Proyecto:

Comparador.

Por:

Isaac Cepas.
Daniel Canales.
Gonzalo Pérez.

INSTITUTO NEBRIJA | Formación Profesional

Página de Agradecimientos

Agradecimientos

Quisiéramos expresar nuestro más sincero agradecimiento a todas las personas y entidades que han hecho posible la realización de este proyecto. Sin su apoyo, guía y colaboración, no habríamos logrado llevar a cabo PZComparator.

A nuestros profesores.

En primer lugar, queremos agradecer a nuestros profesores por su dedicación y paciencia. Su orientación, conocimientos y constantes motivaciones nos han permitido superar numerosos retos y aprender valiosas lecciones. Gracias por compartir su experiencia y por estar siempre disponibles para responder nuestras dudas y guiar nuestros pasos.

A nuestros padres y familias.

A nuestros padres y familias, queremos expresar nuestro profundo agradecimiento por su apoyo incondicional. Han estado a nuestro lado en todo momento, brindándonos el ánimo necesario para seguir adelante incluso en los momentos más difíciles. Su confianza en nuestras capacidades nos ha impulsado a esforzarnos y dar lo mejor de nosotros mismos.

A nuestros compañeros.

A nuestros compañeros de clase, gracias por la camaradería y el trabajo en equipo. Juntos hemos enfrentado desafíos, compartido éxitos y aprendido unos de otros. Su colaboración y amistad han sido esenciales para el desarrollo de este proyecto y para crear un ambiente de aprendizaje enriquecedor y motivador.

Al Centro Universitario Nebrija de Formación Profesional.

Finalmente, queremos agradecer al Centro Universitario Nebrija por brindarnos la oportunidad de crecer académica y profesionalmente. La calidad de la enseñanza y los recursos disponibles nos han proporcionado una base sólida para desarrollar nuestras habilidades y conocimientos. Agradecemos especialmente a los equipos administrativos y técnicos por su apoyo logístico y por crear un entorno propicio para el aprendizaje y la innovación.

Propósito de la Memoria

El propósito de esta memoria es proporcionar una documentación exhaustiva y detallada sobre PZComparator, una aplicación web diseñada para comparar ordenadores portátiles de manera sencilla e intuitiva. Esta memoria busca informar sobre todos los aspectos relevantes relacionados con la aplicación, abarcando desde su concepción hasta su desarrollo y perspectivas futuras.

En particular, esta memoria tiene como objetivos:

1. ***Describir la Aplicación:*** Presentar una visión general de PZComparator, explicando su funcionalidad principal y cómo está diseñada para ayudar a los usuarios a comparar diferentes modelos de ordenadores portátiles de manera eficiente y comprensible.

2. ***Desarrollo del Proyecto:*** Documentar el proceso de desarrollo de la aplicación, detallando cada fase, desde la planificación inicial y el diseño hasta la implementación y las pruebas finales. Esto incluye una descripción de la arquitectura del sistema y el flujo de trabajo seguido para construir la aplicación.

3. ***Problemas y Soluciones:*** Analizar los desafíos y problemas encontrados durante el desarrollo de PZComparator, así como las soluciones implementadas para superarlos. Este apartado es crucial para entender las decisiones técnicas y estratégicas tomadas a lo largo del proyecto.

4. ***Tecnologías Utilizadas:*** Especificar las tecnologías empleadas en el desarrollo de la aplicación, incluyendo PHP para la lógica del servidor, JavaScript para la interactividad del cliente, HTML y CSS para la estructura y el diseño de la interfaz de usuario, y SQL para la gestión de la base de datos.

5. ***Funcionamiento de la Aplicación:*** Explicar detalladamente cómo funciona PZComparator, incluyendo el proceso de comparación de ordenadores, la generación de gráficos comparativos y la interacción del usuario con la aplicación.

6. ***Perspectivas de Futuro:*** Discutir las posibles mejoras y futuras direcciones para el desarrollo de PZComparator, basadas en la retroalimentación de los usuarios y las tendencias tecnológicas emergentes.

7. ***Biografía de los Participantes:*** Incluir una breve biografía de cada uno de los participantes en el proyecto, destacando sus roles, contribuciones y experiencias durante el desarrollo de PZComparator.

En resumen, esta memoria tiene como objetivo ofrecer una visión integral de PZComparator, proporcionando una comprensión profunda de su desarrollo, funcionamiento y potencial futuro, así como reconocer el esfuerzo y dedicación de todos los involucrados en su creación.

Índice

1. Introducción.

1.1. Presentación del Proyecto.

En la era digital actual, la adquisición de un ordenador portátil puede resultar una tarea compleja para muchos consumidores, especialmente aquellos que no poseen un conocimiento técnico profundo sobre hardware y especificaciones. La diversidad y abundancia de opciones disponibles en el mercado puede resultar abrumadora, llevando a decisiones de compra que no siempre satisfacen las necesidades específicas del usuario.

Con el objetivo de simplificar este proceso y hacerlo accesible para todos, nace el proyecto PZComparator. PZComparator es una página web innovadora diseñada para ofrecer una herramienta de comparación de ordenadores portátiles de manera sencilla e intuitiva. Este proyecto se centra en proporcionar una experiencia de usuario amigable, donde cualquier persona, independientemente de su nivel de conocimiento sobre ordenadores, pueda comparar diferentes modelos de portátiles y tomar decisiones informadas.

La página cuenta con un comparador avanzado que permite a los usuarios seleccionar varios ordenadores portátiles y comparar sus especificaciones detalladas. Lo que distingue a PZComparator es su capacidad para presentar esta información de forma clara y visualmente atractiva mediante gráficos comparativos. Estos gráficos permiten una rápida visualización de las diferencias clave entre los modelos seleccionados, facilitando la identificación de la opción más adecuada según las preferencias y necesidades del usuario.

1.2. Objetivos de la Aplicación.

El desarrollo de PZComparator responde a la necesidad creciente de herramientas digitales que ayuden a simplificar la toma de decisiones en un mundo cada vez más tecnológico. Este proyecto no solo busca ser una guía para la compra de portátiles, sino también un recurso educativo que mejore la comprensión de los usuarios sobre las especificaciones y el rendimiento de estos dispositivos. Los objetivos principales de PZComparator son:

1. ***Simplificar la Comparación de Portátiles:*** Facilitar la comparación de diferentes modelos de ordenadores portátiles de manera eficiente y accesible para todos los usuarios.
2. ***Mejorar la Toma de Decisiones:*** Proporcionar a los usuarios la información necesaria para tomar decisiones de compra informadas y adecuadas a sus necesidades específicas.
3. ***Educar a los Usuarios:*** Ayudar a los usuarios a comprender mejor las especificaciones técnicas y el rendimiento de los portátiles, promoviendo una mayor alfabetización tecnológica.
4. ***Ofrecer una Experiencia de Usuario Amigable:*** Desarrollar una interfaz intuitiva y visualmente atractiva que haga la navegación y el uso de la aplicación agradable y sencillo.

1.3. Estructura de la Memoria.

Esta memoria se ha estructurado para proporcionar una visión integral de PZComparator, abarcando todos los aspectos relevantes del proyecto. A continuación, se detalla la estructura de la memoria:

1. ***Introducción.***
    - 1.1. Presentación del Proyecto: Descripción general de PZComparator y su propósito.
    - 1.2. Objetivos de la Aplicación: Objetivos principales y metas del proyecto.
    - 1.3. Estructura de la Memoria: Descripción del contenido y organización del documento.
2. ***Descripción de PZComparator.***
    - 2.1. Funcionalidades Principales
    - 2.2. Público Objetivo
    - 2.3. Beneficios y Usos
3. ***Desarrollo del Proyecto.***
    - 3.1. Fases del Proyecto
        - 3.1.1. Planificación y Análisis de Requisitos
        - 3.1.2. Diseño de la Arquitectura
        - 3.1.3. Implementación
        - 3.1.4. Pruebas y Validación
        - 3.1.5. Despliegue y Mantenimiento
    - 3.2. Metodología de Desarrollo
        - 3.2.1. Enfoque Ágil
        - 3.2.2. Herramientas de Gestión de Proyecto
4. ***Problemas y Soluciones.***
    - 4.1. Desafíos Técnicos
        - 4.1.1. Integración de Tecnologías
        - 4.1.2. Optimización del Rendimiento
    - 4.2. Problemas Encontrados
        - 4.2.1. Errores Comunes y Resolución
        - 4.2.2. Gestión de Datos y Seguridad
    - 4.3. Soluciones Implementadas
        - 4.3.1. Estrategias de Resolución
        - 4.3.2. Innovaciones y Mejores Prácticas
5. ***Tecnologías Utilizadas.***
    - 5.1. Lenguajes de Programación
        - 5.1.1. PHP
        - 5.1.2. JavaScript
        - 5.1.3. HTML y CSS
    - 5.2. Base de Datos
        - 5.2.1. SQL
        - 5.2.2. Diseño del Esquema de Base de Datos
    - 5.3. Herramientas.

6. ***Funcionamiento de la Aplicación.***
    - 6.1. Flujo de Usuario
        - 6.1.1. Proceso de Comparación
        - 6.1.2. Interacción con la Gráfica Comparativa
    - 6.2. Módulos y Componentes
        - 6.2.1. Frontend
        - 6.2.2. Backend
    - 6.3. Ejemplos de Uso
        - 6.3.1. Casos Prácticos
        - 6.3.2. Demostraciones en Vivo
7. ***Perspectivas de Futuro.***
    - 7.1. Mejoras Propuestas
        - 7.1.1. Nuevas Funcionalidades
        - 7.1.2. Optimización y Escalabilidad
    - 7.2. Expansión del Proyecto
        - 7.2.1. Integración con Otros Servicios
        - 7.2.2. Adaptación a Nuevas Tecnologías
8. ***Biografía de los Participantes.***
    - 8.1. Perfil de los Desarrolladores
        - 8.1.1. Roles y Responsabilidades
        - 8.1.2. Contribuciones Individuales
    - 8.2. Experiencias y Aprendizajes
        - 8.2.1. Retos Personales
        - 8.2.2. Habilidades Adquiridas
9. ***Conclusiones.***
    - 9.1. Resumen del Proyecto
    - 9.2. Logros y Éxitos
    - 9.3. Reflexiones Finales
10. ***Referencias.***
    - 10.1. Bibliografía
    - 10.2. Recursos en Línea
    - 10.3. Documentación Técnica
11. ***Anexos.***
    - 11.1. Código Fuente Relevante
    - 11.2. Diagramas y Esquemas
    - 11.3. Documentación Adicional

2. Descripción de PZComparator.

2.1. Funcionalidades Principales.

PZComparator es una aplicación web diseñada para facilitar la comparación de ordenadores portátiles, centrándose en la potencia y calidad de sus componentes en relación al precio. Las funcionalidades principales de PZComparator incluyen:

- C***omparación de Portátiles:*** Permite a los usuarios seleccionar varios modelos de portátiles y comparar sus especificaciones técnicas detalladas, como procesador, memoria RAM, almacenamiento, tarjeta gráfica, duración de la batería, entre otros.
- ***Análisis de Potencia y Calidad:*** La herramienta evalúa y compara la potencia y calidad de los componentes de cada portátil, ayudando a los usuarios a entender mejor el rendimiento y la relación calidad-precio de cada opción.
- ***Inicio de Sesión y Registro:*** Los usuarios pueden registrarse e iniciar sesión en la plataforma para acceder a funciones adicionales. Una de estas funciones es la posibilidad de crear una lista de favoritos, donde pueden guardar los modelos de portátiles que les interesan para compararlos más tarde.
- ***Página de Contacto:*** Los usuarios pueden dejar comentarios en la página de contacto para que el administrador los vea. Además, tienen la opción de contactar al administrador directamente vía correo electrónico o teléfono para consultas más específicas o asistencia personalizada.

2.2. Público Objetivo.

El público objetivo de PZComparator abarca un amplio espectro de usuarios que desean comprar un ordenador portátil, incluyendo:

- ***Usuarios con Conocimientos Limitados de Hardware:*** Principalmente, PZComparator está dirigido a aquellas personas que no tienen un conocimiento técnico profundo sobre hardware y especificaciones de portátiles. La herramienta está diseñada para ser intuitiva y fácil de usar, permitiendo a estos usuarios tomar decisiones informadas sin necesidad de entender detalles técnicos complejos.
- ***Usuarios con Conocimientos Técnicos:*** Aunque la herramienta está especialmente diseñada para usuarios con conocimientos limitados, también es útil para aquellos que tienen un conocimiento más avanzado de hardware. Estos usuarios pueden beneficiarse de la capacidad de PZComparator para proporcionar comparaciones detalladas y gráficos visualmente atractivos que simplifican la evaluación de las diferencias entre modelos de portátiles.

2.3. Beneficios y Usos.

PZComparator ofrece varios beneficios y usos para sus usuarios:

- ***Facilita la Toma de Decisiones:*** Al proporcionar comparaciones detalladas y visualmente claras de diferentes modelos de portátiles, PZComparator ayuda a los usuarios a identificar rápidamente la opción más adecuada según sus necesidades y preferencias.
- ***Ahorro de Tiempo:*** La herramienta reduce significativamente el tiempo que los usuarios necesitan para investigar y comparar distintos modelos de portátiles, centralizando toda la información relevante en un solo lugar.
- ***Educación del Usuario:*** PZComparator no solo ayuda en la decisión de compra, sino que también educa a los usuarios sobre las especificaciones técnicas y el rendimiento de los portátiles, mejorando su comprensión de estos dispositivos.
- ***Interacción y Personalización:*** Con funciones como la creación de listas de favoritos y la posibilidad de dejar comentarios o contactar directamente con el administrador, PZComparator ofrece una experiencia de usuario personalizada y participativa.
- ***Asistencia y Soporte:*** La página de contacto proporciona una vía directa para que los usuarios obtengan asistencia adicional o hagan consultas específicas, mejorando el soporte y la satisfacción del usuario.

3. Desarrollo del Proyecto.

3.1. Fases del Proyecto.

3.1.1. Planificación y Análisis de Requisitos

Desde el inicio de la planificación de PZComparator, tuvimos un objetivo claro: desarrollar una aplicación web que fuera sencilla de ver y usar. Para lograr esto, nos enfocamos en crear una interfaz intuitiva y accesible que explicara claramente el propósito de la página desde el principio.

- ***Interfaz de Usuario Intuitiva:*** Diseñamos una interfaz que presenta de manera explícita el objetivo de la página en la pantalla principal. Incluimos un botón grande y central que actúa como ancla a la herramienta de comparación. Este diseño permite a los usuarios que ya están familiarizados con la página acceder rápidamente a la funcionalidad principal con un solo clic.
- ***Página de Registro:*** La página de registro fue diseñada para validar todos los campos en tiempo real, utilizando peticiones AJAX que interactúan con un script PHP en el servidor. Este script verifica la disponibilidad de los campos en la base de datos.
    - ***Validación en Tiempo Real:*** El correo electrónico y el nombre de usuario se validan en tiempo real para asegurar que no se repitan en la base de datos. Si están disponibles, se muestra un mensaje en verde; si no, en rojo.
    - ***Validación de Nombre y Apellido:*** Utilizamos expresiones regulares (regex) para validar los campos de nombre y apellido, permitiendo nombres compuestos pero sin permitir símbolos ni números.
    - ***Seguridad de la Contraseña:*** Las contraseñas se hashean utilizando bcrypt para mayor seguridad. Además, la aplicación verifica en tiempo real que las contraseñas coincidan cuando se ingresan dos veces.
- ***Proceso de Registro Exitoso:*** Si todos los campos pasan las validaciones y requisitos, se notifica al usuario del registro exitoso y se le redirige a la página de inicio (index).

3.1.2. Diseño de la Arquitectura.

Para asegurar que la aplicación fuera robusta y escalable, diseñamos una arquitectura clara y organizada:

- ***Arquitectura Modular:*** Implementamos una arquitectura Modular para separar los códigos y funcionalidades que funcionan de manera independiente y luego cohesionan haciendo una herramienta completa y funcional.
- ***Base de Datos Relacional:*** Utilizamos una base de datos relacional para gestionar de manera eficiente la información de los usuarios y los portátiles.

3.1.3. Implementación.

Durante la implementación, seguimos un enfoque iterativo y ágil, permitiendo ajustes y mejoras continuas basadas en pruebas y retroalimentación:

- ***Desarrollo Frontend:*** Utilizamos HTML, CSS y JavaScript para construir una interfaz de usuario atractiva y funcional.
- ***Desarrollo Backend:*** Implementamos la lógica del servidor en PHP y JS, asegurando una comunicación eficiente con la base de datos y el manejo seguro de las peticiones de los usuarios.
- ***Integración de AJAX:*** Las peticiones AJAX fueron integradas para mejorar la experiencia del usuario mediante validaciones en tiempo real y actualizaciones dinámicas sin necesidad de recargar la página.

3.1.4. Pruebas y Validación.

Realizamos pruebas exhaustivas para asegurar que la aplicación funcionara correctamente en diferentes escenarios:

- ***Pruebas Unitarias:*** Probamos individualmente cada componente y función para asegurar su correcto funcionamiento.
- ***Pruebas de Integración:*** Verificamos la interacción entre diferentes componentes del sistema para asegurar una integración fluida. Como diferenciación entre usuario normal y administrador, lo cual veremos más adelante.
- ***Pruebas de Usuario:*** Recogimos retroalimentación de usuarios reales para identificar y corregir posibles problemas de usabilidad. Obtuvimos un éxito rotundo en este aspecto.

3.1.5. Despliegue y Mantenimiento.

Finalmente, desplegamos PZComparator en un entorno de producción y establecimos procedimientos para su mantenimiento continuo:

- ***Uso en Lamp/Xampp:*** Primero lo desarrollamos usando este entorno de pruebas en dos sistemas un Ubuntu 24 y un Windows 11 muy distintos en muchos aspectos para asegurarnos de la compatibilidad entre equipos.
- ***Despliegue en Servidor Web:*** Configuramos el servidor web con Ubuntu Server 24 en una maquina Virtual con VirtualBox y conexión puente para alojar la aplicación y aseguramos su disponibilidad, acceso entre ordenadores y un buen rendimiento.

- ***Mantenimiento Regular:*** Establecimos un plan de mantenimiento para asegurar que la aplicación se mantuviera actualizada, segura y funcional. Utilizamos Git y GitHub para mantener vivo y actualizado el proyecto separando y organizando el trabajo y los códigos para no generarnos errores entre nosotros, aunque los hubo igualmente pero lo veremos más adelante.

3.2. Metodología de Desarrollo.

3.2.1. Enfoque Ágil.

Adoptamos una metodología ágil para gestionar el desarrollo del proyecto, permitiendo flexibilidad y adaptación a los cambios:

- ***Sprints:*** Dividimos el desarrollo en sprints cortos, cada uno con objetivos específicos y entregables claros.
- ***Reuniones Diarias:*** Realizamos reuniones diarias para coordinar esfuerzos, discutir avances, tomas de decisiones y resolver problemas rápidamente.

3.2.2. Herramientas de Gestión de Proyecto

Utilizamos diversas herramientas para gestionar y coordinar el desarrollo de PZComparator:

- ***Gestión de Tareas:*** Implementamos herramientas como Trello y Discord para organizar y priorizar las tareas del proyecto.
- ***Control de Versiones:*** Utilizamos Git y GitHub para el control de versiones, permitiendo una colaboración eficiente y el manejo seguro de cambios en el código fuente.
- ***Comunicación y Colaboración:*** Empleamos plataformas como Discord para mantener una comunicación constante y efectiva entre todos los miembros del equipo. Ademas de las reuniones presenciales en clase.

4. Problemas y Soluciones

4.1. Desafíos Técnicos

4.1.1. Integración de Tecnologías

Durante el desarrollo de PZComparator, uno de los desafíos más significativos fue la integración de tecnologías, específicamente la implementación de peticiones AJAX para la validación en tiempo real. Al principio, tuvimos dificultades para validar adecuadamente los datos debido a la falta de comprensión de las funciones asincrónicas. Sin embargo, a medida que avanzamos en el proyecto, nos dedicamos a aprender y aplicar correctamente el uso de funciones asincrónicas en JavaScript. Esto nos permitió realizar validaciones de manera efectiva y mejorar la experiencia del usuario.

4.1.2. Optimización del Rendimiento

Otro desafío técnico importante fue la optimización del rendimiento de la aplicación. La integración de PHP en el código resultó complicada, especialmente en lo que respecta al manejo de conexiones a la base de datos. Inicialmente, usamos mysqli, pero enfrentamos varios problemas de conexión y

repetición de código. Decidimos entonces aprender a utilizar PDO (PHP Data Objects) para manejar las conexiones de manera más eficiente. PDO no solo resolvió nuestros problemas de conexión, sino que también nos permitió escribir un código más limpio y seguro mediante el uso de clases de conexión y consultas preparadas.

4.2. Problemas Encontrados

4.2.1. Errores Comunes y Resolución

A lo largo del desarrollo, nos enfrentamos a numerosos errores comunes relacionados con SQL, phpMyAdmin, MySQL, y la gestión de usuarios y contraseñas. Estos errores incluían problemas de conexión, manejo incorrecto de los datos de entrada y dificultades con la gestión de sesiones de usuario.

Para abordar estos problemas, implementamos extensivas líneas de código para mostrar mensajes de error detallados y empleamos bloques try-catch para gestionar las excepciones. Este enfoque nos permitió identificar y solucionar los errores de manera más eficiente, fortaleciendo la robustez de la aplicación.

4.2.2. Gestión de Datos y Seguridad

La seguridad de los datos fue otra área crítica donde encontramos problemas. Asegurarnos de que las contraseñas de los usuarios se almacenaran de manera segura fue esencial. Para esto, aprendimos a utilizar bcrypt para encriptar las contraseñas antes de almacenarlas en la base de datos. Además,

implementamos validaciones estrictas utilizando expresiones regulares para nombres y apellidos, y verificaciones en tiempo real para evitar duplicados en el registro de correos electrónicos y nombres de usuario.

4.3. Soluciones Implementadas

4.3.1. Estrategias de Resolución

Para superar los desafíos técnicos y de conocimiento, adoptamos varias estrategias de resolución:

- ***Aprendizaje Continuo:*** Nos comprometimos a aprender continuamente sobre las tecnologías que estábamos utilizando. Esto incluyó estudiar la documentación oficial de PHP, JavaScript, AJAX y PDO, así como realizar cursos y tutoriales en línea.
- ***Desarrollo Iterativo:*** Implementamos un enfoque iterativo en el desarrollo, permitiéndonos probar nuevas soluciones rápidamente y ajustarlas según fuera necesario.
- ***Colaboración en Equipo:*** Fomentamos la colaboración constante entre los miembros del equipo, compartiendo conocimientos y soluciones a problemas comunes. Esto nos ayudó a resolver problemas más rápido y de manera más efectiva.

4.3.2. Innovaciones y Mejores Prácticas

A lo largo del proyecto, adoptamos varias innovaciones y mejores prácticas que mejoraron significativamente la calidad de PZComparator:

- ***Uso de PDO:*** La transición de mysqli a PDO para las conexiones a la base de datos no solo resolvió problemas de conexión, sino que también nos permitió escribir un código más seguro y eficiente.
- ***Validación en Tiempo Real con AJAX:*** La implementación de validaciones en tiempo real mejoró la experiencia del usuario al proporcionar retroalimentación inmediata sobre la disponibilidad de los nombres de usuario y correos electrónicos.
- ***Seguridad de Contraseñas:*** La utilización de bcrypt para encriptar contraseñas mejoró significativamente la seguridad de los datos de los usuarios.
- ***Gestión de Errores:*** La implementación de bloques try-catch y mensajes de error detallados nos permitió identificar y solucionar problemas de manera más efectiva, mejorando la estabilidad y robustez de la aplicación.

5. Tecnologías Utilizadas.

5.1. Lenguajes de Programación.

5.1.1. PHP.

Durante el desarrollo de PZComparator, PHP ha sido el lenguaje predominante. Descubrimos la versatilidad de PHP, lo que facilitó la integración con otras tecnologías. Sirvió como un puente eficaz entre los diferentes lenguajes utilizados en el proyecto. Con PHP, gestionamos la conexión con la base de datos, actualizamos el estado de la página según la sesión del usuario, procesamos formularios y llevamos a cabo validaciones de datos. Además, la capacidad de PHP para generar HTML dinámicamente simplificó enormemente la creación y gestión de páginas web.

5.1.2. JavaScript.

JavaScript desempeñó un papel fundamental en la funcionalidad y la experiencia del usuario en PZComparator. Utilizamos JavaScript para implementar la lógica del comparador de portátiles, gestionar los datos en formato JSON que contenían información sobre los portátiles, realizar peticiones AJAX para la validación de campos del formulario y actualizar los datos en tiempo real. La modularidad y flexibilidad de JavaScript nos permitieron desarrollar una aplicación interactiva y dinámica.

5.1.3. HTML y CSS.

HTML y CSS fueron los lenguajes utilizados para definir la estructura y el diseño de las páginas web de PZComparator. Creamos un diseño responsive y simple utilizando HTML y CSS, asegurándonos de que la aplicación fuera legible y agradable en diferentes dispositivos y tamaños de pantalla. Utilizamos CSS para modificar el diseño según el tamaño de la pantalla, garantizando una experiencia de usuario consistente en dispositivos móviles y de escritorio.

Uso de las Tecnologías.

Utilizamos PHP principalmente para la gestión de datos y la lógica del servidor, mientras que JavaScript se utilizó para la interactividad del cliente y la manipulación dinámica de la interfaz de usuario. HTML y CSS proporcionaron la estructura y el diseño visual de la aplicación.

Además, implementamos un iframe para el comparador de portátiles, que funcionaba principalmente con HTML y JavaScript. Diseñamos una interfaz rudimentaria pero efectiva, asegurándonos de que fuera responsive y se adaptara a diferentes tamaños de pantalla. PHP facilitó la conexión entre documentos y la base de datos, así como el procesamiento de formularios y validaciones.

En resumen, PHP sirvió como el lenguaje principal para la gestión del backend y la conexión con la base de datos, mientras que JavaScript se encargó de la funcionalidad del frontend, incluida la interactividad y las actualizaciones en tiempo real. HTML y CSS se utilizaron para la estructura y el diseño visual de la aplicación.

5.2. Base de Datos.

5.2.1. SQL.

SQL (Structured Query Language) ha sido fundamental en el desarrollo de PZComparator para la creación y manipulación de la base de datos. Utilizamos SQL para diseñar y modificar la estructura de la base de datos, así como para realizar consultas para recuperar, insertar, actualizar y eliminar datos. Tanto en entornos como phpMyAdmin como en la terminal de Linux, SQL nos proporcionó las herramientas necesarias para gestionar eficientemente la base de datos del proyecto.

5.2.2. Diseño del Esquema de Base de Datos.

El esquema de la base de datos de PZComparator se diseñó cuidadosamente para garantizar un almacenamiento eficiente y seguro de los datos. Aquí está el diseño básico del esquema de la base de datos:

```
CREATE DATABASE IF NOT EXISTS Comparador;

USE Comparador;


CREATE TABLE IF NOT EXISTS usuarios (
    id INT AUTO_INCREMENT PRIMARY KEY,


    nombre VARCHAR(50) NOT NULL,
    apellidos VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL,
    nombreUsuario VARCHAR(50) NOT NULL,
    contrasenia VARCHAR(100) NOT NULL,
    fechaRegistro DATETIME DEFAULT CURRENT_TIMESTAMP
);


CREATE TABLE IF NOT EXISTS favoritos (
```

```
    id INT AUTO_INCREMENT PRIMARY KEY,
    usuario_id INT NOT NULL,
    nombreOrdenador VARCHAR(100) NOT NULL,
    fechaAgregado DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (usuario_id) REFERENCES usuarios(id) ON DELETE CASCADE
);

CREATE TABLE comentarios (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    sugerencia TEXT NOT NULL,
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

En el diseño de la tabla de usuarios, incluimos campos como nombre, apellidos, correo electrónico, nombre de usuario, contraseña y fecha de registro para almacenar información relevante sobre los usuarios registrados en la plataforma. La tabla de favoritos se utilizó para almacenar los portátiles favoritos de cada usuario, con una referencia a la tabla de usuarios para establecer una relación entre ellas. Por último, la tabla de comentarios se diseñó para permitir a los usuarios enviar comentarios y sugerencias, con campos para el nombre, correo electrónico, sugerencia y fecha de envío del comentario.

Decidimos agregar campos de fecha y hora en algunas tablas para realizar un seguimiento del momento en que se realizan ciertas acciones, como el registro de usuarios, la agregación de portátiles a favoritos y el envío de comentarios. Esta información puede ser valiosa para realizar análisis y estudios sobre el comportamiento de los usuarios en la plataforma.

5.3. Herramientas Utilizadas

Durante el desarrollo de PZComparator, se emplearon diversas herramientas para la programación, la gestión de la base de datos y la administración del servidor. Estas herramientas incluyeron:

- HTML para la estructura básica de las páginas web.
- CSS para el diseño y la presentación visual.
- JavaScript para la interactividad del cliente y la manipulación dinámica de la interfaz de usuario.
- PHP para la lógica del servidor y la conexión con la base de datos.
- SQL para la gestión y consulta de la base de datos.
- phpMyAdmin como interfaz gráfica para administrar la base de datos MySQL.
- Visual Studio Code (VSCode) como el entorno de desarrollo integrado (IDE) principal.
- LAMP (Linux, Apache, MySQL, PHP) como pila de tecnología para el desarrollo y la implementación en el servidor.
- Ubuntu Server como sistema operativo de servidor para alojar la aplicación web.

Estas herramientas proporcionaron un conjunto sólido de recursos para el desarrollo y despliegue eficientes de PZComparator, permitiendo la creación de una aplicación web funcional y robusta.

5.4. Ubuntu Server.

A continuación dejo un mini tutorial sobre como lo hice, puede que me deje algun matiz pero en esencia seria algo así:

Requisitos Previos

- VirtualBox instalado en tu ordenador.
- Imagen ISO de Ubuntu Server.
- Conexión a Internet.
- Página web preparada para el despliegue (HTML, CSS, JS, PHP, etc.).

Paso 1: Crear y Configurar la Máquina Virtual

1. Crear la VM en VirtualBox:
    - Abre VirtualBox y haz clic en Nueva.
    - Asigna un nombre a la VM, elige Linux como tipo y Ubuntu (64-bit) como versión.
    - Asigna al menos 2GB de RAM y 20GB de almacenamiento dinámico.
2. Configurar la Red de la VM:
    - Selecciona la VM creada y haz clic en Configuración.
    - Ve a la sección Red, habilita el Adaptador 1 y selecciona Adaptador Puente. Esto permite que la VM use la misma red que tu ordenador anfitrión.

Paso 2: Instalar Ubuntu Server

1. Iniciar la VM:
    - Selecciona la VM y haz clic en Iniciar.

- Selecciona la imagen ISO de Ubuntu Server cuando se te solicite.

2. Seguir el Proceso de Instalación:
    - Sigue las instrucciones en pantalla para instalar Ubuntu Server.
    - Configura el nombre del servidor, el nombre de usuario y la contraseña.

Paso 3: Configurar el Servidor Web

1. Instalar Apache, MySQL y PHP (LAMP):
    - Inicia sesión en tu servidor.
    - Actualiza los repositorios:

      sudo apt update

      sudo apt upgrade

    - Instalar apache:

      sudo apt install apache2

    - Instala MySQL:

      sudo apt install mysql-server

    - Instala PHP:

      sudo apt install php libapache2-mod-php php-mysql

      Verificar la Instalación de Apache:

    - Abre un navegador (En caso de hacerlo sin interfaz grafica recomiendo “lynx” como navegador, se puede instalar con ‘sudo apt install lynx’ y se ejecuta en terminal con ‘lynx “direccion ip o ruta web”’) en tu ordenador anfitrión e ingresa la dirección IP de tu VM (puedes encontrarla usando ifconfig o ip a en tu VM).
    - Deberías ver la página predeterminada de Apache.

Paso 4: Desplegar tu Página Web

1. Transferir Archivos a la VM:
    - Usa scp o un cliente FTP para transferir tus archivos al directorio web de Apache en la VM (/var/www/html/).
2. Configurar Permisos:
    - Asegúrate de que los archivos tienen los permisos adecuados:

      sudo chown -R www-data:www-data /var/www/html/
      sudo chmod -R 755 /var/www/html/

Probar tu Página Web:

- En un navegador, ingresa la dirección IP de tu VM. Deberías ver tu página web desplegada.

Paso 5: Acceder desde Otros Dispositivos en la Red

1. Encontrar la Dirección IP de la VM:
    - Usa el comando ifconfig o ip addr en tu VM para encontrar su dirección IP.
2. Conectar desde Otros Dispositivos:
    - En cualquier dispositivo conectado a la misma red WiFi, abre un navegador e ingresa la dirección IP de tu VM. Deberías poder acceder a tu página web.

Paso 6: Configuración Adicional (Opcional)

1. Habilitar Firewall (ufw):
    - Instala y habilita ufw para mayor seguridad:

      sudo apt install ufw
      sudo ufw allow 'Apache Full'
      sudo ufw enable

    - 2.Configuración de DNS Dinámico (Opcional):
    - Si deseas un nombre de dominio en lugar de una dirección IP, considera usar un servicio de DNS dinámico.

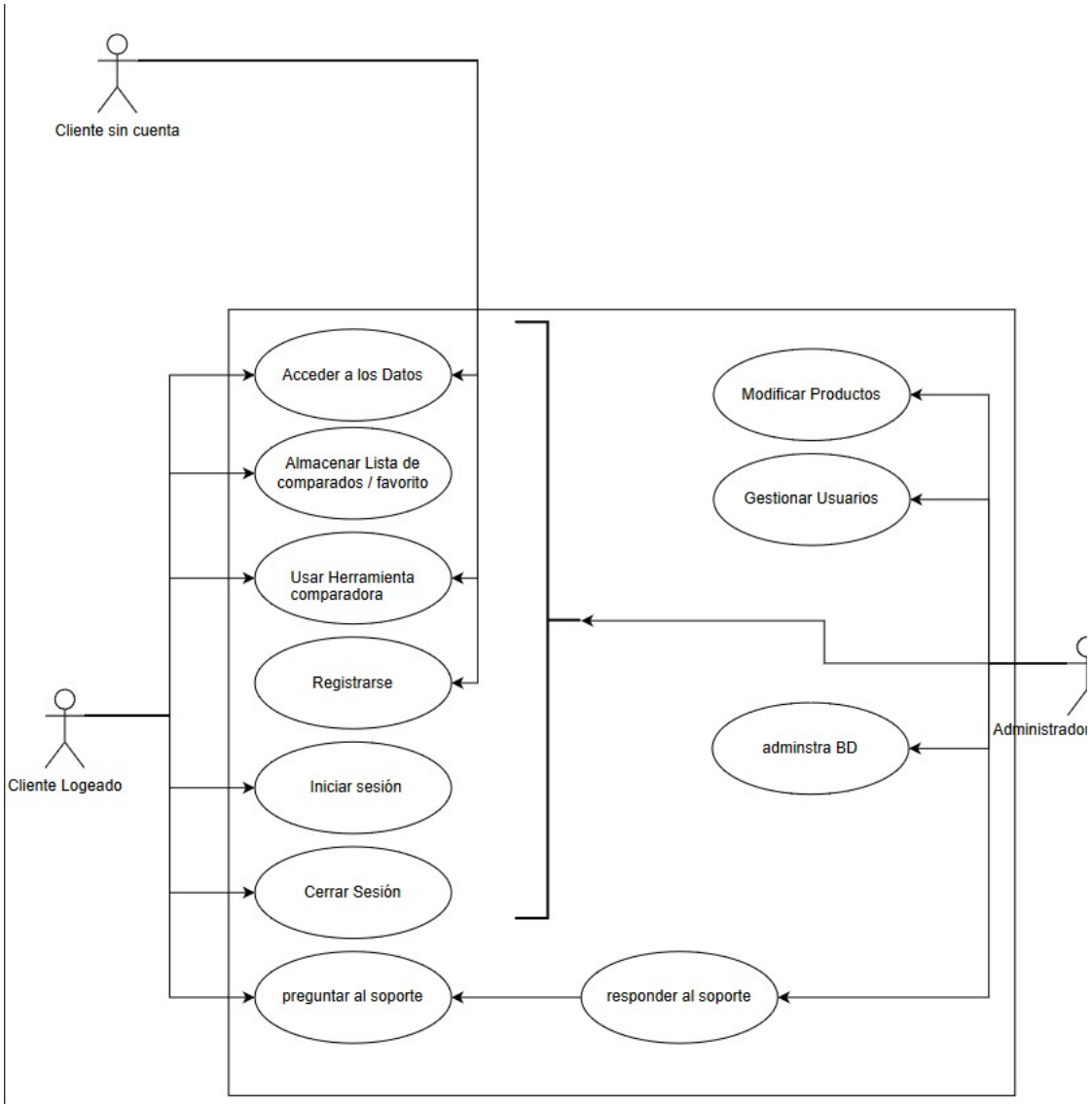
Siguiendo estos pasos, deberías tener una página web funcionando en una máquina virtual con Ubuntu Server, accesible desde cualquier dispositivo en la misma red WiFi.

6. Funcionamiento de la Aplicación

6.1. Flujo de Usuario

El flujo de usuario en PZComparator permite a los usuarios comparar portátiles y utilizar la herramienta de manera general. Aquí se detalla cómo los usuarios interactúan con la aplicación:

Diagrama de Casos de uso:



Nombre: registrarse
Descripción: Los usuarios deben registrarse antes de iniciar sesion
Actores: Usuario
Precondiciones: Ninguna
Curso normal del caso de uso: 1. Pulsar el enlace de inicio de sesion para acceder a la página. 2. Pulsar el botón de registro. 3. Rellenar el formulario con los datos del usuario 4. envía el formulario y te devuleve a la página index.

Nombre: Iniciar sesión
Descripción: Los usuarios registrados deben iniciar sesión para poder guardar ordenadores como favoritos y acceder a su página personal de favoritos
Actores: Usuario
Precondiciones: Registrado
Curso normal del caso de uso: 1. Pulsar el enlace de inicio de sesion para acceder a la página. 2. Introducir el correo elctronico y contraseña de la cuenta. 3. Envía el formulario y te devuleve a la página index.

Nombre: Herramienta comparadora
Descripción: Los usuarios pueden seleccionar 2 o más ordenadores de la lista para ver información más detallada de ellos y comparar sus caracteristicas.
Actores: Usuario
Precondiciones: Ninguna
Curso normal del caso de uso: 1. En el indice, seleccionar 2 o más ordenadores del buscador de ordenadores 2. Pulsar el botón de compara 3. Te manda a una página nueva con la informacíon detallada de los ordenadores seleccionados. 4. Si se han selecionando 2 o más ordenadores, se crea una grafica de radar con las caracteristicas. Si son menos de 2, en vez del grafico sale un mensaje pidiendo que selecciones 2. 5. Pulsando el botón de imprimir, te lleva a la edición de impresion del navegador.

6.1.1. Proceso de Comparación

El proceso de comparación de portátiles es accesible tanto para usuarios registrados como para aquellos que visitan el sitio sin iniciar sesión. Se describe a continuación:

1. ***Inicio de Comparación:*** Los usuarios pueden acceder a la herramienta de comparación desde la página principal de la aplicación. Aquí, pueden explorar la lista de portátiles disponibles y seleccionar los modelos que desean comparar.
2. ***Comparación de Portátiles:*** Una vez seleccionados los portátiles, se muestra una interfaz donde se comparan las especificaciones de cada uno y la gráfica. Esto permite a los usuarios visualizar y analizar las diferencias entre los modelos seleccionados.
3. ***Interacción con la Gráfica Comparativa:*** Los usuarios pueden interactuar con la gráfica comparativa para explorar más a fondo las diferencias entre los portátiles. Pueden utilizar funciones como seleccionar y deseleccionar para obtener una visualización más cómoda y simple.

6.1.2. Funcionalidades Exclusivas para Usuarios Registrados

Para acceder a ciertas funcionalidades adicionales, como agregar portátiles a la lista de favoritos los usuarios deben registrarse e iniciar sesión. Estas funcionalidades se detallan a continuación:

Lista de Favoritos: Después de iniciar sesión, los usuarios pueden agregar portátiles a su lista de favoritos para guardar los modelos que les interesan. Esto les permite acceder y comparar fácilmente estos portátiles en el futuro desde cualquier dispositivo.

7. Perspectivas de Futuro.

7.1. Mejoras Propuestas

7.1.1. Nuevas Funcionalidades

En el futuro, planeamos realizar varias mejoras significativas en PZComparator para ampliar su funcionalidad y usabilidad. Algunas de las nuevas funcionalidades propuestas incluyen:

- ***Gestión de Ordenadores a través de la Base de Datos:*** Actualmente, los ordenadores se cargan desde un archivo JSON. En el futuro, los añadiremos y gestionaremos directamente a través de la base de datos, lo que permitirá una actualización más fácil y eficiente de los modelos disponibles.
- ***Interfaz de Administración:*** Se implementará una interfaz para que los administradores puedan añadir, eliminar y modificar los datos de los ordenadores en la base de datos. Esta herramienta facilitará la gestión de los modelos y sus especificaciones sin necesidad de modificar el código fuente manualmente.
- ***Sistema de Notificaciones:*** Los usuarios registrados podrán optar por recibir notificaciones sobre nuevas comparaciones, actualizaciones de precios y nuevos modelos de portátiles añadidos a la base de datos.

7.1.2. Optimización y Escalabilidad

Para garantizar que PZComparator pueda manejar un creciente número de usuarios y datos, se realizarán las siguientes mejoras en términos de optimización y escalabilidad:

- Optimización del Rendimiento: Mejorar el rendimiento de la aplicación optimizando las consultas SQL y reduciendo los tiempos de carga. Esto incluye el uso de índices en la base de datos y la optimización del código PHP.
- Escalabilidad del Sistema: Diseñar la arquitectura del sistema para ser más escalable, permitiendo que la aplicación maneje un mayor volumen de tráfico y datos sin degradar el rendimiento. Esto podría incluir la implementación de técnicas de caché y el uso de servicios en la nube para la infraestructura.

7.2. Expansión del Proyecto

7.2.1. Integración con Otros Servicios

Para ampliar las capacidades de PZComparator, se planea la integración con otros servicios y API externos:

- ***API de Precios de Venta al Público:*** Implementar una API que recupere información en tiempo real sobre los precios de venta al público de los distintos productos. Esto permitirá a los usuarios ver precios actualizados y comparar opciones basadas no solo en especificaciones técnicas sino también en costos.
- ***Integración con Tiendas en Línea:*** Colaborar con tiendas en línea para mostrar enlaces directos a las páginas de compra de los portátiles comparados. Esto proporcionará una experiencia de compra más fluida para los usuarios.

7.2.2. Adaptación a Nuevas Tecnologías

Para mantenerse relevante y competitiva, PZComparator deberá adaptarse a las nuevas tecnologías emergentes:

- ***Migración a Frameworks Modernos:*** Considerar la migración a frameworks modernos de frontend y backend para mejorar la eficiencia del desarrollo y la experiencia del usuario. Por ejemplo, utilizar frameworks como React o Vue.js para el frontend y Laravel o Symfony para el backend.
- ***Inteligencia Artificial y Machine Learning:*** Explorar el uso de inteligencia artificial y machine learning para ofrecer recomendaciones personalizadas basadas en las preferencias y el historial de búsqueda de los usuarios. Esto puede mejorar la precisión de las sugerencias y proporcionar una experiencia más personalizada.

Estas perspectivas de futuro para PZComparator no solo ampliarán la funcionalidad de la aplicación, sino que también mejorarán significativamente la experiencia del usuario, asegurando que se mantenga relevante y útil en un mercado en constante evolución.

8. Biografía de los Participantes

8.1. Perfil de los Desarrolladores

8.1.1. Roles y Responsabilidades

Isaac Cepas:

- Rol: Desarrollador Backend y Administrador del Sistema
- Responsabilidades:
    - Desarrollo del backend
    - Gestión de la base de datos
    - Implementación de las funcionalidades de registro e inicio de sesión (tanto backend como frontend)
    - Creación de la página de administración para gestionar usuarios
    - Configuración del servidor Ubuntu Server con Apache 2, MySQL y PHP 8.3

Daniel Canales:

- Rol: Desarrollador de la Herramienta de Comparación
- Responsabilidades:
    - Desarrollo de la parte del comparador
    - Implementación de la herramienta (buscador, filtros, visualización de datos, etc.)
    - Normalización de los datos para su correcta visualización en las gráficas
    - Funcionalidad de añadir a favoritos en la base de datos

Gonzalo:

- Rol: Desarrollador de Soporte y Asistente General
- Responsabilidades:
    - Soporte en el desarrollo de diversas partes del proyecto
    - Ayuda en la implementación de funcionalidades adicionales
    - Contribución general al desarrollo del proyecto, asegurando que todas las partes funcionen correctamente

8.1.2. Contribuciones Individuales

Isaac Cepas:

- Diseño y configuración del servidor
- Implementación de la lógica de negocio del backend
- Desarrollo de la funcionalidad de registro e inicio de sesión
- Creación de la interfaz de administración

Daniel Canales:

- Desarrollo del comparador de portátiles
- Implementación de los filtros y visualización de datos en la herramienta
- Gestión de la funcionalidad de favoritos

Gonzalo:

- Soporte en el desarrollo de diversas funcionalidades
- Asistencia en la resolución de problemas y optimización del código
- Aseguramiento de la integración fluida de todas las partes del proyecto

8.2. Experiencias y Aprendizajes

8.2.1. Retos Personales

Isaac Cepas:

- Superar la curva de aprendizaje de PHP y su integración con MySQL
- Configurar y gestionar un servidor Ubuntu Server desde cero
- Implementar medidas de seguridad, como el hashing de contraseñas

Daniel Canales:

- Normalizar y visualizar datos de manera efectiva en gráficas
- Implementar funcionalidades avanzadas de búsqueda y filtrado
- Gestionar la integración de datos en tiempo real para la funcionalidad de favoritos

Gonzalo:

- Apoyar en diversas áreas del desarrollo, asegurando la cohesión del proyecto
- Resolver problemas técnicos en diferentes partes del código
- Colaborar en la implementación de funcionalidades adicionales y su optimización

8.2.2. Habilidades Adquiridas

Isaac Cepas:

- Administración de servidores Linux
- Desarrollo backend con PHP y MySQL
- Implementación de medidas de seguridad en aplicaciones web

Daniel Canales:

- Desarrollo de herramientas de comparación y visualización de datos
- Implementación de búsquedas y filtros avanzados
- Gestión de bases de datos y manipulación de datos.

Gonzalo Pérez:

- Resolución de problemas técnicos y optimización del código
- Gestión de GitHub e integración del código.
- Colaboración efectiva en un equipo de desarrollo
- Implementación de diversas funcionalidades en un proyecto complejo

Este proyecto ha sido una experiencia de aprendizaje significativa para cada uno de los participantes, permitiéndonos adquirir nuevas habilidades y enfrentar desafíos técnicos que han fortalecido nuestras capacidades como desarrolladores.

9. Conclusiones

9.1. Resumen del Proyecto

El proyecto PZComparator se originó con la intención de simplificar el proceso de selección y compra de ordenadores portátiles para usuarios con distintos niveles de conocimiento técnico. Reconociendo la diversidad y complejidad del mercado de portátiles, diseñamos una plataforma que permite a los usuarios comparar especificaciones de diferentes modelos de manera clara, visual y accesible.

Desde el inicio, el objetivo fue crear una herramienta que no solo presentara información técnica de forma comprensible, sino que también ofreciera una experiencia de usuario amigable y atractiva. PZComparator se estructura en torno a un comparador avanzado, el cual permite seleccionar múltiples modelos de portátiles y visualizar sus diferencias clave mediante gráficos comparativos. Estos gráficos facilitan la rápida identificación de la opción más adecuada según las preferencias y necesidades individuales de los usuarios.

El desarrollo del proyecto siguió una metodología ágil, comenzando con una planificación detallada y un análisis de requisitos. Este enfoque nos permitió identificar las funcionalidades esenciales y planificar la arquitectura del sistema de manera efectiva. La interfaz de usuario se diseñó para ser intuitiva, con una navegación sencilla y accesible, permitiendo que tanto usuarios expertos como inexpertos puedan utilizar la plataforma sin dificultades.

La implementación técnica abarcó el desarrollo de un backend robusto utilizando PHP, la gestión de la base de datos con MySQL, y la creación de una interfaz dinámica e interactiva con HTML, CSS y JavaScript. Un aspecto clave del proyecto fue la validación en tiempo real de los campos del formulario de registro mediante AJAX, garantizando una experiencia de usuario fluida y eficiente.

A lo largo del desarrollo, enfrentamos diversos desafíos técnicos, como la integración de tecnologías y la optimización del rendimiento. Estos desafíos se superaron mediante la adquisición de nuevos conocimientos y la implementación de soluciones innovadoras. La utilización de técnicas como el hashing de contraseñas con bcrypt y la validación de datos mediante expresiones regulares aseguró la seguridad y fiabilidad de la aplicación.

En términos de funcionalidad, PZComparator ofrece a los usuarios la capacidad de comparar portátiles y utilizar la herramienta en general sin necesidad de registrarse. Sin embargo, para funcionalidades avanzadas como la creación de una lista de favoritos, el registro e inicio de sesión son necesarios. Además, la opción de imprimir la página de comparación facilita a los usuarios llevarse la información consigo en formato físico.

El proyecto ha sido desplegado y configurado en un servidor Ubuntu Server, con un entorno LAMP, asegurando que la aplicación sea accesible y funcione de manera eficiente. Este proceso incluyó la configuración de Apache, MySQL y PHP, así como la gestión de conexiones y la seguridad de la base de datos.

En resumen, PZComparator se ha desarrollado con éxito como una herramienta integral y eficiente para la comparación de ordenadores portátiles, ofreciendo una experiencia de usuario amigable y

accesible, y solucionando las dificultades que los consumidores enfrentan en la toma de decisiones informadas en un mercado tecnológico complejo.

9.2. Logros y Éxitos

Durante el desarrollo de PZComparator, hemos alcanzado numerosos logros y éxitos tanto a nivel individual como grupal. Estos son algunos de los hitos más destacados:

- Daniel:
    - Herramienta de Comparación: Logró regular la gráfica y normalizar los datos para que la información se presentara de manera clara y visualmente atractiva.
    - Aspecto Visual: Mejoró el diseño visual de la aplicación, manteniendo una estética nostálgica de los años noventa.
    - Gestión de Datos: Implementó con éxito la funcionalidad para enviar datos entre la página del comparador, la lista de favoritos y la base de datos.
- Isaac:
    - Base de Datos: Creó tablas interrelacionadas en la base de datos, facilitando la gestión de usuarios y sus favoritos.
    - Herramientas MySQL: Descubrió y utilizó herramientas de MySQL para almacenar automáticamente la fecha y la hora, y manejó la base de datos desde la terminal de Linux, esencial para la implementación en el servidor.
    - Servidor Ubuntu: Configuró un servidor Ubuntu sin interfaz gráfica, superando la barrera de tener solo 8 meses de experiencia con Linux.
    - Seguridad de la Base de Datos: Implementó conexiones seguras a la base de datos utilizando hashing y prepare statement para prevenir inyecciones SQL.
    - Validaciones en Tiempo Real: Realizó validaciones en tiempo real para evitar la repetición de correos o nombres de usuario en la base de datos.
- Logros Grupales:
    - Uso de Git y GitHub: Aprendimos a utilizar Git y GitHub, herramientas esenciales que salvaron innumerables veces el desarrollo del proyecto, facilitando la colaboración y el control de versiones.

Estos logros reflejan no solo el esfuerzo y la dedicación de cada miembro del equipo, sino también la capacidad de trabajar juntos y aprender de los desafíos enfrentados durante el desarrollo del proyecto.

9.3. Reflexiones Finales

El desarrollo de PZComparator ha sido una experiencia enriquecedora y formativa para todo el equipo. Hemos superado numerosos desafíos técnicos y aprendido a utilizar nuevas tecnologías y metodologías de desarrollo. La colaboración y el trabajo en equipo fueron fundamentales para el éxito del proyecto, permitiéndonos combinar nuestras habilidades y conocimientos para crear una herramienta útil y funcional.

Reflexionando sobre el proceso, hemos reconocido la importancia de una planificación detallada y una comunicación efectiva dentro del equipo. La adopción de un enfoque ágil nos permitió adaptarnos a cambios y mejorar continuamente la aplicación en base a pruebas y retroalimentación.

Además, el proyecto nos ha proporcionado una comprensión más profunda de la importancia de la experiencia de usuario y la seguridad en el desarrollo de aplicaciones web. Las técnicas y conocimientos adquiridos durante este proyecto serán invaluables en nuestros futuros esfuerzos de desarrollo.

En conclusión, PZComparator no solo ha logrado cumplir con los objetivos iniciales, sino que también ha establecido una base sólida para futuras mejoras y expansiones. Estamos orgullosos del trabajo realizado y confiamos en que esta herramienta será de gran ayuda para los usuarios en la toma de decisiones informadas al comprar ordenadores portátiles.

10. Referencias.

10.1. Bibliografía.

- "HTML5, CSS3 y JavaScript" - Esteban Herrera. Anaya Multimedia. Una guía completa y práctica para aprender a desarrollar sitios web modernos y dinámicos utilizando las últimas tecnologías.
- "PHP y MySQL" - Larry Ullman. Anaya Multimedia. Un libro detallado que enseña cómo desarrollar aplicaciones web robustas utilizando PHP y MySQL.
- "SQL" - Chris Fehily. Anaya Multimedia. Un recurso esencial para aprender SQL desde lo básico hasta conceptos avanzados.

10.2. Recursos en Línea.

- W3Schools (www.w3schools.com) - Un recurso muy popular para aprender HTML, CSS, JavaScript y SQL con ejemplos y tutoriales interactivos.
- MDN Web Docs (developer.mozilla.org) - La documentación oficial de Mozilla para desarrolladores web, que abarca HTML, CSS, y JavaScript con ejemplos detallados.
- Stack Overflow (stackoverflow.com) - Una comunidad de preguntas y respuestas donde desarrolladores pueden encontrar soluciones a problemas técnicos y discutir sobre las mejores prácticas.
- PHP: Hypertext Preprocessor (www.php.net) - La documentación oficial de PHP, que proporciona una referencia completa de todas las funciones y características del lenguaje.
- MySQL Documentation (dev.mysql.com/doc) - La documentación oficial de MySQL, que incluye guías de usuario, tutoriales y referencias técnicas.

10.3. Documentación Técnica

- PHP Documentation (www.php.net/docs.php) - La documentación técnica oficial para PHP, que incluye guías, tutoriales y referencias de funciones.
- MySQL Reference Manual (dev.mysql.com/doc/refman) - La guía oficial de referencia para MySQL, que cubre la instalación, configuración y uso de MySQL.
- Bootstrap Documentation (getbootstrap.com/docs) - La documentación oficial de Bootstrap, el framework de frontend más popular para el desarrollo de interfaces web responsivas.
- Git Documentation (git-scm.com/doc) - La guía oficial y completa para Git, una herramienta de control de versiones esencial para la colaboración en proyectos de software.
- Visual Studio Code Documentation (code.visualstudio.com/docs) - La documentación de Visual Studio Code, el editor de código utilizado para el desarrollo de este proyecto, que incluye tutoriales y guías de configuración.
- Ubuntu Server Guide (ubuntu.com/server/docs) - La guía oficial para Ubuntu Server, que cubre la instalación, configuración y administración de servidores Ubuntu.
- DigitalOcean Tutorials (www.digitalocean.com/community/tutorials) - Una colección de tutoriales sobre diversas tecnologías, incluyendo configuraciones de Ubuntu Server y administración de servidores.

10.4. Herramientas de Inteligencia Artificial

- ChatGPT (openai.com/chatgpt) - Utilizado para la generación de texto, resolución de problemas y aprendizaje.
- Gemini (gemini.com) - Otra herramienta de inteligencia artificial empleada para consultas y soporte técnico durante el desarrollo del proyecto.

Estas referencias y herramientas han sido fundamentales para el desarrollo de PZComparator, proporcionando la información y recursos necesarios para superar los desafíos técnicos y construir una aplicación web funcional y segura.

11.Anexos

11.1 Código relevante.

11.1.2. Clases.

- ***Clase db.php:***

```php
<?php
20 references | 0 implementations
class DB {
    2 references
    private $host = 'localhost';
    2 references
    private $usuario = 'admin';
    2 references
    private $contrasenia = '1234';
    2 references
    private $baseDatos = 'comparador';

    18 references | 0 overrides
    public function connect() {
        $dsn = "mysql:host={$this->host};dbname={$this->baseDatos}";
        $conexion = new PDO($dsn, $this->usuario, $this->contrasenia);
        $conexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        return $conexion;
    }

    2 references | 0 overrides
    public function validarUsuario($email, $contrasena) {
        $conexion = $this->connect();
        $sql = "SELECT * FROM usuarios WHERE email = :email";
        $stmt = $conexion->prepare($sql);
        $stmt->bindParam(':email', $email);
        $stmt->execute();
        $usuario = $stmt->fetch(PDO::FETCH_ASSOC);

        if ($usuario && password_verify($contrasena, $usuario['contrasenia'])) {
            return $usuario;
        } else {
            return false;
        }
    }
2 references | 0 overrides
public function agregarAFavoritos($usuarioId, $nombreOrdenador) {
    $conexion = $this->connect();
    $sql = "INSERT INTO favoritos (usuario_id, nombreOrdenador) VALUES (
    $stmt = $conexion->prepare($sql);
    $stmt->bindParam(':usuario_id', $usuarioId);
    $stmt->bindParam(':nombreOrdenador', $nombreOrdenador);
    return $stmt->execute();
}
```

- ***usuario.php***

```php
<?php
6 references | 0 implementations
class Usuario {
    8 references
    private $id;
    6 references
    private $nombre;
    6 references
    private $apellidos;
    6 references
    private $email;
    6 references
    private $nombreUsuario;
    6 references
    private $contrasenia;

    6 references | 0 overrides
    public function __construct($id = null, $nombre, $apellidos, $email, $nombreUsuario, $contrasenia) {
        $this->id = $id;
        $this->nombre = $nombre;
        $this->apellidos = $apellidos;
        $this->email = $email;
        $this->nombreUsuario = $nombreUsuario;
        $this->contrasenia = $contrasenia;
    }

    // Getters
    0 references | 0 overrides
    public function getId() {
        return $this->id;
    }

    2 references | 0 overrides
    public function getNombre() {
        return $this->nombre;
    }

    2 references | 0 overrides
    public function getApellidos() {
        return $this->apellidos;
    }

    2 references | 0 overrides
    public function getEmail() {
```

```php
2 references | 0 overrides
public function getNombreUsuario() {
    return $this->nombreUsuario;
}

2 references | 0 overrides
public function getContrasenia() {
    return $this->contrasenia;
}

// Métodos para actualizar y borrar
2 references | 0 overrides
public function actualizar($conexion) {
    $sql = "UPDATE usuarios SET nombre = :nombre, apellidos = :apellidos, email = :email, nombreUsuario = :nombreUsuario, contrasenia = :contrasenia WHERE id
    $stmt = $conexion->prepare($sql);
    $stmt->bindParam(':nombre', $this->nombre);
    $stmt->bindParam(':apellidos', $this->apellidos);
    $stmt->bindParam(':email', $this->email);
    $stmt->bindParam(':nombreUsuario', $this->nombreUsuario);
    $stmt->bindParam(':contrasenia', $this->contrasenia);
    $stmt->bindParam(':id', $this->id);
    $stmt->execute();
}

2 references | 0 overrides
public function borrar($conexion) {
    $sql = "DELETE FROM usuarios WHERE id = :id";
    $stmt = $conexion->prepare($sql);
    $stmt->bindParam(':id', $this->id);
    $stmt->execute();
}
```

11.1.3 Validaciones en tiempo real.

```
function verificarDisponibilidadEmail() {
    return new Promise((resolve, reject) => {
        var email = document.getElementById('email').value;
        var emailDisponibilidad = document.getElementById('emailDisponibilidad');

        emailDisponibilidad.innerHTML = "";

        if (email.length === 0) {
            resolve(false);
            return;
        }

        var emailRegex = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9-]+\.[a-zA-Z]+$/;

        if (!emailRegex.test(email)) {
            emailDisponibilidad.innerHTML = "<span class='error'>Formato de correo inválido</span>";
            resolve(false);
            return;
        }

        var xhr = new XMLHttpRequest();
        xhr.open("POST", "verificar_disponibilidad.php", true);
        xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");

        xhr.onreadystatechange = function() {
            if (xhr.readyState == 4) {
                if (xhr.status == 200) {
                    try {
                        var respuesta = JSON.parse(xhr.responseText);
                        if (respuesta.disponible) {
                            emailDisponibilidad.innerHTML = "<span class='correcto'>Disponible</span>";
                            resolve(true);
                        } else {
                            emailDisponibilidad.innerHTML = "<span class='error'>No disponible</span>";
                            resolve(false);
                        }
                    } catch (e) {
                        emailDisponibilidad.innerHTML = "<span class='error'>Error en la respuesta del servidor</span>";
                        reject(e);
                    }
                } else {
                    emailDisponibilidad.innerHTML = "<span class='error'>Error en la solicitud</span>";
```

```
    return new Promise((resolve, reject) => {
        xhr.onreadystatechange = function() {
                        reject(new Error("Error en la solicitud"));
                    }
                }
            };

            xhr.send("valor=" + encodeURIComponent(email));
    });
}
```

```php
<?php

require_once 'db.php'; // Incluye la clase de conexión
error_reporting(E_ALL);
ini_set('display_errors', 1);

header('Content-Type: application/json'); // Asegúrate de que la respuesta es JSON

$response = array("disponible" => false);

try {
    if (isset($_POST['valor'])) {
        $email = $_POST['valor'];

        $db = new DB();
        $conexion = $db->connect();

        if ($conexion) {
            $sql = "SELECT * FROM usuarios WHERE email = :email";
            $stmt = $conexion->prepare($sql);
            $stmt->bindParam(':email', $email);
            $stmt->execute();
            $resultado = $stmt->fetch(PDO::FETCH_ASSOC);
            $response["disponible"] = ($resultado === false);
        } else {
            throw new Exception("Error al conectar con la base de datos.");
        }
    } else {
        throw new Exception("No se proporcionó ningún valor de correo electrónico en la solicitud.");
    }
} catch (Exception $e) {
    $response["error"] = $e->getMessage();
}

echo json_encode($response);
```

11.1.4 Herramienta del Comparador.

```javascript
function sacar_categoria() {
    let categoria = [];
    let categoria2 = [];

    // Llenar el array categoria con las marcas
    for(let i = 0; i < res.mejores_ordenadores.length; i++) {
        categoria.push(res.mejores_ordenadores[i].caracteristicas.diseño.marca_original);
    }

    // Ordenar el array categoria alfabéticamente
    categoria.sort();

    // Buscar y guardar elementos únicos en categoria2
    for(let i = 0; i < categoria.length; i++) {
        if (i === 0 || categoria[i] !== categoria[i - 1]) {
            categoria2.push(categoria[i]);
        }
    }

    // Llamar a la función de impresión con categoria2
    impr_categoria(categoria2);
}

function impr_categoria(categorias) {
    let select = '<select name="Categoria" id="categoria"><option value="all">Todo</option>';

    categorias.forEach(categoria => {
        select += '<option value="' + categoria + '">' + categoria + '</option>';
    });

    select += '</select>'
    category.innerHTML = select;
}
```

```
function capturarValor() {
    valorImput = document.getElementById('palabra').value.toUpperCase();
    console.log(valorImput);
    recorrerJson(valorImput);
}

function recorrerJson(valorImput = "") {
    arrayFiltrado = [];  // Vaciar el array antes de llenar con nuevos resultados
    let valor_categoria = document.getElementById('categoria').value;
    let p_min = document.getElementById('pre_min').value;
    let p_max = document.getElementById('pre_max').value;

    if (p_min === "") {
        p_min = 0;
    }
    if (p_max === "") {
        p_max = 999999;
    }

    console.log(p_min, p_max);

    for (let i = 0; i < res.mejores_ordenadores.length; i++) {
        if (valor_categoria === res.mejores_ordenadores[i].caracteristicas.diseño.marca_original || valor_categoria === "all") {
            if (res.mejores_ordenadores[i].descripcion.toUpperCase().includes(valorImput) ||
            res.mejores_ordenadores[i].nombre.toUpperCase().includes(valorImput) ||
            valorImput === "") {

                if (parseFloat(res.mejores_ordenadores[i].precio.replace("Desde ", "")) >= p_min && parseFloat(res.mejores_ordenadores[i].precio.replace("Desde ", "")) <= p_max) {
                    console.log("entró");
                    arrayFiltrado.push(res.mejores_ordenadores[i]);
                }
            }
        }
    }

    imprimirResultado(arrayFiltrado, valor_categoria);
}
```

```
    agregarEventosFavoritos();
}

function agregarEventosFavoritos() {
    for (let i = 0; i < arrayFiltrado.length; i++) {
        let boton = document.getElementById("fav" + i);
        if (boton) { // Verifica que el botón existe
            boton.addEventListener('click', function() {
                let valor = boton.value; // Obtener el valor del botón directamente
                console.log(valor);

                // Enviar el valor al archivo PHP
                fetch('dar_fav.php', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/x-www-form-urlencoded'
                    },
                    body: new URLSearchParams({
                        'valor': valor
                    })
                })
                .then(response => response.json())
                .then(data => {
                    console.log(data.resultado);
                    // Mostrar el resultado en el HTML
                    alert(data.mensaje);
                })
                .catch(error => console.error('Error:', error));
            });
        }
    }
}
```

```
function compValor() {
    let checkboxes = document.getElementsByClassName('comp_check');
    let valoresSeleccionados = [];
    let array_comp = [];

    for (let i = 0; i < checkboxes.length; i++) {
        if (checkboxes[i].checked) {
            valoresSeleccionados.push(checkboxes[i].value);
        }
    }
    console.log(valoresSeleccionados);
    for (let i = 0; i < valoresSeleccionados.length; i++) {
        for(let j = 0; j < arrayFiltrado.length; j++) {
            if (valoresSeleccionados[i] === arrayFiltrado[j].nombre) {
                array_comp.push(arrayFiltrado[j]);
            }
        }
    }

    console.log(array_comp);
    let urlDestino = "comparar.html?array=" + JSON.stringify(array_comp);
    window.open(urlDestino, '_blank');
}
```

```
function obtenerArrayDeURL() {
    // Obtener la cadena de consulta de la URL actual
    let queryString = window.location.search;

    // Obtener los parámetros de la cadena de consulta
    let parametros = new URLSearchParams(queryString);

    // Obtener el valor del parámetro 'array'
    let arrayString = parametros.get('array');

    // Convertir la cadena JSON de array de nuevo a un array JavaScript
    let array = JSON.parse(arrayString);

    return array;
}

// Obtener el array de la URL
let arrayObtenido = obtenerArrayDeURL();

// Obtener el elemento HTML donde mostrar el array
let comparacion = document.getElementById('comparacion');

// Construir una cadena con los elementos del array
let cadena = '';

let datos_comp = []

for (let i = 0; i < arrayObtenido.length; i++) {
```

```
    crear_graf(arrayObtenido, datos_comp)

    function normalizarDimensiones(datos, indicesDimensiones, maxValor) {
    const maxValues = [];

    // Encontrar los valores máximos de las dimensiones específicas
    indicesDimensiones.forEach(index => {
        maxValues[index] = Math.max(...datos.map(d => d[index]));
    });

    // Normalizar las dimensiones específicas
    return datos.map(row => row.map((val, i) => {
        if (indicesDimensiones.includes(i)) {
            return (val / maxValues[i]) * maxValor;
        } else {
            return val;
        }
    }));
}

function crear_graf(arrayObtenido, datos_comp) {
    const etiquetas = ['Precio', 'Pantalla', 'Procesador', 'RAM', 'Disco duro', 'Graficos'];

    const migrafico = document.getElementById("grafico");

    const datasets = [];
    const coloresBorde = ['red', 'blue', 'green', 'orange', 'purple', 'yellow']; // Define colores para el borde
    const coloresFondo = ['rgba(255,0,0,0.3)', 'rgba(0,0,255,0.3)', 'rgba(0,255,0,0.3)', 'rgba(255,165,0,0.3)', 'rgba(128,0,128,0.3)', 'rgba(255,255,0,0.3)'];

    // Recolectar los datos en formato adecuado para la normalización
    const datosParaNormalizar = arrayObtenido.map((elemento, i) => [
        datos_comp[i].precio, datos_comp[i].panta, datos_comp[i].cpu, datos_comp[i].ram, datos_comp[i].hd, datos_comp[i].gpu
    ]);

    // Normalizar solo las dimensiones de Procesador (2) y RAM (3)
    const maxValor = 2000; // Puedes ajustar este valor según tus necesidades
    const datosNormalizados = normalizarDimensiones(datosParaNormalizar, [1, 2, 3], maxValor);

    datosNormalizados.forEach((elemento, i) => {
        datasets.push({
            label: arrayObtenido[i].nombre,
            data: elemento,
            borderColor: coloresBorde[i % coloresBorde.length], // Usamos el operador módulo para asegurarnos de no salirnos del rango de colores
            backgroundColor: coloresFondo[i % coloresFondo.length]
        });
    });

    const datos = {
        labels: etiquetas,
        datasets: datasets
    };

    const configuracion = {
        type: 'radar',
        data: datos
    };

    new Chart(migrafico, configuracion);
}
```

12.Agradecimientos.

Quiero expresar mi más sincero agradecimiento a todas las personas que se han tomado el tiempo de leer esta memoria. Su interés y dedicación para entender el trabajo y los esfuerzos invertidos en este proyecto son profundamente valorados.

Agradezco a todos aquellos que han ofrecido su apoyo y orientación a lo largo del desarrollo del proyecto. Sus conocimientos, paciencia y críticas constructivas han sido esenciales para superar numerosos desafíos y alcanzar los objetivos planteados.

Gracias a quienes han brindado su apoyo incondicional, tanto emocional como logístico, durante este proceso. Su confianza en mis capacidades y su ánimo constante han sido fundamentales para mantener la motivación y perseverar ante las dificultades.

Agradezco también a todos los colaboradores del equipo, cuya cooperación y dedicación han sido vitales para el éxito del proyecto. Juntos hemos aprendido, superado obstáculos y celebrado logros.

Finalmente, agradezco a todos aquellos que han proporcionado un entorno de aprendizaje y recursos que han permitido desarrollar las habilidades y conocimientos necesarios para llevar a cabo este proyecto. La educación y las oportunidades ofrecidas han sido cruciales para la realización de esta memoria.

Este proyecto ha sido una experiencia enriquecedora y desafiante, y todos los involucrados han jugado un papel fundamental en su éxito. A todos ustedes, gracias.