MagiKTCG

Luis Muñoz Barceló
2nFPGS
2024/2025

Índice

1. Introducción.

Este Trabajo de Fin de Grado tiene como objetivo el diseño y desarrollo de una página web comercial llamada **MagiKTCG.es**, orientada a la compraventa de productos de colección relacionados con el universo de los *Trading Card Games* (TCG), en especial cartas, sobres, cajas selladas, *blisters*, *ultra premium boxes*, y otros artículos de la franquicia Pokémon, abarcando todas sus generaciones.

La plataforma se estructura de forma clara e intuitiva, permitiendo a los usuarios navegar por un catálogo clasificado según el idioma de los productos: **inglés**, **japonés**, **español y coreano**, siguiendo este orden según su relevancia y valor en el mercado del coleccionismo. Dentro de cada categoría se ofrecen distintas presentaciones de producto: sobres individuales, cajas completas, *blisters* con cartas promocionales, packs especiales, entre otros.

Cada producto dispone de una ficha individual con precio, descripción e imágenes ilustrativas.

El sistema permite añadir productos a un carrito de compra, donde se calcula el importe total del pedido. Además, el usuario podrá completar un formulario con sus datos de envío, lo que permitirá estimar los costes adicionales asociados al transporte, antes de proceder con el pago.

Este proyecto no solo busca implementar una solución funcional, sino también aplicar buenas prácticas de desarrollo web, empleando tecnologías como **HTML5**, **SCSS**, **JavaScript** y **PHP**, sin el uso de frameworks como Bootstrap, para así mantener un mayor control sobre la estructura y estilo del sitio.

2. Estado del arte.

El comercio electrónico especializado en productos de colección relacionados con juegos de cartas como Pokémon TCG ha experimentado un notable crecimiento en los últimos años. Sitios como **Cardmarket**, **eBay**, **Troll and Toad** o **TCGPlayer** son algunas de las plataformas más consolidadas a nivel internacional. En el ámbito hispano, también existen webs como **CartasPokemon.es** o **Darizard9** que ofrecen productos similares, aunque con menor alcance.

Estas plataformas, si bien son funcionales y robustas, a menudo presentan una interfaz sobrecargada, poco intuitiva o excesivamente genérica, lo que puede dificultar la experiencia de navegación y compra del usuario final. En algunos casos, el sistema de clasificación de productos por idioma o formato no está

suficientemente depurado, lo cual afecta negativamente a usuarios que buscan artículos muy específicos.

A nivel tecnológico, muchas de estas webs se apoyan en frameworks como **React**, **Angular**, o utilizan CMS (Sistemas de Gestión de Contenidos) como **WordPress** con plugins como **WooCommerce** para agilizar el desarrollo. Aunque estas herramientas ofrecen facilidad de implementación, también implican una sobrecarga de recursos y limitan el control sobre el código fuente..

En contraposición, el presente proyecto plantea una solución completamente personalizada desde cero, utilizando tecnologías web puras como **HTML5**, **SCSS**, **JavaScript** y **PHP**, lo que permite mayor optimización, accesibilidad y adaptabilidad. Además, se prioriza la organización clara de productos por idioma y tipo, junto con una experiencia de usuario fluida y coherente, especialmente orientada a coleccionistas y compradores habituales del sector TCG.

Con este planteamiento, MagiKTCG.es se posiciona como una alternativa ligera, eficaz y centrada en el usuario, con un enfoque específico y detallado en el universo Pokémon TCG.

3. Estudio de viabilidad. Método DAFO.

El análisis DAFO (Debilidades, Amenazas, Fortalezas y Oportunidades) permite evaluar los factores internos y externos que pueden afectar al éxito del proyecto:

Debilidades:

- El proyecto es desarrollado por una única persona, lo que puede ralentizar la implementación.
- Falta de experiencia comercial real en gestion de e-commerce y logística.
- No se cuenta con un presupuesto económico elevado.

Amenazas:

- Alta competencia de plataformas ya consolidadas en el sector.
- Cambios en las políticas de envío o costes aduaneros que puedan afectar al modelo de negocio.
- Riesgo de fluctuación de precios en el mercado del coleccionismo.

Fortalezas:

- Desarrollo personalizado sin depender de frameworks pesados o CMSs, lo cual mejora el rendimiento y control.
- Enfoque específico y claro en el coleccionismo de cartas Pokémon , un nicho con demanda estable.
- Experiencia previa en desarrollo web y planificación anticipada del proyecto.

- El control total del código permite escalar y adaptar más fácilmente la plataforma en el futuro.

Oportunidades:
- Nicho aún no completamente explotado en el mercado hispanohablante.
- Posibilidad de escalar la tienda a otros productos TCG u ofrecer servicios complementarios (valoración de cartas, comunidad, etc.).
- Integración futura con redes sociales o canales de venta alternativos (Wallapop, Vinted, Instragram Shopping, Marketplace de Facebook, Tik Tok Shop…).

a. Estudio de mercado.

Actualmente, el mercado de cartas coleccionables, especialmente de Pokémon TCG, vive un auge impulsado tanto por nostalgia como por inversión. Productos de ediciones antiguas o limitadas han alcanzado precios muy altos en plataformas de reventa. Este interés ha favorecido la creación de múltiples webs especializadas, pero la mayoría son demasiado generales o centradas en mercados no hispanos.

MagiKTCG.es busca diferenciarse como un espacio especializado en coleccionismo Pokémon, bien organizado, transparente y con una navegación clara por idiomas y categorías.

i. Viabilidad técnica/económica del proyecto

1. Recursos HW
- Ordenador personal con capacidad suficiente para programación y pruebas locales.
- Posibilidad de desplegar la web en servidores gratuitos durante el desarrollo (ej. 000Webhost, InfinityFree).
- Servidor web local (XAMPP, Laragon)

2. Recursos SW
- Visual Studio Code como entorno de desarrollo.
- Git y GitHub para control de versiones.
- Navegadores modernos para pruebas (Firefox, Chrome, Edge, Safari).
- Software de edición de imágenes (GIMP, Photoshop, Figma).
- MySQL o MariaDB como sistema de bases de datos.
- Apache como servidor web local.
- Lenguajes: HTML5, SCSS, JavaScript, PHP 8+.

3. Recursos humanos
- Desarrollo realizado íntegramente por el alumno.

- Posibilidad de recibir asesoría puntual del tutor del proyecto.

ii.Viabilidad temporal

El desarrollo del proyecto se ha iniciado de forma anticipada (Enero 2025), lo cual permitió una distribución más cómoda del trabajo en fases. Ya se ha completado el análisis inicial, diseño preliminar y parte del código base. Está previsto su finalización para su entrega a principios de Junio

b. Planificación temporal o agenda de trabajo.

Fase	**Periodo estimado**	**Actividades principales**
Análisis y viabilidad	*Enero - Febrero 2025*	DAFO, estudio de mercado, objetivos, estructura general
Diseño base y maqueta	*Febrero - Marzo 2025*	ER, mockups, mapa web, diseño de base de datos
Codificación	*Marzo - Mayo 2025*	Frontend, Backend, conexión BD, funciones principales
Pruebas y ajustes	*Mayo 2025*	Validación de formularios, pruebas de compatibilidad
Documentación escrita	*Mayo - Junio 2025*	Redacción de memoria y manual del usuario
Presentación final	*Junio 2025*	Preparación de presentación oral y entrega final

4. Análisis de requisitos

a. Descripción de requisitos.

El proyecto requiere definir una serie de requisitos funcionales y no funcionales que servirán como base para establecer las funcionalidades del sistema, su comportamiento esperado y las condiciones necesarias para su correcto funcionamiento.

- **Requisitos Funcionales:**

Los requisitos funcionales definen el comportamiento específico del sistema desde el punto de vista del usuario. A continuación, se enumeran los principales:

- **RF1:** El sistema debe permitir al usuario visualizar un catálogo de productos relacionados con cartas coleccionables de Pokémon.
- **RF2:** El sistema debe permitir al usuario filtrar productos por idioma (inglés, japonés, español, coreano).
- **RF3:** El sistema debe mostrar una página individual por cada producto, incluyendo su precio, imágenes, y una breve descripción.
- **RF4:** El sistema debe permitir al usuario añadir productos al carrito de compra.
- **RF5:** El sistema debe mostrar el contenido del carrito, el total acumulado y un formulario para introducir los datos de envío.
- **RF6:** El sistema debe calcular los gastos de envío en base a la localización proporcionada por el usuario.
- **RF7:** El sistema debe simular o integrar un proceso de pago.

- **Requisitos no funcionales**

Los requisitos no funcionales establecen criterios de calidad que deben cumplirse para asegurar el buen rendimiento del sistema:

- **RNF1:** La aplicación web debe ser responsive, adaptándose correctamente a distintos dispositivos y tamaños de pantalla.
- **RNF2:** La web debe ser accesible desde los navegadores más utilizados (Chrome, Firefox, Edge, Safari, Opera).
- **RNF3:** El tiempo de carga de cada página no debe superar los 3 segundos.
- **RNF4:** El sistema debe proteger los datos introducidos por el usuario y evitar vulnerabilidades comunes como inyecciones SQL.

- **RNF5:** El diseño debe ser intuitivo y estéticamente atractivo para un público interesado en el coleccionismo.

i. Diagramas de caso de uso de los más relevantes. Realizando un caso de uso general y si es necesario otros diagramas más específicos.

Actores principales:

- **Usuario visitante:** Persona que navega la web sin estar autenticada.
- **Cliente:** Usuario que añade productos al carrito y realiza una compra.
- **Administrador (futuro opcional):** Persona encargada de gestionar el contenido, productos y pedidos (no implementado en esta primera versión, pero considerado a futuro).

Caso de uso general: Navegación y compra

- **Actor principal:** Usuario visitante / Cliente
- **Objetivo:** Visualizar productos, añadir al carrito y realizar la compra.

Flujo principal:

1. El usuario accede al sitio web.
2. Explora el catálogo de productos desde la página de inicio o mediante los menús de navegación por idioma.
3. Selecciona un producto y accede a su página individual.
4. Añade el producto deseado al carrito.
5. Visualiza el carrito y comprueba el total.
6. Rellena el formulario de datos de envío.
7. Se calculan los gastos de envío automáticamente.
8. Se procede al pago (simulado o redireccionado).
9. Se muestra un mensaje de confirmación de compra.

Casos de uso específicos relevantes:

1. Caso de uso: **Buscar productos**

Actor: Usuario visitante
Descripción: El usuario puede buscar productos usando un buscador por texto.
Precondición: El sitio debe tener un buscador visible en la cabecera.
Resultado esperado: Se muestra una lista de productos que coinciden con el término de búsqueda.

2. Caso de uso: **Filtrar productos por idioma**

Actor: Usuario visitante
Descripción: El usuario puede filtrar los productos en base al idioma de la carta (Inglés, Japonés, Español, etc.).
Resultado esperado: La lista de productos se actualiza mostrando solo los del idioma seleccionado.

3. Caso de uso: **Calcular envío**

Actor: Cliente
Descripción: Tras introducir su ubicación, el cliente puede ver el coste estimado del envío.
Resultado esperado: Se calcula automáticamente un coste de envío en base a la localización ingresada.

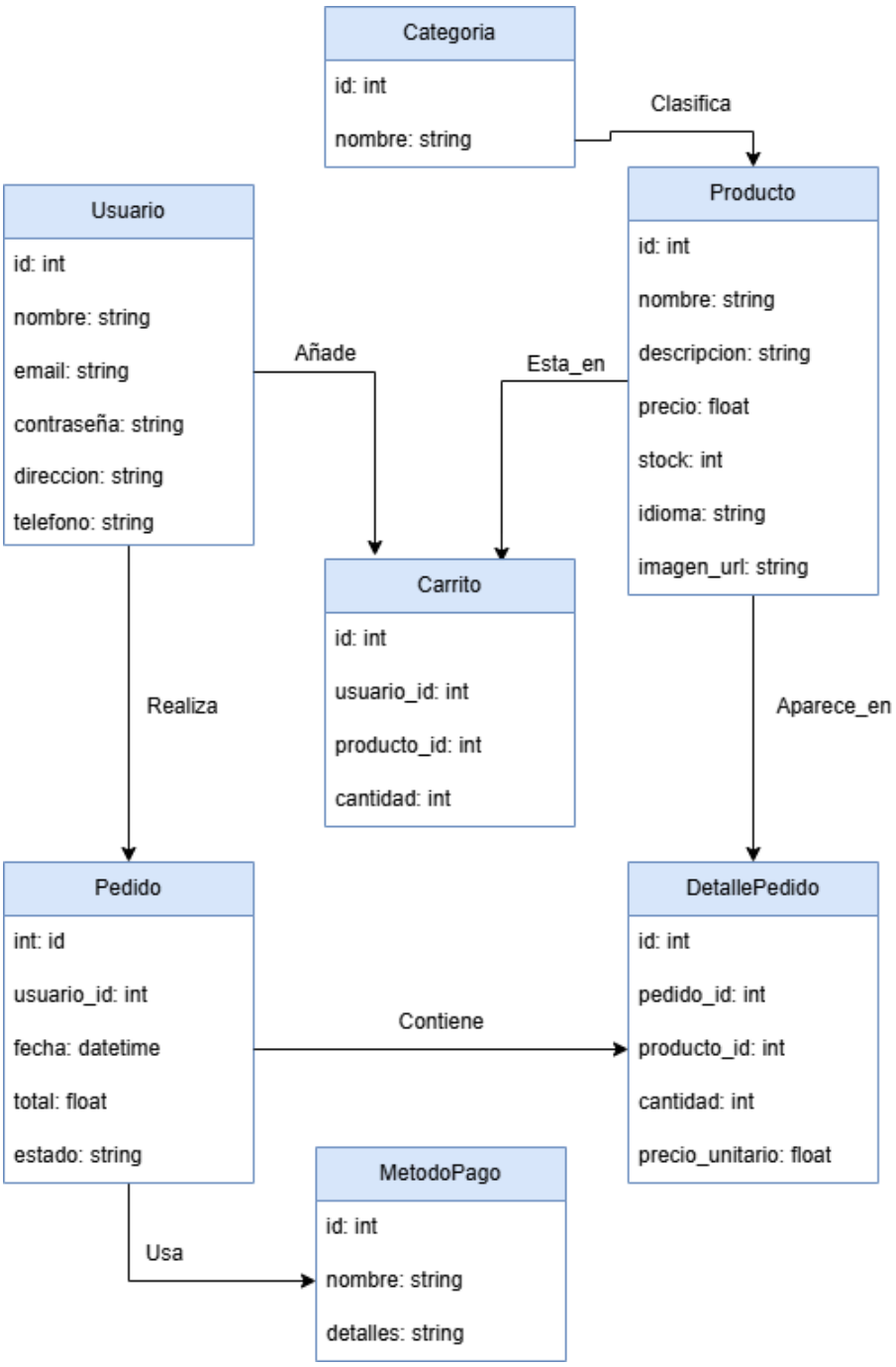
5. Diseño

a. Diseño Conceptual Entidad Relación

- **Usuarios:** representan a los clientes registrados, quienes pueden realizar compras y gestionar su información.
- **Productos:** artículos disponibles para la venta, clasificados por categoría e idioma.
- **Categorías:** agrupan los productos según su idioma.
- **Carrito:** entidad intermedia que representa los productos que un usuario tiene seleccionados antes de finalizar la compra.
- **Pedidos:** contienen información sobre las compras confirmadas por los usuarios.
- **Detalles de Pedido:** descomponen cada pedido en líneas individuales de productos adquiridos.
- **Métodos de Pago:** definen las formas de pago disponibles para el usuario.

b. Diseño Lógico Relacional o Paso a tablas.

- **Usuarios** (id, nombre, email, contraseña, dirección, teléfono)
- **Productos** (id, nombre, descripción, precio, stock, idioma, imagen_url, categoria_id)
- **Categorías** (id, nombre)
- **Carrito** (id, usuario_id, producto_id, cantidad)
- **Pedidos** (id, usuario_id, fecha, total, estado, metodo_pago_id)
- **Detalles_Pedido** (id, pedido_id, producto_id, cantidad, precio_unitario)
- **Métodos_Pago** (id, nombre, detalles)


Categoria
id: int
nombre: string
Clasifica
Producto
id: int
nombre: string
descripcion: string
precio: float
stock: int
idioma: string
imagen_url: string
Usuario
id: int
nombre: string
email: string
contraseña: string
direccion: string
telefono: string
Añade
Esta_en
Carrito
id: int
usuario_id: int
producto_id: int
cantidad: int
Realiza
Aparece_en
Pedido
int: id
usuario_id: int
fecha: datetime
total: float
estado: string
Contiene
DetallePedido
id: int
pedido_id: int
producto_id: int
cantidad: int
precio_unitario: float
MetodoPago
id: int
nombre: string
detalles: string
Usa

c. Diseño Físico o Diagrama Mysql Descripción de las tablas y campos.

- **VARCHAR(255):** para campos de texto como nombre, email, dirección, etc.
- **TEXT:** para descripciones y detalles extensos.
- **DECIMAL(10,2):** para precios y totales.
- **INT / BIGINT:** para identificadores y relaciones.
- **DATETIME**: para la fecha del pedido.
- **ENUM (o VARCHAR controlado):** para el campo estado del pedido (ej: “pendiente”, “enviado”, “entregado”, “cancelado”).

d. Orientación a objetos:

1. Diagramas de clases. Descripción de clases y atributos.

```
class Usuario {
  int id;
  string nombre;
  string email;
  string contraseña;
  string dirección;
  string teléfono;
}

class Producto {
  int id;
  string nombre;
  string descripción;
  float precio;
  int stock;
  string idioma;
  string imagen_url;
  int categoria_id;
}

class Categoria {
  int id;
```

```
  string nombre;
}

class Pedido {
  int id;
  int usuario_id;
  datetime fecha;
  float total;
  string estado;
  int metodo_pago_id;
}

class DetallePedido {
  int id;
  int pedido_id;
  int producto_id;
  int cantidad;
  float precio_unitario;
}

class MetodoPago {
  int id;
  string nombre;
  string detalles;
}

class Carrito {
  int id;
  int usuario_id;
  int producto_id;
  int cantidad;
}
```

2. Diagrama de secuencias. De lo más relevante.

1. El usuario añade un producto al carrito.
2. El sistema actualiza o crea la entrada correspondiente en la tabla **Carrito**.
3. Al confirmar la compra, se crea un nuevo **Pedido**.
4. Se generan las entradas en **Detalles_Pedido** en función de los productos del carrito.
5. Se asocia el método de pago y se calcula el total.

3. Diagrama de actividad. De lo más relevante.

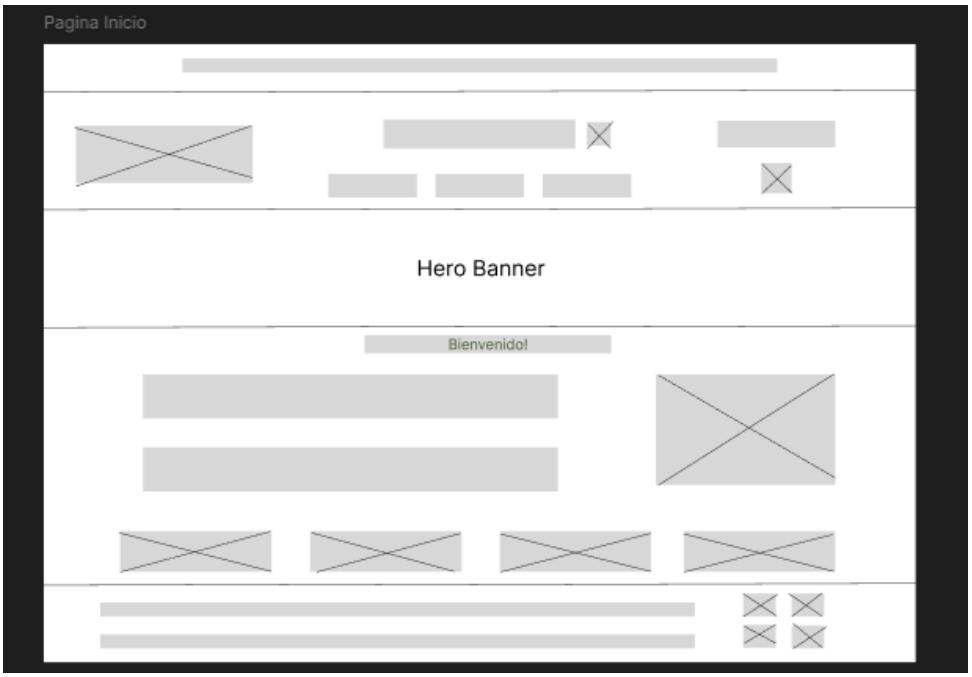
Inicio → Registro/Login → Navegar productos → Añadir al carrito → Ver carrito → Confirmar pedido → Seleccionar método de pago → Finalizar compra.

e. Mapa Web. Gráfico que muestra los enlaces entre páginas.

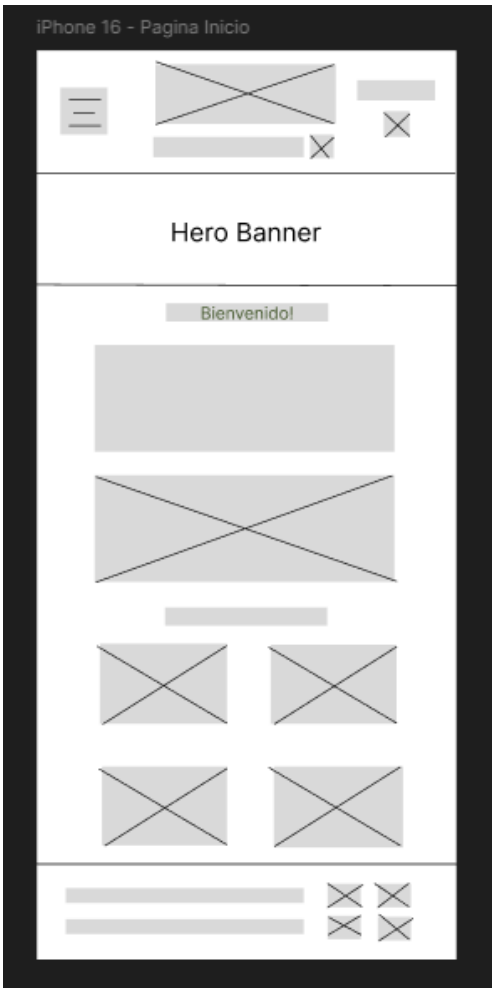
[Inicio]
- [Catálogo]
  - Japonés
  - Inglés
  - Español
  - Coreano
- [Producto: Detalle]
- [Carrito]
- [Formulario de compra]
- [Registro / Login]
- [Contacto / Información]

f. Mockups

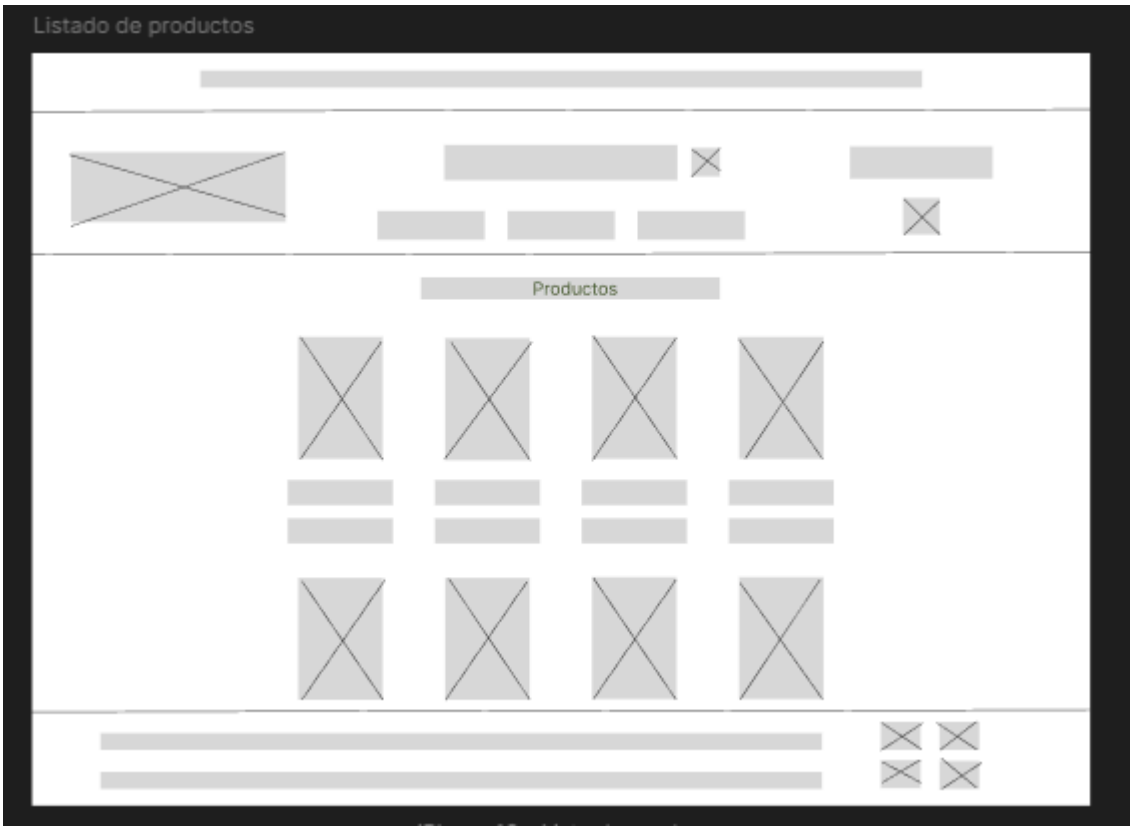
Pagina inicio - Ordenador:



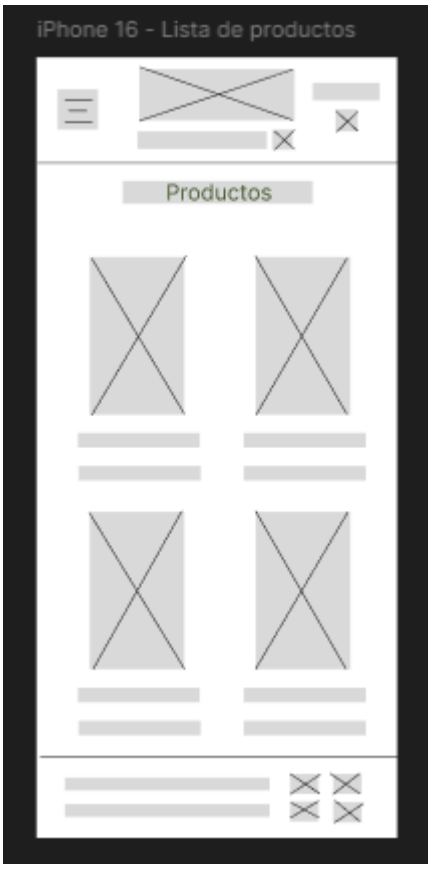
Página inicio - Móvil



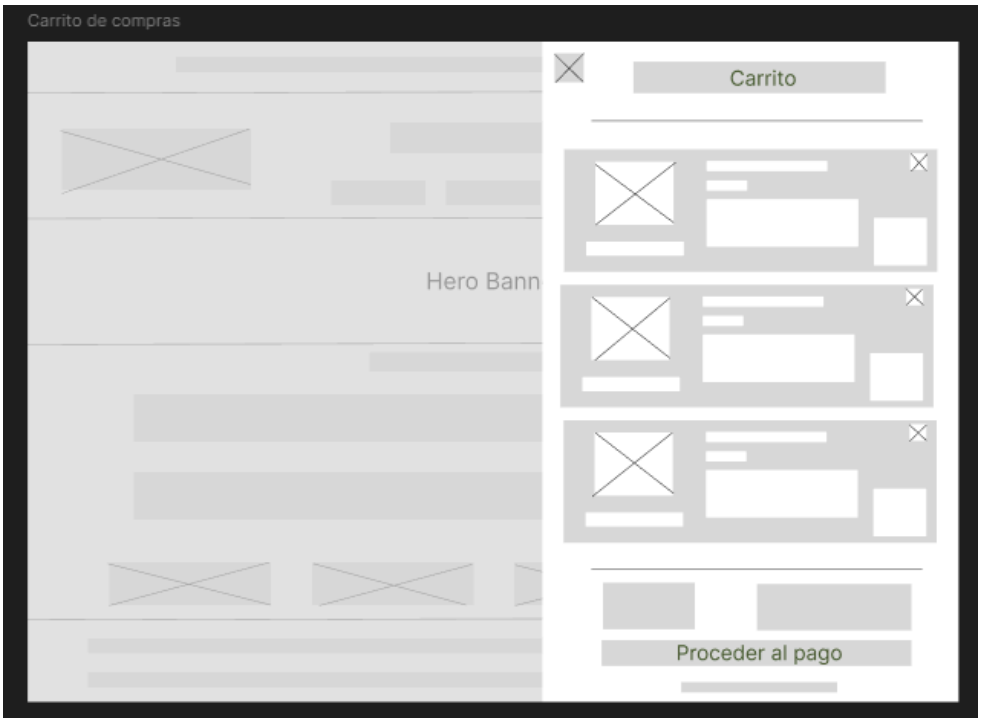
Listado de productos - Ordenador



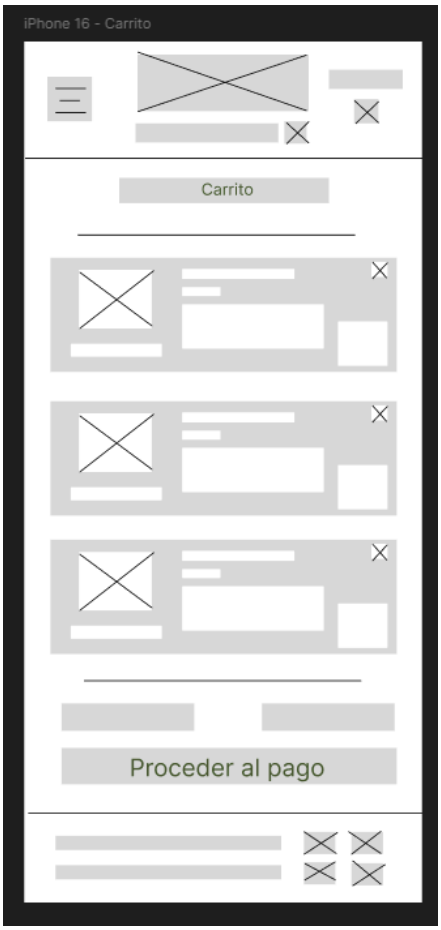
Listado de productos - Móvil



Carrito de compra - Ordenador



Carrito de compra - Móvil



6. Codificación.

a. Tecnologías elegidas y su justificación (lenguajes, frameworks, librerías, etc.)

Para desarrollar este proyecto web he elegido herramientas y lenguajes de programación que ya conocía y que están bien documentados. Mi objetivo era crear una web funcional, fácil de mantener y que pudiera servir como base para futuros proyectos parecidos. A continuación explico las tecnologías que he utilizado:

Front-End (lo que ve el usuario)

- **HTML5**: Es el lenguaje que se usa para crear la estructura de las páginas web. Lo he utilizado para organizar todo el contenido que aparece en pantalla.

- **SCSS**: Es una forma más avanzada de escribir CSS (los estilos de la web), y me ha ayudado a mantener el diseño más limpio y organizado.

- **JavaScript**: Lo he usado para que algunas partes de la web respondan a lo que hace el usuario (por ejemplo, añadir productos al carrito). No he usado librerías ni frameworks, solo JavaScript puro.

Back-End (lo que pasa en el servidor)

- **PHP**: Es el lenguaje que se encarga de gestionar todo lo que pasa detrás de la web, como registrar usuarios, guardar pedidos, etc.

- **MySQL**: Es la base de datos, donde se guarda toda la información como los productos, usuarios, pedidos, etc.

b. Entorno servidor.

Para hacer pruebas y desarrollar la web he usado un servidor local (es decir, en mi propio ordenador) utilizando herramientas como **XAMPP**. Esto me permite simular cómo funcionaría la web en un servidor real sin necesidad de subirla a internet todavía.

En cuanto a la seguridad, he tenido cuidado con cosas como proteger la base de datos usando consultas seguras y evitando errores que puedan mostrar información innecesaria al usuario. También he intentado controlar los posibles fallos para que, si ocurre un error, no afecte al funcionamiento del resto de la web.

c. Entorno cliente.

El diseño de la web está pensado para que funcione en cualquier navegador moderno como **Chrome**, **Firefox**, **Edge**, **Safari** u **Opera**, y también para que se vea bien en ordenadores, móviles y tablets.

La idea principal ha sido hacer una web clara, fácil de navegar y con una estética sencilla, para que cualquier persona pueda entender rápidamente cómo funciona.

d. Documentación interna de código (puede ir en un anexo).

i. Descripción de cada fichero. Autor, función y fecha de creación.
ii. Descripción de cada función. Autor, función y fecha de creación.

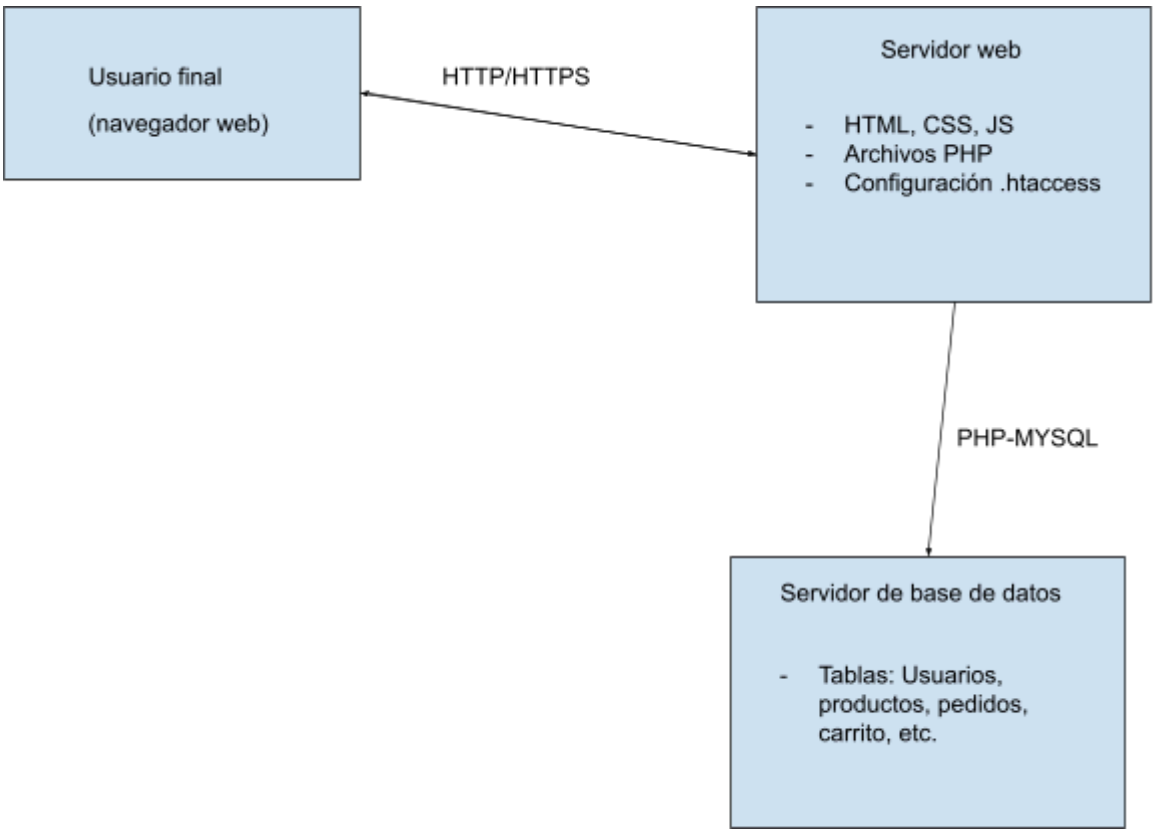
He incluido comentarios dentro del código explicando para qué sirve cada parte. Al principio de cada archivo y de cada función importante, se puede ver quién lo ha hecho, qué hace exactamente y cuándo se creó.

Esto ayuda a que si en el futuro yo mismo, o cualquier otra persona, quiere modificar el proyecto, le resulte más fácil entender cómo está hecho todo.

7. Despliegue

a. Diagramas de despliegue

El siguiente diagrama representa los principales componentes involucrados en la ejecución del sistema MagiKTCG.es, así como su interacción en tiempo de ejecución:



b. Descripción de la instalación o despliegue

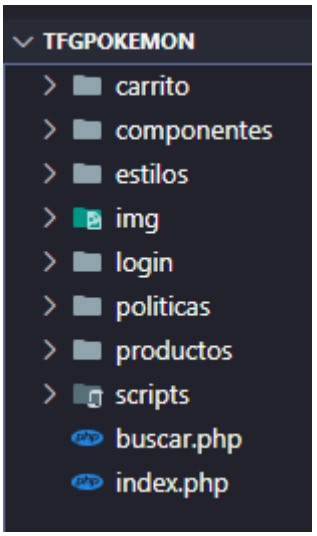
1.Entorno local:

Herramientas necesarias:

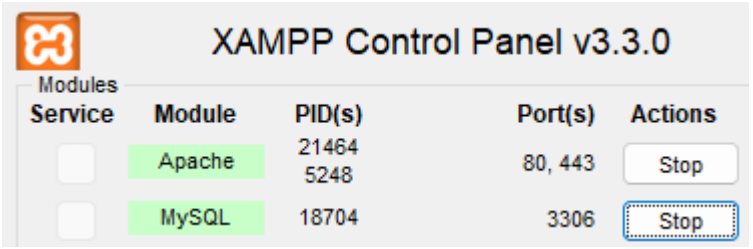
- XAMPP o Laragon (incluye Apache, PHP y MySQL)
- VS Code
- Navegador (Chrome, Firefox, Edge…)

Pasos

1. Clonar o copiar el proyecto en la carpeta htdocs (XAMPP) o en la raíz del servidor local.



2. Asegurarse de que Apache y MySQL estén en ejecución.



3. Crear una base de datos desde **phpMyAdmin**
4. Configurar los parámetros de conexión en un archivo config.php (host, usuario, contraseña, nombre BD).
5. Acceder a **http://localhost/TFGPokemon/index.php** (en este caso) para probar la web.

8. Herramientas de apoyo

a. Control de versiones.

Durante el desarrollo del proyecto, se ha utilizado Git como sistema de control de versiones. Esta herramienta ha permitido realizar un seguimiento preciso de los cambios en el código fuente, facilitando tanto la organización como la reversión de errores. Para el almacenamiento y sincronización del repositorio se ha empleado GitHub, lo cual ha aportado una capa adicional de respaldo y acceso remoto.

Gracias a Git, fue posible implementar distintas ramas para trabajar en nuevas funcionalidades sin afectar el código principal, permitiendo así una integración más ordenada y segura.

b. Sistemas de integración continua.

Debido a la naturaleza individual y académica del proyecto, no se ha implementado un sistema de integración continua automatizado (como Jenkins, Travis CI o GitHub Actions). Sin embargo, el control manual de versiones y las pruebas periódicas han cumplido un rol similar en cuanto a detección de errores y validación funcional antes de cada hito importante.

En futuros desarrollos o ampliaciones, se contempla la posibilidad de integrar herramientas CI/CD para automatizar pruebas y despliegues.

c. Gestión de pruebas

La validación de funcionalidades se ha realizado mediante pruebas manuales estructuradas. Se diseñaron varios casos de prueba que contemplaban tanto escenarios esperados como situaciones límite o erróneas. Entre ellos se destacan:

- Pruebas de navegación desde distintos navegadores y dispositivos.
- Pruebas del carrito de compra (añadir, quitar, modificar cantidad).
- Comprobación de la persistencia de datos (productos, pedidos).
- Validación de formularios de registro, inicio de sesión y compra.
- Pruebas de seguridad básicas (inyecciones SQL, inputs vacíos, etc.).

Estas pruebas han sido documentadas en un anexo para futuras consultas y como evidencia del correcto funcionamiento del sistema.

9. Conclusiones.

a. Conclusiones sobre el trabajo realizado

El proyecto MagiKTCG.es ha permitido consolidar los conocimientos adquiridos a lo largo del ciclo formativo, aplicando conceptos de desarrollo web, diseño de bases de datos, planificación de proyectos y experiencia de usuario. La plataforma resultante es funcional, estructurada y enfocada a un nicho concreto, lo cual cumple con los objetivos iniciales propuestos.

Además, el uso de tecnologías puras como HTML5, SCSS, JavaScript y PHP ha ofrecido un control total sobre el desarrollo, a costa de una mayor carga de trabajo, lo que ha resultado enriquecedor para el aprendizaje.

b. Conclusiones personales

A nivel personal, este trabajo ha supuesto un reto que ha implicado no solo aplicar conocimientos técnicos, sino también organizar tiempos, priorizar tareas y resolver problemas de forma autónoma. Me ha permitido experimentar de forma directa cómo es desarrollar una aplicación web desde cero, enfrentando tanto la parte técnica como aspectos de diseño y experiencia de usuario.

También he comprendido la importancia de documentar bien cada fase del desarrollo y mantener el código limpio, modular y comprensible.

c. Posibles ampliaciones y mejoras

A pesar de que la versión actual es funcional, existen varias posibilidades de mejora que podrían implementarse a futuro:

- Sistema de login para usuarios registrados y seguimiento de pedidos.
- Sistema de valoración de productos por parte de los usuarios.
- Incorporación de un blog o sección de noticias para mejorar el SEO.
- Optimización avanzada del rendimiento web y mejora del posicionamiento en buscadores

10.Bibliografía (comentada)

a. Libros, artículos y apuntes

- Apuntes propios del ciclo formativo de Desarrollo de Aplicaciones Web (1º y 2º curso).

b. Direcciones web

MDN Web Docs. Fuente principal para consultar especificaciones de HTML, CSS y JavaScript.
Sitio oficial de PHP, utilizado para consultar funciones y buenas prácticas.
Guía online de MySQL, útil para construir y optimizar consultas SQL.
Portal educativo con ejemplos prácticos en distintos lenguajes web.
Documentación oficial de Git, consultada para gestionar ramas y versiones.