Prueba Técnica: Sistema de Catálogo de Productos

Esta prueba técnica presenta el desarrollo de un Sistema de Catálogo de Productos, abarcando desde la instalación de herramientas hasta la implementación y prueba del sistema. Se diseñó una aplicación web con una API en PHP para la gestión de productos y categorías, conectada a una base de datos MySQL.

El proyecto inició con la configuración del entorno de desarrollo utilizando XAMPP como servidor local y PDO para la conexión segura a la base de datos. Se creó un esquema relacional en MySQL para almacenar productos y categorías de manera eficiente.

El backend se desarrolló siguiendo una arquitectura REST, implementando endpoints que permiten listar, buscar y filtrar productos. Para el frontend, se construyó una interfaz en HTML, CSS y JavaScript, la cual consume la API mediante fetch(), mostrando los productos dinámicamente y permitiendo la interacción del usuario a través de filtros y búsquedas.

Finalmente, se realizaron pruebas funcionales para validar el correcto funcionamiento de la API, la conexión con la base de datos y la visualización de datos en la interfaz. Este documento detalla cada una de estas etapas, resaltando el código y las decisiones técnicas clave en el desarrollo del sistema.

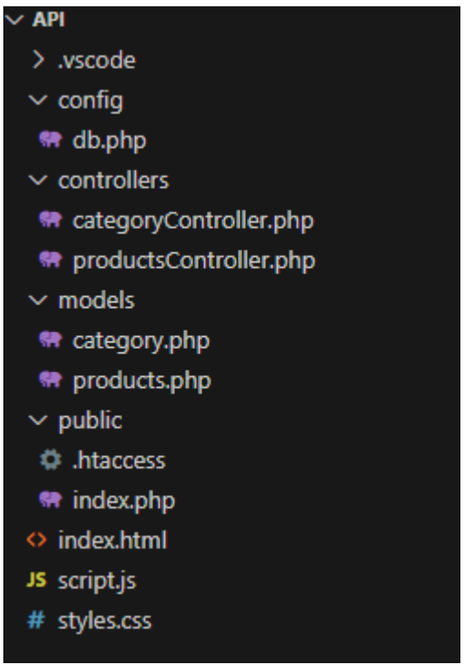
1. Instalación y Configuración

1.1 Instalación del Entorno de Desarrollo

Para desarrollar y ejecutar la aplicación, se configuró un entorno local utilizando XAMPP (Apache, MySQL, PHP) en un sistema Windows.

Pasos realizados:

1. Se descargó e instaló XAMPP desde el sitio web oficial.
2. Se iniciaron los servicios de Apache y MySQL desde el panel de control de XAMPP.
3. Se configuró un directorio de trabajo dentro de htdocs para el proyecto.

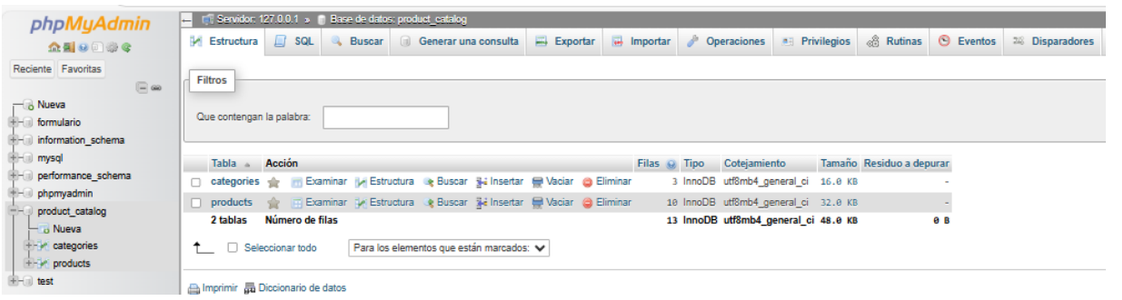


1.2 Creación de la Base de Datos

Se creó la base de datos product_catalog y sus tablas mediante phpMyAdmin. La base de datos contiene dos tablas principales:

- products: almacena los productos del catálogo.

- categories: gestiona las categorías disponibles.



2. Desarrollo del Backend

El backend se implementó en PHP utilizando el patrón MVC (Modelo-Vista-Controlador) para mantener un código organizado y escalable.

2.1 Configuración de la Conexión a la Base de Datos

Se creó un archivo de configuración para establecer la conexión con la base de datos mediante PDO.

2.2 Creación de los Modelos

Se definieron los modelos Products y Category para interactuar con la base de datos. Estos modelos contienen las funciones necesarias para obtener los productos y categorías de la base de datos, realizar búsquedas y filtrados, y manejar la paginación.

2.3 Creación de los Controladores

Se crearon controladores para manejar las solicitudes de la API. Los controladores gestionan la lógica de negocio para obtener los productos, categorías, y realizar búsquedas. Están estructurados para devolver respuestas en formato JSON.

2.4 Implementación de la API REST

Se desarrolló una API REST que permite realizar operaciones sobre los productos y categorías. Los endpoints permiten consultar productos, obtener productos por categoría, y realizar búsquedas por término.

2.5 Configuración de .htaccess

Se configuró un archivo .htaccess para gestionar las rutas amigables, permitiendo que las solicitudes se redirijan correctamente al archivo index.php.

3. Desarrollo del Frontend

Se diseñó una interfaz de usuario simple y moderna utilizando HTML, CSS y JavaScript.

3.1 Estructura del HTML

La página principal contiene un encabezado con el nombre de la aplicación, un campo de búsqueda, un filtro de categorías y una lista de productos que se carga dinámicamente.

3.2 Estilos con CSS

El diseño es responsivo y se utiliza un esquema de colores sencillo, con la opción de cambiar entre modo claro y oscuro para mejorar la experiencia del usuario.

3.3 Lógica con JavaScript

El frontend realiza solicitudes a la API REST utilizando fetch para obtener los productos y categorías, y luego los muestra en la interfaz. También se implementó una funcionalidad para cambiar el tema de la página entre claro y oscuro, guardando la preferencia del usuario en el localStorage.

4. Pruebas

Las pruebas se realizaron en Postman

4.1 Pruebas en Postman

Se probó la API realizando las siguientes operaciones:

- Consultar todos los productos.
- Consultar un producto por su ID.
- Filtrar productos por categoría.

Obtener todos los productos

- Método: GET
- URL: http://localhost/api/productos?page=1&limit=10
- Descripción: Obtiene una lista de productos paginados (en este caso, página 1 y límite de 10 productos).

Obtener un producto por su ID

- Método: GET
- URL: http://localhost/api/productos/1
- Descripción: Obtiene la información de un producto específico mediante su ID.

Obtener productos por categoría

- Método: GET
- URL: http://localhost/api/productos/categoria/Electronica
- Descripción: Obtiene todos los productos que pertenecen a una categoría específica.

Buscar productos por término

- Método: GET
- URL: http://localhost/api/productos/buscar?television
- Descripción: Busca productos cuyo nombre o descripción contengan el término proporcionado.

Obtener todas las categorías

- Método: GET
- URL: http://localhost/api/categorias
- Descripción: Obtiene la lista de todas las categorías disponibles.

Conclusión

La creación de la API para el sistema de catálogo de productos fue un proceso enriquecedor, pero también presentó varios desafíos. Uno de los mayores retos ocurrió durante las pruebas, cuando la API comenzaba a generar errores inesperados. Estos errores se manifestaron principalmente al realizar peticiones a las rutas del backend. Para resolverlo, tuve que revisar las configuraciones de Apache y actualizar los datos en los archivos de configuración, especialmente en .htaccess y config/db.php.

A pesar de estas correcciones, los errores seguían ocurriendo, lo que me llevó a investigar más a fondo utilizando los registros de logs. Aunque esto ayudó a identificar posibles fallos, ninguna de las soluciones comunes fue efectiva. Pasé mucho tiempo ajustando configuraciones en el servidor y reconfigurando las rutas y los parámetros de la API, pero no logré solucionar completamente los problemas.

El backend fue, sin duda, la parte que más tiempo me consumió en el proyecto, especialmente debido a la persistencia de los errores durante las pruebas. Sin embargo, una vez que pude solucionar los problemas técnicos, el proyecto avanzó de manera más fluida, y pude concluir con éxito la implementación y conexión del frontend con la API. A pesar de los obstáculos encontrados, esta experiencia fue muy valiosa y me enseñó la importancia de la depuración y la investigación en el desarrollo de sistemas complejos.

Este proyecto fue realizado por **Andrea Gómez Santiesteban** como parte de una prueba técnica.