Desarrollo web sostenible y seguro: estrategias y tecnologías para una web eficiente y protegida

Gabriel Eduardo Camargo García

Trabajo de grado para optar al título de Ingeniero de Sistemas

Asesor
Raúl Mazo Peña, Ph.D. en Informática

Universidad de Antioquia
Facultad de Ingeniería
Ingeniería de Sistemas
Medellín
2024

Cita	(Camargo García, 2024)
Referencia **Estilo APA 7 (2020)**	(Camargo García, 2024). *Archivo fotográfico de la Universidad de Antioquia: valoración histórica de las fotografías, 1997 - 2003* [Trabajo de Grado]. Universidad de Antioquia, Medellín.

Dedicatoria

A mi mamá, por amarme mucho.

Agradecimientos

A la comunidad universitaria de la ENSTA Bretagne y Universidad de Antioquia,
A los miembros del grupo de investigación de VariaMos,
En especial a Diana, Deisy y Raúl,
A mis amigos y familia.

Tabla de contenido

Lista de figuras

Siglas, acrónimos y abreviaturas

APA	American Psychological Association
CDN	Content Delivery Network
CMS	Content Management System
CSS	Cascading Style Sheets
CSRF	Cross-Site Request Forgery
DI	Dependency Injection
HTML	HyperText Markup Language
IoC	Inversión de Control
MDN	Mozilla Developer Network
MVC	Modelo-Vista-Controlador
MVVM	Model-View-ViewModel
MVP	Model-View-Presenter
SQL	Structured Query Language
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
XSS	Cross-Site Scripting

Resumen

Este trabajo consiste en la construcción de una guía integral que destaca la importancia de la sostenibilidad y la seguridad en el desarrollo web. Se introducen los orígenes y la evolución de la web, destacando la necesidad de prácticas sostenibles para reducir el impacto ambiental y mejorar la seguridad de los datos. En dicha guía se cubren las tecnologías fundamentales como HTML, CSS y JavaScript, ofreciendo ejemplos prácticos y ejercicios para crear aplicaciones accesibles, eficientes.

Además, se analiza la aplicación de patrones de diseño como MVC y sus variantes, enfatizando su importancia en la arquitectura de aplicaciones web. Se realiza un estudio comparativo de diferentes frameworks de desarrollo, incluyendo React, Spring Boot y Laravel, para identificar prácticas y tecnologías que promuevan el cuidado de los recursos y su aplicación mediante ejercicios prácticos.

En otras palabras, este trabajo proporciona una hoja de ruta para desarrolladores interesados en adoptar prácticas más sostenibles, fomentando una mayor conciencia y responsabilidad ambiental en la industria tecnológica. El objetivo es impulsar cambios positivos hacia un uso más eficiente, ético y seguro de los recursos digitales.

Palabras clave: desarrollo web sostenible, aplicaciones web, frameworks, ciberseguridad, sustentabilidad digital, sistemas durables, MVC.

Abstract

This work consists of constructing a comprehensive guide that highlights the importance of sustainability and security in web development. It introduces the origins and evolution of the web, emphasizing the need for sustainable practices to reduce environmental impact and improve data security. This guide covers fundamental technologies such as HTML, CSS, and JavaScript, offering practical examples and exercises to create accessible, efficient, and secure applications.

Additionally, it analyzes the application of design patterns like MVC and its variants, emphasizing their importance in the architecture of web applications. A comparative study of different development frameworks, including React, Spring Boot, and Laravel, is conducted to identify practices and technologies that promote resource conservation and their application through code exercises.

In other words, this work provides a roadmap for developers interested in adopting more sustainable and secure practices, fostering greater environmental and security awareness and responsibility in the technology industry. The goal is to drive positive changes toward a more efficient, ethical, and safe use of digital resources.

*Keywords***:** sustainable web development, web applications, frameworks, cybersecurity, digital sustainability, durable systems, MVC.

Introducción

En un mundo donde las aplicaciones web están en constante evolución y se vuelven cada vez más complejas y demandantes en términos de recursos y seguridad, surge la necesidad de prácticas sostenibles para minimizar el impacto ambiental, mejorar la eficiencia energética y la protección de datos. Este informe abarca los conceptos básicos y tecnologías fundamentales, iniciando desde HTML, CSS y JavaScript, hasta llegar al manejo de frameworks avanzados como React, Spring Boot y Laravel y la aplicación de principios de diseño seguro. La investigación busca no solo buscar maneras de mejorar la eficiencia y reducir el impacto ambiental, sino también promover una cultura de responsabilidad ambiental y seguridad en la industria tecnológica.

El contexto del trabajo se sitúa en la intersección de la sostenibilidad y la seguridad en el desarrollo web. Con la proliferación de aplicaciones web y el aumento de la conciencia ambiental y de seguridad, es imperativo desarrollar metodologías y prácticas que aborden estos desafíos. Este proyecto de investigación se centra en comparar diferentes frameworks de desarrollo para identificar prácticas y tecnologías que contribuyan a un desarrollo más sostenible y seguro. Además, se exploran los principios de ingeniería web y patrones de diseño como Modelo-Vista-Controlador (MVC) y sus variantes, así como la importancia de la seguridad en la arquitectura de aplicaciones web.

En todo momento se busca evaluar la sostenibilidad, eficiencia y seguridad de varios frameworks modernos de desarrollo web mediante la creación y análisis de proyectos web simples. Los objetivos específicos incluyen investigar y seleccionar frameworks web modernos para el estudio, desarrollar aplicaciones y ejercicios web simples utilizando cada framework seleccionado, analizar el consumo de recursos, el desempeño y la seguridad de las aplicaciones desarrolladas, y proponer recomendaciones para un desarrollo con buenas prácticas basado en la investigaciones realizadas.

La metodología empleada en este proyecto combina métodos cualitativos y cuantitativos para evaluar la sostenibilidad y seguridad en el desarrollo web. Se inicia con una revisión de literatura sobre desarrollo sostenible, seguridad en aplicaciones web y tecnologías web, incluyendo

directrices como las del W3C sobre sostenibilidad web. La fase de desarrollo implica la creación de aplicaciones web utilizando las tecnologías de desarrollo web básicas y los frameworks mencionados, implementando técnicas de optimización de recursos y promoviendo la evaluación del desempeño y huella de carbono de una aplicación web con herramientas como Google Lighthouse y análisis de vulnerabilidades.

En otras palabras, este trabajo busca proporcionar recomendaciones prácticas y accesibles para desarrolladores interesados en adoptar prácticas más sostenibles. En última instancia, el trabajo busca fomentar una mayor conciencia y responsabilidad ambiental y de seguridad en la industria tecnológica, impulsando cambios positivos hacia un uso más eficiente, ético y seguro de los recursos digitales.

1 Planteamiento del problema

El problema que se aborda en este trabajo de grado se centra en la identificación y adopción de prácticas y tecnologías que permitan un desarrollo web sostenible y seguro. El incremento en el consumo de recursos y las crecientes amenazas de seguridad plantean un reto significativo para los desarrolladores, que deben equilibrar la funcionalidad y la eficiencia con la sostenibilidad y la protección de datos.

El desarrollo de aplicaciones web, especialmente aquellas que manejan grandes volúmenes de datos y requieren recursos intensivos, contribuye significativamente al consumo de energía y la emisión de carbono. La infraestructura necesaria para soportar estas aplicaciones, incluyendo servidores y centros de datos, consume grandes cantidades de electricidad, muchas veces proveniente de fuentes no renovables. Además, el diseño y la implementación ineficientes de las aplicaciones web pueden llevar a un uso innecesario de recursos, exacerbando el problema del impacto ambiental (MDN Web Docs, 2023; Smashing Magazine, 2023).

Desde el aspecto de la seguridad, las aplicaciones web se han convertido en objetivos frecuentes de ataques cibernéticos. Las vulnerabilidades en el código, la falta de implementación de prácticas de seguridad robustas y el uso de tecnologías obsoletas aumentan el riesgo de brechas de seguridad. Estas brechas pueden tener consecuencias devastadoras, incluyendo la pérdida de datos sensibles, daños a la reputación de las empresas y pérdidas financieras significativas (W3C, 2023).

Ante este contexto, existe una clara necesidad de desarrollar metodologías y prácticas que integren la sostenibilidad y la seguridad en el desarrollo de este tipo de aplicaciones. Esto implica no solo optimizar el uso de recursos y reducir el consumo de recursos, sino también implementar medidas de seguridad eficaces que protejan los datos y las operaciones de las aplicaciones web. La adopción de estas prácticas no solo contribuirá a la sostenibilidad y seguridad del desarrollo web, sino que también mejorará la eficiencia y la responsabilidad ética de las aplicaciones (MDN Web Docs, 2023; Smashing Magazine, 2023).

1.1 Antecedentes

El desarrollo web sostenible es un campo de investigación emergente que ha cobrado relevancia significativa debido a las crecientes preocupaciones sobre el impacto ambiental de las tecnologías digitales. Según datos recientes, la industria digital es responsable de entre el 2% y el 5% de las emisiones globales de carbono, superando incluso a la industria de la aviación. Esta situación ha llevado a la creación de directrices y prácticas específicas para mitigar el impacto ambiental del desarrollo web (W3C, 2023; World Economic Forum, 2021).

Una iniciativa destacada en este ámbito es la publicación de las Web Sustainability Guidelines (WSG) 1.0 por el W3C, que proporciona un conjunto de recomendaciones basadas en investigación medible y evidencia. Estas directrices cubren aspectos desde el diseño centrado en el usuario hasta el desarrollo web, infraestructura y gestión de productos, con el objetivo de reducir el impacto ambiental de los productos y servicios digitales. La estructura de las WSG se inspira en las Web Content Accessibility Guidelines (WCAG) y ofrece un enfoque integral para la sostenibilidad web (W3C, 2023).

En el ámbito académico, los Objetivos de Desarrollo Sostenible (ODS) han tenido un impacto significativo en la producción científica. Desde la adopción de la Agenda 2030 por la Asamblea General de las Naciones Unidas en 2015, ha habido un aumento notable en la investigación relacionada con los ODS. Estos objetivos globales promueven una visión holística del desarrollo, considerando diferentes áreas de impacto como la salud, la energía y la degradación de la tierra (PLOS ONE, 2023; MDPI, 2022).

Además, la implementación de prácticas sostenibles en el desarrollo web no se limita solo a la reducción de emisiones de carbono y la optimización del consumo de energía. También incluye la mejora de la eficiencia y rendimiento de las aplicaciones web mediante técnicas como la minimización de solicitudes HTTP, la optimización de imágenes y el uso de *green hosting*. Estas prácticas benefician tanto al medio ambiente como a la experiencia del usuario, y pueden influir positivamente en el posicionamiento en motores de búsqueda (Sustainable Web Design, 2023).

En términos de seguridad, la creciente digitalización ha convertido a las aplicaciones web en objetivos frecuentes de ataques cibernéticos. La falta de implementación de prácticas de seguridad robustas y el uso de tecnologías obsoletas aumentan el riesgo de brechas de seguridad. Por lo tanto, es esencial integrar medidas de seguridad eficaces que protejan los datos y las operaciones de las aplicaciones web, garantizando así la seguridad y la sostenibilidad del desarrollo web (W3C, 2023).

2 Justificación

El tema se seleccionó debido a la intersección crítica entre la sostenibilidad y la seguridad en el desarrollo web. A medida que las aplicaciones web se vuelven más esenciales en la vida cotidiana, es vital encontrar maneras de hacerlas más eficientes. La selección de este tema permite abordar dos de los desafíos más apremiantes del desarrollo tecnológico actual: la reducción del impacto ambiental y la mejora de la seguridad cibernética. La adopción de prácticas sostenibles no solo beneficiará al medio ambiente, sino que también mejorará la eficiencia operativa y la confianza del usuario en las aplicaciones web (W3C, 2023; Making Science, 2023).

Este trabajo de grado proporciona una contribución significativa a la ingeniería al ofrecer un marco de trabajo y recomendaciones prácticas para el desarrollo web sostenible y seguro. Esto incluye la implementación de técnicas de optimización de recursos, el uso de energía renovable en el alojamiento web, y la adopción de medidas de seguridad avanzadas. Al hacerlo, el proyecto no solo contribuye a la reducción de la huella de carbono de las aplicaciones web, sino que también mejora la resiliencia y protección contra amenazas cibernéticas (MDPI, 2022; Byjus, 2023).

Por ejemplo, las aplicaciones de comercio electrónico enfrentan numerosos desafíos de seguridad. Entre ellos se encuentran los ataques de phishing, donde los atacantes engañan a los usuarios para que proporcionen información sensible a través de correos electrónicos o mensajes que parecen legítimos. Además, el malware, que incluye virus, gusanos, troyanos y spyware, puede instalarse en el dispositivo del usuario sin su conocimiento, robando datos y proporcionando acceso no autorizado. El fraude con tarjetas de crédito también es prevalente, involucrando el uso no autorizado de la información de la tarjeta para realizar compras o retirar fondos (Abou Ghazaleh & Nehme, 2023).

Otro problema de las plataformas de comercio electrónico es el uso excesivo de centros de datos. Los centros de datos son instalaciones físicas que albergan una gran cantidad de servidores, sistemas de almacenamiento y equipos de red necesarios para procesar, almacenar y transmitir grandes volúmenes de datos. Estas infraestructuras son esenciales para el funcionamiento de

diversas aplicaciones y servicios en línea, desde sitios web hasta servicios de almacenamiento en la nube.

Los centros de datos son grandes consumidores de energía, lo que genera una cantidad significativa de emisiones de carbono. El artículo de Xu y Buyya (2020) mencionó que en el 2013, los centros de datos en los Estados Unidos consumieron 91 mil millones de kWh de electricidad, equivalente al consumo de energía de dos años de los hogares de la ciudad de Nueva York. Para el 2020 se estimó que este consumo creciera hasta los 140 mil millones de kWh, generando 150 millones de toneladas de emisiones de carbono. Este consumo elevado se debe a la infrautilización y sobrecarga de recursos, como el procesamiento, almacenamiento, redes y sistemas de refrigeración, lo que no permite una eficiencia energética adecuada (Xu & Buyya, 2020).

Y esta parte solo aplica para una subsección de las aplicaciones web. La investigación y las prácticas propuestas pueden servir como guía para desarrolladores y empresas tecnológicas, fomentando una mayor responsabilidad ambiental y de seguridad en la industria.

3 Objetivos

3.1 Objetivo general

Evaluar la sostenibilidad, eficiencia y seguridad de varios frameworks modernos de desarrollo web mediante la creación y análisis de proyectos web simples, con el fin de identificar prácticas y tecnologías que contribuyan a un desarrollo web más sostenible y seguro.

3.2 Objetivos específicos

- Comparar el desempeño de diferentes frameworks de desarrollo web, como React, Spring Boot y Laravel, en términos de consumo de recursos.
- Analizar las mejores prácticas en el desarrollo web sostenible y su impacto en la reducción de la huella de carbono de las aplicaciones web.
- Evaluar la efectividad de las técnicas de optimización de recursos implementadas en las aplicaciones web desarrolladas.
- Identificar las vulnerabilidades comunes en los frameworks web seleccionados y proponer soluciones de seguridad para mitigarlas.
- Desarrollar ejemplos prácticos que demuestren cómo integrar la sostenibilidad y la seguridad en el ciclo de vida del desarrollo web.
- Proponer recomendaciones para desarrolladores y empresas tecnológicas sobre cómo adoptar prácticas más sostenibles en el desarrollo de aplicaciones web.

5 Marco teórico

El desarrollo web ha evolucionado significativamente desde sus inicios, llevando consigo tanto avances tecnológicos como retos ambientales. La creciente demanda de aplicaciones web más complejas y funcionales ha incrementado el consumo de recursos, lo cual ha generado una necesidad urgente de enfoques más sostenibles. El desarrollo web sostenible, por lo tanto, surge como una disciplina que busca minimizar el impacto ambiental de las aplicaciones digitales, optimizando el uso de recursos y mejorando la eficiencia energética (Mazo & Camargo, 2024).

5.1 Desarrollo web sostenible

El desarrollo web sostenible aboga por la minimización del uso de recursos naturales y la reducción de la huella de carbono asociada con los servicios web. Esto incluye prácticas como la optimización del código para mejorar la eficiencia energética, el uso de servidores con energías renovables y el diseño de sitios web accesibles y eficientes que reduzcan la necesidad de transferencias de datos excesivas (Querne & Widloecher, 2009). Para ello, una de las filosofías centrales de la sostenibilidad es la construcción de páginas con diseños frugales.

Una buena página sostenible parte desde un diseño minimalista, o frugal. El **diseño frugal** es una filosofía de desarrollo que prioriza la simplicidad y la eficiencia. Se enfoca en eliminar elementos innecesarios y optimizar los recursos utilizados, tanto en el diseño de la interfaz de usuario como en la arquitectura del sistema. Este enfoque no solo mejora la sostenibilidad, sino que también puede conducir a mejores experiencias de usuario al reducir los tiempos de carga y simplificar la navegación (W3C, 2023). A continuación, se analizarán técnicas clave para el ahorro de recursos.

5.1.1 Técnicas para el ahorro de recursos energéticos

La eficiencia energética en el desarrollo web implica diseñar y codificar aplicaciones de manera que utilicen la menor cantidad posible de recursos energéticos. Esto incluye optimizar el código para reducir el número de solicitudes HTTP, comprimir archivos y utilizar técnicas de **carga**

diferida (lazy loading) para mejorar los tiempos de carga y reducir el consumo de energía (W3C, 2023). Además de la eficiencia en el código, el uso de proveedores de alojamiento que operan con energía renovable, conocido como green hosting, también es esencial (Mazo & Camargo, 2024).

El **green hosting** se refiere al uso de proveedores de alojamiento web que operan con energía renovable. Estos proveedores se comprometen a minimizar la huella de carbono de sus centros de datos utilizando fuentes de energía como la solar, eólica o hidroeléctrica. Esta práctica es fundamental para reducir el impacto ambiental del alojamiento de aplicaciones web (Siegman, 2023).

Las **Redes de Entrega de Contenido** (CDN) distribuyen el contenido a través de una red de servidores ubicados estratégicamente en todo el mundo. Esto reduce la distancia que los datos deben viajar, disminuyendo el consumo de energía y mejorando los tiempos de carga. Proveedores sostenibles de CDN, como Cloudflare y Akamai, ofrecen soluciones eficientes para la entrega de contenido (OpenReplay, 2023). Las CDN no solo mejoran la velocidad de entrega del contenido sino que también aumentan la redundancia y la fiabilidad, asegurando que el contenido esté disponible incluso si un servidor falla.

La **minimización de recursos** es otra técnica que implica reducir al máximo el uso de materiales y energía en todas las etapas del desarrollo web, desde la programación hasta la implementación y el mantenimiento. Esto puede lograrse a través de la reutilización de componentes, la adopción de prácticas de codificación eficientes y el uso de tecnologías que faciliten la reducción del consumo energético (Siegman, 2023). Para que un sistema sea verdaderamente sostenible, también debe ser duradero. Sin embargo, ¿qué características debe cumplir un sistema durable?.

5.1.2 Sistema durable

Un sistema durable es aquel que está diseñado para perdurar y funcionar de manera eficiente y efectiva a lo largo del tiempo, adaptándose a cambios y manteniendo un uso óptimo de los

recursos (Mazo & Camargo, 2024). La durabilidad de un sistema se evalúa mediante su capacidad de evolucionar, ser económico y confiabilidad:

- **Evolutivo**: Un sistema durable debe ser adaptable para responder a las necesidades crecientes o cambiantes. Esto incluye la capacidad de añadir nuevas funcionalidades o modificar las existentes sin necesidad de reescribir el sistema completo y la habilidad de manejar una carga incrementada sin degradación del rendimiento. Este concepto está fundamentado en los principios de diseño de sistemas flexibles y escalables. Según Sommerville (2016), la adaptabilidad y la evolución son esenciales para asegurar la longevidad de un sistema de software.
- **Económico**: La eficiencia en el uso de los recursos es crucial. Un sistema durable debe minimizar la utilización de energía y otros recursos, optimizando su funcionamiento para ser lo más eficiente posible. Esto se alinea con los principios de Green IT, que buscan reducir el impacto ambiental de las tecnologías de la información. Lamb (2009) enfatiza la importancia de diseñar sistemas que sean sostenibles desde el punto de vista económico y medioambiental.
- **Confiabilidad**: La capacidad de un sistema para gestionar fallos, errores y ataques, y recuperarse rápidamente en caso de problemas, es fundamental para su durabilidad. Los sistemas deben estar diseñados para funcionar en modo degradado y mantener la integridad y disponibilidad de los datos. Según Avizienis et al. (2004), la fiabilidad y la resiliencia son características clave para los sistemas críticos, asegurando su funcionamiento continuo y seguro.

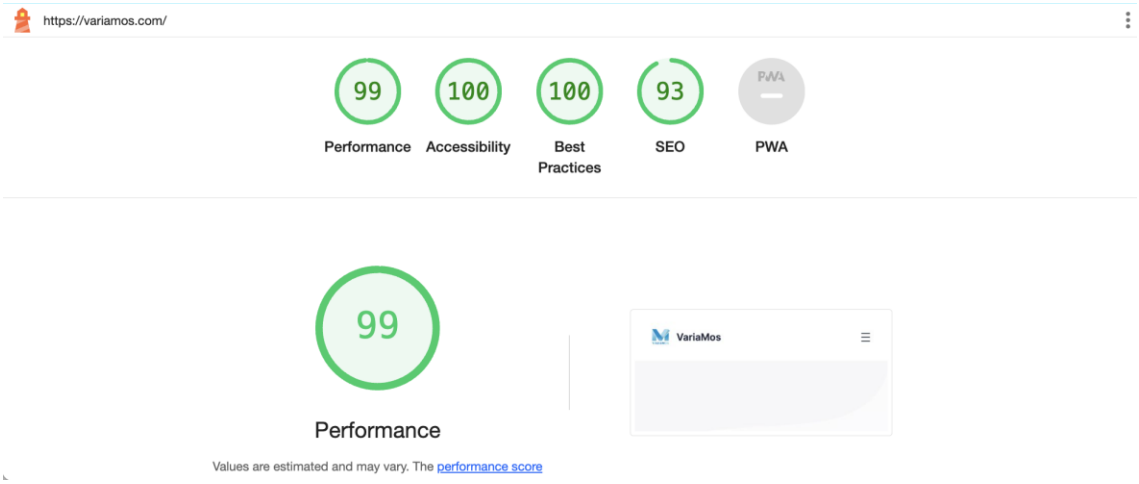
5.1.2 Herramientas para la evaluación de sostenibilidad

- **Lighthouse**: Es una herramienta de auditoría de código abierto incluida en Chrome DevTools que ayuda a los desarrolladores a mejorar la calidad de las páginas web. Proporciona auditorías de rendimiento, accesibilidad, mejores prácticas, SEO y aplicaciones web progresivas (PWA). Lighthouse simula una carga de página y analiza varios aspectos, generando un informe detallado con puntuaciones y recomendaciones para mejorar la eficiencia y sostenibilidad de las aplicaciones web. La herramienta también ofrece sugerencias específicas para optimizar la carga de la página, reducir el tiempo de

respuesta del servidor y mejorar la accesibilidad general del sitio (Google Developers, 2023). Por ejemplo, en la Fig. 1 se aprecia el puntaje que obtiene el sitio web https://variamos.com.

Figura 1

Vista principal de Lighthouse



Nota. Reporte general de Lighthouse de la página variamos.com.

- **Website Carbon Calculator** es una herramienta popular que estima la huella de carbono de una página web. Esta herramienta calcula las emisiones de CO2 generadas por cada vista de página, tomando en cuenta el consumo de electricidad de los centros de datos, las redes de transmisión y los dispositivos del usuario. Por ejemplo, el promedio global de una página web produce aproximadamente 0.8 gramos de CO2 por vista de página. Utilizar esta herramienta puede ayudar a los desarrolladores a identificar áreas de mejora para reducir la huella de carbono de sus sitios web (Website Carbon, 2023).

5.2 Patrón Modelo-Vista-Controlador

El patrón de diseño MVC es fundamental en el desarrollo de aplicaciones web seguras. Al separar la aplicación en tres componentes interconectados, Modelo, Vista y Controlador, el patrón ayuda a aislar las preocupaciones, haciendo que la aplicación sea más mantenible y escalable.

El Modelo maneja los datos y la lógica del negocio, mientras que la Vista gestiona la capa de presentación, y el Controlador procesa la entrada del usuario y coordina entre el Modelo y la Vista.

Esta separación de intereses ayuda a implementar medidas de seguridad de manera más efectiva. Por ejemplo, la validación de datos y las reglas de negocio pueden imponerse en el Modelo, mientras que el Controlador gestiona la lógica de autenticación y autorización. La Vista se enfoca en presentar los datos de manera segura, minimizando el riesgo de exponer información sensible.

Por otro lado, la inversión de control (IoC) permite una gestión eficiente de las dependencias entre estos componentes, promoviendo la construcción de aplicaciones modulares y eficaces.

5.2.1 Inversión de control

La IoC es un principio de diseño en el cual el flujo de control de un programa se transfiere desde los componentes específicos a un contenedor o framework externo. Tradicionalmente, los objetos son responsables de crear sus propias dependencias, pero con IoC, esta responsabilidad se invierte.

En lugar de que los objetos gestionen la creación y la gestión de sus dependencias, un contenedor de IoC como Spring (y Spring Boot) se encarga de estas tareas. Esto permite que los objetos se conviertan en participantes pasivos dentro del sistema, confiando en el contenedor para proporcionar y gestionar todas las dependencias necesarias.

El propósito principal de la IoC es reducir el acoplamiento entre componentes, aumentar la reutilización del código y facilitar el mantenimiento y las pruebas. Esto se logra al permitir que el contenedor gestione la creación, la configuración y el ciclo de vida de los objetos, lo que resulta en una mayor flexibilidad y modularidad en la aplicación. . IoC también facilita la implementación de patrones de diseño como la **Inyección de Dependencias** (DI), que es una forma específica de IoC (Ladd, Davison, Yates, & Devijver, 2006).

La DI funciona proporcionando las dependencias necesarias a un objeto desde el exterior, en lugar de que el objeto las cree por sí mismo. La DI puede realizarse mediante constructores, métodos setter o interfaces, lo que permite una configuración flexible y un alto grado de pruebas.

5.3 Ciberseguridad

Los frameworks como Laravel y Spring Boot ofrecen herramientas robustas y mejores prácticas para mejorar la seguridad de las aplicaciones web, convirtiéndolos en opciones cruciales para los desarrolladores que buscan construir sistemas seguros.

Por ejemplo, Laravel proporciona un conjunto completo de características de seguridad diseñadas para proteger las aplicaciones web de vulnerabilidades comunes. Una de las características clave es la protección contra ataques de Cross-Site Scripting (XSS).

5.3.1 Cross-Site Scripting

El XSS es un tipo de vulnerabilidad de seguridad común en las aplicaciones web que permite a los atacantes inyectar scripts maliciosos en contenido web visto por otros usuarios. Esta vulnerabilidad se aprovecha al introducir código JavaScript malicioso en sitios web que no validan o sanitizan adecuadamente las entradas del usuario. Como resultado, el script malicioso puede ejecutar en el navegador de la víctima, robando información sensible como cookies, credenciales de inicio de sesión y otros datos personales (Gupta & Gupta, 2015).

Un ejemplo conocido de un ataque XSS es el incidente de 2012 en la aplicación de sistema de pagos, PayPal. En este caso, los atacantes encontraron una vulnerabilidad XSS en las páginas de inicio de sesión de PayPal. Utilizando una URL especialmente diseñada, los atacantes pudieron inyectar scripts maliciosos que redirigían a los usuarios a páginas de inicio de sesión falsificadas, donde se robaban las credenciales de los usuarios (Gupta & Gupta, 2015).

Laravel implementa varias medidas para proteger las aplicaciones contra ataques de XSS. Una de las principales técnicas es el escapado automático de salidas en sus vistas Blade, donde cualquier dato pasado a las vistas se escapa automáticamente usando dobles llaves {{ }}. Esto convierte los caracteres especiales en entidades HTML seguras. El framework también promueve el uso de su sistema de validación de entradas para asegurar que los datos recibidos a través de formularios

estén correctamente validados y saneados. Implementar reglas de validación como string y regex ayuda a garantizar que solo se acepten datos apropiados (Laravel, s.f.).

5.3.2 Cross-Site Request Forgery

CSRF (Cross-Site Request Forgery) es un tipo de ataque en el que un atacante engaña a un usuario para que ejecute acciones no deseadas en una aplicación web en la que está autenticado. Estos ataques explotan la confianza que una aplicación web tiene en el navegador de un usuario. CSRF puede llevar a la ejecución de acciones no autorizadas, como transferencias de fondos, cambios de contraseña o la compra de productos sin el consentimiento del usuario.

Para prevenir ataques CSRF, es fundamental implementar tokens de verificación únicos y secretos que se envían con cada solicitud que puede cambiar el estado. Estos tokens aseguran que la solicitud se originó desde la aplicación web legítima y no desde un sitio malicioso. Laravel utiliza este método incorporando tokens CSRF en sus formularios y solicitudes AJAX, asegurando así la protección contra tales ataques (Mazo, s.f., diapositiva 172).

Por otro lado, Spring Boot posee la misma funcionalidad de la mano de Spring Security, un framework de seguridad robusto para aplicaciones Java y que se puede incorporar como una librería. Este mecanismo se basa en asegurar que cada solicitud que cambie el estado de la aplicación vaya acompañada de un token único que el servidor puede verificar. Spring Security genera automáticamente estos tokens CSRF y los incluye en los formularios como campos ocultos. Por ejemplo, en un formulario HTML, se inserta un campo oculto *_csrf* con un valor de token único. Cuando se envía el formulario, Spring Security intercepta la solicitud y valida el token; si está ausente o es inválido, la solicitud es rechazada. Esta validación garantiza que solo se procesen solicitudes legítimas.

5.3.3 Inyección SQL

La inyección SQL permite a los atacantes acceder sin restricciones a las bases de datos subyacentes y a la información sensible que contienen. Este tipo de ataque ocurre cuando los datos

proporcionados por el usuario se incluyen en una consulta SQL de manera que parte de la entrada del usuario se trata como código SQL, lo que permite a los atacantes enviar comandos SQL directamente a la base de datos (Halfond, Viegas, & Orso, 2006).

Un ejemplo común de inyección SQL es la utilización de un formulario de inicio de sesión vulnerable. Si una aplicación web construye dinámicamente una consulta SQL a partir de las entradas del usuario sin una validación adecuada, un atacante puede inyectar código malicioso. Por ejemplo, al enviar *'; DROP TABLE users;--* en un campo de nombre de usuario, la consulta SQL resultante podría eliminar la tabla de usuarios completa en la base de datos (Halfond, Viegas, & Orso, 2006).

Para prevenir los ataques de inyección SQL, Laravel utiliza su ORM Eloquent, que emplea el enlace de parámetros PDO. Este mecanismo asegura que las consultas SQL se ejecuten con parámetros enlazados, protegiendo la aplicación de entradas de usuario maliciosas que podrían alterar la estructura de la consulta prevista. Además, la protección CSRF integrada de Laravel genera un token único para cada sesión de usuario, que debe incluirse en todas las presentaciones de formularios. Este token ayuda a evitar que comandos no autorizados sean ejecutados por atacantes.

Por otro lado, Spring Boot utiliza varias estrategias para prevenir ataques de inyección SQL de manera efectiva. Uno de los métodos principales es mediante el uso de JPA (Java Persistence API) y Spring Data JPA. Estas herramientas abstraen las interacciones con la base de datos y evitan la necesidad de concatenar consultas SQL manualmente, reduciendo así el riesgo de inyección. Al usar repositorios y métodos de consulta, los desarrolladores pueden escribir código de acceso a la base de datos sin incrustar directamente consultas SQL, asegurando que la entrada del usuario se maneje de manera segura. Además, Spring Boot enfatiza el uso de consultas preparadas y consultas parametrizadas a través de JdbcTemplate y otras herramientas de acceso a bases de datos. Las consultas preparadas aseguran que las consultas SQL se precompilen y las entradas del usuario se traten como datos en lugar de código ejecutable (Baeldung, s.f.).

6 Metodología

Se utilizará un enfoque mixto que combinará métodos cualitativos y cuantitativos para evaluar la sostenibilidad en el desarrollo web. La primera fase consiste en la definición del concepto y el marco teórico, donde se revisará literatura existente sobre desarrollo sostenible y tecnologías web, incluyendo directrices como el W3C sobre sostenibilidad web. Además, se consultarán libros adicionales recomendados como *Sustainable Web Design In 20 Lessons* de Michael Andersen, *The Ethical Design Handbook* de Trine Falbe, *World Wide Waste* de Gerry McGovern, *Green IT et accessibilité* de Hervé Boisgontier, y *Ecoconception web: les 115 bonnes pratiques* de Frédéric Bordage.

En la fase de inmersión en el trabajo, se desarrollará una página web aplicando principios de sostenibilidad, utilizando frameworks modernos como React. Este desarrollo incluirá información sobre técnicas de optimización de recursos en formato de recomendaciones y ejercicios de código para el lector. Simultáneamente, se realizará una revisión sistemática de literatura para consolidar conocimientos teóricos y prácticos sobre desarrollo web sostenible, identificando mejores prácticas y tendencias actuales.

La fase de desarrollo y documentación implicará la implementación de proyectos web utilizando frameworks como React, con una evaluación detallada del desempeño y el consumo de recursos. Se utilizarán herramientas como Google Lighthouse para medir la eficiencia y el impacto ambiental de las aplicaciones desarrolladas. Esta fase permitirá identificar áreas de mejora y optimización en la implementación de prácticas sostenibles.

Finalmente, se realizará un análisis de los resultados obtenidos, se propondrán recomendaciones basadas en los hallazgos y se expondrán dichas ideas en el trabajo de grado. Las propuestas de mejora se orientarán a fomentar un desarrollo web más eficiente y responsable con el medio ambiente, contribuyendo así a la sostenibilidad en la industria tecnológica.

7 Resultados

Los resultados de este trabajo dieron lugar a la creación de dos libros en inglés titulados *Sustainable Web Development* y *Designing and Implementing Modern Web Applications - Patterns, Frameworks and OOP Design Principles*, los cuales buscan ser una guía para introducir a los desarrolladores en el desarrollo de aplicaciones web sostenibles y seguras, respectivamente. Desde el contenido teórico hasta los ejercicios abarcan las temáticas mencionadas a lo largo del informe.

7.1 Desarrollo web sostenible

El principal objetivo del libro *Sustainable Web Development* es enseñar a los desarrolladores web cómo crear aplicaciones sostenibles y eficientes. La estructura del libro está cuidadosamente diseñada para guiar al lector desde los conceptos más básicos del desarrollo sostenible hasta las técnicas más avanzadas.

7.1.1 Fundamentos de la web y la sostenibilidad

Desde los primeros capítulos se proporciona una visión general de la importancia de la sostenibilidad en el desarrollo web. Se discuten las implicaciones ambientales de las tecnologías web y se presentan estrategias para reducir el impacto ecológico de las aplicaciones web. El capítulo también aborda cómo las decisiones de diseño y desarrollo pueden influir en la eficiencia energética de los sitios web.

A lo largo del libro, se implementa un sistema de consejos y recomendaciones para asegurar que los sitios web sean sostenibles desde las tres perspectivas de un sistema durable: facilidad de evolución, frugalidad en el uso de recursos y confiabilidad.

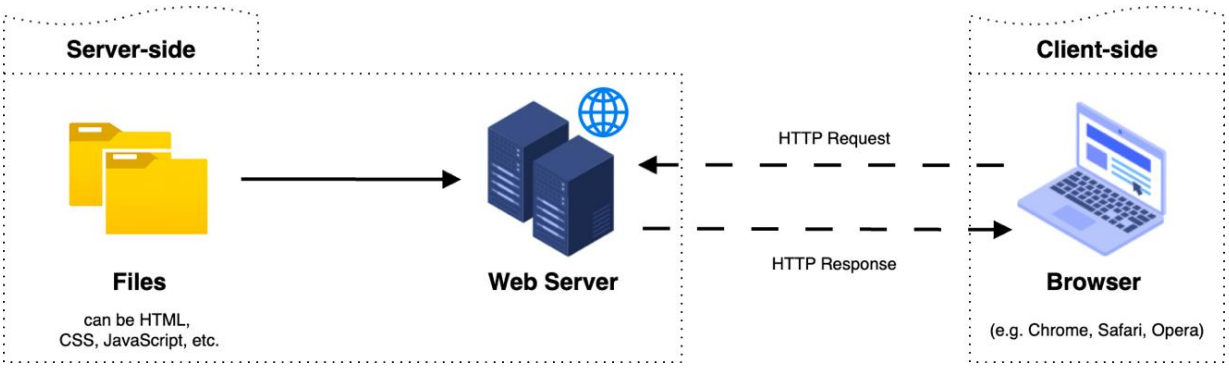
Por otro lado, se expone una base sólida sobre las aplicaciones web en general, introduciendo arquitecturas como la de cliente-servidor y cómo facilita la gestión eficiente de recursos. Se hace una mención a los sitios estáticos, recalcando su utilidad cuando se tiene un número reducido de

páginas y se desea enviar el mismo contenido a todos los usuarios, pero que terminan por volverse ineficientes a medida que el número de páginas crece.

Se ilustra la carga de un sitio estático a través de una simple petición, tal como se aprecia en la Fig. 2.

Figura 2

Proceso de carga de un sitio estático

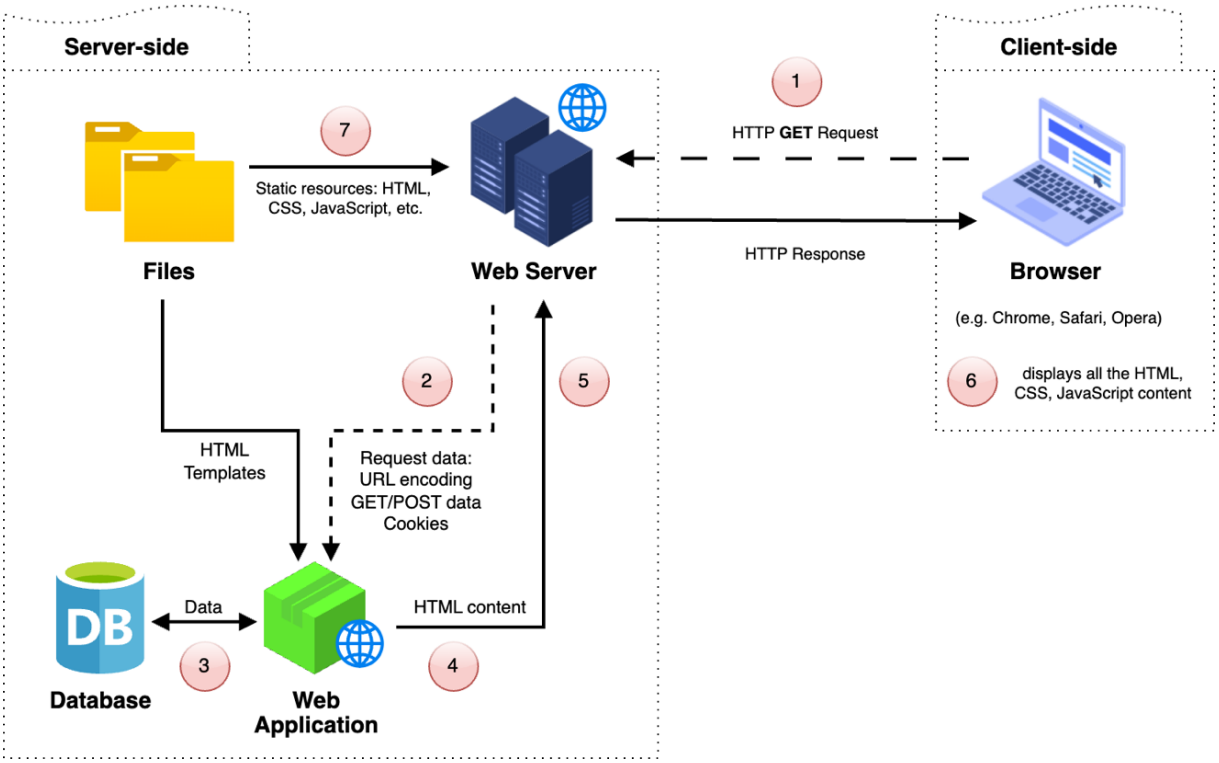


Nota. Solicitud de una petición HTTP para la carga de una página estática.

En contraste, se mencionan los sitios dinámicos y cómo estos pueden generar y devolver contenido basado en la URL específica y los datos solicitados, en lugar de devolver siempre el mismo archivo codificado para una URL particular. En la Fig. 3, se ilustra el proceso para que un usuario pueda visualizar una página dinámica al momento de cargarla en su navegador

Figura 3

Proceso de carga de un sitio dinámico



Nota. Solicitud de una petición HTTP para la carga de la página de una aplicación web.

7.1.2 Fundamentos del desarrollo web

Mientras el libro avanza, se introducen los conceptos básicos de HTML y CSS. Estos capítulos están diseñados para que el lector entienda la importancia de una estructura de código limpia y semántica, así como las mejores prácticas para la accesibilidad y el rendimiento.

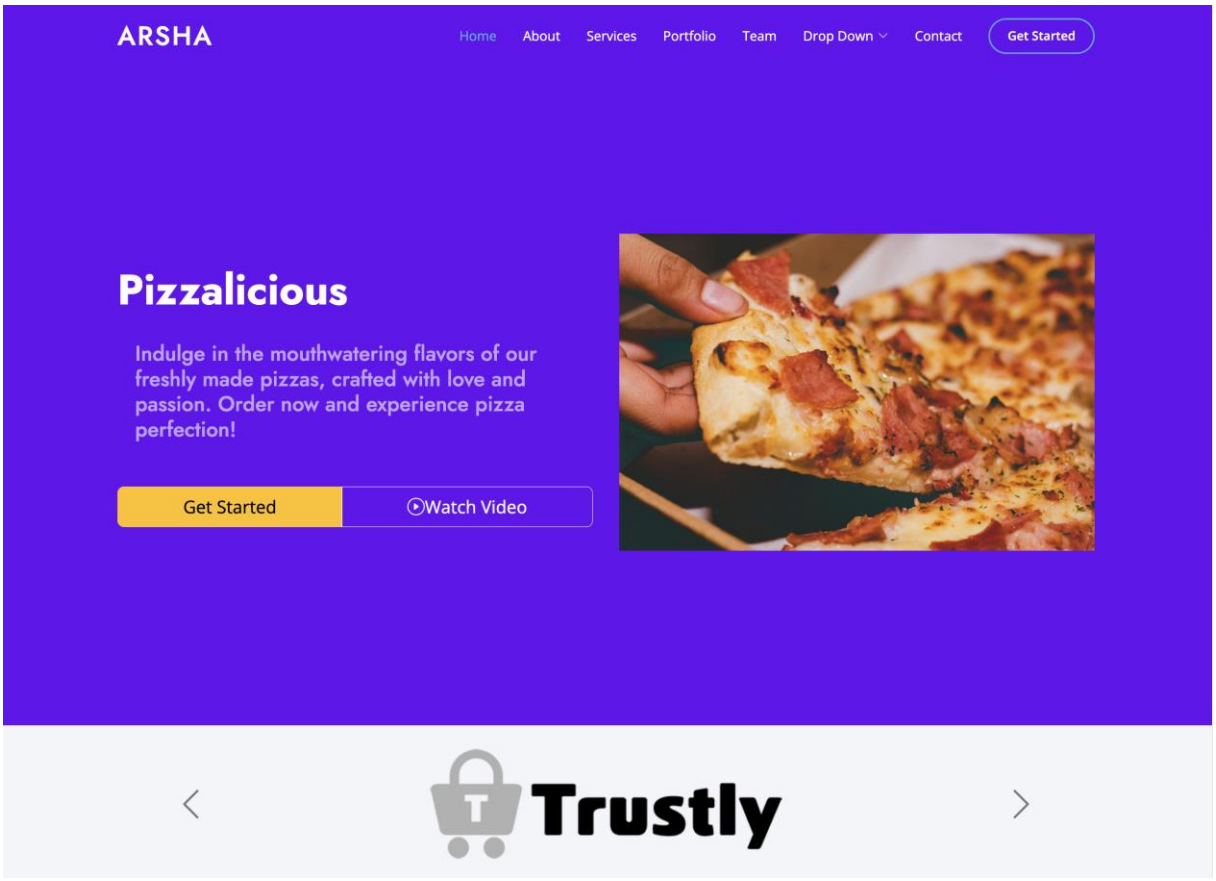
El lector comienza con ejercicios prácticos como la creación de una página HTML simple que debe incluir elementos básicos como encabezados, párrafos, imágenes y enlaces para crear una calculadora relacionada con estadísticas de peso, en el cual se enfatiza el uso de aplicar etiquetas semánticas de HTML5, que no solo organizan el contenido de manera lógica, sino que también mejoran la accesibilidad y la visibilidad en el algoritmo del motor de búsqueda.

En cuanto a CSS, los ejercicios incluyen el diseño de una página web adaptable enfatizando técnicas como media queries y flexbox. Los ejercicios propuestos proponen hacer uso del

framework de frontend, Bootstrap, para crear el diseño de la página de un comercio electrónico que se adapte a diferentes tamaños de pantalla, asegurando que la página sea visualmente atractiva y funcional en dispositivos móviles, tabletas y computadoras de escritorio. Para este caso, en la Fig. 4 se visualiza el resultado final del diseño adaptable propuesto.

Figura 4

Página de inicio de Pizzalicious



Nota. Página web realizada utilizando una plantilla de Bootstrap.

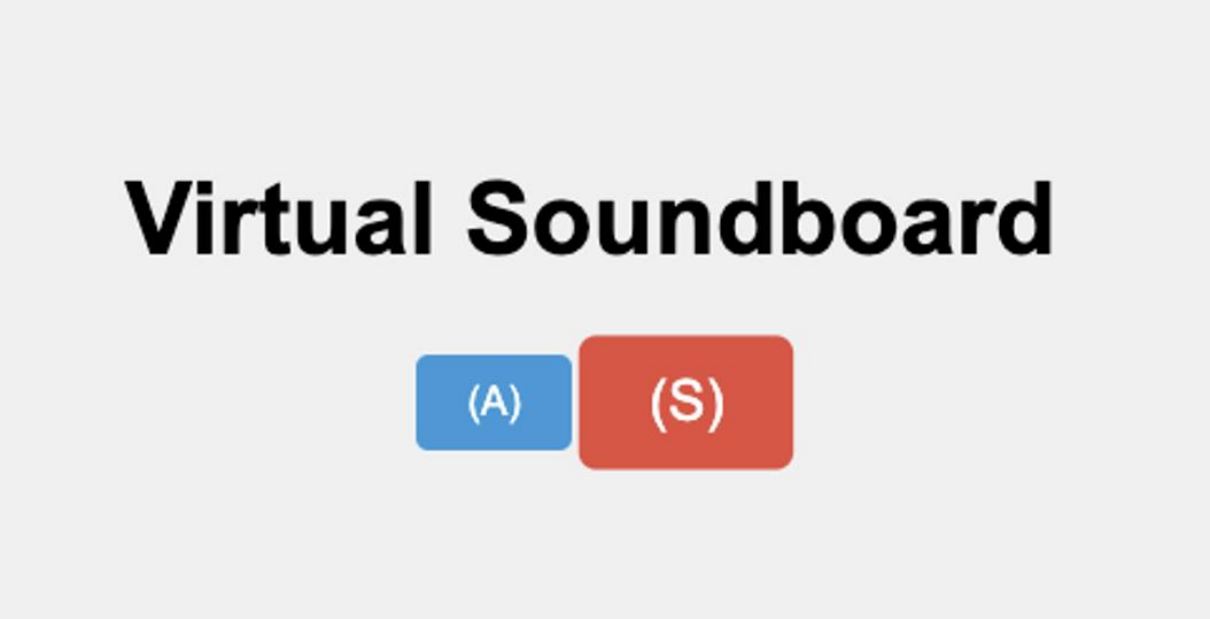
7.1.3 Fundamentos de programación

A medida que el lector avanza, se profundiza en JavaScript, para enfocarse en la creación de interacciones dinámicas en las páginas web. Aquí se cubren desde las bases de la sintaxis y la lógica de programación hasta técnicas más complejas como la manipulación del DOM (Document Object Model) y la gestión de eventos.

Los ejercicios propuestos incluyen la creación de *scripts* que manipulan el DOM para cambiar el contenido y el estilo de los elementos en respuesta a las acciones del usuario. Por ejemplo, los lectores deben desarrollar una aplicación que haga usa de la librería que genera números aleatorios en JavaScript, una página que contenga una lista de botones que permita reproducir sonidos al oprimir un determinado botón, como se ilustra en la Fig. 5.

Figura 5

Página de consola de audio virtual



Nota. Página web realizada como ejercicio avanzado de JavaScript.

Además, se introducen técnicas avanzadas de gestión de eventos utilizando *addEventListener* para implementar interacciones más complejas, como formularios que validan la entrada del usuario en tiempo real.

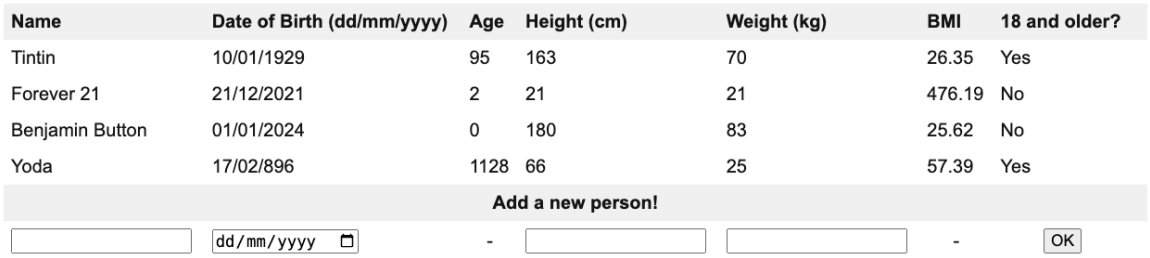
La importancia de la tipificación estática se introduce con TypeScript, un lenguaje derivado de JavaScript que agrega tipos estáticos para mejorar la mantenibilidad y reducir errores en el código. TypeScript se convierte en una herramienta crucial para proyectos de gran escala, permitiendo un desarrollo más estructurado y menos propenso a fallos.

Los ejercicios en esta sección requieren que los lectores conviertan proyectos JavaScript existentes a TypeScript, añadiendo tipos explícitos a variables, funciones y objetos. Un ejercicio típico

incluye la refactorización de un comercio electrónico, donde los lectores deben definir interfaces para los productos y las funciones de manipulación del inventario. También se les pide implementar una listado con información personal que permite añadir, editar y eliminar dicha información dinámicamente. El resultado de dicho ejercicio se aprecia en la siguiente Fig. 6.

Figura 6

Listado de información personal



Name	Date of Birth (dd/mm/yyyy)	Age	Height (cm)	Weight (kg)	BMI	18 and older?
Tintin	10/01/1929	95	163	70	26.35	Yes
Forever 21	21/12/2021	2	21	21	476.19	No
Benjamin Button	01/01/2024	0	180	83	25.62	No
Yoda	17/02/896	1128	66	25	57.39	Yes

Add a new person!

dd/mm/yyyy - - OK

Nota. Página web realizada como ejercicio avanzado de TypeScript.

7.1.4 React

El libro culmina con un capítulo dedicado a React, donde se enseñan técnicas avanzadas para crear aplicaciones web utilizando componentes funcionales y hooks. React es un framework que permite construir interfaces de usuario de manera declarativa y eficiente, facilitando la creación de aplicaciones web interactivas y de alto desempeño.

El capítulo comienza con una introducción a los conceptos fundamentales de React, explicando su filosofía de desarrollo basada en componentes. Asimismo, se destaca la importancia de pensar en la UI como una colección de componentes reutilizables que encapsulan su propio estado y lógica.

Posteriormente, se explica el manejo del estado y las props en React. Se menciona cómo el estado local permite manejar datos internos de los componentes y cómo las props facilitan la comunicación entre componentes.

Con ello, el capítulo culmina al introducir los hooks, un concepto clave que permite utilizar el estado y otras funcionalidades de React en componentes funcionales. Se cubren hooks como *useState, useEffect* y *useContext*.

Para afianzar este concepto, se propone crear una aplicación de lista de tareas (to-do list) donde pueden agregar, editar y eliminar tareas. Otros ejercicios a desarrollar es un cronómetro, utilizando hooks para manejar el tiempo y controlar el intervalo de tiempo, junto con botones para iniciar, detener y reiniciar el cronómetro.

Finalmente, se propone crear una aplicación del clima que obtenga datos de una API, mostrando información meteorológica basada en la entrada del usuario, utilizando hooks para gestionar el estado y efectos secundarios. Estos ejercicios refuerzan el manejo del estado, la creación de componentes reutilizables y la optimización del rendimiento en React.

7.2. Implementación de patrones y frameworks para aplicaciones web

El objetivo del libro *Designing and Implementing Modern Web Applications - Patterns, Frameworks and OOP Design Principles* es proporcionar una guía sobre el desarrollo de aplicaciones web con una arquitectura de software sólida. El contenido del libro incluye la implementación de principios y prácticas de seguridad desde las etapas iniciales del desarrollo, abarcando temas como la correcta utilización de patrones de diseño como MVC, el manejo de datos sensibles de los usuarios y la implementación de mecanismos de seguridad contra amenazas como CSRF.

7.2.1 Introducción a la ingeniería web

En el libro se comienza con una introducción a la ingeniería web como una rama de la ingeniería de software, dedicada a la aplicación de conceptos, métodos, técnicas y herramientas para el análisis de requerimientos, diseño, implementación, prueba, operación y mantenimiento de aplicaciones web de alta calidad.

Por otro lado, se aborda el concepto de la "crisis web", debido a la omnipresencia de las aplicaciones web y la creciente interdependencia entre sistemas y usuarios a nivel global,

destacando la importancia de contrarrestar este fenómeno al diseñar aplicaciones que no solo sean funcionales sino también escalables.

7.2.2 Principios y patrones de software

En el segundo libro se introduce al lector los principios de software esenciales, sobre todo destacando los principios SOLID, que son fundamentales para la creación de aplicaciones robustas, mantenibles y escalables. Cada principio es explicado en detalle, con ejemplos prácticos que ilustran cómo se pueden aplicar en el desarrollo de aplicaciones web o de escritorio en Java.

En las siguientes subsecciones, se incluirá la información que incluye cada uno de los principios SOLID.

7.2.2.1 Principio de responsabilidad única. Para este principio se establece que una clase debe tener una, y solo una, razón para cambiar. Este principio se enfoca en asegurar que cada clase tenga una única responsabilidad o propósito.

De ejemplo práctico, se presenta un caso donde una clase *ClientManager* originalmente maneja tanto la lógica de autenticación como la lógica de manejo de perfiles de usuario. En el libro se guía al lector para dividir esta clase en dos: *AuthManager* para la autenticación y *ProfileManager* para el manejo de perfiles, demostrando cómo esta separación mejora la claridad y la mantenibilidad del código.

7.2.2.2 Principio abierto/cerrado. En este principio, se sostiene que las entidades de software (clases, módulos, funciones, etc.) deben estar abiertas para la extensión, pero cerradas para la modificación.

Para explicar este principio, se incluye un ejercicio que trata acerca de un sistema de transportes que posee una clase base llamada *Transport*. En este sistema de transportes se desea implementar un método para reabastecer de combustible a los medios de transporte. Sin embargo, no todos los medios de transporte (clases derivadas) pueden ser reabastecidos, tales como la bicicleta. Para

solucionar ello, se invita al lector a crear una nueva interfaz llamada *Motorized*, específica para las subclases que poseen motor y puedan implementar las características que conlleva tener uno. De esta manera se extiende el comportamiento de la superclase *Transport*, sin tener que afectar las entidades previamente implementadas y evitando la violación del principio.

7.2.2.3 Principio de sustitución de Liskov. Este principio indica que los objetos de una clase derivada deben ser reemplazables por objetos de la clase base sin alterar el correcto funcionamiento del programa.

Como ejemplo, se presenta una jerarquía de clases con *Bird* como clase base y *Penguin* como clase derivada. Dado que los pingüinos no pueden volar, una implementación incorrecta haría que el método *fly* en *Penguin* lance una excepción, violando el principio. El libro guía al lector a redefinir la jerarquía de clases para crear una clase *FlyingBird* intermedia, separando las capacidades de vuelo y asegurando que *Penguin* no necesite implementar fly.

7.2.2.4 Principio de segregación de interfaces. El principio sugiere que los clientes no deben estar forzados a depender de interfaces que no utilizan.

Para explicar el principio, se muestra una interfaz *Worker* con métodos *work*, *eat*, y *sleep*. Para adherirse al principio, se sugiere dividir esta interfaz en varias más específicas (*Workable*, *Eatable*, *Sleepable*), permitiendo que diferentes clases de trabajadores (por ejemplo, *OfficeWorker* y *RobotWorker*) implementen solo las interfaces que realmente necesitan, reduciendo la complejidad y adhiriéndose al principio.

7.2.2.5 Principio de inversión de dependencias. Establece que las dependencias deben ser sobre abstracciones, no sobre entidades concretas.

Para ilustrar este principio, se muestra una clase *Car* que depende de una interfaz *Engine* en lugar de una implementación concreta como *GasEngine*. El libro muestra cómo utilizar inyección de dependencias para pasar una instancia de *Engine* al constructor de *Car*, permitiendo cambiar

fácilmente la implementación del motor sin modificar la clase *Car*. Esto hace que el sistema sea más flexible y fácil de mantener.

7.2.3 Frameworks web y la inversión de control

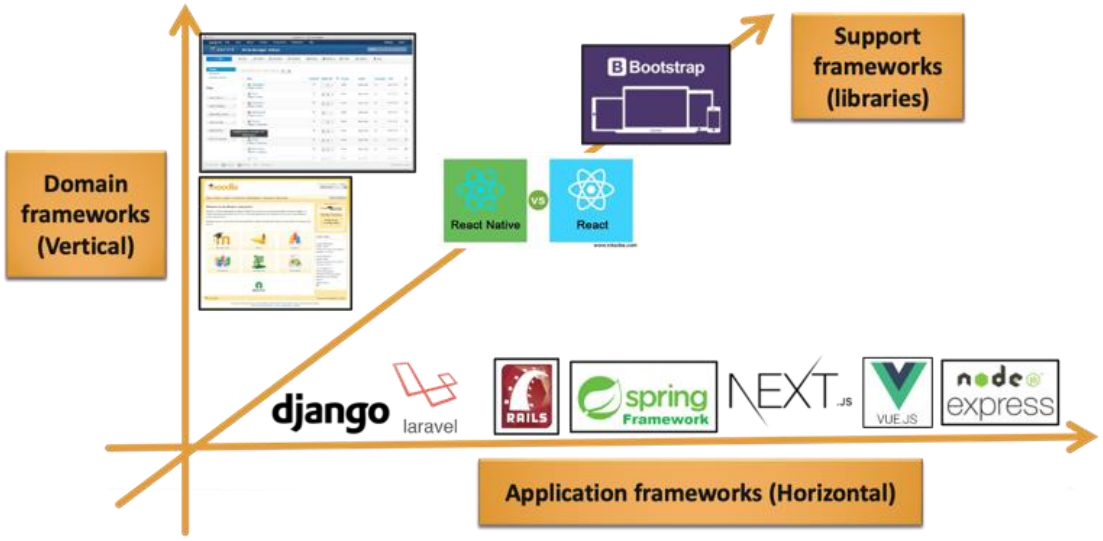
El libro comienza un capítulo mencionando que las librerías y los frameworks son términos que cumplen propósitos distintos.

Las bibliotecas son colecciones de códigos preescritos que los desarrolladores pueden utilizar para realizar tareas específicas, otorgándoles el control sobre cuándo y dónde se usan estas funciones en su código. Por ejemplo, jQuery es una biblioteca de JavaScript que simplifica la manipulación del documento HTML y el manejo de eventos. Sin embargo, mantener actualizadas las bibliotecas y abordar las posibles vulnerabilidades de seguridad puede incrementar la carga de trabajo en el desarrollo.

A diferencia de las bibliotecas (también llamadas frameworks de soporte), que están diseñadas para resolver problemas específicos o añadir características particulares a un programa, los frameworks ofrecen algo más genérico y reutilizable. Ejemplos de frameworks incluyen Vue.js, que ofrece un conjunto completo de herramientas para construir interfaces de usuario y aplicaciones de una sola página.

Un framework puede abarcar todo lo necesario para el desarrollo de aplicaciones y se puede ver como un esqueleto que incluye códigos preescritos que ayudan a los desarrolladores a construir aplicaciones de alta calidad, ya sea para un dominio específico (frameworks de dominio) o para cualquier tipo de aplicaciones web (frameworks de aplicación). En la siguiente Fig. 7, se puede apreciar las tecnologías que corresponden a cada tipo de framework.

Figura 7
Diagrama de los diferentes tipos de framework



Nota. Diagrama de algunos frameworks y el tipo de framework correspondiente.

En este contexto, la IoC se refiere a cómo un framework controla el flujo de la aplicación, llamando al código del desarrollador en puntos específicos durante la ejecución. Con las bibliotecas, los desarrolladores mantienen el control, llamando a las funciones de la biblioteca según sea necesario. En cambio, los frameworks gestionan el flujo de control, y los desarrolladores integran su código en el framework, que luego toma el control del proceso de ejecución.

7.2.3.1 Frameworks de dominio. Los frameworks de dominio, también conocidos como frameworks verticales, se enfocan en dominios o industrias específicas, proporcionando soluciones adaptadas para tipos de aplicaciones particulares. Estos frameworks facilitan el desarrollo al ofrecer módulos preconstruidos y funcionalidades diseñadas para satisfacer los requisitos únicos de sus respectivos dominios. Se incluyen ejemplos como Moodle para e-learning y WordPress para la publicación de contenido.

7.2.3.2 Frameworks de aplicación. Los frameworks de aplicación, o frameworks horizontales, son herramientas versátiles que ofrecen una amplia gama de características y funcionalidades aplicables en diversos dominios e industrias. Ejemplos de estos frameworks incluyen Next.js para aplicaciones web renderizadas en el servidor y Django para desarrollo rápido con Python. Se

destaca que, aunque los frameworks de aplicación ofrecen ventajas significativas en términos de estructura y productividad, presentan posibles problemas de arquitectura poco limpia.

7.2.4 Patrón de diseño MVC

Uno de los patrones más destacados abordados en el libro es el MVC. Este patrón es crucial para la arquitectura de aplicaciones web, ya que separa la aplicación en tres componentes interrelacionados: el Modelo, la Vista y el Controlador.

Para explicar el patrón, se hace uso de 2 frameworks: Spring Boot y Laravel. A través de estos frameworks, se ilustra la implementación de cada componente del MVC y se sugieren consejos para proteger cada componente frente a ataques cibernéticos, y así, crear una aplicación web segura y escalable.

7.2.4.1 Modelo. El modelo maneja los datos y la lógica del negocio. En el libro, se profundiza en cómo estructurar el modelo de manera eficiente. Para ello, se discuten técnicas para validar y sanitizar datos, y aplicar las reglas de negocio.

7.2.4.2 Vista. La vista es responsable de presentar la interfaz de usuario. En esta sección, el libro explica cómo diseñar vistas que no solo sean estéticamente agradables, sino también que inspiren confianza en el usuario. Se abordan temas como la sanitización de salidas para prevenir Cross-Site Scripting (XSS). Se incluyen ejercicios prácticos que guían al lector en la creación de interfaces de usuario accesibles a través de los motores de plantilla de cada framework, Blade y Thymeleaf.

7.2.4.3 Controlador. El controlador maneja la entrada del usuario y coordina las interacciones entre el modelo y la vista. El libro detalla cómo implementar controladores que gestionen de manera segura las solicitudes y respuestas.

7.2.5 Protección contra Ataques CSRF

En el libro se mencionan ataques como el CSRF, un tipo de ataque en el que un atacante engaña a un usuario autenticado para que ejecute acciones no deseadas en una aplicación web. En ello, se explican diversas técnicas que poseen los frameworks para proteger las aplicaciones contra estos ataques, incluyendo la implementación de tokens CSRF y el uso de cabeceras HTTP adecuadas.

7.2.6 Variantes del Patrón MVC

En el libro también se exploran varias variantes del patrón MVC que han surgido para abordar diferentes necesidades y desafíos en el desarrollo de aplicaciones de software.

7.2.6.1 Model-View-ViewModel (MVVM). MVVM es una variante de MVC que se utiliza comúnmente en aplicaciones que requieren una separación más clara entre la lógica de negocios y la interfaz de usuario. En MVVM, el ViewModel actúa como un intermediario entre el Modelo y la Vista, manejando la lógica de presentación y facilitando el enlace de datos bidireccional. Esta separación adicional ayuda a mejorar la testabilidad y mantenibilidad del código, permitiendo que los desarrolladores trabajen en la lógica de la aplicación sin preocuparse por los detalles de la interfaz de usuario.

7.2.6.2 Model-View-Presenter (MVP). MVP es otra variante de MVC que se enfoca en la separación de responsabilidades entre la lógica de presentación y la interfaz de usuario. En MVP, el Presentador reemplaza al Controlador y se encarga de manipular el Modelo y actualizar la Vista. Esto permite que la lógica de presentación sea más fácil de probar, ya que el Presentador no tiene ninguna dependencia directa de la Vista. Finalmente, se comenta que el MVP es particularmente útil en aplicaciones donde la lógica de presentación es compleja y necesita ser probada de manera aislada.

7.2.7 Escalabilidad, testabilidad y mantenibilidad

El libro explora los siguientes conceptos para evaluar la efectividad y desempeño de los patrones de diseño mencionados:

- Escalabilidad: Una aplicación web escalable es aquella que puede manejar un aumento en la carga de trabajo sin comprometer el rendimiento. Esto se logra mediante una arquitectura bien diseñada y la implementación de prácticas como la optimización de bases de datos, el uso de caching y la distribución de la carga entre múltiples servidores.

- Testabilidad: La testabilidad se refiere a la facilidad con la que se pueden realizar pruebas en una aplicación para asegurar su correcto funcionamiento. Una aplicación con buena testabilidad permite realizar pruebas unitarias, de integración y funcionales de manera eficiente. Se destaca la importancia de utilizar patrones de diseño que faciliten la testabilidad, como MVVM.

- Mantenibilidad: La mantenibilidad es la capacidad de una aplicación para ser modificada y actualizada con facilidad. Esto incluye la corrección de errores, la adición de nuevas funcionalidades y la mejora del rendimiento. Se enfatiza la necesidad de seguir principios de diseño sólidos para asegurar la mantenibilidad a largo plazo.

8 Discusión

El desarrollo de aplicaciones web sostenibles es un tema de gran relevancia en la actualidad, dado el crecimiento exponencial de la web y la creciente conciencia sobre el impacto ambiental y la seguridad cibernética. A lo largo de este trabajo de grado, se abordaron estos dos, integrando prácticas y frameworks que promueven un desarrollo responsable y seguro.

La sostenibilidad en el desarrollo web se enfoca en reducir el impacto ambiental de las aplicaciones digitales. El primer libro destaca la importancia de optimizar el uso de recursos y mejorar la eficiencia energética de las aplicaciones web. Los ejemplos prácticos incluidos en el libro demuestran cómo estas técnicas pueden aplicarse en proyectos reales. Por ejemplo, la creación de una página HTML simple que utiliza etiquetas semánticas de HTML5 asegura que una página sea eficiente y accesible en la Internet. Estas prácticas no solo mejoran la sostenibilidad sino que también benefician la experiencia del usuario al reducir los tiempos de carga y mejorar la navegación.

El segundo libro se enfoca en la implementación de principios y buenas prácticas desde las etapas iniciales del desarrollo de aplicaciones web. La correcta utilización de patrones de diseño como MVC y la implementación de mecanismos de seguridad como la protección contra CSRF y la inyección SQL son fundamentales para garantizar la seguridad de las aplicaciones. Asimismo, se discuten en detalle los principios SOLID y su aplicación en el desarrollo de software seguro y mantenible. Por último, se presentan ejemplos del desarrollo de aplicaciones seguras con frameworks como Spring Boot y Laravel, mostrando cómo estos frameworks facilitan la implementación de los patrones de diseño y mejoran la eficiencia del desarrollo de aplicaciones web.

9 Conclusiones

El trabajo realizado en este proyecto de grado proporciona una comprensión profunda de la importancia de integrar la sostenibilidad y la seguridad en el desarrollo de aplicaciones web. A través de los libros *Sustainable Web Development* y *Designing and Implementing Modern Web Applications - Patterns, Frameworks and OOP Design Principles*, se han presentado principios, prácticas y herramientas que permiten a los desarrolladores crear aplicaciones web eficientes y sostenibles.

La implementación de prácticas sostenibles en el desarrollo web puede reducir drásticamente la huella de carbono de las aplicaciones, contribuyendo a la mitigación del cambio climático. El libro proporciona técnicas específicas y ejemplos prácticos que los desarrolladores pueden adoptar para hacer sus aplicaciones más eficientes y menos dependientes de recursos intensivos en carbono. Según un estudio de Aslan et al. (2018), la adopción de prácticas de desarrollo sostenible puede reducir hasta en un 20% el consumo energético de las aplicaciones web.

En cuanto a la seguridad, la guía detallada sobre la implementación de principios de seguridad y patrones de diseño como MVC ayuda a proteger las aplicaciones contra amenazas cibernéticas comunes. Esto es crucial en un contexto donde los ataques cibernéticos son cada vez más frecuentes y sofisticados. Un estudio de Verizon (2021) indica que el 43% de los ataques cibernéticos están dirigidos a aplicaciones web, subrayando la necesidad de mejores prácticas de seguridad en el desarrollo de software.

Este trabajo de grado no solo contribuye a la teoría y práctica del desarrollo web sostenible y seguro, sino que también proporciona una base para futuras investigaciones y desarrollos en esta área. En conclusión, integrar la sostenibilidad y la seguridad en el desarrollo de aplicaciones web es no solo una necesidad ambiental y de seguridad, sino también una oportunidad para mejorar la eficiencia, la responsabilidad ética y la experiencia del usuario en la industria tecnológica.

Referencias

Aslan, J., Mayers, K., Koomey, J. G., & France, C. (2018). Electricity Intensity of Internet Data Transmission: Untangling the Estimates. *Journal of Industrial Ecology, 22*(4), 785-798. https://doi.org/10.1111/jiec.12630

Baeldung. (s.f.). SQL Injection. Recuperado el 3 de julio de 2024, de https://www.baeldung.com/sql-injection

Google Developers. (2023). Lighthouse. Recuperado el 3 de julio de 2024, de https://developer.chrome.com/docs/lighthouse/overview

Gupta, S., & Gupta, B. B. (2015). Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art. *International Journal of System Assurance Engineering and Management, 6*(4), 500-509. https://doi.org/10.1007/s13198-015-0376-0

Halfond, W. G., Viegas, J., & Orso, A. (2006, Marzo). A Classification of SQL Injection Attacks and Countermeasures. En *Proceedings of the International Symposium on Software Security*. https://doi.org/10.1007/11856214_17

Ladd, S., Davison, D., Yates, C., & Devijver, S. (2006). *Expert Spring MVC and Web Flow*. Apress.

Mazo R., & Camargo G. (2024). Sustainable Web Development, ISBN: 979-8327610835, Brest, France, https://www.amazon.com/dp/B0D6GX35BT

Mazo, R. (s.f.). Web development I (diapositiva 172). Recuperado el 3 de julio de 2024.

MDN Web Docs. (2023). Introduction to Web Sustainability. Recuperado de MDN Web Docs. https://developer.mozilla.org/

MDPI. (2022). The Trends and Content of Research Related to the Sustainable Development Goals. Recuperado de MDPI. https://www.mdpi.com/

OpenReplay. (2023). Green Web Design: Bridging Tech and Sustainability. Recuperado de https://openreplay.com

PLOS ONE. (2023). Impact of the Sustainable Development Goals on the academic research agenda. Recuperado de PLOS ONE. https://journals.plos.org/plosone/

Sustainable Web Design. (2023). Guidelines for Web Sustainability. Recuperado de Sustainable Web Design. https://www.sustainablewebdesign.org/

Smashing Magazine. (2023). Sustainable Web Development Strategies Within An Organization. Recuperado de Smashing Magazine. https://www.smashingmagazine.com/

Verizon. (2021). 2021 Data Breach Investigations Report. Recuperado de https://www.verizon.com/business/resources/reports/dbir/

Website Carbon Calculator. (2023). What's your site's carbon footprint? Recuperado de Website Carbon. https://www.websitecarbon.com/

World Economic Forum. (2021). What are the challenges in making tech more sustainable? Recuperado de World Economic Forum. https://www.weforum.org/

W3C. (2023a). New Web Sustainability Guidelines Report. Recuperado de W3C. https://www.w3.org/

W3C. (2023b). Introducing Web Sustainability Guidelines. Recuperado de W3C. https://www.w3.org/

Xu, M., & Buyya, R. (2020). Managing renewable energy and carbon footprint in multi-cloud computing environments. *Journal of Parallel and Distributed Computing, 135*, 191-202. https://doi.org/10.1016/j.jpdc.2019.09.015