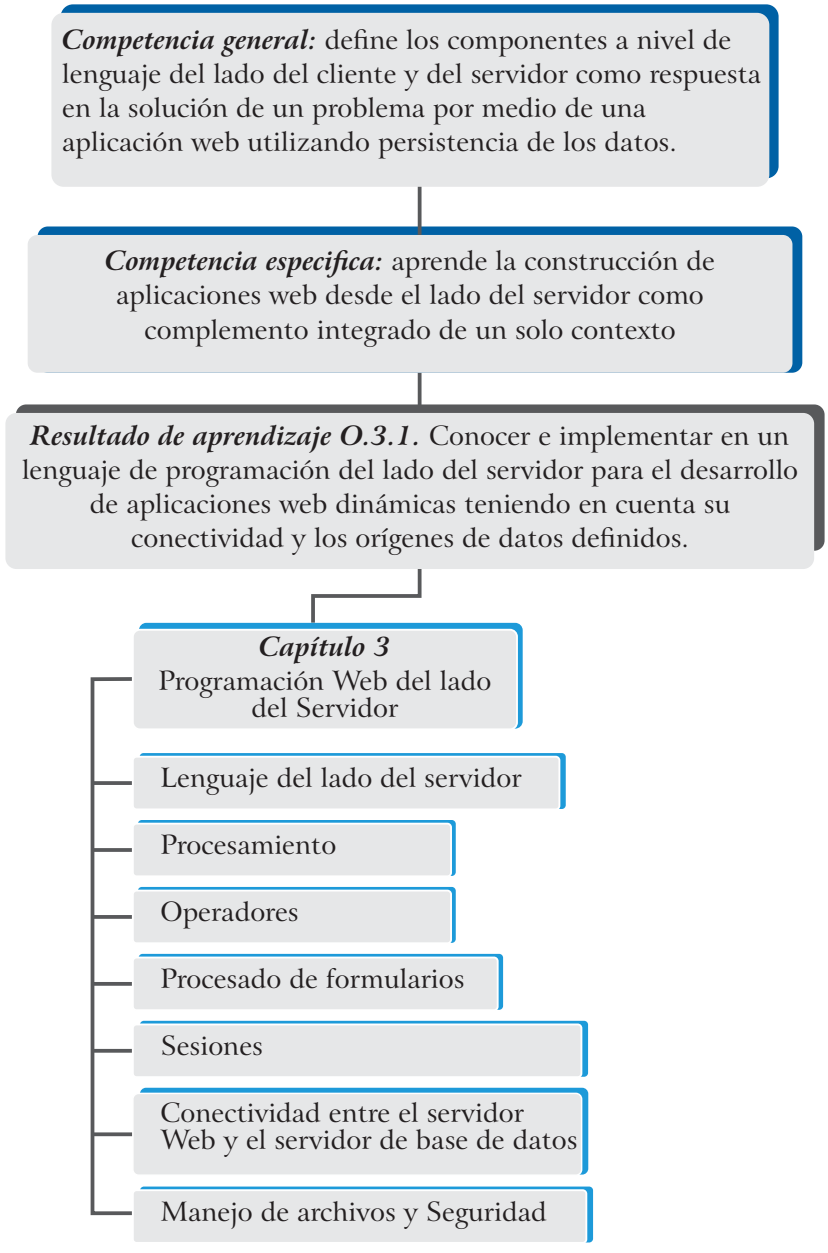
PROGRAMACIÓN WEB DEL LADO DEL SERVIDOR

Sebastián Ariza González
Edwin Eduardo Millán Rojas
Denis Lorena Álvarez Guayara

[1]*Estudiante del programa de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad de la Amazonia, correo: se.ariza@udla.edu.co*
[2]*Docente de tiempo completo del programa de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad de la Amazonia, correo: e.millan@udla.edu.co*
[3]*Docente de tiempo completo del programa de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad de la Amazonia, correo: d.alvarez@udla.edu.co*



ESQUEMA | CAPÍTULO 3.



COMPETENCIA Y RESULTADO DE APRENDIZAJE | 3. CAPÍTULO

La competencia general está establecida de la siguiente forma: "Define los componentes a nivel de lenguaje del lado del cliente y del servidor como respuesta en la solución de un problema por medio de una aplicación web utilizando persistencia de los datos.". Asociada a la siguiente competencia específica para la cual se desarrolla en el capítulo tres, definida como: "Aprende la construcción de aplicaciones web desde el lado del servidor como complemento integrado de un solo contexto". El resultado de aprendizaje asociado a la competencia del capítulo tres, se estableció de la siguiente forma O3.1. Conocer e implementar en un lenguaje de programación del lado del servidor para el desarrollo de aplicaciones web dinámicas teniendo en cuenta su conectividad y los orígenes de datos definidos. Los niveles de desempeño definidos para el resultado de aprendizaje son expuestos en la tabla 3.

Tabla 3.
Niveles de desempeños del resultado de aprendizaje O3.1

Avanzado (5-4,8)	Intermedio (4,7-4)	Básico (3.9 -3)	Bajo (2.9-0)
Desarrolla la aplicación del lado del servidor con la implementación de componentes eficientes que dan respuesta en su operatividad e integridad del sistema.	Desarrolla la aplicación del lado del servidor con la implementación de componentes no optimizados que den respuesta eficiente en su operatividad y usabilidad.	Desarrolla la aplicación del lado del servidor con la implementación de componentes, pero no evidencia el dominio teórico que optimice la operatividad y usabilidad del sistema.	Presenta dificultad en el desarrollo de la aplicación del lado del servidor con la implementación de elementos y componentes efectivos que den respuesta en su operatividad e integridad del sistema.

Nota. *Trabajo realizado a partir del desarrollo de resultados de aprendizaje. Fuente: elaboración propia.*

Las actividades de evaluación determinadas para el resultado de aprendizaje son las siguientes: entrega de un proyecto relacionado con un sistema de información web utilizando los conceptos teóricos y prácticos del presente capítulo (Código fuente y sustentación).

1.1. Métodos y Materiales de Aprendizaje

Los estudiantes estarán expuestos a entornos de aprendizaje donde los enfoques teóricos se mezclan adecuadamente con aplicaciones prácticas. Las lecturas requeridas, los recursos en línea como tutoriales y documentación oficial, y los artículos relacionados serán parte de la lectura obligatoria. Los contenidos cubrirán las áreas principales: principios de la programación del

lado del servidor, uso de lenguajes específicos como PHP, Python (con Django), y JavaScript (con Node.js), y la gestión de bases de datos; las mejores prácticas en seguridad web.

Se dará un ejemplo práctico con ejercicios a los estudiantes de manera que apliquen lo que han aprendido a través de los proyectos a realizar. Estos incluyen actividades prácticas sobre cómo crear y gestionar formularios web, configurar y operar una base de datos con los requisitos actuales para implantar seguridad en una aplicación web, y eventualmente desarrollar una aplicación web completa capaz de incorporar todas las funcionalidades del lado del servidor.

Se ofrece a los estudiantes el uso de herramientas y tecnologías recientes, incluidos servidores web para pruebas locales, y el marco de trabajo para Django y Express. Esto se basará en la calidad y la completitud de las prácticas ejercitadas y exitosamente implementadas a través de un proyecto final, donde el estudiante será capaz de demostrar las mejores prácticas de desarrollo, eficiencia en la codificación, y la habilidad para integrar funcionalidades complejas en una aplicación web dinámica.

Este enfoque de aprendizaje está diseñado para preparar a los estudiantes no solo para comprender la teoría detrás de la programación del lado del servidor, sino también para aplicar ese conocimiento en el desarrollo de soluciones web reales, equipándolo con las habilidades necesarias para contribuir efectivamente en entornos profesionales de desarrollo web.

1.2. Desarrollo Temático del Capítulo

Este capítulo, "Programación Web del lado del Servidor", se desarrolla en torno al enfoque temático de las técnicas y herramientas esenciales requeridas por los desarrolladores, creando aplicaciones web dinámicas y seguras. Esto contribuye al estudiante en la investigación de la programación básica, conceptos avanzados y lenguajes, incluyendo PHP, Python con Django, y JavaScript con Node.js. Describe a los estudiantes las maneras de gestionar formularios, sesiones y la seguridad en las aplicaciones web. Se hace énfasis en la conectividad con las Bases de datos. Entre algunos de los problemas más críticos en el desarrollo web, el aspecto de la seguridad será enfatizado. Se cubrirá exhaustivamente para hacer que los estudiantes estén seguros de las vulnerabilidades comúnmente encontradas y las mejores prácticas para su corrección. El curso también cubrirá herramientas y prácticas modernas de desarrollo, preparando a los estudiantes para enfrentar los desafíos del desarrollo web del lado del servidor de hoy en día y competencia. Este curso está diseñado para proporcionar a los estudiantes el conocimiento teórico y las habilidades prácticas que son directamente aplicables en entornos profesionales de desarrollo web.

3.4.1 Lenguaje del lado del servidor

Son aquellos que se ejecutan en el servidor web, en lugar de en el navegador web del cliente (Usuario). Estos lenguajes se utilizan para procesar solicitudes HTTP, gestionar las bases de datos, así mismo ejecutar la lógica del negocio, todo ello invisible para el usuario final (Caballero, 2016). "Como Celi Párraga et al. (2023) describen en su análisis sobre la programación web, 'Cuando se habla de Back-end se refiere al interior de las aplicaciones que viven en el servidor' (p. 6)". Este enfoque describe la importancia crítica del back-end en el desarrollo de aplicaciones web, donde se manejan aspectos fundamentales como la lógica de negocio y la interacción con bases de datos. Algunos de los lenguajes de lado de servidor más populares son:

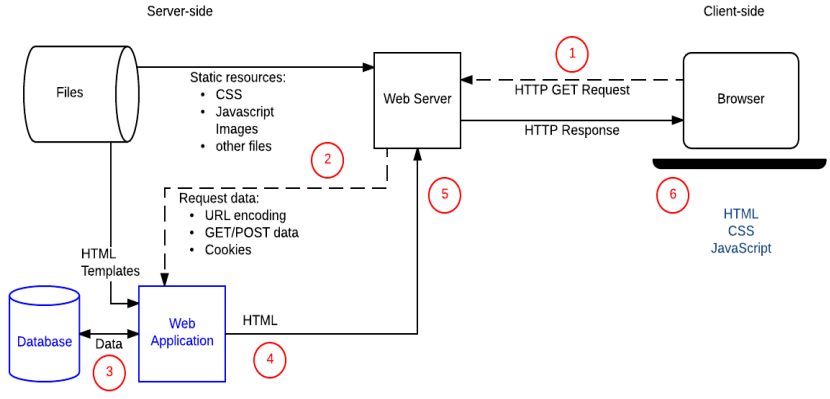
- **PHP.** Es uno de los lenguajes de servidor más antiguos y ampliamente empleados, especialmente en el desarrollo de sitios web. Ofrece compatibilidad con la mayoría de los sistemas operativos y servidores web disponibles.
- **ASP.NET (C#, VB.NET).** Es un conjunto de herramientas y bibliotecas proporcionadas por Microsoft para construir aplicaciones y servicios web, utilizando principalmente lenguajes de programación como C# y Visual Basic .NET.
- **Node.js (JavaScript)**. Node.js posibilita la ejecución de JavaScript en el servidor, lo que otorga a los desarrolladores la flexibilidad de utilizar un único lenguaje (JavaScript) tanto en el lado del cliente como en el servidor.
- **Python (Django, Flask)**. Python es un lenguaje de programación versátil con frameworks populares como Django y Flask, los cuales se usan para la creación de aplicativos web del lado del servidor.

Estos lenguajes del lado del servidor se utilizan para procesar la lógica de negocios, consultar Bases de datos, generar contenido dinámico y enviar la respuesta final al navegador web del cliente. Por otro lado, estos lenguajes hacen de los desarrollos web, un sitio dinámico, según (Smith, 2024, p. 157) define una web dinámica como “un conjunto de datos almacenados en una base de datos (BD) – gestionada por la capa de software conocida como sistema de gestión de base de datos (SGBD)”. Esta explicación resalta cómo la programación del lado del servidor facilita la interacción entre la interfaz del usuario y la base de datos que reside en el servidor.

Al hacerlo, se diferencia de los sitios web estáticos, los cuales permanecen iguales cada vez que son visitados. En cambio los sitios web dinámicos, pueden cambiar y adaptarse en respuesta a las acciones del usuario o a los datos almacenados. Este proceso no solo permite que sean interactivos, sino que también son capaces de ofrecer contenido personalizado y actualizado. Esto contribuye notablemente a mejorar la experiencia del usuario durante la navegación en el sitio web.

A continuación, se ilustra el funcionamiento de una sitio web dinámico en la figura 1. Esta representación muestra una arquitectura básica utilizada. Los navegadores web envían solicitudes HTTP al servidor, el cual las procesa y devuelve una respuesta utilizando el mismo protocolo.

Figura 1.
Arquitectura simple para un sitio web dinámico.



Nota. *Esta figura muestra el funcionamiento básico de un sitio web dinámico. Tomado de MDN Web Docs. (2024).*

El flujo general muestra que cuando un usuario solicita una petición a través de su navegador (paso 1), el servidor web decide si debe enviar un archivo estático directamente (paso 2) o si debe pasar la solicitud a una aplicación web(código del lado del servidor) para generar una respuesta dinámica (pasos 3), si es así, se recuperará la información necesaria de la Base de datos, hace una combinación de los datos con plantillas HTML(paso 4), envía una respuesta que incluye el HTML generado según la solicitud del usuario (paso 5 y 6). Este proceso ilustra la naturaleza dinámica de los sitios web modernos, que pueden adaptarse y cambiar en respuesta a la interacción del usuario y a los datos en tiempo real.

3.4.2 Sitios web de refuerzo para los temas.

1. https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Introduction
El documento aborda el ámbito del desarrollo web en el servidor, pieza clave para forjar experiencias en línea que son personalizadas como dinámicas. Resalta la importancia de dominar la creación de sitios web que no solo sean interactivos, sino también capaces de interactuar de forma inteligente con las peticiones de los usuarios mediante el protocolo HTTP. Se enfatiza el valor de herramientas específicas como Django para Python y Express para Node.js/JavaScript, destacando su papel vital en el universo del desarrollo web.

2. https://www.w3schools.com/
W3Schools proporciona tutoriales paso a paso sobre programación del lado del servidor, cubriendo lenguajes y tecnologías como PHP, ASP.NET, y Node.js. Este sitio es ideal para principiantes que buscan entender los conceptos fundamentales de la programación back-end, con ejemplos de código y prácticas interactivas.

3. https://stackoverflow.com/
Aunque no es un tutorial tradicional, Stack Overflow es una comunidad de desarrolladores que ofrece respuestas a preguntas específicas sobre programación del lado del servidor. Es una fuente para resolver problemas, entender mejores prácticas y descubrir patrones de diseño emergentes en la programación del lado del servidor, especialmente en lenguajes como Python, Ruby, y JavaScript en el entorno Node.js.

3.4.3 Procesamiento

El trabajo del lado del servidor desempeña un papel fundamental en el desarrollo de aplicaciones web dinámicas y eficientes, encargándose de realizar operaciones complejas en segundo plano, como la generación dinámica de contenido para las páginas web, la gestión de sesiones de usuarios y la autenticación. Este proceso permite la actualización instantánea del contenido de las páginas web en función de las acciones del usuario y las circunstancias cambiantes, asegurando una experiencia segura y personalizada para cada usuario. Además, es esencial para facilitar la interacción con las bases de datos, lo que permite a las aplicaciones web gestionar información en tiempo real y llevar a cabo tareas como la validación y el procesamiento de formularios.

En este contexto, los frameworks de desarrollo web se destacan como herramientas esenciales, diseñadas para facilitar y acelerar la construcción de aplicaciones web. Un framework proporciona una infraestructura de soporte que incluye una serie de componentes y servicios preestablecidos. Esto permite a los programadores concentrarse en las particularidades de su proyecto, evitando el reto de desarrollar desde cero y las complicaciones asociadas a ello. Al ofrecer un entorno de desarrollo "listo para usar", los frameworks simplifican el proceso al evitar la repetición innecesaria de tareas comunes como la autenticación de usuarios, la gestión de bases de datos y la creación de formularios. Además, los frameworks no solo promueven un desarrollo más eficiente y seguro mediante la reutilización de código y la estandarización de prácticas, sino que también mejoran la escalabilidad y la mantenibilidad del software. Esto hace que el desarrollo web sea más accesible, manejable y adaptable a las necesidades cambiantes de proyectos de cualquier tamaño, desde pequeñas aplicaciones hasta soluciones empresariales complejas. A continuación, se presentan tres frameworks populares para la programación del lado del servidor:

- **Django (Python).** Django es un framework para el desarrollo web con Python, reconocido por su eficiencia y simplicidad en la creación de aplicaciones. Desde su lanzamiento en 2005 por un grupo de desarrolladores web, ha ganado popularidad al ofrecer un entorno "listo para usar" que elimina la necesidad de comenzar desde cero gracias a sus numerosas funcionalidades integradas. Estas incluyen la autenticación de usuarios, la administración de bases de datos, la creación de formularios y la gestión de sesiones, permitiendo a los desarrolladores enfocarse en desarrollar funcionalidades específicas sin tener que construir la infraestructura subyacente.

  El diseño de Django se basa en el patrón Modelo-Vista-Plantilla (MVT), una adaptación del Modelo-Vista-Controlador (MVC), que organiza la estructura de la aplicación en tres componentes interrelacionados. Este enfoque no solo facilita un desarrollo más ordenado y modular, sino que también mejora la mantenibilidad del código. El modelo gestiona la estructura de datos y las reglas del negocio, la vista se encarga de mostrar los datos y gestionar la interacción con el usuario, y la plantilla define el diseño visual de las páginas web, garantizando una clara separación entre la lógica de la aplicación y su presentación.

Además de fomentar una alta productividad,

Django se destaca por su fuerte enfoque en la seguridad y la escalabilidad. Incorpora robustos mecanismos de seguridad para proteger contra vulnerabilidades comunes como inyecciones SQL y ataques XSS, asegurando el desarrollo de aplicaciones web seguras. Su diseño también permite una fácil escalabilidad, facilitando la gestión eficiente de grandes volúmenes de tráfico y usuarios concurrentes, lo que lo convierte en una opción ideal para proyectos de cualquier tamaño, desde pequeñas aplicaciones hasta soluciones empresariales de gran envergadura. A continuación, se presentarán una serie de pasos simples para instalar Django:

Paso 1: Instalar Python.

En Windows:

1.Descarga Python desde python.org.

2.Ejecuta el instalador y sigue las instrucciones en pantalla. Asegúrate de seleccionar la opción para agregar Python al PATH durante la instalación.

En macOS/Linux:

1.En la mayoría de los sistemas macOS y Linux, Python ya viene preinstalado. Puedes verificar su instalación abriendo una terminal y escribiendo:

css

Copiar código

python3 --version

Si Python no está instalado, sigue los pasos correspondientes a tu sistema:

macOS

- Instala Python usando Homebrew. Si Homebrew no está instalado, primero instálalo siguiendo las instrucciones en brew.sh, y luego ejecuta:

Copiar código

brew install python

Linux

- Utiliza el gestor de paquetes de tu distribución. Por ejemplo, en Ubuntu, puedes instalar Python con:

sql

Copiar código

sudo apt-get update

sudo apt-get install python3

Después de asegurarte de que Python está instalado correctamente, puedes proceder con la instalación de Django utilizando pip, el gestor de paquetes de Python.

Paso2: Instalar pip.

Para instalar pip, se deben seguir los pasos según el sistema operativo:

Windows:

- Pip generalmente se instala automáticamente junto con Python. Para verificar que pip está instalado, se debe abrir la línea de comandos y ejecutar:

css

Copiar código

pip --version

Si pip no está instalado, se puede descargar el script get-pip.py desde bootstrap.pypa.io y ejecutarlo con Python utilizando:

arduino

Copiar código

python get-pip.py

macOS/Linux:

- En muchos sistemas, pip viene preinstalado con Python. Para verificar su instalación, se debe ejecutar:

css

Copiar código

pip3 --version

Si pip no está presente, puede instalarse con el siguiente comando:

Copiar código

sudo easy_install pip

En caso de que easy_install no esté disponible, una alternativa es:

csharp
Copiar código
sudo apt-get install python3-pip # En sistemas basados en Debian, como Ubuntu
o, en macOS, utilizando Homebrew:
Copiar código
brew install pip

Una vez que pip esté instalado, se podrá usar para instalar Django y otros paquetes de Python de manera sencilla.

Paso 3: Instalación de Django.

1. Para instalar Django, se debe ejecutar el siguiente comando en la terminal:
Copiar código
pip install Django
2. Para verificar que la instalación se realizó correctamente, se puede comprobar la versión de Django con el siguiente comando:
css
Copiar código
django-admin --version
Estos pasos instalarán Django y permitirán verificar que está listo para ser utilizado en el desarrollo de aplicaciones web.

- **Express (Node.js)**. Express.js es un framework para el desarrollo web en Node.js que facilita la creación de aplicaciones web y APIs mediante una capa adicional de abstracción para gestionar solicitudes HTTP y la navegación entre páginas. Inspirado en la filosofía de Django en Python, Express.js se centra en la productividad y simplicidad, proporcionando un conjunto de herramientas y funcionalidades que reducen la necesidad de código repetitivo y simplifican la construcción de aplicaciones web complejas. Este enfoque de "minimalismo inteligente" dota a los desarrolladores de las herramientas esenciales sin imponer estructuras rígidas, lo que permite gran flexibilidad y adaptabilidad en la organización del código.

  La flexibilidad y modularidad son características distintivas de Express.js, que lo diferencian de otros frameworks más prescriptivos. Los desarrolladores tienen la libertad de estructurar sus aplicaciones de la manera que mejor se adapte a sus necesidades, gracias al uso de middleware. Estas funciones intermedias se integran en el flujo de solicitud/respuesta para realizar diversas tareas, desde el análisis de datos entrantes hasta la autenticación de usuarios y la manipulación de encabezados HTTP, contribuyendo así a una arquitectura altamente personalizable y eficiente. Además, Express.js se beneficia de una comunidad activa de desarrolladores y un ecosistema rico en complementos y middleware, lo que amplía aún más sus

capacidades y ofrece múltiples posibilidades para la creación de aplicaciones web avanzadas.

La participación comunitaria en el desarrollo de bibliotecas y herramientas adicionales no solo expande las funcionalidades de Express.js, sino que también proporciona a los desarrolladores recursos y soporte para sus proyectos.

A continuación, se presentan unos pasos simples para instalar Express:
Asegúrese de tener Node.js instalado. Si no está instalado, descárguelo desde nodejs.org.

Una vez Node.js esté instalado, abra la terminal y ejecute:
css
Copiar código
npm install express --save
Para confirmar que Express está correctamente instalado, cree un archivo JavaScript (por ejemplo, app.js) y agregue el siguiente código básico para iniciar un servidor Express:
javascript
Copiar código
```
const express = require('express');
const app = express();
const port = 3000;
app.get('/', (req, res) => {
res.send('¡Express está funcionando!');
});
app.listen(port, () => {
  console.log(`Servidor escuchando en http://localhost:${port}`);
});
```
Ejecute el archivo con Node.js usando:
Copiar código
node app.js

Este procedimiento instalará Express y confirmará su funcionalidad mediante la creación y ejecución de un servidor básico.

- **Laravel (PHP).** Laravel es un framework para el desarrollo web en PHP, lanzado en 2011, que ha sido ampliamente reconocido en la comunidad de desarrolladores por su sintaxis elegante, su amplia gama de funcionalidades y su enfoque en la facilidad de uso y productividad. Este framework se destaca por su enfoque de "baterías incluidas", proporcionando un conjunto completo de herramientas listas para usar que abordan aspectos comunes del desarrollo web, como la autenticación de usuarios, enrutamiento flexible, gestión de sesiones y migraciones de bases de datos, entre otros.

Basado en el patrón de diseño Modelo-Vista-Controlador (MVC), Laravel facilita la organización del código y separa eficientemente la lógica de negocio de la presentación de datos, lo que permite un desarrollo más limpio y mantenible. Su sintaxis expresiva y elegante permite a los desarrolladores escribir código de manera más intuitiva y legible, mejorando la eficiencia y la calidad del desarrollo. Además, Laravel integra Composer, un sistema de gestión de dependencias que simplifica la inclusión y gestión de bibliotecas y paquetes de terceros en los proyectos.

Otro gran beneficio de Laravel es su activa y colaborativa comunidad de desarrolladores, que constantemente contribuye con nuevas bibliotecas, paquetes y herramientas que amplían las funcionalidades del framework. Esta comunidad, junto con la extensa documentación oficial, tutoriales y cursos en línea, hace de Laravel una opción atractiva tanto para novatos como para expertos en desarrollo web, proporcionando los recursos necesarios para desarrollar aplicaciones web avanzadas de manera eficiente y efectiva.

Pasos para instalar Laravel

1. Instalar PHP

Windows
Utilizar XAMPP o WAMP, que incluyen PHP junto con otros componentes necesarios.

macOS
Si se usa Homebrew, instalar PHP con:
Copiar código
brew install php

Linux (Ubuntu/Debian)
Instalar PHP utilizando el gestor de paquetes:
arduino
Copiar código
sudo apt-get install php

2. Instalar Composer:
Composer es un gestor de dependencias para PHP. Para instalarlo, se debe visitar getcomposer.org y seguir las instrucciones que se ofrecen en el sitio web.

3. Instalación de Laravel:
- Una vez Composer esté instalado, abrir la terminal y ejecutar:

lua
Copiar código
composer create-project --prefer-dist laravel/laravel nombreDelProyecto
- Navegar a la carpeta del proyecto recién creado con:

bash

Copiar código
cd nombreDelProyecto

Estos pasos configuran Laravel y preparan el entorno para comenzar a desarrollar aplicaciones web, proporcionando una estructura de proyecto básica y todas las herramientas necesarias para aprovechar al máximo las capacidades del framework.

Para construir una vista sencilla en HTML que muestre un mensaje en pantalla, sigue los siguientes pasos:

1. Crear un Archivo de Vista HTML:

- Dentro de tu proyecto Laravel, las vistas se encuentran generalmente en la carpeta resources/views.
- Crea un nuevo archivo llamado welcome.blade.php dentro de la carpeta resources/views.

2. Escribir el Código HTML en la Vista:

- Abre el archivo welcome.blade.php y añade el siguiente código HTML para mostrar un mensaje en pantalla:

html
Copiar código

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Mensaje de Bienvenida</title>
</head>
<body>
    <h1>¡Bienvenido a Laravel!</h1>
    <p>Este es un mensaje sencillo mostrado en pantalla.</p>
</body>
</html>
```

3. Configurar la Ruta para Mostrar la Vista:

- Abre el archivo de rutas routes/web.php y añade una ruta para mostrar la vista que acabas de crear:

php
Copiar código

```
use Illuminate\Support\Facades\Route;

Route::get('/', function () {
    return view('welcome');
});
```

4. Probar la Vista:

- Asegúrarse de que el servidor Laravel está corriendo. Se puede iniciar el servidor usando el siguiente comando en la terminal dentro del proyecto Laravel:

Copiar código
php artisan serve

- Abra un navegador y visite http://localhost:8000. Deberías ver la página con el mensaje "¡Bienvenido a Laravel!" en la pantalla.

Este proceso crea una vista básica en HTML que muestra un mensaje en pantalla utilizando Laravel. La estructura Blade (.blade.php) es el motor de plantillas de Laravel que permite añadir más funcionalidades y lógica a las vistas de manera sencilla.

Figura 2.
Formulario Hola Mundo

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Hola Mundo - Django</title>
</head>
<body>
    <h1>¡Hola Mundo desde Django!</h1>
</body>
</html>
```

Nota. *La figura muestra la estructura del formulario del Hola Mundo. Elaboración propia.*

El siguiente código en Python representa una vista básica en Django que devuelve un saludo "¡Hola Mundo!" como respuesta HTTP cuando se accede a través de una solicitud web. Utiliza la clase HttpResponse del módulo django.http para generar la respuesta. La función index toma un argumento request, que representa la solicitud HTTP entrante, y devuelve un objeto HttpResponse con el mensaje "¡Hola Mundo!" como contenido de la respuesta. Esta vista puede integrarse en una aplicación Django para manejar solicitudes web y generar respuestas dinámicas.

Código de la vista en Django:
python
Copiar código

```
from django.http import HttpResponse
def index(request):
    return HttpResponse("¡Hola Mundo!")
```

Explicación:

1. **Importación de HttpResponse:** La clase HttpResponse se importa desde el módulo django.http. Esta clase se utiliza para crear respuestas HTTP en Django.
2. **Función index:** Se define una función llamada index que toma un argumento request, el cual representa la solicitud HTTP entrante. Dentro de esta función, se retorna un objeto HttpResponse que contiene el texto "¡Hola Mundo!".
3. **Integración en una Aplicación Django:**

- Para integrar esta vista en una aplicación Django, se debe definir una ruta en el archivo urls.py de la aplicación que apunte a esta función. A continuación, se muestra un ejemplo de cómo configurar la ruta:

python

Copiar código

```
from django.urls import path
from . import views
urlpatterns = [
    path('', views.index, name='index'),
]
```

- Este código en urls.py establece que cuando se accede a la raíz del sitio (/), Django llamará a la función index y devolverá la respuesta "¡Hola Mundo!".

4. **Ejecutar el Servidor:**

- Inicia el servidor de desarrollo de Django usando el comando:

Copiar código

```
python manage.py runserver
```

- Luego, visita **http://localhost:8000** en el navegador para ver la respuesta "¡Hola Mundo!".

Este sencillo ejemplo muestra cómo crear una vista básica en Django que devuelve una respuesta HTTP con un saludo, ilustrando el funcionamiento fundamental de las vistas en el framework Django.

Figura 3.
Hola Mundo en Django.

```python
from django.http import HttpResponse

def index(request):
    return HttpResponse("¡Hola Mundo desde Django!")
```

Nota. *La figura muestra una vista básica del Hola Mundo. Elaboración propia.*

Ahora se hará el formulario en Express.

Figura 4.
Formulario del Hola Mundo.

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Hola Mundo - Express</title>
</head>
<body>
    <h1>¡Hola Mundo desde Express!</h1>
</body>
</html>
```

Nota. *La figura muestra la estructura del formulario del Hola Mundo. Elaboración propia.*

El siguiente código es un ejemplo de cómo crear un servidor web básico utilizando Express en Node.js. En primer lugar, se incluye el módulo Express, se inicia una aplicación Express y se define el puerto en el cual el servidor atenderá las solicitudes recibidas. A continuación, se establece una ruta principal '/' utilizando el método app.get(), que maneja las solicitudes GET a la raíz del servidor. Cuando se accede a esta ruta, el servidor responde con el mensaje "¡Hola Mundo desde Express!". Finalmente, se inicia el servidor y se pone en marcha escuchando en el puerto especificado, mientras que un mensaje se imprime en la consola para indicar que el servidor está funcionando correctamente.

Figura 5.
Hola Mundo en Express.

```javascript
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => {
  res.send('¡Hola Mundo desde Express!');
});

app.listen(port, () => {
  console.log(`Servidor escuchando en http://localhost:${port}`);
});
```

Nota. *La figura muestra la estructura del formulario del Hola Mundo. Elaboración propia.*

Por último, se hace un ejemplo en Laravel.

Figura 6.
Formulario del Hola Mundo.

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Hola Mundo - Laravel</title>
</head>
<body>
    <h1>¡Hola Mundo desde Laravel!</h1>
</body>
</html>
```

Nota. *La figura muestra la estructura del formulario del Hola Mundo. Elaboración propia.*

El siguiente código proporciona un ejemplo de una ruta básica en Laravel que responde a las solicitudes GET en la raíz del servidor. Utiliza la clase Route para definir la ruta '/', la cual devuelve el mensaje "¡Hola Mundo desde Laravel!" cuando se accede a ella. Esta ruta es gestionada mediante una función anónima que se pasa como segundo argumento al método get() de la clase Route. Así, cuando un usuario accede a la raíz del servidor, Laravel devuelve este mensaje como respuesta.

Figura 7.
Hola Mundo en Laravel.

```php
<?php

use Illuminate\Support\Facades\Route;

Route::get('/', function () {
    return '¡Hola Mundo desde Laravel!';
});
```

Nota. *La figura muestra una vista básica del Hola Mundo. Elaboración propia.*

Para encontrar información adicional y recursos educativos sobre estos y otros frameworks del lado del servidor, se recomienda visitar los siguientes tres sitios web:

Sitios web

1. https://docs.djangoproject.com/en/
Este material es fundamental para los desarrolladores que trabajan con Django, un framework para el desarrollo web en Python, diseñado para facilitar una construcción ágil y un diseño orientado a resultados claros y prácticos. Incluye una documentación exhaustiva que abarca desde los pasos iniciales de instalación, pasando por tutoriales dirigidos a principiantes, hasta temas más avanzados como la optimización del rendimiento y la seguridad de las bases de datos. Mantenida de manera regular por la Django Software Foundation, esta guía garantiza que los programadores tengan acceso a la información más actualizada y relevante para sus proyectos.

2. https://expressjs.com/
Esta página web es el principal recurso para Express, un framework para aplicaciones web en Node.js, conocido por ser minimalista y adaptable, diseñado para el desarrollo ágil de aplicaciones web y APIs. Ofrece tutoriales para una rápida puesta en marcha, documentación exhaustiva sobre la API y un amplio conjunto de recursos comunitarios. La documentación es esencial tanto para programadores que se inician en Express como para aquellos con experiencia que buscan profundizar en funcionalidades específicas o aprender las mejores prácticas en el desarrollo con este framework.

3. https://laravel-news.com/
Este portal se ha consolidado como un recurso fundamental para la comunidad de Laravel, ofreciendo las últimas noticias, tutoriales y paquetes relacionados con este framework, reconocido como uno de los más populares para el desarrollo de aplicaciones web en PHP. Aunque no es el sitio oficial de Laravel, Laravel News destaca por su contenido de alta calidad, abarcando desde las actualizaciones y novedades más recientes del framework hasta consejos y técnicas para optimizar el desarrollo de proyectos. Es una fuente indispensable para desarrolladores que buscan mantenerse al día con las últimas tendencias y mejores prácticas en Laravel.

3.4.5 Operadores.

En el lado del servidor, la programación emplea operadores clave que son esenciales para la manipulación efectiva de datos, el control del flujo de ejecución y la implementación de lógica en aplicaciones web de gran escala. Salazar (2022) afirma que "un operador es una entidad que actúa sobre los datos para cambiarlos", subrayando el papel crucial de los operadores en la programación. Estos operadores varían ampliamente, desde operaciones aritméticas simples hasta comparaciones lógicas complejas. En lenguajes de programación populares como Python, JavaScript (Node.js) y PHP, los operadores aritméticos son fundamentales para realizar cálculos directos con

números, lo que facilita la construcción de la lógica dentro de las aplicaciones.

Operadores Aritméticos.
Los operadores aritméticos permiten la ejecución de cálculos matemáticos básicos, como la adición, sustracción, multiplicación, división y la determinación del residuo. Estos operadores, representados por "+, -, *, /, %", funcionan de manera similar a una calculadora: el operador + suma dos valores, - resta el segundo valor del primero, * multiplica ambos valores, / divide el primer valor entre el segundo, y % devuelve el residuo de la división del primer valor por el segundo. Estos operadores son fundamentales en programación, ya que facilitan la realización de cálculos directos con números, de manera similar a como lo haría una calculadora (Izquierdo, 2023).

Figura 8.
Operadores de aritmética básica.

***Nota**. La figura muestra la representación visual de cada operador. Tomado de Fundamentos de Programación en Lenguaje C (p.14), por Chávez, J. D. (2019).*

Operadores de Asignación.
Los operadores de asignación se utilizan para asignar valores a las variables. El operador más básico es el "=", que asigna el valor de la expresión a la derecha del operador a la variable situada a la izquierda. Existen también operadores compuestos, como "+=" para sumar y asignar, y "-=" para restar y asignar, entre otros. En estos casos, primero se realiza la operación aritmética y luego se asigna el resultado a la variable. Por ejemplo, x += 3 es equivalente a x = x + 3. Estos operadores simplifican la manipulación de variables y mejoran la claridad del código, haciéndolo más conciso y fácil de leer.

Operadores de Incremento/Decremento.
Los operadores "++" y "--" están diseñados para incrementar o decrementar en uno el valor de una variable numérica, respectivamente. La posición de estos operadores en relación con la variable influye en cómo se realiza la operación: si se colocan antes de la variable (++x), incrementan o decrementan el valor de la variable antes de utilizarla en cualquier operación; si se colocan después de la variable (x++), el valor se incrementa o decrementa

después de su uso inicial. Estos operadores son especialmente útiles en contextos de operaciones repetitivas y bucles iterativos, donde permiten gestionar de manera eficiente los contadores y el control del flujo de ejecución.

Operadores Lógicos.
Los operadores lógicos permiten construir expresiones lógicas complejas combinando valores booleanos (verdadero/falso). Entre ellos se incluyen operadores como "!", "&&" y "||". El operador "!" niega el valor booleano, convirtiendo true en false y viceversa. El operador "&&" (AND) devuelve true solo si ambos operandos son true. Por otro lado, el operador "||" (OR) devuelve true si al menos uno de los dos operandos es true. Estos operadores lógicos son esenciales para la toma de decisiones en el flujo del programa, ya que permiten evaluar condiciones y ejecutar acciones basadas en esas evaluaciones.

Operadores Relacionales.
Estos operadores se utilizan para realizar comparaciones entre dos valores, devolviendo un resultado booleano (true o false). Incluyen operadores como >, <, ==, y !=. El operador > verifica si el valor a la izquierda es mayor que el valor a la derecha, mientras que < comprueba si es menor. El operador == evalúa si ambos operandos son iguales, y != determina si son distintos. Estos operadores generan un resultado de verdadero o falso, lo cual es fundamental para implementar decisiones condicionales en la programación. A continuación, se muestra una breve demostración de cómo se utilizan los operadores aritméticos en JavaScript:

Figura 9.
Operadores lógicos en JavaScript.

```
// Operadores aritméticos
let p = 2;              // Asignación: p recibe el valor de 2
let q = 3;              // Asignación: q recibe el valor de 3
let suma = p + q;       // Suma: suma de p y q
let resta = p - q;      // Resta: resta de p y q
let multiplicacion = p * q; // Multiplicación: producto de p y q
let division = p / q;   // División: división de p entre q
let modulo = p % q;     // Módulo: resto de la división de p sobre q
```

Nota. *Uso básico de los operadores. Fuente: elaboración propia.*

El código define dos variables, p y q, con valores 2 y 3, respectivamente, y luego realiza varias operaciones aritméticas con ellas, asignando los resultados a variables separadas: suma contiene la suma de p y q, resta almacena la diferencia entre p y q, multiplicación guarda el producto de p y q, división contiene el cociente de p dividido por q, y modulo guarda el resto de la división de p entre q.

A continuación, se presenta una breve demostración de cómo se utilizan los operadores de asignación y los operadores lógicos en Python:

Figura 10.
Operadores lógicos y de asignación en Python.

```python
# Operadores de asignación compuesta
r = 7  # Asignación: r recibe el valor de 7
r += p  # Asignación compuesta de suma: r incrementa en el valor de p
r *= q  # Asignación compuesta de multiplicación: r se multiplica por el valor de q

# Operadores lógicos
edad = 30  # Asignación: edad recibe el valor de 30
es_adulto = edad >= 18 and edad <= 65  # Lógico AND: verdadero si edad está entre 18 y 65
```

Nota. *Uso básico de los operadores de lógicos y de asignación. Fuente: elaboración propia.*

En el primer fragmento, se están utilizando operadores de asignación compuesta en Python. Inicialmente, se asigna el valor 7 a la variable r. A continuación, se utiliza el operador += para incrementar el valor de r sumándole el valor de p. Posteriormente, se emplea el operador *= para multiplicar el valor actual de r por el valor de q. Esto significa que el valor de r primero se incrementa en p y luego se multiplica por q. En el segundo fragmento, se emplean operadores lógicos en Python para evaluar una condición. Se asigna el valor 30 a la variable edad, y se utiliza el operador and para verificar si edad se encuentra entre 18 y 65. La expresión edad >= 18 and edad <= 65 devolverá True si el valor de edad está dentro de ese rango, lo que indica que la persona es considerada adulta según esta definición.

Figura 11.
Operadores de comparación en PHP.

```php
<?php
// Operadores de comparación
$s = 15;  // Asignación: s recibe el valor de 15
$t = 20;  // Asignación: t recibe el valor de 20
$menor_que = ($s < $t);  // Menor que: verdadero si s es menor que t
$igual_b = ($s == $p);  // Igualdad: verdadero si s es igual a p
?>
```

Nota. *Uso básico de los operadores de asignación. Fuente: elaboración propia.*

En este fragmento de código PHP, se utilizan operadores de comparación para evaluar condiciones entre variables. Se asigna el valor 15 a la variable $s y 20 a la variable $t. A continuación, se emplea el operador < para comprobar si $s es menor que $t, asignando el resultado booleano a la variable $menor_que. Además, se intenta verificar si $s es igual a $p utilizando el operador ==. Sin embargo, dado que la variable $p no está definida previamente en este contexto, la comparación no se realiza correctamente y la variable $igual_b será false debido a la falta de definición de $p, lo cual podría generar un aviso de error en PHP.

Para evitar este tipo de problemas, es importante asegurarse de que todas las variables estén definidas antes de realizar comparaciones. Para profundizar en este tema, se sugiere consultar estos tres recursos:

Sitios web

1. https://j2logo.com/python/tutorial/operadores-en-python/
   Este sitio ofrece una guía completa sobre los operadores en Python, un lenguaje de programación fundamental en áreas como el desarrollo web y el análisis de datos. La guía detalla el uso de operadores aritméticos básicos, operadores de comparación y operadores lógicos, los cuales son esenciales para la construcción de estructuras de control de flujo. Este conocimiento es crucial para que los programadores puedan crear código eficiente y fácil de entender. Además, la guía incluye ejemplos prácticos que ayudan a aplicar estos conceptos en proyectos de programación, facilitando así su comprensión y uso en contextos reales.
2. https://uniwebsidad.com/libros/javascript/capitulo-3/operadores
   Este sitio ofrece una guía detallada sobre JavaScript, en la cual se proporciona información exhaustiva acerca de los operadores en JavaScript, incluyendo operadores de asignación, incremento y decremento, lógicos, matemáticos y relacionales. Se explican conceptos clave como la asignación de valores a variables, el incremento o decremento de valores numéricos, el uso de operadores para realizar decisiones lógicas, la ejecución de operaciones matemáticas básicas y la comparación de valores para determinar igualdad, diferencia y relaciones de tamaño. Esta explicación es esencial para comprender cómo manipular datos y tomar decisiones dentro de programas en JavaScript. Para obtener más información, se puede visitar el sitio web directamente.
3. https://desarrolloweb.com/articulos/operadores-php.html
   Este sitio proporciona una guía completa y estructurada para comprender el funcionamiento de los diferentes tipos de operadores en el lenguaje de programación PHP. La guía cubre categorías como operadores aritméticos, de asignación, de comparación, lógicos, de incremento y decremento, y operadores de cadenas. Además, resalta la importancia de entender la precedencia de los operadores para escribir códigos claros y eficientes. Este recurso es valioso tanto para principiantes que están aprendiendo PHP como para programadores avanzados que deseen refrescar y profundizar sus conocimientos.

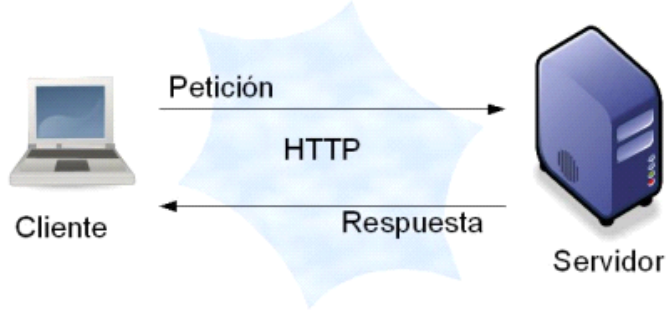
3.4.6 Procesado de formularios

En el desarrollo web y las aplicaciones digitales, los formularios funcionan como interfaces interactivas esenciales que permiten a los usuarios compartir información. Según Londoño-Rojas et al. (2021), "los formularios son usados comúnmente para proporcionar la interacción de los usuarios con el sitio web". Estos elementos se estructuran con HTML, se estilizan con CSS para mejorar su presentación visual y a menudo se enriquecen con JavaScript para añadir validación de datos y mayor interactividad. Este proceso de interac-

ción es fundamental para la experiencia del usuario, facilitando funciones clave como el registro de usuarios, la realización de transacciones, la recolección de feedback y el inicio de un proceso integral de procesamiento de formularios.

El procedimiento comienza con la creación y el envío del formulario por parte del usuario y continúa con la transmisión de los datos al servidor mediante los métodos GET o POST. El servidor recibe, valida y procesa estos datos, realizando operaciones como registros en bases de datos o actualizaciones, y responde al usuario con confirmaciones, mensajes de error o redirecciones según el resultado. La seguridad en este proceso es crucial, ya que protege los datos contra vulnerabilidades como la inyección de SQL y los ataques XSS, subrayando la importancia de implementar prácticas de seguridad adecuadas en el manejo de formularios.

Figura 12.
Operadores de comparación en PHP.



Nota. *La figura muestra el flujo básico de comunicación consistiendo en que el cliente envía peticiones al servidor, éste la procesa y responde al cliente. Tomado de Sansano Miralles (2017).*

Los formularios y su procesamiento destacan la interacción significativa entre usuarios y plataformas digitales, subrayando su importancia en la web moderna. Los aspectos prácticos del manejo y la seguridad de los formularios son fundamentales para la integridad de las aplicaciones. En este contexto, los métodos GET y POST son los más utilizados para el procesamiento de formularios, aunque existen otros métodos como PUT, DELETE y PATCH, que son menos comunes en este ámbito y se utilizan principalmente en aplicaciones RESTful y API.

El método GET se utiliza para solicitar recursos del servidor sin modificarlo, como ocurre al navegar mediante enlaces o realizar búsquedas básicas. Sin embargo, por razones de seguridad, no es recomendable su uso en aplicaciones que impliquen cambios en el servidor, ya que la información se transfiere a través de la URL, incluyendo los parámetros en esta (Ríos Pérez et al., 2017). En contraste, el método POST es una petición HTTP utilizada para enviar datos al servidor con el fin de ser procesados de cierta manera. A diferencia de GET, los datos se transmiten dentro del cuerpo de la petición HTTP, lo que proporciona mayor seguridad y lo hace ideal para manejar información

sensible como contraseñas o datos financieros, ya que estos no se exponen en la URL. POST es comúnmente utilizado para acciones que implican modificaciones en el servidor, como el envío de formularios de inscripción o la realización de transacciones en línea.

Figura 13.
Formulario para inicio de sesión.

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Formulario de Inicio de Sesión</title>
</head>
<body>
    <h2>Iniciar Sesión</h2>
    <form action="/submit" method="post">
        <label for="username">Nombre de usuario:</label>
        <input type="text" id="username" name="username" required><br><br>
        <label for="password">Contraseña:</label>
        <input type="password" id="password" name="password" required><br><br>
        <input type="submit" value="Iniciar Sesión">
    </form>
</body>
</html>
```

Nota. *La figura muestra la estructura del formulario de inicio de sesión. Fuente: elaboración propia.*

Este fragmento de código HTML configura la interfaz de un formulario de inicio de sesión, solicitando al usuario que ingrese su nombre de usuario y contraseña. Al enviar el formulario, los datos se envían a la ruta /submit mediante el método POST, donde serán procesados por una aplicación basada en Flask, según el script que se proporcionará a continuación.

Figura 14.
Procesado de formulario en Flask.

```python
# Importa las clases Flask y request del módulo flask
from flask import Flask, request

# Crea una instancia de la clase Flask, representando la aplicación web
app = Flask(__name__)

# Define una ruta '/submit' que solo acepta solicitudes POST
@app.route('/submit', methods=['POST'])
def submit_form():
    # Extrae el valor del campo 'username' del formulario enviado en la solicitud POST
    username = request.form['username']
    # Extrae el valor del campo 'password' del formulario enviado en la solicitud POST
    password = request.form['password']
    # Realiza procesamiento adicional de los datos recibidos (comentario simulado)

    # Devuelve un mensaje de saludo al cliente con el nombre de usuario y la contraseña
    return f'¡Hola {username}! Tu contraseña es {password}.'

# Verifica si el script se está ejecutando directamente
if __name__ == '__main__':
    # Inicia el servidor web de Flask en el puerto predeterminado (5000) en modo de depuración
    app.run(debug=True)
```

Nota. *La figura muestra cómo se procesa un formulario en Flask. Fuente: elaboración propia.*

Este código utiliza el framework Flask para crear un servidor web que maneja solicitudes POST enviadas al endpoint /submit. Cuando se recibe una solicitud POST con los campos del formulario 'username' y 'password', la función submit_form() extrae estos datos y devuelve un mensaje de saludo al cliente que incluye el nombre de usuario y la contraseña. El servidor se inicia con el método app.run(), ejecutándose en modo de depuración para facilitar el desarrollo y la detección de errores.

Figura 15.
Formulario de inicio de sesión.

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Formulario de Inicio de Sesión</title>
</head>
<body>
    <h2>Iniciar Sesión</h2>
    <form action="/submit" method="post">
        <label for="username">Nombre de usuario:</label>
        <input type="text" id="username" name="username" required><br><br>
        <label for="password">Contraseña:</label>
        <input type="password" id="password" name="password" required><br><br>
        <input type="submit" value="Iniciar Sesión">
    </form>
</body>
</html>
```

Nota. *La figura muestra la estructura del formulario de inicio de sesión. Fuente: elaboración propia.*

El código HTML presenta un formulario de inicio de sesión que incluye campos para introducir el nombre de usuario y la contraseña. Al enviar el formulario, la información es enviada a la ruta /submit mediante el método POST. La aplicación Flask, descrita a continuación, utiliza esta ruta para manejar las solicitudes POST y procesar los datos ingresados, como el nombre de usuario y la contraseña, con el propósito de realizar la autenticación o verificación de las credenciales proporcionadas.

Figura 16.
Procesado de formulario en Express.

```
// Importa el módulo express, que es un framework web para Node.js
const express = require('express');
// Importa el módulo bodyParser, que se utiliza para analizar los datos de las solicitudes HTTP
const bodyParser = require('body-parser');

// Crea una instancia de la aplicación Express
const app = express();
// Utiliza el middleware bodyParser.urlencoded para analizar los datos de los formularios
app.use(bodyParser.urlencoded({ extended: true }));

// Define una ruta '/submit' que responde a las solicitudes POST
app.post('/submit', (req, res) => {
    // Extrae el valor del campo 'username' del formulario enviado en la solicitud POST
    const username = req.body.username;
    // Extrae el valor del campo 'password' del formulario enviado en la solicitud POST
    const password = req.body.password;
    // Realiza procesamiento adicional de los datos recibidos (comentario simulado)

    // Envía una respuesta al cliente con un mensaje de saludo que incluye el nombre de usuario y la contraseña
    res.send(`¡Hola ${username}! Tu contraseña es ${password}.`);
});

// Inicia el servidor Express en el puerto 3000 y muestra un mensaje en la consola cuando está escuchando
app.listen(3000, () => console.log('Servidor escuchando en el puerto 3000'));
```

Nota. *La figura muestra cómo se procesa un formulario en Express. Fuente: elaboración propia.*

El siguiente código en Node.js con Express establece un servidor web que gestiona solicitudes POST en la ruta '/submit'. Al recibir una solicitud POST, el servidor extrae los datos del formulario, como el nombre de usuario y la contraseña, del cuerpo de la solicitud y los almacena en variables. Posteriormente, se puede realizar cualquier procesamiento adicional necesario. Finalmente, se envía una respuesta al cliente con un mensaje de saludo que incluye los datos del formulario. El servidor está configurado para escuchar en el puerto 3000 y muestra un mensaje en la consola cuando está en funcionamiento.

Figura 17.
Formulario de inicio de sesión.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Formulario de Inicio de Sesión</title>
</head>
<body>
    <h2>Iniciar Sesión</h2>
    <form action="" method="post">
        <label for="username">Nombre de usuario:</label>
        <input type="text" id="username" name="username" required><br><br>
        <label for="password">Contraseña:</label>
        <input type="password" id="password" name="password" required><br><br>
        <input type="submit" value="Iniciar Sesión">
    </form>
</body>
</html>
```

Nota. *La figura muestra la estructura del formulario de inicio de sesión. Fuente: elaboración propia.*

Este fragmento de código HTML configura una interfaz para un formulario de autenticación, similar al mencionado anteriormente, pero implementado en PHP. Solicita al usuario que introduzca su identificador y clave de seguridad. Al enviar el formulario, la información se redirige al mismo script PHP que gestiona el procesamiento. Después de analizar los datos, se muestra un mensaje de bienvenida que incluye el nombre de usuario y la contraseña.

Figura 18.
Procesado de formulario en PHP.

```php
<?php
// Verifica si la solicitud es de tipo POST
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Extrae el valor del campo 'username' del formulario enviado mediante POST
    $username = $_POST['username'];
    // Extrae el valor del campo 'password' del formulario enviado mediante POST
    $password = $_POST['password'];
    // Procesamiento adicional de los datos recibidos (comentario simulado)

    // Imprime un mensaje de saludo con los datos del formulario
    echo "¡Hola $username! Tu contraseña es $password.";
}
?>
```

Nota. *La figura muestra cómo se procesa un formulario en PHP. Fuente: elaboración propia.*

Este código PHP se encarga de manejar solicitudes POST en el servidor. Primero, verifica si la solicitud actual es de tipo POST utilizando $_SERVER["REQUEST_METHOD"]. Si es una solicitud POST, extrae los valores de los campos 'username' y 'password' del formulario enviado mediante POST utilizando la superglobal $_POST. Luego, se puede realizar cualquier procesamiento adicional necesario, como la validación de datos o la autenticación del usuario. Finalmente, imprime un mensaje de saludo al cliente que incluye los datos del formulario ('username' y 'password'). Este código PHP asegura que solo se procesen formularios enviados mediante POST y proporciona una respuesta adecuada al cliente basada en los datos recibidos. Aquí se presentan tres sitios web donde se puede profundizar más en el tema:

Sitios web

1. https://desarrolloweb.com/articulos/317.php
   En el sitio web de DesarrolloWeb con la URL proporcionada, se explican dos métodos comunes para el intercambio de datos en aplicaciones web desarrolladas en PHP. Primero, se detalla cómo pasar variables entre páginas mediante la URL utilizando el método GET. Se describe la sintaxis para incluir variables en un enlace hipertextual y cómo estas son recibidas en la página de destino a través del array superglobal $_GET, abordando también consideraciones sobre seguridad y claridad del código.
   En segundo lugar, se proporciona información sobre cómo enviar y recibir datos a través de formularios utilizando el método POST. Se incluye la

construcción de formularios en HTML, la asignación de nombres a los campos de entrada y la recepción y procesamiento de estos datos en una página PHP utilizando $_POST. Además, se ofrecen ejemplos de código y se discuten prácticas antiguas en comparación con las actuales.

2. https://pythonbasics.org/flask-http-methods/
En el sitio web de Python Basics, se ofrece un tutorial sobre los métodos HTTP en Flask, centrado en la gestión de solicitudes GET y POST. Se proporciona un ejemplo de cómo crear un formulario HTML que utiliza el método POST para enviar datos a un servidor Flask, y cómo configurar una ruta en Flask para manejar tanto solicitudes GET como POST. Además, se muestra cómo obtener los datos enviados por el usuario a través de estos métodos, utilizando request.form para POST y request.args para GET. El tutorial ilustra con ejemplos de código cómo redirigir a los usuarios y mostrar mensajes de bienvenida basados en los datos recibidos.

3.https://expressjs.com/en/guide/routing.html
En la página de Express sobre enrutamiento, se encuentra información detallada sobre cómo las aplicaciones Express manejan las rutas de acceso. Se explica el uso de métodos del objeto app de Express que corresponden a métodos HTTP, como app.get() para solicitudes GET y app.post() para solicitudes POST. También se discute cómo manejar múltiples métodos HTTP y el uso de parámetros de ruta para capturar valores específicos de la URL. Además, se proporcionan ejemplos de código para ilustrar cómo se implementa el enrutamiento en Express.

Sesiones.

Las sesiones constituyen un componente crítico y multifacético en el desarrollo web, diseñadas para mantener el estado y la información del usuario a través de múltiples solicitudes HTTP en un protocolo inherentemente sin estado. Según Marqués (2020), estas sesiones se describen como cookies temporales almacenadas en el navegador durante la visita de un usuario, y son esenciales para proporcionar una experiencia de navegación continua y coherente. Vergara (2023) amplía esta perspectiva al destacar cómo las sesiones de usuario facilitan el reconocimiento en un sitio web, permitiendo que acciones como el inicio de sesión o la selección de opciones se mantengan consistentes mientras el usuario navega entre páginas. Además, Boettner (2022) subraya la importancia de las sesiones desde una perspectiva de seguridad, especialmente en sitios web bancarios y financieros, donde su uso con una duración limitada ayuda a minimizar los riesgos de seguridad. La implementación efectiva de las sesiones no solo enriquece la interacción en línea al personalizar la experiencia del usuario, sino que también introduce medidas de seguridad esenciales para proteger al usuario y a la aplicación web contra accesos no autorizados. Esto se logra a través de técnicas como el cifrado HTTPS y la generación segura de tokens de sesión. Más allá de la usabilidad y la seguridad, la gestión de sesiones enfrenta desafíos relacionados

con el rendimiento, la escalabilidad y la personalización, donde estrategias como el uso de almacenamiento en memoria con Redis o Memcached se vuelven cruciales para mantener la eficiencia bajo cargas de tráfico elevadas.

Las sesiones permiten una personalización detallada y un análisis profundo del comportamiento del usuario, facilitando mejoras significativas en la interfaz y la funcionalidad de las aplicaciones, y aprovechando el historial de navegación del usuario para enriquecer su experiencia. Sin embargo, en el marco de regulaciones de privacidad como el GDPR, el manejo de las sesiones requiere un enfoque meticuloso en la legalidad y la privacidad, asegurando que la transparencia y el consentimiento sean piedras angulares en el uso de cookies y el tratamiento de datos de sesión. Estas prácticas no solo hacen la web más intuitiva y segura para los usuarios, sino que también representan desafíos significativos y oportunidades para los desarrolladores web en áreas como el rendimiento, la escalabilidad y el cumplimiento legal. La gestión cuidadosa de las sesiones es crucial para el éxito y la sostenibilidad de las plataformas digitales, equilibrando el enriquecimiento de la experiencia del usuario con las crecientes demandas de seguridad y privacidad.

Figura 19.
Interfaz de usuario simple iniciar sesión.

```php
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Iniciar sesión</title>
</head>
<body>
    <h2>Iniciar sesión</h2>
    <?php if(isset($error)) { ?>
        <p><?php echo $error; ?></p>
    <?php } ?>
    <form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>">
        <label for="username">Nombre de usuario:</label>
        <input type="text" id="username" name="username"><br><br>
        <label for="password">Contraseña:</label>
        <input type="password" id="password" name="password"><br><br>
        <input type="submit" value="Iniciar sesión">
    </form>
</body>
</html>
```

Nota. *Formulario para iniciar sesión. Fuente: elaboración propia.*

Después, se construirá el código en PHP que permitirá manejar el procedimiento mediante el cual el usuario inicia sesión en un sistema o plataforma. El código comprobará si se ha enviado un formulario de inicio de sesión, validará las credenciales proporcionadas y configurará una variable de sesión para indicar un inicio de sesión exitoso. Si las credenciales son incorrectas, se mostrará un mensaje de error.

Figura 20.
Manejo del proceso de autenticación en PHP.

```php
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Iniciar sesión</title>
</head>
<body>
    <h2>Iniciar sesión</h2>
    <?php if(isset($error)) { ?>
        <p><?php echo $error; ?></p>
    <?php } ?>
    <form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>">
        <label for="username">Nombre de usuario:</label>
        <input type="text" id="username" name="username"><br><br>
        <label for="password">Contraseña:</label>
        <input type="password" id="password" name="password"><br><br>
        <input type="submit" value="Iniciar sesión">
    </form>
</body>
</html>
```

Nota. *Esta representación visual ilustra el manejo de autenticación en PHP. Fuente: elaboración propia.*

Este código PHP está diseñado para controlar el proceso de acceso de un usuario a un sitio web. Comienza con session_start() para iniciar una sesión PHP. Luego, verifica si el formulario de acceso ha sido enviado utilizando el método POST ($_SERVER["REQUEST_METHOD"] == "POST"). Las variables $username y $password se utilizan para almacenar las credenciales del usuario. Si las credenciales proporcionadas son correctas, se asigna el valor verdadero a $_SESSION["loggedin"] para indicar que el usuario ha ingresado correctamente y se redirige al usuario a una página de bienvenida mediante header("location: bienvenida.php"). Si las credenciales no son válidas, se guarda un mensaje de error en $error para notificar al usuario del fallo en el inicio de sesión.

Ahora, aplicaremos el mismo ejemplo en otros frameworks para ofrecer una visión general de cómo se maneja este tema en diferentes entornos. Al igual que en la plantilla anterior, este ejemplo utiliza un bloque condicional en Django para mostrar un mensaje de error si existe alguno. Además, la página permite a los usuarios ingresar sus credenciales y enviarlas al servidor para iniciar sesión en la aplicación web.

Cuando un usuario solicita iniciar sesión accediendo a la URL '/iniciar_sesion/', Django llama a la función de vista iniciar_sesion, la cual está definida en el archivo views.py. Esto significa que, cuando el usuario realiza una solicitud HTTP a la ruta URL '/iniciar_sesion/', Django ejecutará la función iniciar_sesion y llevará a cabo todas las operaciones relacionadas con el inicio de sesión, como la verificación de credenciales y la autenticación del usuario.

Figura 21.
Interfaz de usuario simple para iniciar sesión.

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Iniciar sesión</title>
</head>
<body>
    <h2>Iniciar sesión</h2>
    {% if error %}
        <p>{{ error }}</p>
    {% endif %}
    <form method="post">
        {% csrf_token %}
        <label for="username">Nombre de usuario:</label>
        <input type="text" id="username" name="username"><br><br>
        <label for="password">Contraseña:</label>
        <input type="password" id="password" name="password"><br><br>
        <input type="submit" value="Iniciar sesión">
    </form>
</body>
</html>
```

Nota. *Formato del formulario para iniciar sesión. Fuente: elaboración propia.*

Figura 22.
Manejo del proceso de autenticación en Django.

```python
from django.urls import path
from .views import iniciar_sesion

urlpatterns = [
    path('iniciar_sesion/', iniciar_sesion, name='iniciar_sesion'),
    # Otras URL de la aplicación
]
```

Nota. *La figura ilustra el redireccionamiento básico en Django. Fuente: elaboración propia.*

El siguiente código define una vista en Django para manejar el inicio de sesión de un usuario. Primero, importa las funciones render y redirect de django.shortcuts. La función iniciar_sesion toma un objeto request que contiene la información sobre la solicitud HTTP recibida. Verifica si la solicitud es de tipo POST, lo que indica que se ha enviado un formulario de inicio de sesión.

Luego, valida las credenciales del usuario comparándolas con credenciales predefinidas. Si las credenciales son correctas, establece una variable de sesión llamada loggedin y redirige al usuario a una página de bienvenida. En caso de que las credenciales no sean válidas, presenta un mensaje de advertencia. Finalmente, carga la plantilla inicio_sesion.html, incorporando los datos del contexto, incluidos cualquier aviso de error.

Figura 23.
Manejo del proceso de autenticación en Django.

```python
from django.shortcuts import render, redirect

def iniciar_sesion(request):
    error = None

    if request.method == 'POST':
        # Verificar las credenciales del usuario (este es un ejemplo muy básico)
        username = "usuario"  # Nombre de usuario válido
        password = "contraseña"  # Contraseña válida

        # Verificar si las credenciales enviadas coinciden con las credenciales válidas
        if request.POST.get('username') == username and request.POST.get('password') == password:
            # Establecer una variable de sesión para indicar que el usuario ha iniciado sesión
            request.session['loggedin'] = True

            # Redirigir al usuario a una página de bienvenida
            return redirect('bienvenida')
        else:
            # Mostrar un mensaje de error si las credenciales son incorrectas
            error = "Nombre de usuario o contraseña incorrectos."

    return render(request, 'inicio_sesion.html', {'error': error})
```

Nota. *La figura muestra ilustra el manejad el proceso de iniciar sesión en Django.*
Fuente: elaboración propia.

Por último, se implementará el mismo procedimiento en Node.js utilizando Express. De manera similar, se utilizará una plantilla que incorpora un bloque condicional de Express para mostrar un mensaje de error si existe alguno. Además, permitirá al usuario enviar sus credenciales para iniciar sesión.

Figura 24.
Interfaz de usuario simple para iniciar sesión.

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Iniciar sesión</title>
</head>
<body>
    <h2>Iniciar sesión</h2>
    <% if (error) { %>
        <p><%= error %></p>
    <% } %>
    <form method="post" action="/login">
        <label for="username">Nombre de usuario:</label>
        <input type="text" id="username" name="username"><br><br>
        <label for="password">Contraseña:</label>
        <input type="password" id="password" name="password"><br><br>
        <input type="submit" value="Iniciar sesión">
    </form>
</body>
</html>
```

Nota. *La figura ilustra la estructura del formulario de inicio de sesión. Fuente: elaboración propia.*

Este código configura un servidor web utilizando Express, un framework de Node.js. Incluye la importación de módulos necesarios como Express para la creación del servidor, Express-session para el manejo de sesiones de usuario, y Body-parser para el procesamiento de datos de formularios. Se inicializa una aplicación Express y se define el puerto en el que el servidor se ejecutará. La configuración de la sesión de usuario emplea una clave secreta para la firma de cookies de sesión. Además, se establece un manejador de solicitudes POST para procesar el inicio de sesión, evaluando las credenciales del usuario. Si las credenciales son correctas, se asigna una variable de sesión y se redirige al usuario a una página de bienvenida; en caso contrario, se muestra un mensaje de error. Se proporciona una página para el inicio de sesión y se arranca el servidor para que escuche en el puerto definido. Este fragmento de código es fundamental para el desarrollo de un sistema de inicio de sesión seguro y eficiente en aplicaciones web que operan con Node.js.

Figura 25.
Gestión del proceso para iniciar sesión en Express.

```
// Importar los módulos necesarios
const express = require('express'); // Framework web
const session = require('express-session'); // Sesiones de usuario
const bodyParser = require('body-parser'); // Analizar cuerpos de solicitud

// Crear instancia de la aplicación Express
const app = express();

// Puerto en el que se ejecutará el servidor
const port = 3000;

// Configurar sesión de usuario
app.use(session({
    secret: 'mi_secreto_secreto', // Clave secreta para firmar la cookie de sesión
    resave: true, // Forzar a la sesión a guardar
    saveUninitialized: true // Guardar la sesión incluso si está vacía
}));

// Configurar body-parser para analizar datos de formulario
app.use(bodyParser.urlencoded({ extended: true }));

// Manejar solicitud POST para inicio de sesión
app.post('/login', (req, res) => {
    // Obtener datos de formulario (nombre de usuario y contraseña)
    const { username, password } = req.body;

    // Credenciales válidas predefinidas
    const validUsername = 'usuario'; // Nombre de usuario válido
    const validPassword = 'contraseña'; // Contraseña válida

    // Verificar credenciales
    if (username === validUsername && password === validPassword) {
        // Establecer variable de sesión para indicar inicio de sesión
        req.session.loggedin = true;

        // Redirigir al usuario a una página de bienvenida
        res.redirect('/bienvenida');
    } else {
        // Mostrar mensaje de error si las credenciales son incorrectas
        res.render('login', { error: 'Nombre de usuario o contraseña incorrectos.' });
    }
});

// Servir página de inicio de sesión
app.get('/', (req, res) => {
    // Renderizar página de inicio de sesión, pasando un mensaje de error vacío
    res.render('login', { error: '' });
});

// Iniciar el servidor y escuchar en el puerto especificado
app.listen(port, () => {
    console.log(`Servidor iniciado en http://localhost:${port}`);
});
```

Nota. *La figura ilustra la configuración del servidor web y el manejo de inicio de sesión. Fuente: elaboración propia.*

3.4.7 Sitios web para abordar los temas vistos anteriormente como material de apoyo

1. https://developer.okta.com/blog/2019/05/02/everything-you-wanted-to-know-about-node-dot-js-sessions
   En la página de Okta, se puede encontrar una guía exhaustiva sobre la gestión de sesiones en Node.js. El artículo aborda temas importantes como las posibles amenazas de seguridad relacionadas con la gestión de sesiones, incluyendo la predicción de sesiones, el sniffing de sesiones, los ataques de hombre en el medio y los ataques del lado del cliente. También discute las mejores prácticas para la gestión de sesiones, como el uso de un secreto de sesión para mejorar la seguridad, la expiración de sesiones y la regeneración del ID de sesión tras el inicio de sesión del usuario. Además, ofrece una visión detallada sobre cómo utilizar bibliotecas de gestión de sesiones en Node.js, con un enfoque específico en express-session, una de las bibliotecas más maduras y utilizadas para este propósito en el ecosistema de Node.js.

2. https://www.php.net/manual/es/book.session.php
   En la página de PHP.net dedicada a las sesiones, se puede encontrar la documentación oficial sobre el manejo de sesiones en PHP. Este recurso abarca desde los fundamentos de las sesiones, cómo iniciar y destruir una sesión, hasta cómo configurar y utilizar variables de sesión para mantener el estado a lo largo de múltiples solicitudes. Es un recurso esencial para desarrolladores que buscan comprender y aplicar prácticas de gestión de sesiones seguras y eficaces en sus aplicaciones web PHP.

3. https://docs.djangoproject.com/en/3.2/topics/http/sessions/

En esta página, se encuentra la documentación de Django que ofrece una guía detallada sobre cómo Django maneja las sesiones. La sección explica el flujo de trabajo de las sesiones en Django, incluyendo cómo almacenar, recuperar y eliminar datos de sesión. También se abordan temas avanzados como la configuración de la sesión, la personalización del almacenamiento de sesión y la protección contra ataques. Esta guía es invaluable para los desarrolladores que trabajan con Django y buscan aprovechar las capacidades de gestión de sesiones del framework para construir aplicaciones web robustas y seguras.

Conectividad entre el servidor Web y el servidor de base de datos

La interacción entre el servidor web y el servidor de base de datos es un componente esencial en el desarrollo y operación de aplicaciones web, facilitando el flujo de datos necesario para responder a las solicitudes de los usuarios. En este contexto, scripts en lenguajes como PHP, Python o Node.js, alojados en el servidor web, realizan operaciones de lectura, escritura, actualización y borrado en bases de datos mediante sistemas de gestión como MySQL, PostgreSQL y MongoDB. Cada uno de estos sistemas ofrece

características únicas, adecuadas para distintos tipos de aplicaciones web.

Chávez (2020) proporciona un análisis profundo de PostgreSQL, destacando su organización multinivel con un clúster en el nivel más alto, lo que permite una administración de datos y operaciones de base de datos eficientes. Resalta la capacidad de PostgreSQL para manejar consultas complejas y un almacenamiento de datos avanzado, ideal para aplicaciones que demandan un procesamiento intensivo de datos.

Por su parte, Rawat et al. (2021) identifican a MySQL como el software de base de datos más utilizado en el mundo, respaldado por Oracle Corporation. MySQL es conocido por su rendimiento, confiabilidad y simplicidad, lo que lo hace preferido para muchas aplicaciones web. Krogh (2020) explica que, cuando una aplicación necesita ejecutar una consulta o almacenar datos en MySQL, envía una solicitud a través de la red hacia MySQL. Este, a su vez, trabaja junto con el sistema operativo y utiliza recursos del sistema, como la memoria y los discos, para procesar la solicitud. Una vez que el proceso ha terminado y el resultado está listo, MySQL lo envía de vuelta a la aplicación a través de la red.

La elección adecuada de un sistema de gestión de bases de datos y la aplicación de buenas prácticas en la conectividad entre las bases de datos y los servidores web son decisivas para el éxito de las aplicaciones web, impactando directamente en su eficiencia, seguridad y la experiencia del usuario. La integración efectiva y una gestión cuidadosa de esta conexión son esenciales, con sistemas como MySQL, PostgreSQL y MongoDB ofreciendo soluciones robustas para estos requerimientos.

Para configurar una conexión efectiva con MySQL, se deben seguir pasos críticos que incluyen la adecuada configuración de MySQL, la instalación de los drivers o conectores necesarios en el servidor web y el uso de extensiones o módulos específicos que faciliten la comunicación con MySQL, variando según el lenguaje de programación utilizado. La conexión se establece mediante la especificación de parámetros clave como el host, el nombre de usuario, la contraseña y el nombre de la base de datos.

A continuación, se desarrollarán ejemplos prácticos en PHP, Django (Python) y Node.js para ilustrar la forma de establecer una conexión efectiva entre un servidor web y un servidor de base de datos MySQL, y se explicarán los pasos esenciales para configurar esta conexión y asegurar un inicio adecuado.

Este fragmento de código PHP inicia una conexión a una base de datos MySQL, empleando parámetros de conexión como el host, el nombre de la base de datos, el nombre de usuario y la contraseña. Procede a comprobar si la conexión se ha establecido correctamente. En caso de encontrar un error de conexión, despliega un mensaje de error y detiene la ejecución del script. Si la

Figura 26.
Conexión a un almacenamiento de datos con PHP.

```php
<?php
// Datos de conexión a la base de datos MySQL
$servername = "localhost"; // Nombre del servidor de base de datos
$username = "usuario"; // Nombre de usuario de la base de datos
$password = "contraseña"; // Contraseña de la base de datos
$dbname = "nombre_base_de_datos"; // Nombre de la base de datos

// Crear conexión
$conn = new mysqli($servername, $username, $password, $dbname);

// Verificar conexión
if ($conn->connect_error) {
    // Si hay un error de conexión, muestra un mensaje de error y termina el script
    die("Error de conexión: " . $conn->connect_error);
}

// Si la conexión es exitosa, muestra un mensaje de éxito
echo "Conexión exitosa a la base de datos MySQL";

// Cerrar conexión
$conn->close(); // Cierra la conexión con la base de datos para liberar recursos
?>
```

Nota. *La figura ilustra la configuración en PHP para conectar con una base de datos MySQL.*
Fuente: elaboración propia.

conexión es exitosa, muestra un mensaje indicando el éxito y, posteriormente, cierra la conexión para liberar los recursos utilizados. A continuación, se abordará una tarea similar utilizando Django:

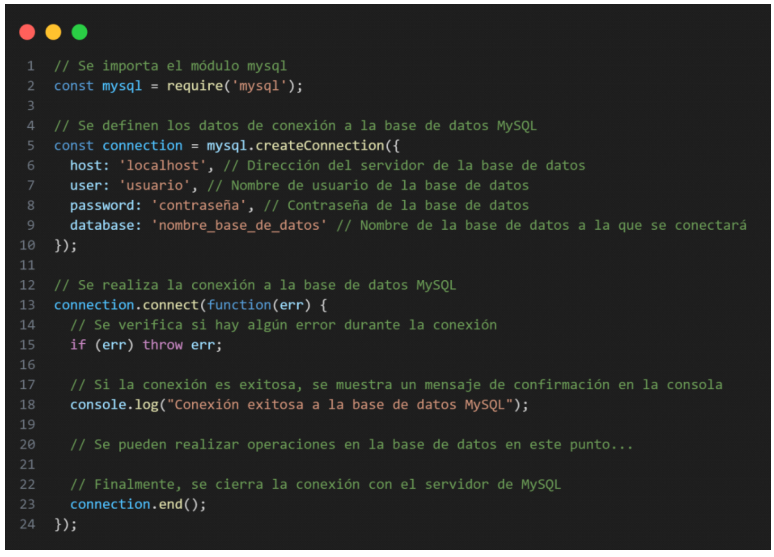
Figura 27.
Gestión del proceso para iniciar sesión en Express.

```python
# Se importa el módulo mysql.connector
import mysql.connector

# Se definen los datos de conexión a la base de datos MySQL
conexion = mysql.connector.connect(
    host='localhost',  # Dirección del servidor de la base de datos
    user='usuario',  # Nombre de usuario de la base de datos
    password='contraseña',  # Contraseña de la base de datos
    database='nombre_base_de_datos'  # Nombre de la base de datos a la que se conectará
)

# Se verifica si la conexión fue exitosa
if conexion.is_connected():
    print("Conexión exitosa a la base de datos MySQL")

    # Se pueden realizar operaciones en la base de datos en este punto...

    # Finalmente, se cierra la conexión con el servidor de MySQL
    conexion.close()
```

Nota. *La figura muestra la configuración en Django para conectar con una base de datos MySQL.*
Fuente: elaboración propia.

Este código realiza la conexión a una base de datos MySQL utilizando el módulo mysql.connector en Python, especificando los detalles de conexión como el host, el nombre de la base de datos, el nombre de usuario y la contraseña. Posteriormente, se verifica el éxito de la conexión y, en caso afirmativo, se emite un mensaje de confirmación. Para concluir, se emplea el método close() para cerrar la conexión. Por último, se llevará a cabo la misma tarea utilizando Node.js.

Figura 28.
Conexión a un almacenamiento de datos con Node.js.



```
// Se importa el módulo mysql
const mysql = require('mysql');

// Se definen los datos de conexión a la base de datos MySQL
const connection = mysql.createConnection({
  host: 'localhost', // Dirección del servidor de la base de datos
  user: 'usuario', // Nombre de usuario de la base de datos
  password: 'contraseña', // Contraseña de la base de datos
  database: 'nombre_base_de_datos' // Nombre de la base de datos a la que se conectará
});

// Se realiza la conexión a la base de datos MySQL
connection.connect(function(err) {
  // Se verifica si hay algún error durante la conexión
  if (err) throw err;

  // Si la conexión es exitosa, se muestra un mensaje de confirmación en la consola
  console.log("Conexión exitosa a la base de datos MySQL");

  // Se pueden realizar operaciones en la base de datos en este punto...

  // Finalmente, se cierra la conexión con el servidor de MySQL
  connection.end();
});
```

Nota. *La figura muestra la configuración en Node.js para conectar con una base de datos MySQL.*
Fuente: elaboración propia.

Este fragmento de código crea una conexión a una base de datos MySQL utilizando el módulo mysql de Node.js, especificando los parámetros necesarios para la conexión, incluyendo el host, el nombre de la base de datos, el nombre de usuario y la contraseña. Posteriormente, ejecuta la conexión empleando el método connect. En caso de establecerse la conexión exitosamente, se visualiza un mensaje de confirmación en la consola. Para finalizar, se procede a cerrar la conexión con el método end. Por último, se proporcionarán tres sitios web donde se puede profundizar más sobre el tema:

Sitios web

1. https://www.php.net/manual/es/mysqli.quickstart.connections.php
   En el sitio web se puede encontrar información detallada sobre cómo establecer conexiones a bases de datos MySQL utilizando la extensión mysqli de PHP. La guía rápida de conexiones introduce los pasos necesarios, mientras que la sección de opciones de conexión proporciona detalles sobre la configuración de parámetros como el nombre del host, el usuario,

la contraseña y el nombre de la base de datos. Además, se explica el proceso para establecer la conexión y cómo manejar errores. La página también incluye ejemplos de código para diferentes escenarios y enlaces a recursos adicionales, como el manual completo de PHP, tutoriales sobre mysqli y foros de PHP donde los desarrolladores pueden encontrar asistencia. En resumen, este sitio web es un recurso valioso para cualquier desarrollador que desee aprender a usar mysqli para conectarse a bases de datos MySQL con PHP.

2. https://parzibyte.me/blog/2019/07/30/tutorial-django-bases-datos-migraciones-modelos/
En el sitio web se puede encontrar una guía completa sobre cómo trabajar con bases de datos en Django, el framework web de Python. La guía cubre aspectos fundamentales como la configuración de bases de datos, la creación de modelos, el uso del sistema de migraciones, la realización de consultas a la base de datos y ejemplos prácticos para aplicar los conceptos aprendidos. Con esta información, los desarrolladores pueden construir aplicaciones web robustas y escalables en Django, gestionando la información de manera eficiente y estableciendo una sólida base de conocimientos sobre bases de datos relacionales.

3. https://keepcoding.io/blog/conexion-a-bases-de-datos-en-node-js/
En el sitio web mencionado, se encuentra un tutorial completo sobre cómo conectar Node.js a una base de datos MySQL. El tutorial cubre los fundamentos de las bases de datos, proporciona una introducción a Node.js como entorno de ejecución para JavaScript en el lado del servidor, y destaca las ventajas de utilizar Node.js para esta tarea. Además, se enseña a instalar el módulo mysql de Node.js, crear una conexión a la base de datos, manejar errores de conexión y realizar consultas SQL, incluyendo SELECT, INSERT, UPDATE y DELETE. Se incluyen ejemplos prácticos de código para comprender y aplicar los conceptos. Por último, se ofrecen recursos adicionales, como la documentación oficial de Node.js y del módulo mysql, así como un foro de la comunidad Node.js. Este tutorial proporciona una base sólida para desarrollar aplicaciones web robustas y escalables con una gestión eficiente de la información.

3.4.8 Manejo de archivos y Seguridad

En el ámbito de las aplicaciones web, la administración de archivos y la seguridad son fundamentales para garantizar la protección y eficacia en el manejo de datos transmitidos y almacenados. Según Gomez (2023), la seguridad informática se define como un conjunto de prácticas, herramientas y técnicas destinadas a proteger la confidencialidad, integridad y acceso de los datos e información almacenados y gestionados en un sistema informático. Esta perspectiva destaca la necesidad de un enfoque integral que abarque tanto la funcionalidad como la seguridad de las aplicaciones web, para mantener la integridad, disponibilidad y confidencialidad de la información frente a un panorama de amenazas constantes y cambiantes.

La gestión de archivos en la web, que incluye operaciones de carga y descarga por parte de los usuarios, así como el almacenamiento seguro y la gestión adecuada de estos archivos, subraya la importancia de implementar interfaces de usuario intuitivas y mantener una estructura organizada que optimice tanto el rendimiento como la accesibilidad. Esto implica organizar los archivos en estructuras de directorios coherentes y aplicar estrategias de respaldo. Según López (2022), las vulnerabilidades en los sistemas son características propias que afectan a las organizaciones, volviéndolas susceptibles a amenazas.

En términos de seguridad, las aplicaciones web enfrentan desafíos únicos, como la vulnerabilidad a ataques de inyección de código. Bugoi et al. (2023) describen cómo el proceso de carga de archivos (file upload) puede convertirse en un punto crítico al permitir a los usuarios subir archivos al servidor sin una validación adecuada. Para mitigar estos riesgos, es esencial implementar controles de acceso rigurosos, cifrar los archivos para proteger datos sensibles, y aplicar procedimientos de validación y sanitización para prevenir la ejecución de código malicioso.

Se construirá una vista en HTML que permite a los usuarios cargar archivos en una aplicación web. Esta vista incluye un formulario con un campo de entrada de tipo "file", que permite a los usuarios seleccionar un archivo de su dispositivo local. Cuando se envía el formulario, el archivo seleccionado se envía al servidor web para su procesamiento. La etiqueta "enctype" con el valor "multipart/form-data" se utiliza para habilitar la carga de archivos en el formulario. Esta vista es parte de la interfaz de usuario de la aplicación y ofrece una forma intuitiva para que los usuarios interactúen con la funcionalidad de carga de archivos.

Figura 29.
Interfaz de usuario para cargar archivos.

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Subir Archivo</title>
</head>
<body>
    <h2>Subir Archivo</h2>
    <form action="upload.php" method="post" enctype="multipart/form-data">
        <label for="archivo">Seleccionar archivo:</label>
        <input type="file" id="archivo" name="archivo"><br><br>
        <input type="submit" value="Subir Archivo">
    </form>
</body>
</html>
```

Nota. *La figura muestra la estructura del formulario para la carga de archivos. Fuente: elaboración propia.*

Luego se manejará la carga del archivo enviado desde el formulario HTML utilizando un script en PHP. El script comienza verificando si se ha enviado un archivo mediante la comprobación del array $_FILES. Extrae detalles del archivo, como el nombre original y la ruta temporal en el servidor. A continuación, define una lista de extensiones permitidas y verifica si la extensión del archivo se encuentra en esa lista. Si la extensión es válida, el archivo se mueve a una ubicación específica en el servidor mediante la función move_uploaded_file(). En caso de que la extensión del archivo no sea permitida, se muestra un mensaje de error que indica los tipos de archivos aceptados. Este script es esencial para aplicaciones web que requieren validar y gestionar los archivos cargados por los usuarios, asegurando que solo se acepten tipos de archivos permitidos y evitando la carga de archivos potencialmente dañinos.

Figura 30.
Manipulación para la carga de archivos en PHP.

```php
<?php
// Verificar si se ha enviado un archivo
if(isset($_FILES['archivo'])){
    $file_name = $_FILES['archivo']['name']; // Nombre del archivo
    $file_tmp = $_FILES['archivo']['tmp_name']; // Ruta temporal del archivo
    $file_destination = "uploads/" . $file_name; // Ruta de destino del archivo en el servidor

    // Extensiones permitidas
    $allowed_extensions = array('jpg', 'jpeg', 'png', 'gif', 'pdf', 'doc', 'docx', 'txt');

    // Obtener la extensión del archivo
    $file_extension = strtolower(pathinfo($file_name, PATHINFO_EXTENSION));

    // Verificar si la extensión está permitida
    if(in_array($file_extension, $allowed_extensions)){
        // Mover el archivo del directorio temporal al directorio de destino
        move_uploaded_file($file_tmp, $file_destination);
        echo "El archivo se ha cargado correctamente.";
    } else {
        echo "Error: Solo se permiten archivos JPG, JPEG, PNG, GIF, PDF, DOC, DOCX y TXT.";
    }
}
?>
```

Nota. *La figura muestra cómo se manipula la carga de archivos en PHP. Fuente: elaboración propia.*

De manera similar, se implementará una vista en Flask para manejar la carga de archivos con algunas adaptaciones. Se creará una ruta en Flask que responda a solicitudes POST en la URL /upload. El formulario HTML asociado permitirá a los usuarios seleccionar y enviar un archivo al servidor. El atributo enctype="multipart/form-data" se usará en el formulario para habilitar el envío de archivos binarios.

Para manejar la carga de archivos en Flask, podemos definir una función para verificar los tipos de archivos permitidos, establecer una ruta para manejar la carga y guardar los archivos en una carpeta específica en el servidor. A continuación se muestra cómo se puede implementar esto en un proyecto Flask:

Figura 31.
Interfaz de usuario para cargar archivos.

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Subir Archivo</title>
</head>
<body>
    <h2>Subir Archivo</h2>
    <form action="/upload" method="post" enctype="multipart/form-data">
        <label for="archivo">Seleccionar archivo:</label>
        <input type="file" id="archivo" name="archivo"><br><br>
        <input type="submit" value="Subir Archivo">
    </form>
</body>
</html>
```

Nota. *La figura muestra la estructura del formulario para la carga de archivos.*
Fuente: elaboración propia.

Figura 32.
Manipulación para la carga de archivos en Express.

```javascript
const express = require('express'); // Importa el módulo Express
const multer = require('multer'); // Importa el módulo Multer para el manejo de archivos
const path = require('path'); // Importa el módulo Path para el manejo de rutas de archivos

const app = express(); // Crea una instancia de la aplicación Express
const PORT = 3000; // Puerto en el que se ejecutará el servidor

// Configuración de Multer para la carga de archivos en el directorio 'uploads/'
const upload = multer({ dest: 'uploads/' });

// Extensiones de archivo permitidas
const allowedExtensions = ['jpg', 'jpeg', 'png', 'gif', 'pdf', 'doc', 'docx', 'txt'];

// Función para verificar si la extensión del archivo es permitida
function allowedFile(filename) {
    const ext = path.extname(filename).toLowerCase();
    return allowedExtensions.includes(ext.substring(1));
}

// Middleware para servir archivos estáticos desde el directorio 'public/'
app.use(express.static('public'));

// Ruta para manejar la carga de archivos mediante POST
app.post('/upload', upload.single('archivo'), (req, res) => {
    if (!req.file) {
        return res.send('No se ha seleccionado ningún archivo.');
    }

    if (!allowedFile(req.file.originalname)) {
        return res.send('Error: Solo se permiten archivos JPG, JPEG, PNG, GIF, PDF, DOC, DOCX y TXT.');
    }

    res.send('El archivo se ha cargado correctamente.');
});

// Inicia el servidor y escucha en el puerto especificado
app.listen(PORT, () => {
    console.log(`Servidor iniciado en http://localhost:${PORT}`);
});
```

Nota. *La figura muestra cómo se manipula la carga de archivos en Express. Fuente: elaboración propia.*

a. Definir la Función para Verificar Tipos de Archivos Permitidos:
- Se crea una función para verificar si la extensión del archivo es una de las permitidas.

b. Configurar la Ruta para Manejar la Carga de Archivos:
- Se define una ruta que maneje tanto solicitudes GET (para mostrar el formulario de carga) como POST (para procesar la carga del archivo).

c. Guardar el Archivo en el Servidor:
- Se guarda el archivo en una carpeta específica y se muestra un mensaje de éxito o error basado en el resultado de la operación.

Para manejar la carga de archivos en Flask, donde los archivos se envían al servidor utilizando el método HTTP POST y se codifican como datos de formulario multipartes (multipart/form-data), se puede seguir el siguiente proceso:

1. Definir la Ruta y Función para Manejar la Carga:
   - Configura la ruta en Flask que manejará tanto las solicitudes GET (para mostrar el formulario) como POST (para procesar la carga del archivo).
   - La función asociada verificará el tipo de archivo, lo guardará en una carpeta específica y proporcionará mensajes de éxito o error.
2. Crear el Formulario HTML:
   - Define un formulario HTML que permita a los usuarios seleccionar un archivo y enviarlo al servidor.

Figura 33.
Interfaz de usuario para cargar archivos.

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Subir Archivo</title>
</head>
<body>
    <h2>Subir Archivo</h2>
    <form action="/" method="post" enctype="multipart/form-data">
        <label for="archivo">Seleccionar archivo:</label>
        <input type="file" id="archivo" name="archivo"><br><br>
        <input type="submit" value="Subir Archivo">
    </form>
</body>
</html>
```

Nota. *La figura muestra la estructura del formulario para la carga de archivos.*
Fuente: elaboración propia.

El código configura un servidor web utilizando Express y el middleware Multer para gestionar la carga de archivos. Se establece una lista de extensiones de archivo permitidas y se verifica que los archivos cargados cumplan con estas extensiones. Además, el servidor sirve archivos estáticos desde un

directorio denominado 'public'.

Configuración del Middleware Multer:

1. Multer se utiliza para manejar la carga de archivos y se configura para almacenar los archivos en un directorio específico (uploads).
2. Se define una lista de extensiones permitidas y se crea una función de filtro para asegurarse de que los archivos cargados sean del tipo permitido.

Archivo de Configuración:

- El servidor escucha en el puerto 3000.
- Se incluye una página HTML para la carga de archivos, utilizando el atributo enctype="multipart/form-data" para enviar archivos binarios al servidor.
- Cuando un archivo es cargado, el middleware verifica si cumple con las extensiones permitidas y, en consecuencia, devuelve un mensaje de éxito o un error si el archivo no está permitido.

El código proporciona un enfoque seguro para la carga de archivos al servidor, asegurando que solo se acepten tipos de archivos específicos. Este proceso incluye la configuración de Multer para la validación de archivos, el servicio de archivos estáticos desde el directorio 'public', y la ejecución del servidor en el puerto 3000.

Figura 33.

Manipulación para la carga de archivos en Flask.

```python
from flask import Flask, render_template, request
import os

app = Flask(__name__)

UPLOAD_FOLDER = 'uploads/'
ALLOWED_EXTENSIONS = {'jpg', 'jpeg', 'png', 'gif', 'pdf', 'doc', 'docx', 'txt'}

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

@app.route('/', methods=['GET', 'POST'])
def upload_file():
    if request.method == 'POST':
        if 'archivo' not in request.files:
            return 'No se ha seleccionado ningún archivo.'

        file = request.files['archivo']
        if file.filename == '':
            return 'No se ha seleccionado ningún archivo.'

        if file and allowed_file(file.filename):
            filename = file.filename
            file.save(os.path.join(UPLOAD_FOLDER, filename))
            return 'El archivo se ha cargado correctamente.'
        else:
            return 'Error: Solo se permiten archivos JPG, JPEG, PNG, GIF, PDF, DOC, DOCX y TXT.'

    return render_template('upload.html')

if __name__ == '__main__':
    app.run(debug=True)
```

Nota. *La figura muestra cómo se manipula la carga de archivos en Flask. Fuente: elaboración propia.*

Aquí se presentan tres sitios web recomendados para profundizar en el manejo de archivos y la seguridad en aplicaciones web:

Sitios Web

1. https://code.tutsplus.com/es/how-to-upload-a-file-in-php-with-example--cms-31763t
En este sitio web se ofrece una guía completa sobre cómo subir archivos a un servidor utilizando PHP. El contenido abarca desde los fundamentos básicos de la subida de archivos hasta ejemplos prácticos y consideraciones de seguridad. Se exploran técnicas para subir archivos tanto de forma simple como a través de múltiples campos, con ejemplos de código que facilitan la comprensión de los conceptos y su aplicación en proyectos propios. Además, se proporcionan recursos adicionales para profundizar en el tema y desarrollar aplicaciones web robustas y seguras.

2. https://www.delftstack.com/es/howto/django/django-upload-file-or-image/
El sitio web proporciona una guía detallada sobre cómo cargar archivos e imágenes en Django, un framework de desarrollo web en Python. La guía abarca desde la configuración inicial del entorno de Django hasta la creación de un modelo para almacenar los archivos cargados. Incluye la configuración de un formulario para la carga de archivos y la visualización de estos archivos en la aplicación. Además, se aborda la configuración necesaria en el archivo settings.py para manejar archivos multimedia, así como la creación de vistas para procesar la carga y mostrar los archivos. Esta guía está diseñada como un tutorial práctico para desarrolladores interesados en implementar funcionalidades de carga de archivos en sus aplicaciones web con Django.

3. https://ed.team/blog/como-subir-archivos-al-servidor-con-nodejs
En el sitio web de EDteam, se presenta un artículo que explica detalladamente cómo subir archivos al servidor utilizando Node.js. El artículo cubre el uso de librerías como Multer y Express-fileupload para facilitar el proceso. A través de un tutorial paso a paso, el autor guía a los lectores en la creación de un proyecto Node.js, la instalación de las dependencias necesarias y la configuración de una ruta para cargar archivos. Además, se muestra cómo guardar los archivos en un directorio específico del servidor y cómo crear un formulario HTML para la carga de archivos. Este recurso es útil para desarrolladores que buscan implementar la funcionalidad de carga de archivos en sus aplicaciones web usando Node.js.

Actividades de Evaluación

Este documento educativo se enfoca en el desarrollo de habilidades fundamentales en la programación web del lado del servidor, dividiéndose en dos partes esenciales: la programación de formularios y la conectividad con bases

de datos. A continuación, se describen los componentes y objetivos de aprendizaje de cada sección.

Programación de Formularios

Componentes Clave:

- Creación de un Formulario Web: Diseño de un formulario utilizando HTML que incluya campos para el nombre de usuario, contraseña, y una selección que represente una consulta o preferencia del usuario, junto con un botón de envío.
- Procesamiento del Formulario en el Servidor: Implementación de lógica en el servidor para recibir y procesar los datos del formulario, incluyendo validación de los campos ingresados y generación de una respuesta acorde.
- Respuesta del Servidor: Envío de un mensaje de bienvenida con el nombre de usuario y su preferencia seleccionada si la validación es exitosa, o indicación de los errores a corregir si falla.
- Seguridad: Incorporación de medidas de seguridad como la sanitización de datos para prevenir ataques de inyección SQL y XSS.

Objetivos de Aprendizaje:

- Diseñar formularios web seguros y eficientes.
- Desarrollar la lógica del lado del servidor para el procesamiento seguro de datos de formularios.
- Implementar las validaciones de datos en el servidor para asegurar la integridad de los datos.
- Aplicar las medidas de seguridad en el procesamiento de formularios.

Conectividad entre el Servidor Web y el Servidor de Base de Datos.

Componentes Clave:

- Establecimiento de Conexión: Desarrollo de un script en el servidor para conectar de manera segura con una base de datos, especificando los parámetros de conexión necesarios.
- Operaciones CRUD: Implementación de funciones para realizar operaciones CRUD en la base de datos, valorando la claridad y eficiencia del código.
- Integración con el Front-end: Creación de un formulario web y elementos de interacción para que los usuarios realicen operaciones CRUD, con una interfaz intuitiva y feedback adecuado.
- Seguridad y Validación de Datos: Implementación de medidas de seguridad para proteger la conexión y los datos, incluyendo la sanitización y validación de los datos del formulario para prevenir ataques.

Objetivos de Aprendizaje:

- Realizar conexiones seguras entre el servidor web y las bases de datos.
- Ejecutar operaciones CRUD desde el servidor web.

- Integrar la lógica del servidor con interfaces de usuario para aplicaciones dinámicas.
- Implementar la seguridad y validación de datos para proteger la información y la integridad de la aplicación.

Esta actividad permite a los estudiantes demostrar su habilidad para desarrollar componentes web interactivos y seguros, enfatizando la comunicación efectiva entre el cliente y el servidor y la adopción de prácticas de seguridad críticas en el desarrollo web.

Referencias Bibliográficas

Arango Gómez, O. D. (2023). El ABC de la seguridad informática: guía práctica para entender la seguridad digital. Recuperado de https://www.autoreseditores.com/libro/22997/oscar-dario-arango-gomez/el-abc-de-la-seguridad-informatica-guia-practica-para-entender.html.

Bugoi, R. C., & Esquinas Puche, P. (2023). Sitio web seguro frente a ciberataques. Recuperado el 7 de abril de 2024 https://docta.ucm.es/rest/api/core/bitstreams/d0496fa4-17e0-4f55-9538-636d7260600b/content

Chávez Vergara, L. D. (2023). Análisis y evaluación de las vulnerabilidades y riesgos asociados con las cookies de inicio de sesión y de terceros en sitios web (Trabajo de licenciatura, Babahoyo: UTB-FAFI). Recuperado de: http://dspace.utb.edu.ec/bitstream/handle/49000/14976/E-UTB-FAFI-SIST.INF-000162.pdf?sequence=1&isAllowed=y

Chávez, J. D. (2020). Cliente PSQL de PostgreSQL. Venezuela: JEASS Editores. Recuperado de https://www.researchgate.net/profile/Jorge-Domingu ez-Chavez/publication/340548281_CLIENTE_PSQL_DE_POSTGRESQL/links/5e904f72299bf130798dba80/CLIENTE-PSQL-DE-POSTGRESQL.pdf

Cloudflare. (2024). ¿Qué significa lado del cliente y lado del servidor? Lado del cliente vs. Lado del servidor. Recuperado el 31 de marzo de 2024, de https://www.cloudflare.com/es-es/learning/serverless/glossary/client-side-vs-server-side/

DelftStack. (2023). Cómo subir un archivo o imagen en Django. Recuperado el 31 de marzo de 2024, de https://www.delftstack.com/es/howto/django/django-upload-file-or-image/

DesarrolloWeb.com. (2020). El operador de concatenación en PHP. Recuperado el 31 de marzo de 2024, de https://desarrolloweb.com/articulos/317.php

DesarrolloWeb.com. (2016). Operadores en PHP. Recuperado el 31 de marzo de 2024, de https://desarrolloweb.com/articulos/operadores-php.html

Django Documentation. (2024). Recuperado el 31 de marzo de 2024, de https://docs.djangoproject.com/en/5.0/topics/http/sessions/

Django Documentation. (2024). Recuperado el 31 de marzo de 2024, de https://docs.djangoproject.com/en/5.0/contents/ -:~:text=Django%20documentation%20contents%20%C2%B6%201%20Getting%20started%20Django,code%20Models%20and%20databases%20Models%20Making%20queries%20

ED.team. (2018). Cómo subir archivos al servidor con Node.js. Recuperado el 31 de marzo de 2024, de https://ed.team/blog/como-subir-archivos-al-servidor-con-nodejs

Express.js. (2017). Recuperado de https://expressjs.com/

J2Logo. (2023). Tutorial de operadores en Python. Recuperado el 31 de marzo de 2024, de https://j2logo.com/python/tutorial/operadores-en-python/

KeepCoding.io. (2024). Conexión a bases de datos en Node.js. Recuperado el 31 de marzo de 2024, de https://keepcoding.io/blog/conexion-a-bases-de-datos-en-node-js/

Krogh, J. W., & Krogh, J. W. (2020). MySQL Workbench. Ajuste de rendimiento de consultas MySQL 8: Un método sistemático para mejorar la velocidad de ejecución, 199-226. https://doi.org/10.1007/978-1-4842-5584-1

Laravel News. (2024). Laravel News. Recuperado el 31 de marzo de 2024, el https://laravel-news.com/

Llerena Izquierdo, J. (2023). Guía de aprendizaje de programación. https://dspace.ups.edu.ec/handle/123456789/24037

López Sempere, A. (2022). Identificación y detección de vulnerabilidades (Doctoral dissertation, Universitat Politècnica de València). Recuperado el 31 de marzo de 2024, de https://riunet.upv.es/bitstream/handle/10251/183781/Lopez%20-%20Identificacion%20y%20deteccion%20de%20vulnerabilidades.pdf?sequence=1&isAllowed=y

Londoño-Rojas, L. F., Tabares-Morales, V., Rosecler-Bez, M., & Duque-Méndez, N. D. (2021). Guías prácticas y herramientas para apoyar el desarrollo de sitios web accesibles. Revista Científica, (41), 225-241. Recuperado de http://www.scielo.org.co/pdf/cient/n41/2344-8350-cient-41-225.pdf

MDN Web Docs. (2022). Programación lado servidor. Recuperado el 31 de marzo de 2024, de https://developer.mozilla.org/es/docs/Learn/Server-side/First_steps/Introduction

Node.js. (2024). Recuperado de https://nodejs.org/

Parzibyte.me. (2019). Tutorial de Django: Bases de datos, migraciones y modelos. Recuperado el 31 de marzo de 2024, de https://parzibyte.me/blog/2019/07/30/tutorial-django-bases-datos-migraciones-modelos/

Rawat, B., & Purnama, S. (2021). Sistema de Gestión de Bases de Datos MySQL (DBMS) en el Sitio FTP LAPAN Bandung. International Journal of Cyber and IT Service Management, 1(2), 173-179. Recuperado el 31 de marzo de 2024, de https://iiast.iaic-publisher.org/ijcitsm/index.php/IJCITSM/article/view/47/16

PHP.net. (2024). PHP Sessions. Recuperado de https://www.php.net/manual/es/book.session.php

PHP.net. (2024). Quickstart: Conexiones a la base de datos MySQLi en PHP. Recuperado el 31 de marzo de 2024, de https://www.php.net/manual/es/mysqli.quickstart.connections.php

Python.org. (2024). Recuperado el 31 de marzo de 2024, de https://www.python.org/

Ríos Pérez, F. E., Polanco Carrillo, F., & Moreno Vega, V. (2017). Servidor web empotrado en un FPGA para configurar un Controlador Maestro del Sistema Inteligente de Tráfico Cubano. Revista Cubana de Ciencias Informáticas, 11(2), 16-28. http://scielo.sld.cu/pdf/rcci/v11n2/rcci02217.pdf

Salazar Guaña, C. E. (2022). Lógica de Programación. Recuperado el 31 de marzo de 2024, de https://dspace.itsjapon.edu.ec/jspui/bitstream/123456789/3764/1/LOGICA_DE_PROGRAMACION.pdf

Sansano Miralles, H. (2017). Creación de un software de pruebas de carga y funcionamiento. Recuperado el 31 de marzo de 2024, de https://rua.ua.es/dspace/bitstream/10045/69468/1/Creacion_de_un_software_de_pruebas_de_carga_y_funcio_SANSANO_MIRALLES_HECTOR.pdf

Stack Overflow - where developers learn, share, & build careers. (2024). Stack Overflow. https://stackoverflow.com/

TutsPlus. (2020). Cómo subir un archivo en PHP con ejemplo. Recuperado el 31 de marzo de 2024, de https://code.tutsplus.com/es/how-to-upload-a-file-in-php-with-example--cms-31763t

Uniwebsidad. (2024). Capítulo 3: Operadores en JavaScript. Recuperado el 31 de marzo de 2024, de https://uniwebsidad.com/libros/javascript/capitulo-3/operadores

Valdivia-Caballero, J. J. (2016). Modelo de procesos para el desarrollo del front-end de aplicaciones web. Interfases, (009), 187-208. https://revistas.ulima.edu.pe/index.php/Interfases/article/view/1245/1205

W3Schools online web tutorials. (2024). Recuperado el 31 de marzo de 2024, de https://www.w3schools.com/

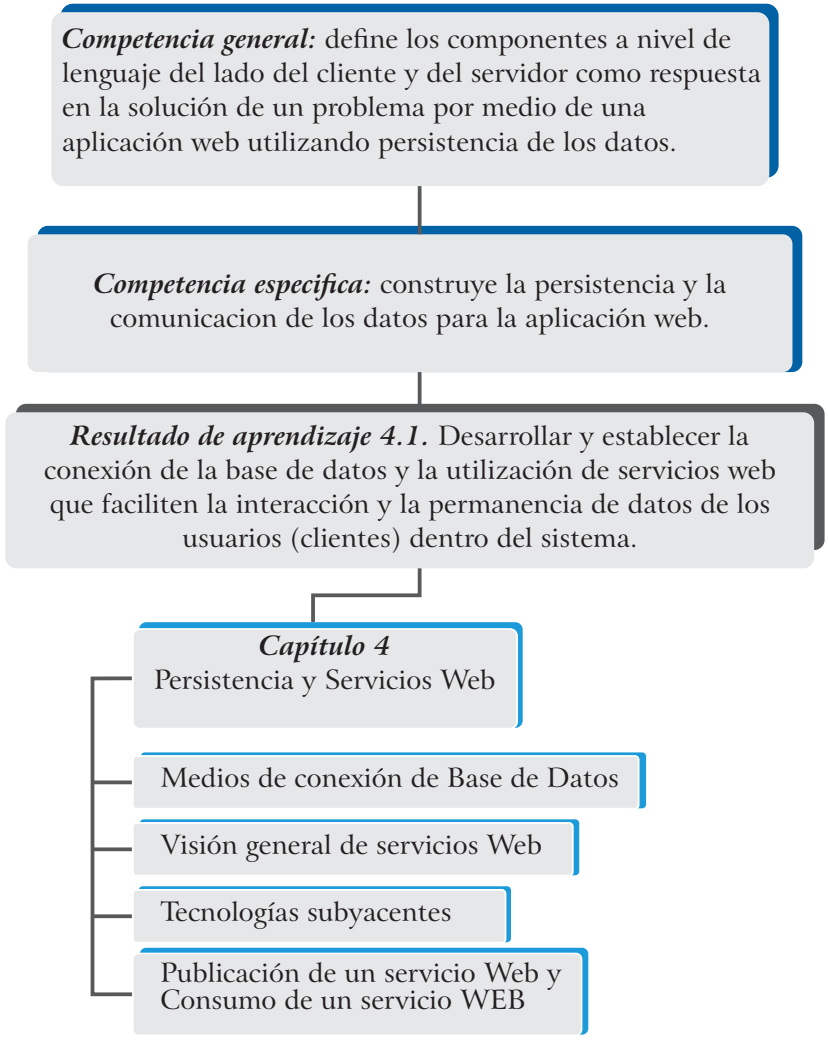
CAPÍTULO 4.

PERSISTENCIA Y SERVICIOS WEB

Juan Sebastián Mejía Suaza
Fredy Antonio Verástegui González
Denis Lorena Álvarez Guayara

[1]*Estudiante del programa de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad de la Amazonia, correo: juans.mejia@udla.edu.co*
[2]*Docente de tiempo completo del programa de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad de la Amazonia, correo: f.verastegui@udla.edu.co*
[3]*Docente de tiempo completo del programa de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad de la Amazonia, correo: d.alvarez@udla.edu.co*

ESQUEMA | CAPÍTULO 4.



COMPETENCIA Y RESULTADO DE APRENDIZAJE | 4. CAPÍTULO

La competencia general está establecida de la siguiente forma: "Define los componentes a nivel de lenguaje del lado del cliente y del servidor como respuesta en la solución de un problema por medio de una aplicación web utilizando persistencia de los datos.". Asociada a la siguiente competencia específica para la cual se desarrolla en el capítulo cuatro, definida como: "Construye la persistencia y la comunicación de los datos para la aplicación web". El resultado de aprendizaje vinculado al capítulo cuatro de competencia se ha formulado de la siguiente manera: resultado de aprendizaje 4.1. Se trata de desarrollar y precisar una conexión con la base de datos y emplear servicios web, lo que facilitará la comunicación y el almacenamiento de la información de los usuarios (clientes) dentro del sistema. Los estándares de desempeño establecidos para este objetivo de aprendizaje se presentan en la tabla número 4.1.

Tabla 4.1.
Niveles de desempeños del resultado de aprendizaje O3.1

Avanzado (5-4,8)	Intermedio (4,7-4)	Básico (3.9 -3)	Bajo (2.9-0)
Construye un sistema de persistencia y comunicación de los datos con un comportamiento eficiente en la operatividad de la aplicación web.	Construye un sistema de persistencia y comunicación de los datos con un comportamiento efectivo en la operatividad de la aplicación web.	Construye un sistema de persistencia y comunicación de los datos con un comportamiento no eficiente en la operatividad de la aplicación web.	Presenta dificultad en la construcción de un sistema de persistencia y comunicación de los datos con un ineficiente comportamiento operativo de la aplicación web.

Nota. *Trabajo realizado a partir del desarrollo de resultados de aprendizaje. Fuente: elaboración propia.*

Las actividades de evaluación determinadas para el resultado de aprendizaje son las siguientes: entrega de un proyecto relacionado con un sistema de información web utilizando los conceptos teóricos y prácticos del presente capítulo (Código fuente, documentación y sustentación).

Métodos y Materiales de Aprendizaje

La metodología adoptada en este capítulo de "Persistencia y Servicios Web" está diseñada para maximizar la eficacia del aprendizaje autónomo y apoyar la autonomía del estudiante en el dominio de las tecnologías de bases de datos y servicios web. A continuación, se detallan los enfoques y recursos utilizados: Lecturas Dirigidas y Material en Línea:

Se asignarán lecturas específicas y recursos en línea que cubren los fundamentos de la persistencia de datos y la arquitectura de servicios web, incluyendo

ODBC y JDBC. Los estudiantes deberán revisar estos materiales para prepararse para las sesiones interactivas y las evaluaciones.

Uso de Librerías y Herramientas de Programación:
Se fomentará el uso de diversas bibliotecas de programación como PyMySQL, psycopg2, y sqlite3 en Python, así como Connector/J para Java. Esto proporcionará una experiencia práctica en la conexión y manejo de bases de datos desde diferentes entornos de desarrollo.

Ejercicios Prácticos:
Para consolidar la teoría, se implementarán ejercicios prácticos que requieren el uso de ODBC y JDBC, permitiendo a los estudiantes diseñar y manipular bases de datos en escenarios reales y controlados.

Foros de Discusión:
Se establecerán foros en línea donde los estudiantes podrán discutir problemas, compartir recursos y resolver dudas de manera colaborativa, promoviendo un aprendizaje profundo y diversificado.

Evaluaciones Continuas:
Se realizarán evaluaciones periódicas para monitorear el progreso de los estudiantes a través de cuestionarios, proyectos prácticos y pruebas cortas centradas en los principales temas de la unidad.

Sesiones de Retroalimentación:
Después de cada evaluación mayor, los estudiantes recibirán retroalimentación detallada sobre su desempeño, con sugerencias específicas para mejorar en áreas clave.

Recursos Complementarios:
Se proporcionarán enlaces a tutoriales, webinars, y conferencias que complementen los temas tratados, tales como arquitecturas avanzadas de JDBC y mejores prácticas en el diseño de APIs.

Esta metodología está diseñada no solo para enseñar a los estudiantes cómo trabajar con tecnologías de bases de datos y servicios web, sino también para incitarlos a pensar críticamente sobre cómo estos sistemas se integran y funcionan en entornos reales y aplicados.

Desarrollo Temático del Capítulo
En esta unidad, se iniciará el estudio abordando las infraestructuras tecnológicas que permiten la persistencia de datos y la implementación de servicios web en aplicaciones contemporáneas. Se explorará detenidamente cómo las metodologías y tecnologías de conectividad, como ODBC y JDBC, facilitan la interacción eficiente entre aplicaciones y sistemas de bases de datos. Esta introducción proporcionará una base sólida para profundizar en las conexio-

nes específicas y su aplicación en el desarrollo de software.

Medios de conexión de Base de Datos

Un medio de conexión a un servidor es un protocolo de comunicaciones que facilita la interacción entre una aplicación cliente y un servidor de bases de datos es fundamental para la gestión y el tratamiento de la información. Este canal de enlace desempeña un papel crítico en la ejecución de operaciones de bases de datos tales como la consulta, inserción, actualización o eliminación de datos. De acuerdo con Connoly & Morris (2019), la importancia de este medio de conexión radica en su capacidad para viabilizar el flujo de datos requeridos por una aplicación, habilitando así la correspondiente solicitud y modificación de datos de acuerdo con las necesidades operativas. Entre estos métodos están los siguientes:

Librerías específicas de lenguajes de programación

Con respecto a la interacción entre aplicaciones y bases de datos utilizando Python, existen múltiples bibliotecas diseñadas para este fin. Por ejemplo, para conectar con bases de datos MySQL se puede emplear PyMySQL, mientras que psycopg2 y sqlite3 son utilizadas para PostgreSQL y SQLite, respectivamente. Estas herramientas son esenciales en el ámbito del desarrollo, tal como lo señalan Chauhan et al. (2019) al proporcionar una interfaz eficaz entre los programas y los sistemas de almacenamiento de datos. Además, estos autores abordan cómo la integración del framework Flask en Python facilita la creación de sistemas robustos para la administración de bases de datos en sectores educativos, resaltando la versatilidad y potencia de Python en este campo. Las librerías generalmente son creadas por las comunidades de programación, pero también son proporcionadas por los fabricantes de las bases de datos, por ejemplo, tenemos Connector/J, el cual fue creado por el fabricante de MySQL para la conexión en el lenguaje Java.

Open Database Connectivity (ODBC)

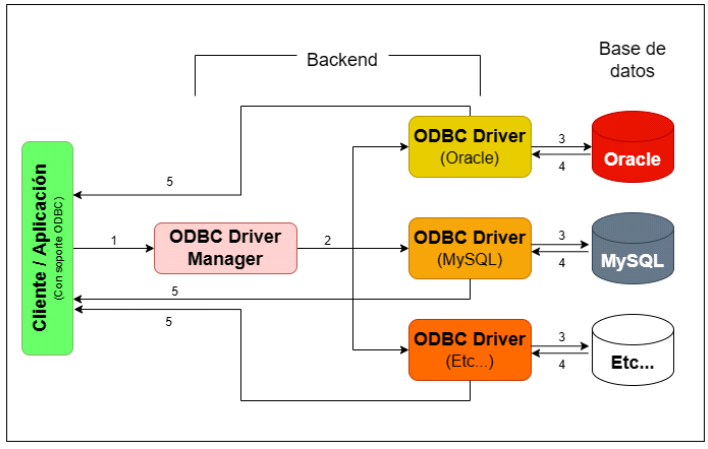
La Conectividad de Base de Batos abierta o ODBC (por sus siglas en inglés, Open Database Connectivity), explicada en detalle por Shum (1996) es un estándar de conexión diseñado para maximizar la interoperabilidad entre distintas fuentes de datos y aplicaciones. La idea de ODBC, tal como la introduce Shum, representa una evolución en las técnicas de interoperabilidad de datos. Este concepto se fundamenta en la facilitación de un lenguaje común entre distintas aplicaciones y sus respectivas bases de datos. La implementación de ODBC se lleva a cabo mediante el uso de controladores adaptados para cada uno de los sistemas de gestión de bases de datos que se podrían emplear en un proyecto determinado, lo cual es esencial para la uniformidad en el acceso a los datos.

Arquitectura ODBC

En el terreno de los negocios y la tecnología de la información, la

Arquitectura ODBC (Conectividad Abierta de Bases de Datos) se ha establecido como un estándar esencial. Este framework ha sido elaborado cuidadosamente para facilitar el diálogo entre aplicaciones diversas y los sistemas manejadores de bases de datos (DBMS), promoviendo así una interoperabilidad sólida y de confianza. La interacción con la base de datos a través de ODBC comienza con el ODBC Driver Manager, el cual es el componente esencial en la gestión de los diferentes controladores requeridos para cada tipo específico de base de datos en uso. La funcionalidad clave del ODBC Driver Manager radica en su capacidad para administrar y asegurar que el controlador adecuado esté en su lugar, lo cual es fundamental para la compatibilidad y operación correcta del sistema. Este proceso es el que permite a clientes y aplicaciones comunicarse efectivamente con la base de datos.

Figura 1.
Arquitectura de ODBC



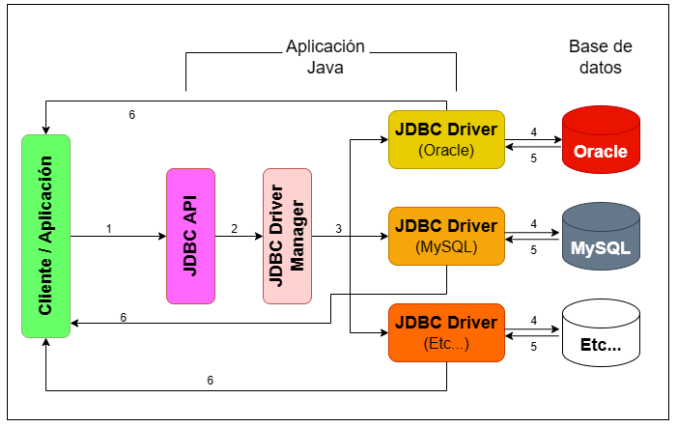
Nota. *Arquitectura básica de un OBDC para ser utilizada en u proyecto web.*
Fuente: elaboración propia.

Java Database Connectivity (JDBC)

JDBC, como se discute en el trabajo de Ramesh Fadatare (2018), facilita la interacción entre aplicaciones Java y bases de datos relacionales. El desarrollo de JDBC y su importancia en el acceso y manipulación de datos de manera eficiente y segura es fundamental en el entorno de aplicaciones empresariales. Fadatare destaca cómo JDBC actúa como un puente entre aplicaciones Java y bases de datos. Fundamentalmente, JDBC sirve de enlace entre el entorno de Java y las bases de datos, ofreciendo una interfaz CLI que habilita a los desarrolladores para realizar operaciones como consultas SQL y la manipulación de datos, incluyendo su creación y eliminación. Este método, que se alinea con estándares internacionales, facilita la creación de aplicaciones que pueden operar en diversas plataformas y asegura que el código pueda ser transferido sin problemas entre diferentes sistemas gestores de bases de datos. JDBC opera mediante una serie de interfaces y clases suministradas por el paquete java.sql. Estos elementos son cruciales, ya que brindan a los progra-

madores las herramientas necesarias para crear conexiones con bases de datos, realizar consultas y manejar los resultados obtenidos de forma eficiente. Además, JDBC facilita una gama de controladores específicos para distintos sistemas de manejo de bases de datos, conocidos como controladores JDBC, que sirven como intermediarios entre las aplicaciones Java y las bases de datos que operan en el fondo.

Figura 2.
Arquitectura Java Database Connectivity (JDBC)

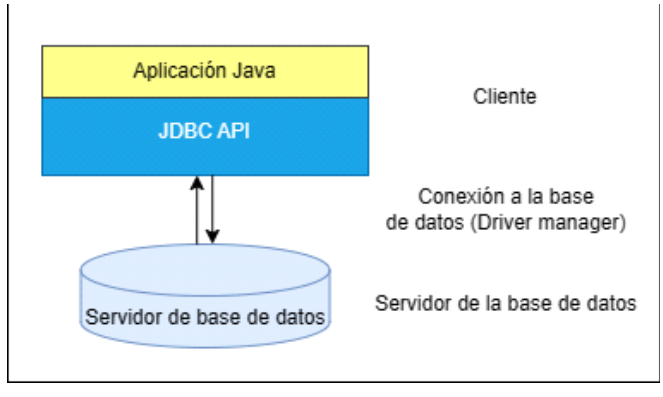


Nota. *Tomado de Java JDBC API Overview, de Ramesh Fadatare, Fuente: Java JDBC API Overview (javaguides.net)*

Tipos de arquitectura JDBC

Según *Oracle Documentation* (2022), en JDBC existen dos arquitecturas, de dos niveles o tres niveles. La arquitectura de dos niveles consiste en que la aplicación envía las peticiones directamente al servidor de la base de datos y esta misma devuelve el resultado de la petición al cliente. Todo esto con ayuda del controlador JDBC que traduce las peticiones del usuario en lenguaje SQL.

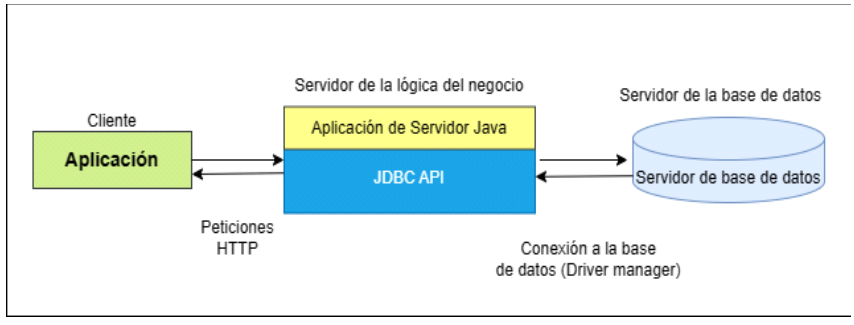
Figura 3.
Arquitectura Java Database Connectivity (JDBC)



Nota. *Tomado de Java JDBC API Overview, de Ramesh Fadatare, Fuente: Java JDBC API Overview (javaguides.net)*

La arquitectura de tres niveles agrega un intermediario entre la aplicación y la base de datos. Este intermediario es quien administra las solicitudes HTTP y las respuestas de la base de datos. Adicionalmente, el intermediario agrega una capa de seguridad ya permite tener control sobre el acceso de los datos.

Figura 4.
Arquitectura JDBC de tres niveles



Nota. *Tomado de The Java™ Tutorials, de Oracle, Fuente: JDBC Architecture (The Java™ Tutorials)*

Como realizar una conexión a Bases de Datos
En esta sección se explicará como realizar una conexión con un servidor de base de datos utilizando MySQL y Java/C#.

MySQL Java
La comunicación entre Java y una base de datos MySQL requiere el uso de clases específicas que Java proporciona para facilitar esta interacción, como java.sql.Connection y java.sql.DriverManager. La clase Connection ofrece métodos para interactuar con la base de datos, permitiendo ejecutar consultas y procesar los resultados obtenidos. Por su parte, DriverManager se encarga de gestionar los distintos drivers de bases de datos y establece la conexión con la base de datos deseada a través del método getConnection. En el código mostrado en la Figura 4.5, el proceso comienza definiendo la URL de conexión y las credenciales necesarias (nombre de usuario y contraseña) para acceder a MySQL. Esta URL incluye el protocolo específico de JDBC (jdbc:mysql://), la ubicación del servidor de la base de datos (por ejemplo, localhost:3306) y el nombre de la base de datos a la cual se desea conectar (como prueba).

Con la URL y las credenciales definidas, el siguiente paso es cargar el controlador JDBC de MySQL utilizando el método Class.forName, que es un requisito para versiones antiguas de JDBC; sin embargo, en versiones modernas, este paso puede omitirse. Posteriormente, el código intenta establecer una conexión utilizando DriverManager.getConnection, pasando como argumentos la URL de conexión, el usuario y la contraseña. Si la conexión se establece con éxito, se muestra el mensaje "Conexión exitosa a la base de datos

MySQL!" en la consola. A continuación, la conexión se cierra mediante connection.close(), y se imprime "Sesión cerrada" para indicar que la conexión ha finalizado correctamente. Si ocurre algún problema durante el proceso de conexión, se captura la excepción correspondiente y se imprime un mensaje de error, seguido del rastreo de la pila de la excepción para facilitar la depuración (Bales & GiantDino, 2002).

Figura 5.
Conectarse a una base de datos MySQL con JAVA

```java
//Librerias a usar
import java.sql.Connection;
import java.sql.DriverManager;


// Conexion a la base de datos
//Parametros para conectarse a la base de datos
                                //IP : PUERTO / Nombre de la base de datos
        String url = "jdbc:mysql://localhost:3306/prueba";
        String username = "root";
        String password = "";

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            //Iniciando conexion
            Connection connection = DriverManager.getConnection(url, username, password);
            System.out.println("Conexion exitosa a la base de datos MySQL!");

            //Cerrando conexion
            connection.close();
            System.out.println("Sesion cerrada");
        } catch (Exception ex) {
            System.out.println("Error al conectarse a la base de datos MySQL.");
            ex.printStackTrace();
        }
```

Nota. *Código elaborado para realizar la conexión en el lenguaje de programación de java para conectar la base de Datos. Fuente: elaboración propia.*

MySQL C#

Para realizar una comparación con Java, se muestra cómo establecer una conexión a un servidor de bases de datos MySQL utilizando el lenguaje de programación C#. El código presentado en la Figura 6 es un ejemplo de cómo conectar una aplicación .NET con una base de datos MySQL, empleando la librería MySql.Data.MySqlClient, que facilita la interacción con la base de datos. Para comenzar, es necesario definir los parámetros esenciales para una conexión efectiva: estos incluyen la dirección del servidor, el nombre de la base de datos, las credenciales de acceso del usuario (nombre de usuario y contraseña), el número de puerto y la configuración del modo SSL. Estos parámetros se combinan en una cadena de conexión, que se utiliza para crear una instancia de MySqlConnection.

El programa intenta abrir una sesión con la base de datos mediante el método Open de la instancia MySqlConnection (Figura 6). Si la conexión se establece correctamente, se muestra un mensaje de confirmación y luego se cierra la conexión con el método Close para liberar recursos. En caso de que la conexión falle, se captura una excepción de tipo MySqlException, mostrando el mensaje de error y los detalles de la cadena de conexión para facilitar la depuración. Este proceso está encapsulado dentro de un método estático

llamado connect, el cual es invocado desde el método Main. Esta práctica refleja un enfoque común en la gestión de conexiones a bases de datos en aplicaciones .NET, garantizando una estructura clara y un manejo adecuado de recursos en las operaciones de conexión a MySQL (Muhammad Maisam Abbas, 2023).

Figura 6.
Estableciendo conexión con una base de datos MySQL usando C#

```
using System;
using MySql.Data.MySqlClient;

namespace mysql {
    class Program {
        // Declaramos una variable privada para la conexión MySqlConnection
        private MySqlConnection conn;

        // Método estático para establecer la conexión
        static void connect() {
            // Definimos los parámetros de conexión
            string server = "localhost";
            string database = "mysqldb1";
            string user = "root";
            string password = "u1s2e3r4";
            string port = "3306";
            string sslM = "none";

            // Construimos la cadena de conexión usando los parámetros definidos
            string connString =
                String.Format("server={0};port={1};user id={2}; password={3}; database=
{4}; SslMode={5}",
                              server, port, user, password, database, sslM);

            // Creamos una nueva instancia de MySqlConnection usando la cadena de conexión
            conn = new MySqlConnection(connString);
            try {
                // Intentamos abrir la conexión
                conn.Open();

                // Si la conexión es exitosa, mostramos un mensaje
                Console.WriteLine("Connection Successful");

                // Cerramos la conexión
                conn.Close();
             } catch (MySqlException e) {
                 // En caso de error, mostramos el mensaje de error
                  Console.WriteLine(e.Message + connString);
             }
         }

         static void Main(string[] args) {
             // Llamamos al método connect() para establecer la conexión
             connect();
         }
     }
 }
```

Nota. *Tomado de Conexión MySQL en C#, de Muhammad Maisam Abbas en 2023. Fuente:* *Conexión MySql en C# | Delft Stack*

Si se desea complementar la información presentada anteriormente, se recomienda visitar los siguientes sitios web, donde se puede profundizar aún más en el tema de conexiones a bases de datos:

Núcleo Visual ofrece un manual exhaustivo para integrar una base de datos con un sitio web. El contenido cubre desde la selección adecuada de la base de datos hasta la arquitectura web necesaria para la integración. Se destacan MySQL y PostgreSQL como las opciones preferidas, y se detalla un procedimiento que abarca desde la creación de cuentas de usuario hasta la implementación de la conexión mediante lenguajes de programación como PHP o

Python. Además, el manual subraya la importancia de cerrar correctamente la conexión a la base de datos después de su uso para mantener la eficiencia y seguridad del sistema. Para obtener más información, puedes visitar el siguiente enlace:¿Cómo conectar una página web a una base de datos? Nucleo Visual (Nucleo Visual, 2023)

Informatec Digital. Presenta un manual esencial para quienes inician en el manejo de Python para la gestión de datos. El manual comienza con instrucciones sobre la instalación de Python y del sistema gestor de bases de datos de preferencia, y continúa con el aprendizaje de tareas fundamentales, como la creación de tablas, el ingreso de datos y la realización de consultas. Se enfatiza el rol significativo de Python en áreas como el desarrollo web, el análisis de datos, y su aplicación en el sector empresarial y en la investigación de datos. Para explorar más sobre este recurso, se puede acceder al sitio web a través del siguiente enlace:https://informatecdigital.com/bases-de-datos/python-y-bases-de-datos-guia-definitiva-para-principiantes/ (TecnoDigital, 2024)

Think Big Empresas. El progreso de las bases de datos migrando al entorno de la nube destaca beneficios significativos, como la flexibilidad en el manejo de la carga de trabajo, la reducción de costos operativos y la facilidad de acceso remoto. Este enfoque proporciona un manual fundamental para la administración de bases de datos en la nube, que abarca desde la elección de servicios y proveedores adecuados hasta la implementación de medidas de seguridad y procedimientos de respaldo. Se enfatiza la creciente tendencia de trasladar datos al entorno cloud, impulsada por el auge de tecnologías emergentes como el Internet de las Cosas (IoT) y el análisis de grandes volúmenes de datos (Big Data). Para explorar más sobre este tema, se puede acceder al sitio web a través del siguiente enlace: https://empresas.blogthinkbig.com/bases-de-datos-web-guia-basica-para-su-gestion/ (Moncho Terol, 2021)

Visión general de servicios Web

Un Servicio Web es un software que permite la comunicación entre máquinas a través de una red. Se basa en un conjunto de protocolos y estándares que facilitan la interacción y el intercambio de información entre diferentes aplicaciones, independientemente del lenguaje de programación en el que estén escritas (IBM, 2024). Los componentes clave que definen un servicio web son:

XML (Extensible Markup Language)

Utilizado para codificar mensajes en un formato que es legible tanto para humanos como para máquinas, facilitando el intercambio de datos estructurados.

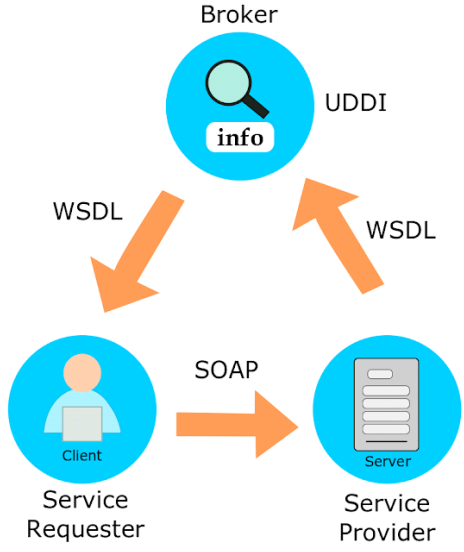
El 'SOAP' (Protocolo Simple de Acceso a Objetos), que se fundamenta en XML, facilita el intercambio de información necesaria para los servicios web. Este protocolo posibilita que las aplicaciones interactúen entre sí sobre varias

redes, estando concebido para operar de forma agnóstica con respecto a los distintos lenguajes de programación o plataformas.

El **'WSDL'** (Lenguaje de Descripción de Servicios Web), también basado en XML, actúa como una herramienta para articular los detalles de los servicios proporcionados por un servicio web, detallando cómo acceder a estos y qué protocolos de comunicación se emplean.

Por último, el **'UDDI'** (Descripción, Descubrimiento e Integración Universales), es una norma establecida para la publicación y localización de información sobre servicios web, lo que facilita a las aplicaciones el hallazgo y la integración de dichos servicios en Internet.

Figura 7.
Arquitectura y elementos de un Servicio web



Nota. *Tomado de: ¿Que es un Servicio Web? (Web services) - Elementos y aplicación, de (Luis Mario, 2019). Fuente: ¿Que es un Servicio Web? (Web services) - Elementos y aplicación (my-entertainment-place.blogspot.com)*

Objetivos de un Servicio Web

Los servicios web están concebidos con el objetivo de facilitar la comunicación entre aplicaciones variadas, que pueden estar construidas sobre múltiples plataformas y escritas en diferentes lenguajes de programación. La clave para alcanzar esta interoperabilidad reside en la adopción de estándares de la industria y en la implementación de protocolos que utilizan XML. Esto asegura la correcta transferencia y manipulación de la información entre sistemas que de otro modo serían incompatibles.

Reutilización de Servicios

Facilitar la reutilización de funcionalidades de negocio existentes, permitien-

do que se expongan como servicios web que pueden ser invocados por diferentes aplicaciones, reduciendo así el tiempo y el costo asociado al desarrollo de nuevas soluciones.

Integración de Aplicaciones
Simplificar la integración de aplicaciones tanto dentro de una organización como entre diferentes organizaciones. Un servicio web permite la comunicación bidireccional de datos entre aplicaciones dispares, lo cual es clave para la creación de sistemas de información corporativos que necesitan interactuar con una variedad de otras aplicaciones y sistemas.

Facilidad de Uso y Accesibilidad
Ofrecer una manera sencilla y accesible para que las aplicaciones accedan a servicios y funcionalidades específicas a través de Internet. Los servicios web utilizan tecnologías y protocolos ampliamente soportados y comprendidos, como HTTP, facilitando su implementación y consumo.

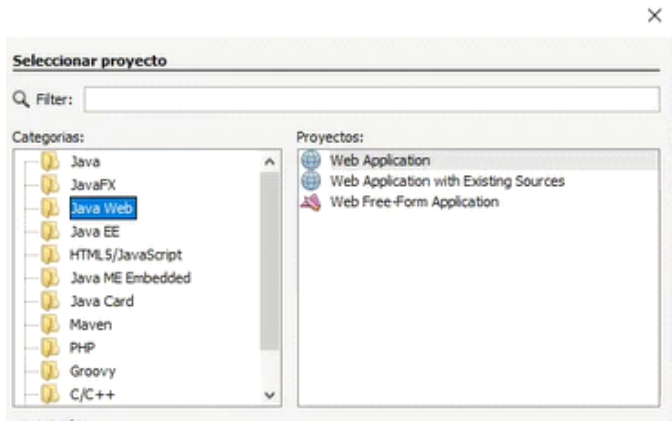
Escalabilidad y Mantenimiento
Permitir la escalabilidad y el mantenimiento eficiente de aplicaciones mediante la separación de las interfaces de servicio de las implementaciones subyacentes. Esto significa que los servicios web pueden actualizarse o modificarse sin afectar negativamente a las aplicaciones cliente que los consumen, siempre y cuando se mantenga la compatibilidad de las interfaces.

Seguridad
Proveer mecanismos para asegurar la comunicación entre aplicaciones, incluyendo la autenticación, autorización, integridad de los datos y confidencialidad. Los servicios web adoptan estándares de seguridad como WS-Security para proteger los datos y asegurar las transacciones.

Como crear un Servicio Web con Java
A continuación, se mostrará como crear un servicio web paso a paso en el IDE Java Netbeans 8.

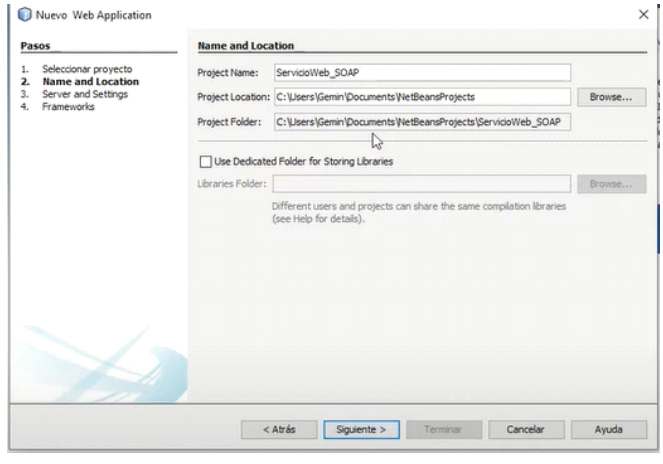
Primero, se crea un nuevo proyecto y se busca la categoría Java Web, aparecerán tres opciones, pero se escoge la que dice Web Application.

Figura 8.
Arquitectura y elementos de un Servicio web



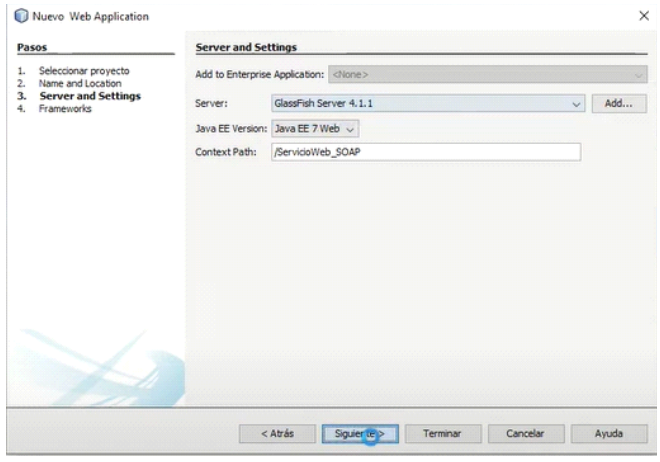
Nota. *Se escoge el nombre y la ruta la cual tendrá el nuevo proyecto a crear. Fuente: elaboración propia a partir del uso de NetBeans.*

Figura 9.
Paso 2: Nombre y ruta del proyecto



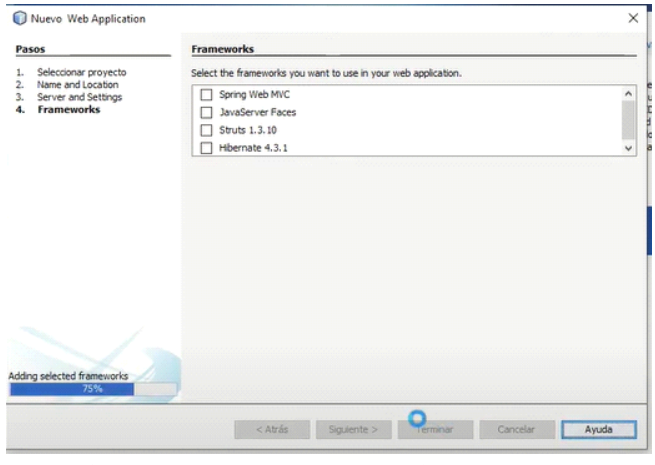
Nota. *Se escoge el servidor para usar en la aplicación, se usa la versión de Java EE. Fuente: elaboración propia a partir del uso de NetBeans.*

Figura 10.
Paso 3: Elegir el servidor y la versión de Java EE



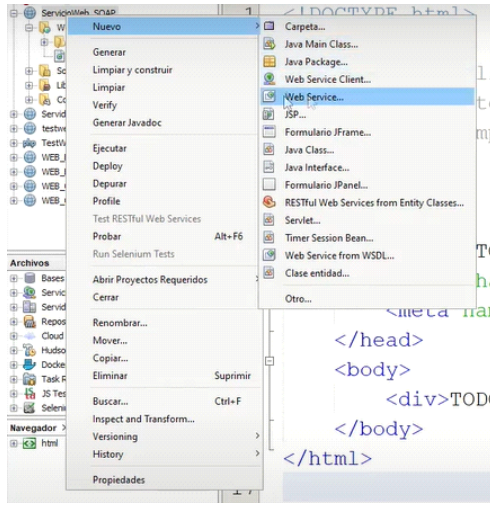
Nota. *Si se desea utilizar algún Framework para realizar la Web Application, en este caso no se utilizarán Frameworks. Fuente: elaboración propia a partir del uso de NetBeans.*

Figura 11.
Paso 4: Elegir el Framework



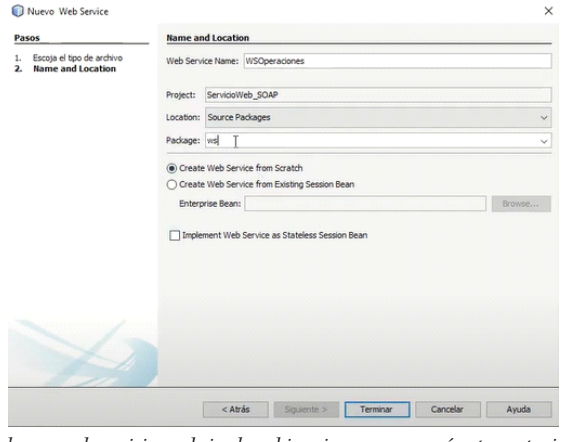
Nota. *Con esto ya solo falta crear el servicio web y la lógica del uso. Para esto le damos clic derecho al package del proyecto y se crea un nuevo Web Service. Fuente: elaboración propia a partir del uso de NetBeans.*

Figura 12.
Paso 5: Creación del web service



Nota. *Aparecerá una nueva ventana para poder escribir el nombre que llevará el servicio y el nombre del package donde estará localizado el servicio. Fuente: elaboración propia a partir del uso de NetBeans.*

Figura 13.
Paso 6: Elegir el nombre y ruta del servicio web



Nota. *Al momento de crear el servicio y abrir el archivo .java, aparecerá este contenido. Fuente: elaboración propia a partir del uso de NetBeans.*

Figura 14.
Contenido inicial de un servicio web de Java

```java
package ws;

import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;

/**
 *
 * @author Gemin
 */
@WebService(serviceName = "WSOperaciones")
public class WSOperaciones {

    /**
     * This is a sample web service operation
     */
    @WebMethod(operationName = "hello")
    public String hello(@WebParam(name = "name") String txt) {
        return "Hello " + txt + " !";
    }
}
```

Nota. *El contenido dentro de la clase se elimina para permitir la escritura de la propia lógica de negocio. Este proceso no difiere mucho del uso estándar de Java, con la excepción de una sintaxis adicional que se introduce en la creación de los métodos. Fuente: elaboración propia a partir del uso de NetBeans.*

Figura 15.
Paso 7: Escribir la lógica del negocio de nuestro servicio

```java
/**
 * Web service operation
 */
@WebMethod(operationName = "Login")
public Boolean Login(@WebParam(name = "usuario") String usuario, @WebParam(name = "contrasena") String contrasena)

    if(usuario.equals("Pavel") && contrasena.equals("Pavel2020")){
        return true;
    } else {
        return false;
    }
}

/**
 * Web service operation
 */
@WebMethod(operationName = "ProcesarPago")
public int ProcesarPago(@WebParam(name = "total") int total, @WebParam(name = "pago") int pago) {
    if(pago>=total){
        //retornar vuelto
        return pago-total;
    } else {
        return -1;
    }
```

Nota. *El primer método es un login básico, donde compara los parámetros que recibe con los almacenados en el equals y retorna un booleano; el segundo método compara si el parámetro recibido del pago es mayor al parámetro recibido del total a pagar, si es así retorna lo que le sobra al usuario, si no, da error. Fuente: elaboración propia a partir del uso de NetBeans.*

Para tener acceso al servicio web, se debe “Levantar” el proyecto, es decir, poner en funcionamiento el servicio, para esto se debe dar clic derecho al proyecto y buscar Deploy, así comenzara a ejecutarse el servicio en el puerto 8080 (este es el puerto por defecto).

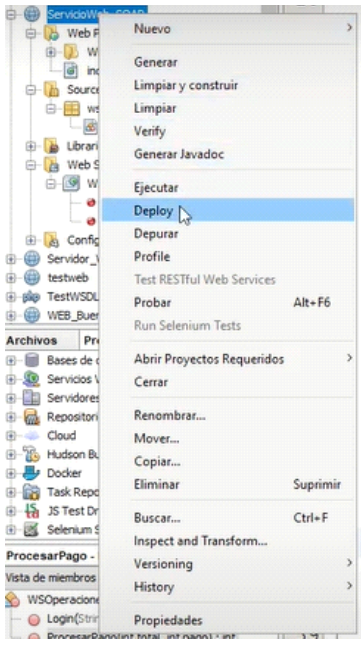
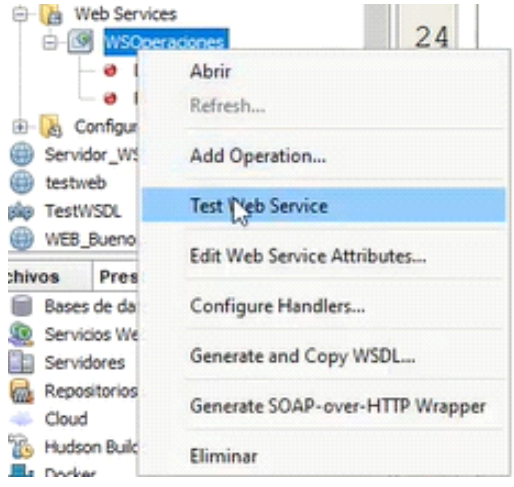


Figura 16.
Paso 8: Levantar el proyecto

Nota. *Construyendo el servicio web en el proyecto definido de form previa. Fuente: elaboración propia a partir del uso de NetBeans.*

Por último, se procederá a probar el servicio web creado. NetBeans ofrece un servicio de Testing que facilita este proceso al proporcionar una interfaz simple que muestra todas las respuestas del servicio. Para llevar a cabo esta prueba, se debe hacer clic derecho en el archivo del servicio web y seleccionar la opción "Test Web Service". Esto abrirá una ventana en el navegador donde se podrá interactuar y probar el servicio web.

Figura 17.
Paso 9: Probar el servicio web



Nota. *Ejecución de la prueba que trae el IDE por defecto para verificar el funcionamiento del web service. Fuente: elaboración propia a partir del uso de NetBeans.*

Figura 18.
Resultado de la creación del servicio web

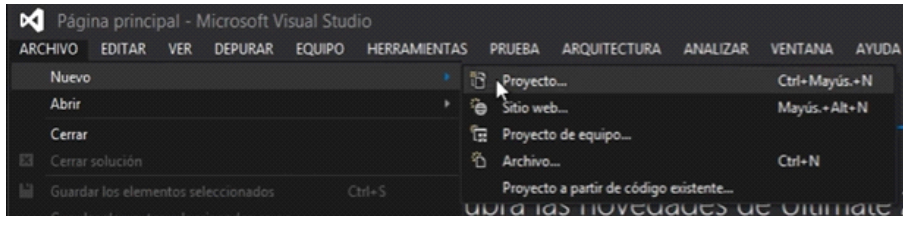
Nota. *Interfaz generada ese el IDE para probar el Web service creado. Fuente: elaboración propia a partir del uso de NetBeans.*

¿Cómo se crea un Web Service con C#?

Para crear un servicio web con C#, primero se debe de tener instalado el Visual Studio, este programa podrán descargarlo gratuitamente.

Ya teniendo instalado el Visual Studio, se crea un nuevo proyecto

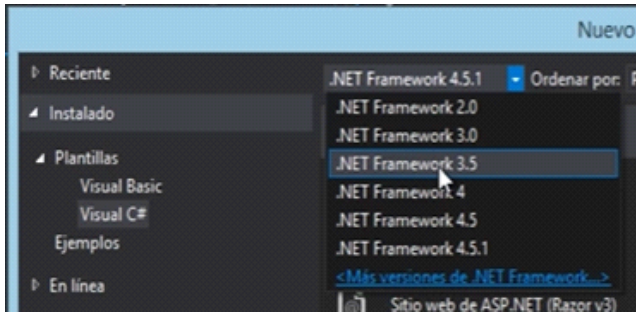
Figura 19.
Paso 1: Creación del proyecto.



Nota. *Se dirige a archivo y se selecciona nuevo proyecto. Fuente: elaboración propia a partir de Visual Studio.*

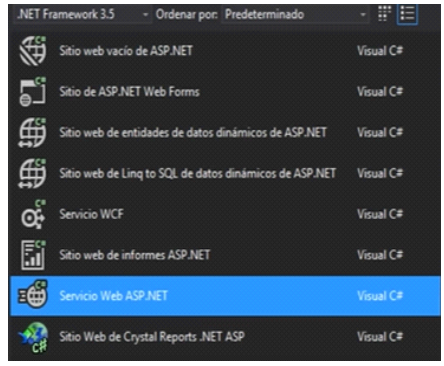
Se seleccionará la versión de .NET Framework para el proyecto; en este caso, se utilizará la versión 3.5.

Figura 20.
Paso 2: Escoger la versión del Framework.



Nota. *Al seleccionar la versión 3.5 de .NET Framework, deja escoger la opción de crear un proyecto de Servicio Web ASP .NET. Fuente: elaboración propia a partir de Visual Studio.*

Figura 21.
Paso 3: Elegir el tipo de proyecto a crear.



Nota. *Una vez que el servicio se haya creado, solo restará implementar la lógica de negocio, de la misma manera que se haría en Java. Fuente: elaboración propia a partir de Visual Studio.*

Programas para implementar servicios webs

Es posible que se pregunten: "¿Por qué es tan complicado realizar un servicio web? ¿No hay una manera más sencilla de hacerlo?". Y, de hecho, sí la hay. Muchos programadores han trabajado para simplificar este proceso, desarrollando frameworks especializados para la creación de servicios web. Algunos ejemplos de estos frameworks son:

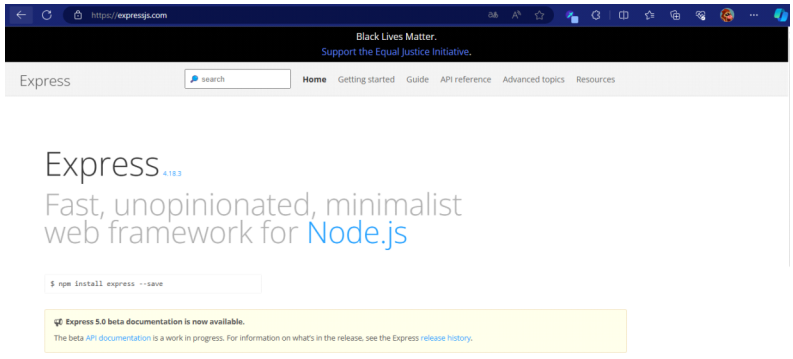
1. **Spring Boot (Java):** Simplifica la creación de aplicaciones basadas en Java, incluyendo servicios web, mediante configuraciones automáticas y herramientas integradas.
2. **ASP.NET Core (.NET):** Ofrece un framework ligero y modular para construir servicios web y aplicaciones basadas en la nube, con soporte para múltiples plataformas.
3. **Flask (Python):** Un microframework minimalista que permite crear servicios web de manera rápida y sencilla, ideal para proyectos pequeños y prototipos.
4. **Django (Python):** Framework de alto nivel que incluye una serie de herramientas para la creación de aplicaciones web completas, incluyendo servicios web RESTful.
5. **Express (Node.js):** Framework minimalista y flexible para Node.js que facilita la creación de servicios web con una gran variedad de complementos y middleware.

Estos frameworks permiten a los desarrolladores crear servicios web de manera más eficiente y con menos complicaciones, acelerando el desarrollo y mejorando la calidad del software producido.

Por último, uno de los frameworks más utilizados para la creación de aplicaciones web y sus servicios web es Express.js, una librería de Node.js escrita en JavaScript. Express.js ofrece un conjunto robusto de características para desarrollar aplicaciones web y móviles, incluyendo un sistema de enrutamien-

to poderoso, soporte para middleware y la capacidad de integrar diversos motores de plantillas, lo que facilita la creación de páginas web dinámicas. Su arquitectura ligera y su amplia comunidad de desarrolladores hacen de Express.js una opción excelente tanto para proyectos simples como para aplicaciones complejas y escalables.

Figura 22.
Página de inicio de Express.com



Nota. *Guía de instalación, guías de usuario y documentación para el perfecto uso ExpressJS, Fuente: https://expressjs.com/*

Tecnologías subyacentes

Las tecnologías subyacentes en el desarrollo y consumo de servicios web abarcan un conjunto de estándares, protocolos y herramientas que facilitan la comunicación bidireccional entre sistemas. Estas tecnologías incluyen protocolos de comunicación como HTTP y HTTPS, formatos de intercambio de datos como XML y JSON, y estándares de interfaz de servicio como SOAP y REST. Además, herramientas y librerías específicas para el desarrollo de servicios web, como WSDL para describir servicios web SOAP o GraphQL para consultas eficientes, juegan un papel crucial. El correcto entendimiento y aplicación de estas tecnologías son fundamentales para el diseño, implementación, y consumo eficiente de servicios web, permitiendo así la integración y cooperación entre aplicaciones de manera fluida y segura.

Fuentes teóricas para explorar estas tecnologías incluyen:

a) **"RESTful Web Services" por Leonard Richardson y Sam Ruby:** Este libro ofrece una comprensión profunda de los servicios web RESTful, explicando cómo usar HTTP de manera efectiva para construir servicios web escalables y mantenibles.
b) **"SOAP and RESTful Web Services" por Radu Vunvulea:** Aunque no es un libro, varios artículos y trabajos de Radu Vunvulea profundizan en las diferencias y casos de uso entre SOAP y REST, ofreciendo una visión equilibrada de estas tecnologías para desarrolladores y arquitectos de software.
c) **"Learning GraphQL and Relay" por Samer Buna:** Este libro proporciona una introducción a GraphQL, una tecnología emergente para el

desarrollo de APIs que permite a los clientes solicitar datos específicos, mejorando el rendimiento y flexibilidad en comparación con REST.

Ejemplos de tecnologías subyacentes en la práctica:

a) **HTTP/HTTPS:** Protocolos fundamentales para la transmisión de datos en servicios web, permitiendo la comunicación entre cliente y servidor a través de solicitudes y respuestas.(Cloudflare, 2023)
b) **XML y JSON:** Formatos de intercambio de datos ampliamente utilizados en servicios web, donde XML ha sido tradicionalmente usado en SOAP y configuraciones de servicios web, mientras que JSON se ha popularizado por su simplicidad y eficacia en APIs RESTful.
c) **WSDL (Web Services Description Language):** Utilizado para describir los servicios web y sus operaciones en un formato XML, facilitando la interoperabilidad entre diferentes sistemas al definir cómo deben interactuar con el servicio.

Sitios web con información útil sobre tecnologías subyacentes:

a) **W3Schools (https://www.w3schools.com/):** Proporciona tutoriales y referencias sobre tecnologías web, incluyendo HTML, XML, JSON, y HTTP. Es un recurso educativo útil para principiantes y desarrolladores experimentados que buscan refrescar sus conocimientos.
b) **MDN Web Docs (https://developer.mozilla.org/):** Ofrece documentación exhaustiva y tutoriales sobre tecnologías web, incluyendo detalles sobre protocolos HTTP/HTTPS, formatos de datos como JSON y XML, y guías sobre cómo consumir y crear servicios web.
c) **GraphQL.org (https://graphql.org/):** El sitio oficial de GraphQL proporciona una documentación completa sobre esta tecnología, incluyendo tutoriales, herramientas y casos de estudio. Es un recurso esencial para aquellos que desean aprender cómo GraphQL puede mejorar el desarrollo de APIs en comparación con REST.

Publicación de un servicio Web

La publicación de un servicio web implica hacer disponible una aplicación o función de software en la red, permitiendo que otros sistemas o aplicaciones accedan a sus funcionalidades o datos a través de internet. Utilizando estándares web como HTTP, REST o SOAP, los servicios web se diseñan para soportar la interoperabilidad entre diferentes sistemas, facilitando así la integración y comunicación en entornos heterogéneos. Este enfoque de desarrollo y despliegue de software es fundamental en la arquitectura orientada a servicios (SOA), permitiendo que las empresas y desarrolladores construyan aplicaciones más flexibles y escalables.

Para comprender mejor el concepto y la implementación de servicios web, se pueden consultar fuentes teóricas como:

a) **"Web Services: Principles and Technology" por Michael P. Papazoglou:** Este libro es una guía exhaustiva sobre los principios y

tecnologías subyacentes a los servicios web, incluyendo los protocolos de comunicación, formatos de mensaje y patrones de diseño. Proporciona una base sólida para entender cómo se construyen, despliegan y consumen los servicios web en diversas plataformas.

b) **"RESTful Web Services" por Leonard Richardson y Sam Ruby:** Centrado en los servicios web RESTful, este libro ofrece una visión profunda de cómo diseñar servicios web siguiendo los principios REST. Es una lectura esencial para aquellos interesados en desarrollar servicios web eficientes, escalables y fáciles de usar.

c) **"Service-Oriented Architecture:** Concepts, Technology, and Design" por Thomas Erl: Este libro aborda los servicios web desde la perspectiva de la arquitectura orientada a servicios (SOA), detallando cómo los servicios web se integran en arquitecturas empresariales más amplias para mejorar la agilidad y eficiencia organizacional.

Ejemplos de servicios web implementados con diferentes lenguajes de programación incluyen:

a) **Un servicio web RESTful en Java con Spring Boot:** Spring Boot facilita la creación de servicios web RESTful robustos y escalables en Java, proporcionando herramientas integradas para el manejo de peticiones HTTP, serialización de datos y autenticación.

b) **API GraphQL en JavaScript con Node.js:** GraphQL ofrece una alternativa a REST para el desarrollo de servicios web, permitiendo a los clientes solicitar exactamente los datos que necesitan. Node.js, junto con bibliotecas como Express y Apollo Server, permite construir eficientemente estas APIs.

c) **Servicios web SOAP en C# con .NET Framework:** .NET Framework de Microsoft ofrece amplio soporte para la construcción de servicios web basados en SOAP, una alternativa a REST que sigue siendo popular en entornos empresariales que requieren protocolos de comunicación más estrictos y funcionalidades como transacciones y seguridad mejorada.

A continuación se presentan sitios web donde se puede encontrar información relacionada con la publicación de servicios web:

a) **Documentación oficial de Spring (https://spring.io/):** Este sitio ofrece guías completas y documentación técnica sobre cómo desarrollar servicios web RESTful con Spring Boot, incluyendo ejemplos de código y mejores prácticas.

b) **GraphQL.org (https://graphql.org/):** Aquí se encuentra una amplia documentación sobre GraphQL, incluyendo tutoriales para principiantes, documentación de referencia y estudios de caso. Es el sitio web oficial de GraphQL y un recurso esencial para desarrolladores interesados en esta tecnología para APIs.

c) **Microsoft Docs (https://docs.microsoft.com/):** En este sitio se ofrece documentación y guías sobre el desarrollo de servicios web SOAP y

RESTful utilizando el .NET Framework y C#. Es un recurso valioso para desarrolladores que trabajan en el ecosistema de Microsoft, proporcionando ejemplos prácticos, referencias de API y consejos de mejores prácticas.

Consumo de un servicio WEB

El consumo de un servicio web se refiere al proceso por el cual una aplicación cliente realiza solicitudes a un servicio web para obtener datos o ejecutar operaciones específicas. Esto se hace a través de protocolos estandarizados de internet, como HTTP, utilizando formatos de intercambio de datos como XML o JSON. Este proceso es fundamental en la arquitectura de aplicaciones modernas, permitiendo que sistemas dispares se comuniquen y colaboren entre sí, independientemente de las tecnologías subyacentes o plataformas de implementación. Los servicios web RESTful y SOAP son los más comunes, cada uno con sus propias convenciones para la solicitud y recepción de datos.

Para profundizar en el tema del consumo de servicios web, se pueden considerar las siguientes fuentes teóricas:

a) **"Programming Web Services with SOAP" por James Snell, Doug Tidwell, y Pavel Kulchenko:** Este libro proporciona una introducción detallada al desarrollo y consumo de servicios web utilizando SOAP, un protocolo estándar para el intercambio de mensajes de solicitud-respuesta en un entorno distribuido.
b) **"RESTful Web Clients: Enabling Reuse Through Hypermedia" por Mike Amundsen:** Se centra en el consumo de servicios web RESTful, explicando cómo construir clientes web que interactúan eficientemente con servicios web a través de HTTP. Este libro cubre desde los fundamentos de REST hasta prácticas avanzadas como el manejo de hipermedios.
c) **"Learning GraphQL and Relay" por Samer Buna:** Aunque se centra más en GraphQL, una alternativa a REST y SOAP, este libro también proporciona una excelente visión sobre cómo consumir servicios web, explicando cómo las aplicaciones pueden utilizar GraphQL para solicitar y actualizar datos de manera más eficiente.

Ejemplos de consumo de servicios web con diferentes lenguajes de programación incluyen:

a) **Consumo de una API REST con Python usando Requests:** Python, con su biblioteca Requests, es ampliamente utilizado para consumir servicios web RESTful debido a su simplicidad y facilidad de uso. Permite a los desarrolladores hacer solicitudes HTTP y manejar respuestas de manera eficiente con pocas líneas de código.
b) **Uso de Retrofit para consumir un servicio web en Android con Kotlin:** Retrofit es una biblioteca tipo cliente HTTP para Android y Java que facilita el consumo de servicios web RESTful. Se integra perfectamente con Kotlin, ofreciendo una manera declarativa de definir la API y

deserializar respuestas automáticamente.

c) **Fetch API en JavaScript para consumir servicios web:** La Fetch API proporciona una forma moderna de realizar solicitudes de red en el navegador o en entornos de Node.js. Es especialmente útil para consumir APIs RESTful, permitiendo a los desarrolladores manejar fácilmente solicitudes y respuestas asincrónicas.

Sitios web con información útil sobre el consumo de servicios web:

a) **MDN Web Docs (https://developer.mozilla.org/):** Ofrece documentación completa sobre la Fetch API y otras tecnologías web relevantes para el consumo de servicios web. Es una fuente confiable para desarrolladores web, con guías, tutoriales y referencias sobre las últimas tecnologías web.

b) **Postman Learning Center (https://learning.postman.com/):** Postman es una herramienta popular para probar APIs, y su centro de aprendizaje proporciona una gran cantidad de recursos educativos sobre cómo consumir servicios web, incluyendo guías paso a paso, tutoriales y consejos de mejores prácticas.

c) **Official Python Requests library documentation (https://docs.python-requests.org/):** La documentación oficial de la biblioteca Requests de Python es un recurso indispensable para desarrolladores que utilizan Python para consumir servicios web. Ofrece una guía exhaustiva sobre cómo realizar solicitudes HTTP, manejar respuestas y trabajar con sesiones y cookies.

Actividades de Evaluación

La primera actividad propuesta en esta sesión consiste en establecer una conexión a una base de datos, aplicando los conocimientos adquiridos en el capítulo actual. Para ello, se deberán seguir los pasos detallados previamente, como la configuración de los parámetros de conexión, la selección del sistema gestor de base de datos adecuado, y la implementación de la lógica de conexión utilizando el lenguaje de programación correspondiente. Esta actividad no solo reforzará los conceptos teóricos aprendidos, sino que también proporcionará una experiencia práctica valiosa, permitiendo a los participantes identificar y resolver posibles desafíos que puedan surgir durante la conexión a bases de datos en un entorno real.

1. Conexión de Base de Datos en Java

Ejercicio: Desarrollar un software en Java capaz de establecer comunicación con una base de datos MySQL. El sistema debe permitir el manejo integral de registros (Adición, Consulta, Modificación, Supresión) en una entidad denominada Usuarios, la cual está compuesta por atributos como identificador, nombreDeUsuario, y correoElectrónico. Se exigirá el uso del API JDBC para todas las interacciones con la base de datos.

Aspectos por evaluar:

- Establecimiento exitoso del enlace con el sistema de base de datos.
- Desarrollo completo de funcionalidades CRUD.
- Administración eficaz de situaciones excepcionales.
- Terminación adecuada de la conexión y gestión óptima de los elementos de la base de datos.

2. Integración de Base de Datos con C#

Ejercicio: construir una solución en C# que se integre con un sistema de base de datos SQL Server para ejecutar tareas de mantenimiento de datos en una entidad Productos, que contenga propiedades tales como IDdeProducto, DenominaciónDelProducto, y Coste. Será esencial el empleo de ADO.NET para el enlace y la manipulación de los datos dentro de la base de datos.

Aspectos para evaluar:

- Iniciación adecuada de la conexión con la base de datos.
- Realización efectiva de las operaciones de mantenimiento de datos.
- Uso correcto de herramientas como SqlConnection y SqlCommand, entre otros componentes pertinentes de ADO.NET.
- Control apropiado de excepciones y recursos del sistema.

3. Crear un Servicio Web en Java

Ejercicio: Implementar un servicio web RESTful en Java usando el framework Spring Boot, que gestione una colección de Libros. El servicio debe ofrecer endpoints para agregar, obtener, actualizar y eliminar libros. Cada Libro debe tener un id, titulo, y autor.

Aspectos por evaluar:

- Configuración adecuada de Spring Boot y sus dependencias.
- Definición correcta de los endpoints REST y los métodos HTTP correspondientes.
- Uso de JpaRepository para la persistencia de datos.
- Manejo de estados HTTP y mensajes de respuesta.

4. Crear un Servicio Web en C#

Ejercicio: Desarrollar un servicio web RESTful en C# utilizando .NET Core, que administre una lista de Empleados. El servicio debe proporcionar endpoints para las operaciones CRUD sobre los empleados. Cada Empleado debe contener EmpleadoID, Nombre, y Departamento.

Aspectos por evaluar:

- ·Configuración correcta del entorno .NET Core y sus dependencias.
- ·Creación adecuada de los endpoints y métodos HTTP.
- ·Implementación de Entity Framework Core para operaciones de base de datos.

- ·Gestión efectiva de los códigos de estado HTTP y las respuestas.

5. Consumir un Servicio Web en Java

Ejercicio: Escribir un cliente en Java que consuma el servicio web creado en el ejercicio 3 (el servicio web de Libros). El cliente debe ser capaz de realizar solicitudes HTTP para consumir cada uno de los endpoints del servicio (agregar, obtener, actualizar, eliminar).

Aspectos por evaluar:

- ·Uso correcto de la librería HttpClient de Java para realizar solicitudes HTTP.
- ·Capacidad para consumir y procesar las respuestas del servicio web.
- ·Manejo adecuado de JSON para el envío y recepción de datos.
- ·Implementación de una interfaz de usuario sencilla para interactuar con el servicio web (opcional).

6. Consumir un Servicio Web en C#

Ejercicio: Desarrollar una aplicación cliente en C# que consuma el servicio web RESTful creado en el ejercicio 4 (el servicio de Empleados). La aplicación debe hacer uso de HttpClient para llamar a los endpoints del servicio web y realizar las operaciones CRUD sobre los empleados.

Aspectos por evaluar:

- ·Correcta configuración y uso de HttpClient para realizar llamadas al servicio web.
- ·Procesamiento y manejo eficiente de las respuestas del servicio web.
- ·Utilización de JSON para el intercambio de datos con el servicio web.
- ·Creación de una interfaz gráfica o consola para la interacción con el servicio web (opcional).

Referencias Bibliográficas

Bales, D., & GiantDino, B. (2002). Java Programming with Oracle JDBC pages.

Chauhan, N., Singh, M., Verma, A., Parasher, A., & Budhiraja, G. (2019). Implementation of database using python flask framework. International Journal of Engineering and Computer Science, 8(12), 24894–24899. https://doi.org/10.18535/ijecs/v8i12.4390

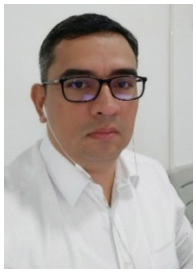
Cloudflare. (2023). What is HTTPS? Https://Www.Cloudflare.Com/Learning/Ssl/What-Is-Https/.

Connoly, C., & Morris, S. (2019). Database systems: design, implementation and management. (13th ed.). Cengage Learning.

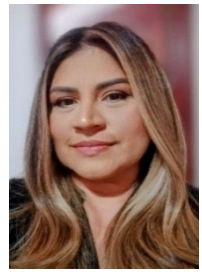
IBM. (2024). ¿Qué es un servicio web? Https://Www.Ibm.Com/Docs/Es/Integration-Bus/10.1?Topic=services-What-Is-Web-Service.

Luis Mario. (2019). ¿Qué es un Servicio Web? (Web services) - Elementos y aplicación. Https://My-Entertainment-Place.Blogspot.Com/2019/10/Que-Es-Un-Servicio-Web-Web-Services.Html.
Moncho Terol. (2021). Bases de datos web: guía básica para su gestión. Https://Empresas.Blogthinkbig.Com/Bases-de-Datos-Web-Guia-Basica-Para-Su-Gestion/.
Muhammad Maisam Abbas. (2023). Conexión MySql en C#. Https://Www.Delftstack.Com/Es/Howto/Csharp/Mysql-Connection-in-Csharp/.
Nucleo Visual. (2023). ¿Cómo conectar una página web a una base de datos? Https://Nucleovisual.Com/Como-Conectar-Una-Pagina-Web-a-Una-Base-de-Datos/.
Oracle Documentation. (2022). JDBC Architecture. Https://Docs.Oracle.Com/Javase/Tutorial/Jdbc/Overview/Architecture.Html.
Ramesh Fadatare. (2018). Java JDBC API Overview. Https://Www.Javaguides.Net/2018/10/Java-Jdbc-Api-Overview.Html.
Shum, A. C. Y. (1996). Open Database Connectivity Development of the Context Interchange System.
TecnoDigital. (2024). Python y Bases de Datos: Guía Definitiva para Principiantes. Https://Informatecdigital.Com/Bases-de-Datos/Python-y-Bases-de-Datos-Guia-Definitiva-Para-Principiantes/.

Los Autores



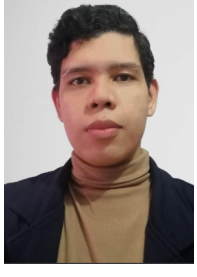
Edwin Eduardo Millán Rojas
Doctor en Ingeniería de la Universidad Distrital Francisco José de Caldas. Profesor Titular programa de Ingeniería de Sistemas de la Universidad de la Amazonia. Docente Investigador del Grupo de Investigación GITUA de la Universidad de la Amazonia.
Correo: e.millan@udla.edu.co



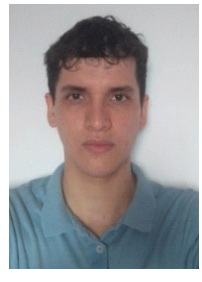
Denis Lorena Álvarez Guayara
Candidata a Doctora en Educación y Cultura Ambiental de la Universidad de la Amazonia, Profesora Asistente de la Facultad de Ingeniería de la Universidad de la Amazonia, Investigadora del Grupo de Investigación GITUA.
Correo: d.alvarez@udla.edu.co,

Fredy Antonio Verástegui González.
Magíster en Ciencias de la Información y las Comunicaciones. Profesor Titular del programa de Ingeniería de Sistemas de la Universidad de la Amazonia. Docente Investigador del Grupo de Investigación GITUA de la Universidad de la Amazonia.
Correo:**f.verastegui@udla.edu.co**

Gustavo Andrés Macías Ramos.
Estudiante del programa de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad de la Amazonia, Grupo de Investigación – GIECOM.
Correo: gu.macias@udla.edu.co

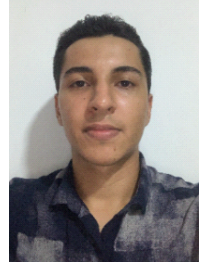


Juan Sebastián Mejía Suaza
Estudiante del programa de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad de la Amazonia, Grupo de Investigación – GIECOM.
Correo: **juans.mejia@udla.edu.co**



Sebastián Ariza González
Estudiante del programa de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad de la Amazonia, Grupo de Investigación – GIECOM.
Correo: **se.ariza@udla.edu.co**

Heriberto Fernando Vargas Losada.
Doctor en Educación y cultura Ambiental de la Universidad de la Amazonia. Profesor Titular, Programa Ingeniería de Sistemas de la Facultad de Ingenierías de la Universidad de la Amazonia, Investigar Junior Minciencias, Líder del Grupo de Investigación – GIECOM.
Correo: **heri.vargas@udla.edu.co**



Michael Mauricio Orjuela Cepeda
Estudiante del programa de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad de la Amazonia, Grupo de Investigación – GIECOM.
Correo: **mi.orjuela@udla.edu.co**