|  |  |
| --- | --- |
| 3 | **2025** |
|  | IES San Vicente  Jorge Valero y Juan Lorenzo  1º ASIR Semipresencial  10-5-2025 |

|  |
| --- |
| **[aplicación web de calzado]** |
| Desarrollo de una aplicación web para la gestión de calzado. |

Contenido

[1. Introducción 2](#_Toc197792007)

[2. Antecedentes 3](#_Toc197792008)

[3. Análisis 4](#_Toc197792009)

[4. Diseño 5](#_Toc197792010)

[5. Implementación Técnica 6](#_Toc197792011)

[• Base de Datos y Configuración: 6](#_Toc197792012)

[• Gestión de Usuarios (Básica): 6](#_Toc197792013)

[• Interfaz de Usuario (Frontend HTML, CSS, JS): 7](#_Toc197792014)

[• Gestión de Productos (Backend PHP): 8](#_Toc197792015)

[6. Resultados 9](#_Toc197792016)

[7. Conclusiones y Mejoras Futuras 10](#_Toc197792017)

[Anexo I – Proyecto de Empresa 11](#_Toc197792018)

[Herramientas de gestión del proyecto 12](#_Toc197792019)

Imágenes………………………………………………………………………………………………………………………………..13

# 1. Introducción

En el contexto actual del comercio, la gestión eficiente de inventarios y el acceso rápido a productos son esenciales para una experiencia de compra óptima. Este proyecto tiene como objetivo desarrollar una aplicación web centrada en la gestión de stock de zapatillas mediante un servidor propio, ofreciendo una solución rápida, segura y escalable.  
  
La solución está orientada tanto a tiendas como a usuarios, permitiendo una actualización en tiempo real del stock y facilitando la consulta y administración del inventario. Con un servidor dedicado, se garantiza mayor seguridad, autonomía y velocidad.  
  
Este trabajo se centra en el desarrollo completo del sistema: diseño de interfaz, estructura de base de datos, programación del backend, y pruebas funcionales. Además, se incorporan criterios de accesibilidad, diseño responsive y optimización de carga.

# 2. Antecedentes

El comercio de zapatillas ha experimentado una transformación significativa con la digitalización. Tradicionalmente basado en tiendas físicas, el sector ha debido adaptarse al entorno online, impulsado por la demanda de rapidez, disponibilidad y variedad.  
  
Numerosas plataformas han emergido utilizando servicios de terceros, pero esto puede limitar el control y la seguridad. Por ello, muchas empresas optan por servidores propios, integraciones personalizadas y gestión directa del stock.  
  
Este proyecto parte de esa necesidad, ofreciendo una solución enfocada en zapatillas deportivas actualmente, con control completo sobre el inventario y personalización según las necesidades del negocio.

# 3. Análisis

El sistema implementado sigue una arquitectura cliente-servidor utilizando tecnologías como PHP, MySQL, HTML/CSS y JavaScript. El servidor almacena la base de datos del inventario, mientras que el cliente accede a la información a través del navegador.  
  
Flujo de trabajo básico:  
1. El usuario accede al catálogo mediante filtros (marca, talla, color, stock).  
2. Se muestran productos disponibles con detalles e imágenes.  
3. El administrador puede agregar productos mediante un formulario protegido.  
4. Las operaciones modifican la base de datos y el stock en tiempo real.  
  
La solución busca eficiencia, velocidad de acceso y facilidad de uso para administradores y usuarios.

# 4. Diseño

La plataforma utiliza un diseño responsive basado en principios de UX/UI modernos. La navegación se realiza mediante una barra superior fija con acceso a secciones clave. El sistema incorpora filtros dinámicos y mensajes de stock en tiempo real.  
  
Visualmente, se emplean colores neutros y tipografía sans-serif para asegurar claridad. Las imágenes son relevantes, optimizadas y organizadas.  
  
El formulario de productos incluye validación, y el login asegura acceso restringido. Toda la experiencia está diseñada para ser fluida tanto en escritorio como en dispositivos móviles.

# 5. Implementación Técnica

La estructura de archivos se divide en carpetas organizadas por función:  
- `/php/`: contiene scripts como conexión (`config.php`), inserción (`guardar\_producto.php`) y consulta (`get\_products.php`).  
- `/js/`: contiene `main.js` y `validation.js` para interactividad y validaciones.  
- `/images/`: contiene imágenes de los productos.  
- `/web/`: archivos `.html` y `.php` con interfaz de usuario.  
  
La base de datos incluye las tablas `products`, `usuarios` y el control del stock. Se ha configurado phpMyAdmin en el entorno local (WAMP) para su gestión.

### • Base de Datos y Configuración:

* La base de datos se llama zapatostyle.
* El archivo db\_init.php está diseñado para **inicializar esta base de datos** y crear la tabla productos si no existe. La tabla productos almacena información como nombre, descripción, precio, talla, stock, marca, color e imagen.
* El archivo config.php centraliza los **parámetros de conexión a la base de datos**, como el host, usuario, contraseña y nombre de la base de datos.  
  Se mencionan **recomendaciones de seguridad importantes** para este archivo en entornos de producción, como no usar el usuario root, usar contraseñas seguras y moverlo fuera del directorio público.

### • Gestión de Usuarios (Básica):

* El archivo login.php maneja el **inicio de sesión**. Verifica si el usuario ya está autenticado mediante la sesión.  
  Si no lo está, muestra un formulario.  
  La autenticación válida las credenciales ("admin" / "1234"), lo cual **solo es para realizar pruebas, no es recomendable para producción**.  
  Si el login es exitoso, se establece una variable de sesión $\_SESSION['login'] = true y se redirige a add\_product.php.
* El archivo logout.php se encarga de **cerrar la sesión**. Simplemente destruye la sesión activa (session\_destroy()) y redirige al usuario de vuelta a login.php.
* Los archivos add\_product.php y login.php utilizan session\_start() para gestionar el estado de la sesión.

### • Interfaz de Usuario (Frontend HTML, CSS, JS):

* El archivo index.html es la **página principal**. Contiene la estructura básica, una barra de navegación, una sección “hero”, una sección para mostrar productos y un pie de página.  
  Incluye **elementos clave para los filtros** (selects con IDs como #brandFilter) y un contenedor para la lista de productos (#productsList).  
  También tiene una **plantilla HTML** (#productTemplate) que main.js clona para mostrar cada producto.
* El archivo contacto.html es la **página de contacto**. Contiene información de contacto y un formulario.  
  Se menciona que este formulario no envía datos reales, sino que su funcionalidad está **simulada en JavaScript**.
* El archivo sobre\_nosotros.html es la página **"Sobre Nosotros"**. Incluye información textual sobre la tienda, su historia y características clave con iconos.
* El archivo style.css maneja todo el **estilo visual de la aplicación**. Está bien organizado, utiliza **variables CSS** para colores y espacios, e implementa un diseño **responsive** adaptable a distintos tamaños de pantalla.  
  Define estilos para los componentes como la cabecera, los productos, filtros, botones, etc.
* El archivo main.js es el **motor dinámico del frontend**.  
  Controla el menú móvil. Gestiona la interacción con los filtros dinámicos, llamando a loadProducts() cuando se aplican.  
  La función loadProducts() realiza una **petición fetch** a get\_products.php para obtener los productos filtrados.  
  Lee la respuesta en **JSON** y **actualiza las opciones de los filtros** (updateFilterOptions) y muestra los productos en la interfaz (displayProducts) usando la plantilla HTML.  
  También simula el **envío del formulario** de contacto.
* El archivo validation.js se enfoca en la **validación del formulario de añadir productos** (posiblemente el usado con guardar\_producto.php).  
  Proporciona validación para cada campo (nombre, marca, color, talla, precio, stock).  
  Incluye una función validateForm() que valida todos los campos.  
  La función submitForm() envía los datos del formulario usando fetch() a php/add\_product.php o guardar\_producto.php (según sea el caso) y maneja la respuesta JSON para mostrar mensajes de éxito o error al usuario.

### • Gestión de Productos (Backend PHP):

* Hay dos enfoques para añadir productos descritos en los archivos:
  + add\_product.php: Este archivo recibe datos por POST, verifica si hay una sesión iniciada, valida mínimamente que nombre, precio y talla estén presentes, e **inserta el producto en la base de datos** usando mysqli y prepare para seguridad.  
    Después de la inserción, redirige al usuario a index.php. Contiene también el formulario HTML para añadir el producto.
  + guardar\_producto.php: Esta es una versión alternativa diseñada para ser usada como **API**. Recibe datos por POST (con nombres en inglés), es validada con validation.js, realiza una validación mínima y devuelve una **respuesta en formato JSON** indicando éxito o error. También utiliza prepare para inserciones seguras.
* El archivo get\_products.php es crucial para mostrar productos. Consulta la **tabla productos en la base de datos**.  
  Puede aplicar **filtros dinámicamente** (por marca, color, talla, o si tienen stock) recibidos por GET.  
  Utiliza prepare y bind\_param para seguridad.  
  Devuelve los datos como una **lista de valores únicos para los filtros** en **formato JSON**, lo cual es ideal para ser consumido por JavaScript en el frontend.

#### Diseño del servidor y sistema de copias

La aplicación web se encuentra alojada en un servidor Windows server 2022 configurado con un sistema RAID1, que garantiza la **tolerancia a fallos** al mantener los datos duplicados en dos discos físicos. En este entorno se han instalado los servicios web necesarios (Apache, PHP y MySQL) para ejecutar la plataforma.

Además, se ha diseñado una rutina de **copias de seguridad automatizadas** que se ejecutan diariamente. Estas copias se almacenan en el segundo disco del RAID, y se realiza una verificación de integridad al finalizar. El objetivo es asegurar la persistencia de datos ante fallos imprevistos del sistema.

#### Script de monitorización y alertas

Se ha implementado un **script de monitorización** del servidor que se ejecuta una vez al día tras el cierre de jornada. Este script comprueba parámetros clave del sistema como el uso de CPU, memoria, espacio en disco y carga media. Si se detectan valores por encima de un umbral crítico, se genera una **alerta automática** que puede ser enviada por correo electrónico o registrada en un archivo de log para su revisión.

Esta funcionalidad permite anticiparse a posibles fallos del sistema y garantiza una supervisión proactiva del entorno.

# 6. Resultados

Se ha logrado implementar una solución funcional para la visualización y gestión de stock. El sistema permite:  
- Agregar productos con imagen, talla, color, marca y precio.  
- Consultar zapatillas disponibles mediante filtros.  
- Mostrar productos sin stock y destacar los disponibles.  
  
El sistema es accesible localmente, presenta tiempos de carga bajos y ofrece una experiencia clara y sencilla al usuario.

# 7. Conclusiones y Mejoras Futuras

El proyecto ha demostrado que es posible construir una plataforma personalizada para la gestión de stock de calzado con tecnologías abiertas y control total sobre el servidor.  
  
Dificultades superadas incluyeron la validación de formularios, estructura de base de datos y diseño responsive. Como mejoras se proponen:  
- Añadir edición y eliminación de productos.  
- Generar informes y gráficos de stock en PDF.  
- Publicación online del sitio y conexión con pasarelas de pago.

# Anexo I – Proyecto de Empresa

Análisis DAFO:  
- Debilidades: Recursos técnicos limitados, dependencia inicial del entorno local.  
- Amenazas: Competencia con plataformas ya consolidadas.  
- Fortalezas: Control total del sistema, escalabilidad, personalización.  
- Oportunidades: Expansión a nuevos nichos, integración de inteligencia artificial.  
  
Plan económico básico (estimado):  
- Desarrollo y mantenimiento inicial: 300€.  
- Tiempo estimado de trabajo: 80 horas.

# Herramientas de gestión del proyecto

Para la coordinación del equipo, el control del código y la planificación de tareas se han utilizado las siguientes herramientas:

* **Jira**  
  Hemos utilizado Jira como herramienta principal de planificación y gestión ágil de tareas. El proyecto se estructuró en épicas, historias de usuario y tareas técnicas, organizadas por sprint. Esto nos permitió distribuir el trabajo y realizar seguimiento de los avances.  
  <https://juanlorenzo87.atlassian.net/jira/software/projects/SCRUM/boards/1/backlog>
* **GitHub**  
  El código fuente de la aplicación, los scripts de copias de seguridad y la documentación técnica se alojan en un repositorio privado en GitHub. También se ha usado para el control de versiones y la colaboración simultánea entre los miembros del equipo.  
  <https://github.com/panakosan3000/proyecto-stock-servidor>
* **Slack**  
  Slack ha sido la herramienta de comunicación interna del equipo. Permite mantener un canal continuo donde resolver dudas, compartir errores en tiempo real, y coordinar reuniones o sesiones conjuntas de trabajo.  
  <https://app.slack.com/client/T08LQQ6AADV/C08M3ABM3HB>

**Bibliografía y recursos consultados:**

* <https://www.php.net/>
* <https://www.w3schools.com/>
* Apuntes del módulo de Proyecto Integrado – 1º ASIR





  
  
  
  
  
  
  
  






  
  
  
  
  
  
  


  
  
  
  
  


  
  
  
  


  
  


  
  
  
  
  
  
  


