

# All in Toys



## ÍNDICE:

### 1. Descripción general del proyecto

- a. Detalle del problema planteado indicando punto por punto todos los detalles a gestionar.
- b. Indicar las áreas generales en las que se va a centrar el proyecto

### 2. Perfiles de usuario

Descripción de todos los perfiles de usuario que habrá en la aplicación

### 3. Diagrama Entidad/Relación

- a. Diseñar el diagrama Entidad/Relación
- b. Explicar origen de los atributos
- c. Explicar por qué se han elegido esas claves primarias

### 4. Paso a tablas

- a. Realizar el paso a tablas del diagrama Entidad/Relación

### 5. Implantación de la base de datos

- a. Código SQL que genera las tablas

### 6. Diseño de la web

- a. Estudio de la competencia
- b. Diagramación: mapa web y menús de navegación
- c. Prototipos de bajo nivel: Wireframes
- d. Referentes estéticos y colores
- e. Tipografía
- f. Imágenes: Logotipo, iconos, botones y formularios
- g. Diseño gráfico de la interfaz

### 7. Programación de la web

- a. Funcionalidades programadas con PHP
- b. Funcionalidades programadas con AJAX
- c. APIs utilizadas

### 8. Manuales de usuario

## DESCRIPCIÓN GENERAL DEL PROYECTO

Mi trabajo de final de grado va a consistir en la creación de una página web sobre una tienda de juguetes que se llama All in Toys que le falta una página web donde mostrar sus productos y venderlos por internet.

La aplicación web se estructurará en varias páginas:

- Inicio: Presentación de la tienda, sección de los juguetes más vendidos, preguntas frecuentes y promociones.
- Catálogo: Listado completo de los juguetes de la base de datos, con la opción de filtrar por categoría o sección.
- Marcas: Visualización de las marcas con las que trabaja la tienda y los juguetes asociados a cada una.
- Registro/Login: Formulario de registro y acceso para usuarios.
- Carrito de compra: Disponible solo para usuarios registrados, donde podrán gestionar sus compras.

## PERFILES DE USUARIO

Los diferentes tipos de usuario son los siguientes:

- Usuario sin registrar:

Este Usuario puede navegar por toda la página, ver todos los juguetes, pero no podrá comprar ninguno ya que no está registrado.

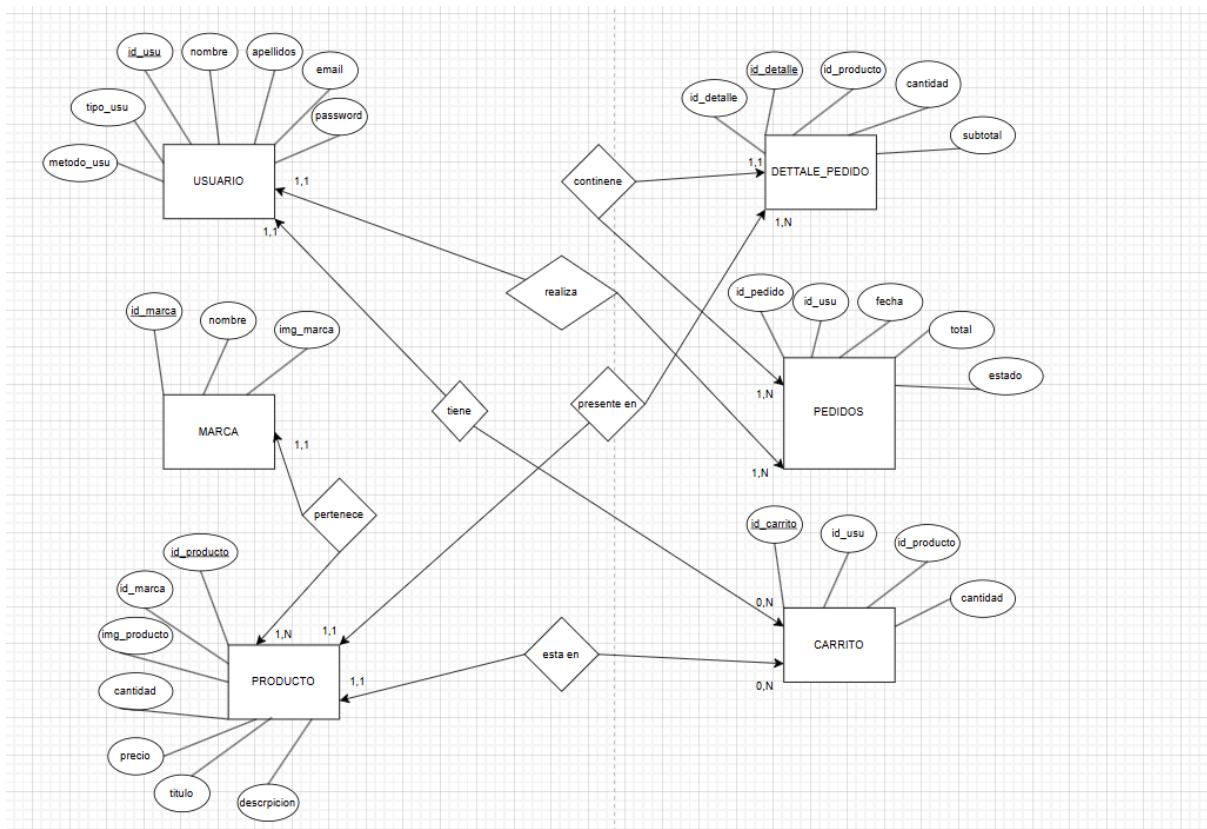
- Usuario registrado:

Este Usuario puede navegar por toda la página y una vez registrado, podrá realizar compras de juguetes.

- Administrador:

Este Usuario podrá realizar los cambios de los productos en cuanto a precio, compañía a la que pertenece, añadir o eliminar productos.

## DIAGRAMA ENTIDAD/RELACIÓN



Explicación de porqué he elegido los atributos anteriores para cada tabla de mi base de datos:

- **USUARIOS:**
  - **id\_usu:** Su función es darle un valor identificador único a cada usuario.
  - **nombre:** Almacena el nombre del usuario.
  - **apellidos:** Contiene los apellidos del usuario.
  - **correo:** Dirección de correo electrónico del usuario, empleada como credencial de acceso.
  - **password:** Cadena de caracteres que representa la contraseña del usuario, necesaria para control de acceso.
  - **metodo\_pago:** Campo destinado a almacenar una referencia o detalle del método de pago preferido del usuario.
  - **tipo\_usu:** Define el rol de acceso del usuario dentro de la página (registrado, administrador, sin registrar).

- MARCA:

- **id\_marca:** Su función es darle un valor identificador único a cada marca de juguetes. Necesario para categorizar los productos por su fabricante.
- **nombre:** Nombre comercial de la marca.
- **imagen\_marca:** URL o (ruta de almacenamiento del logotipo) de la imagen representativa de la marca, para su visualización en la web.

- PRODUCTO:

- **id\_producto:** Su función es darle un valor identificador único a cada juguete.
- **titulo:** Nombre del producto.
- **precio:** Valor monetario del producto para su venta.
- **imagen\_producto:** URL o (ruta de almacenamiento del logotipo) de la imagen representativa del juguete, para su visualización en la web.
- **descripcion:** Texto detallado que describe las características y funcionalidades del juguete.
- **id\_marca:** Foreign key que establece la relación con la tabla **marca**, indicando la marca a la que pertenece cada producto.
- **cantidad:** Representa el stock disponible del producto, indicando cuántas unidades hay en almacén.

- PEDIDOS:

- **id\_pedido**: Su función es darle un valor identificador único a cada orden de compra realizada. Permite el seguimiento individual de cada transacción.
- **id\_usu**: Foreign key que vincula el pedido con el **usuario** que lo ha generado. Crucial para el historial de compras del cliente.
- **total**: Suma total del importe de todos los productos y posibles gastos de envío asociados al pedido.
- **fecha**: Fecha exacta en que el pedido fue registrado en el sistema.
- **estado**: Indica la fase actual en la que se encuentra el pedido (ej., 'pendiente', 'procesando', 'enviado', 'entregado', 'cancelado'). TENGO Q MIRAR ESTO XQ NS COMO PONERLO

- DETALLE\_PEDIDO:

- **id\_detalle**: Su función es darle un valor identificador único a cada línea de un pedido. Permite diferenciar los diferentes productos que pueden componer un único pedido.
- **id\_pedido**: Foreign key que enlaza este detalle con el **pedido** principal al que pertenece.
- **id\_producto**: Foreign key que asocia esta línea de detalle con el **producto** específico del que se trata.
- **cantidad**: Número de unidades del producto específico que se incluyen en esta línea del pedido.
- **subtotal**: Costo total de los productos de esta línea de detalle ( $\text{cantidad} * \text{precio del producto al momento de la compra}$ ). Su almacenamiento es importante para mantener un registro histórico del precio.

- CARRITO:
  - **id\_carrito**: Su función es darle un valor identificador único a cada ítem dentro del carrito de un usuario.
  - **id\_usu**: Foreign key que asocia esta línea del carrito al **usuario** propietario. Permite que cada usuario tenga su propio carrito personal.
  - **id\_producto**: Foreign key que vincula este ítem del carrito con el **producto** que el usuario desea comprar.
  - **cantidad**: Cantidad de unidades de un producto específico que el usuario ha añadido a su carrito.

En las tablas anteriores he puesto como claves principales los siguientes atributos:

- **id\_usu** (en **usuarios**): Es un identificador numérico autoincremental que garantiza que cada usuario registrado tenga una identidad única en el sistema.
- **id\_marca** (en **marca**): Similar a **id\_usu**, asegura que cada marca de juguetes sea única, ya que si lo ponemos al nombre este puede en un futuro ser modificado.
- **id\_producto** (en **producto**): Proporciona un identificador único para cada juguete, permitiendo un control preciso del inventario, ventas y referencias en otras tablas.
- **id\_pedido** (en **pedidos**): Cada pedido es una transacción única y debe tener un identificador que lo diferencie de cualquier otro, facilitando su seguimiento y gestión.
- **id\_detalle** (en **detalle\_pedido**): Aunque un pedido puede contener múltiples productos, cada línea de producto dentro de un pedido necesita su propio identificador único para ser gestionada de forma independiente y así no haya errores a la hora de gestionar las cantidades o precios del artículo pedido.
- **id\_carrito** (en **carrito**): Cada entrada de un producto en el carrito de un usuario debe ser única.

## PASO A TABLAS

A continuación, se detalla la conversión de las entidades del Diagrama Entidad/Relación en tablas relacionales, especificando las columnas que componen cada una, así como sus primary key(PK) y sus foreign key(FK)

- **usuarios:** `id_usu` (PK), nombre, apellidos, correo, contraseña, metodo\_pago, tipo\_usu
- **marca:** `id_marca` (PK), nombre, imagen\_marca
- **producto:** `id_producto` (PK), titulo, precio, imagen\_producto, descripcion, `id_marca` (FK), cantidad
- **pedidos:** `id_pedido` (PK), `id_usu` (FK), total, fecha, estado
- **detalle\_pedido:** `id_detalle` (PK), `id_pedido` (FK), `id_producto` (FK), cantidad, subtotal
- **carrito:** `id_carrito` (PK), `id_usu` (FK), `id_producto` (FK), cantidad

## IMPLANTACIÓN DE LA BASE DE DATOS

-- 1. Tabla: usuarios -- Almacena la información de los usuarios registrados y administradores.

```
CREATE TABLE usuarios (  
id_usu INT AUTO_INCREMENT PRIMARY KEY, -- Clave Primaria, autoincremental  
nombre VARCHAR(100) NOT NULL,  
apellidos VARCHAR(100) NOT NULL,  
correo VARCHAR(255) NOT NULL UNIQUE, -- Correo único para cada usuario  
password VARCHAR(255) NOT NULL, -- Almacenar hash de la contraseña  
tipo_usu ENUM('registrado', 'administrador') NOT NULL DEFAULT  
'registrado' -- Rol del usuario );
```

-- 2. Tabla: marca -- Contiene los datos de las marcas de juguetes.

```
CREATE TABLE marca (  
id_marca INT AUTO_INCREMENT PRIMARY KEY, -- Clave Primaria,  
autoincremental  
nombre VARCHAR(100) NOT NULL UNIQUE, -- Nombre de la marca debe ser único  
imagen_marca VARCHAR(255) -- Ruta o URL de la imagen de la marca );
```



-- 3. Tabla: producto -- Almacena los detalles de los juguetes disponibles.

```
CREATE TABLE producto (  
id_producto INT AUTO_INCREMENT PRIMARY KEY, -- Clave Primaria,  
autoincremental  
titulo VARCHAR(255) NOT NULL,  
precio DECIMAL(10, 2) NOT NULL, -- Precio con 2 decimales  
imagen_producto VARCHAR(255), -- Ruta o URL de la imagen del producto  
descripcion TEXT, -- Campo de texto largo para la descripción  
id_marca INT NOT NULL, -- Clave Foránea a la tabla 'marca'  
cantidad INT NOT NULL DEFAULT 0, -- Stock disponible, por defecto 0  
FOREIGN KEY (id_marca) REFERENCES marca(id_marca)  
ON UPDATE CASCADE ON DELETE RESTRICT -- Acciones en cascada  
para actualización, restrict para borrado );
```

-- 4. Tabla: pedidos -- Registra las transacciones de compra realizadas por los usuarios.

```
CREATE TABLE pedidos (  
id_pedido INT AUTO_INCREMENT PRIMARY KEY, -- Clave Primaria,  
autoincremental  
id_usu INT NOT NULL, -- Clave Foránea a la tabla 'usuarios'  
total DECIMAL(10, 2) NOT NULL, -- Monto total del pedido  
fecha DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP, -- Fecha y  
hora del pedido  
estado ENUM('pendiente', 'procesando', 'enviado', 'entregado',  
'cancelado') NOT NULL DEFAULT 'pendiente', -- Estado del pedido  
FOREIGN KEY (id_usu) REFERENCES usuarios(id_usu)  
ON UPDATE CASCADE ON DELETE RESTRICT );
```

-- 5. Tabla: detalle\_pedido -- Desglosa los productos individuales incluidos en cada pedido.

```
CREATE TABLE detalle_pedido (  
id_detalle INT AUTO_INCREMENT PRIMARY KEY, -- Clave Primaria,  
autoincremental  
id_pedido INT NOT NULL, -- Clave Foránea a la tabla 'pedidos'  
id_producto INT NOT NULL, -- Clave Foránea a la tabla 'producto'  
cantidad INT NOT NULL, -- Cantidad de este producto en el detalle  
subtotal DECIMAL(10, 2) NOT NULL, -- Subtotal de esta línea (cantidad *  
precio_unitario_al_momento_de_compra)  
FOREIGN KEY (id_pedido) REFERENCES pedidos(id_pedido)  
ON UPDATE CASCADE ON DELETE CASCADE, -- Si se borra un pedido,  
sus detalles también  
FOREIGN KEY (id_producto) REFERENCES producto(id_producto)  
ON UPDATE CASCADE ON DELETE RESTRICT -- No borrar producto si  
está en un pedido );
```

```
-- 6. Tabla: carrito -- Almacena los productos que un usuario ha
añadido a su carrito de compra temporalmente.
CREATE TABLE carrito (
  id_carrito INT AUTO_INCREMENT PRIMARY KEY, -- Clave Primaria,
  autoincremental
  id_usu INT NOT NULL, -- Clave Foránea a la tabla 'usuarios'
  id_producto INT NOT NULL, -- Clave Foránea a la tabla 'producto'
  cantidad INT NOT NULL, -- Cantidad del producto en el carrito -- Restricción para que
  un usuario no tenga el mismo producto duplicado en el carrito
  UNIQUE KEY (id_usu, id_producto),
  FOREIGN KEY (id_usu) REFERENCES usuarios(id_usu)
  ON UPDATE CASCADE ON DELETE CASCADE, -- Si se borra un usuario,
  su carrito se limpia
  FOREIGN KEY (id_producto) REFERENCES producto(id_producto) ON
  UPDATE CASCADE ON DELETE CASCADE -- Si se borra un producto, se
  elimina de los carritos );
```

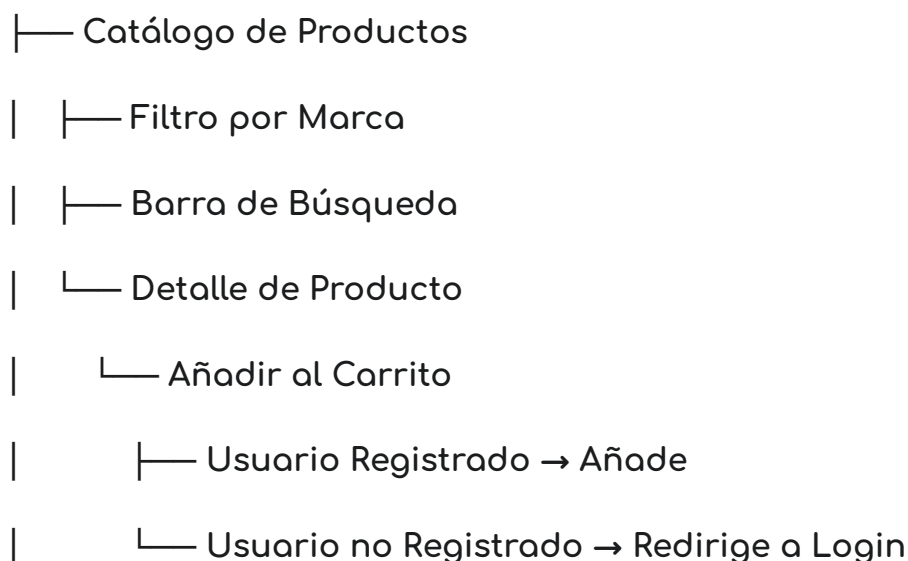
## DISEÑO DE LA WEB

Para realizar mi estudio de mercado, he seleccionado tres competidores directos o plataformas que venden juguetes online.

1. **Juguettos:** Es una de las cadenas de jugueterías más grandes de España. Su sitio web ofrecerá una visión de las funcionalidades básicas de una tienda online de juguetes, como navegación por categorías, fichas de producto y carrito de compra.
  - Puntos a analizar: Diseño general, facilidad de búsqueda y filtrado, proceso de compra, gestión de categorías, presentación de productos.

2. **Toy Planet** : Otra importante cadena de jugueterías con amplia presencia en el mercado español. Su modelo de negocio, a menudo con tiendas franquiciadas, y su estrategia online complementan la visión del sector
  - Puntos a analizar: Organización en sus secciones, la facilidad para el usuario de moverse por la web y la respuesta del sitio y existencia de blogs, noticias, o contenido educativo relacionado con los juguetes.
  
3. **El Corte Inglés Juguetes**: Otra gran superficie con una importante sección de juguetes online. Ofrecerá una perspectiva sobre la integración de un catálogo de juguetes dentro de una tienda más grande, y cómo manejan la marca y la confianza del cliente.
  - Puntos a analizar: Navegación en un catálogo amplio, equilibrio entre diseño y funcionalidad, promociones y ofertas, servicio al cliente.

## MAPA WEB



- |— Marcas

- | |— Ver productos por marca

- |— Acceso / Registro (solo usuarios no registrados)

- | |— Login

- | |— Registro

- |— Carrito (solo usuarios registrados)

- |— Ver productos

- |— Modificar cantidad

- |— Eliminar producto

- |— Procesar pedido

- |— Cerrar sesión

- |  Panel de Administración (solo admin)

- |— Dashboard

- |— Gestión de Usuarios

- | |— Ver listado

- | |— Buscar

- | |— Eliminar

- |— Gestión de Productos

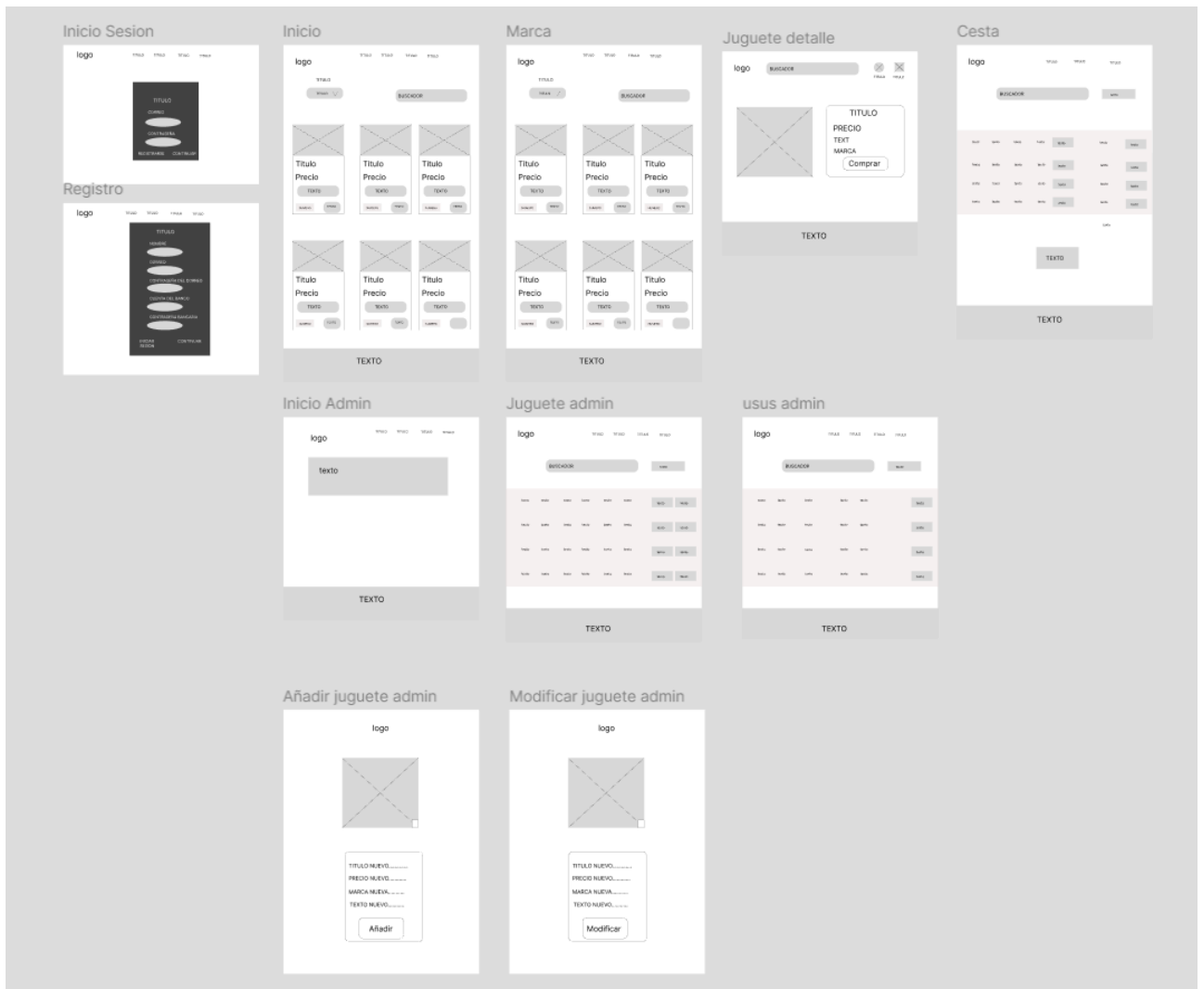
- | |— Ver listado

- | |— Añadir

- | |— Editar

| └─ Eliminar

└─ Cerrar sesión



## Referentes Estéticos:

Para inspirar el diseño visual de la web, he considerado principalmente referentes que transmiten una sensación de diversión, alegría y confianza, adecuada para una tienda de juguetes online con un enfoque moderno y accesible. El objetivo es crear una interfaz limpia, intuitiva y visualmente atractiva, donde los productos sean los protagonistas. Se busca un equilibrio entre un estilo lúdico y la profesionalidad necesaria para una plataforma de comercio electrónico.

## Paleta de Colores:

- Color Primario (Fondo de contenido):
  - Azul celeste: **#D6F0FF**
- Colores Secundarios (Cabecera, enlaces y footer):
  - Azul relajado para cabecera: **#BFEFFF**
  - Azul intenso para footer: **#00AEEF**
  - Rojo vibrante (Enlaces activos/Botones): **#FF4747**
- Colores Neutros (Base y texto):
  - Gris oscuro (Texto principal): **#333333**
  - Amarillo claro (Texto secundario/bordes): **#FFD93D**
- Colores de Estado:
  - Rojo (Acción peligrosa/Error): **#DC3545**

## Filosofía Tipográfica

Para "All in Toys", se busca un texto visual que sea:

- Amigable y Cercana: Que refleje la diversión y accesibilidad del mundo de los juguetes.
- Clara y Legible: Priorizando que la información sea fácil de leer, especialmente para los padres, pero también intuitiva para los niños.

- Moderna y Limpia: En línea con las tendencias de diseño web actuales.

Función de las fuentes:

- Encabezados Principales (H1):
  - Estilo: Poppins Bold
  - Tamaño: 30px
  - Propósito: Utilizado para los títulos más importantes de cada sección o página.
- Encabezados Secundarios (H2):
  - Estilo: Poppins SemiBold
  - Tamaño: 22px
  - Propósito: Empleado para subtítulos y divisiones de contenido importantes.
- Texto Normal (Cuerpo de Texto):
  - Estilo: Poppins Regular
  - Tamaño: 16px
  - Propósito: Usado para las descripciones de productos, bloques de texto informativos, párrafos y todo el contenido general de la web.
- Etiquetas y Menús (Navegación):
  - Estilo: Poppins Medium
  - Tamaño: 14px
  - Propósito: Aplicado a las etiquetas de formulario, elementos de menú de navegación y otros elementos de interfaz.

- Botones de Acción:
  - Estilo: Poppins SemiBold
  - Tamaño: 16px
  - Propósito: Diseñado para la tipografía dentro de los botones de acción (ej. "Comprar", "Añadir al Carrito", "Modificar", "Eliminar").

### Logotipo de "All in Toys":

El logotipo es la representación visual central de la marca y se ubicará prominentemente en la cabecera de todas las páginas de la aplicación.

- Descripción: El logotipo de "All in Toys" combina un diseño legible con elementos gráficos que evocan la temática de los juguetes. Incorpora un diseño que sugiere diversión y confianza, alineado con el público objetivo.
- Propósito: Proporcionar una identidad visual clara a la marca y servir como un punto de anclaje para la navegación, enlazando siempre a la página de inicio.

### Diseño de Formularios:

Los formularios son esenciales para la interacción del usuario, desde el registro y el login hasta la actualización de datos de productos. Su diseño debe ser claro y fácil de usar.

- Campos de Entrada: Como se observa en la sección de "Modificar juguete admin", los campos de texto tendrán un diseño limpio, con un borde sutil y un fondo blanco.
  - Claridad: Las etiquetas de los campos serán claras y estarán asociadas directamente con el campo correspondiente.
  - Feedback: Se diseñarán estados visuales para los campos (ej. borde resaltado al enfocar, indicación visual para campos requeridos, mensajes de error en rojo).



## Gestión de Imágenes de Producto/Marca:

La organización y presentación de las imágenes de los productos y marcas son fundamentales para una tienda online.

- Organización: Como se muestra en la estructura de carpetas, las imágenes se organizarán lógicamente por marca (Barbie, Playmobil, Nancy, Lego) dentro de una carpeta **productos** y **marcas** en el directorio público del proyecto. Esto facilita la gestión y el acceso.

## DISEÑO WEB DEL TRABAJO


### Nuestro Catálogo de Juguetes

CatálogoIniciar SesiónRegistrarseCarrito


#### Filtros y Búsqueda

Filtrar por Marca: Todas las Marcas


Buscar:




**Barbie Dreamhouse**  
199.99 €



**Barbie Chef de Pasteles**  
24.99 €



**Barbie Coche Descapotable Rosa**  
39.99 €



**Barbie Fashionista con Silla de Ruedas**  
29.99 €

### Inicia Sesión

Correo Electrónico:

manuj

Contraseña:

\*\*\*

☐ Recordarme

Iniciar Sesión

¿No tienes cuenta? [Regístrate aquí](#)

### Registra una nueva cuenta

Nombre:

Apellidos:

Correo Electrónico:

Contraseña:

Confirmar Contraseña:

☐ Mostrar contraseñas

Registrarse

Tu Carrito de Compras, m

[Volver al Catálogo](#) [Bienvenido, m](#) [Cerrar Sesión](#)

Contenido de tu Carrito

Producto	Imagen	Cantidad	Precio Unitario	Subtotal	
Barbie Coche Descapotable Rosa		<div>2</div> <div>Actualizar</div>	39.99 €	79.98 €	<a href="#">Eliminar</a>
Barbie Fashionista con Silla de Ruedas		<div>1</div> <div>Actualizar</div>	29.99 €	29.99 €	<a href="#">Eliminar</a>
Total del Carrito:				109.97 €	

Procesar Pedido

Barbie Coche Descapotable Rosa – Detalles

[Catálogo](#) [Iniciar Sesión](#) [Registrarse](#) [Carrito](#)

Barbie Coche Descapotable Rosa

39.99 €

Un elegante coche descapotable para la muñeca Barbie con espacio para dos muñecas.

1

Añadir al Carrito

[← Volver al Catálogo](#)

© 2025 Tu Tienda. Todos los derechos reservados.

## ARCHIVOS DE LA PAGINA

### Index.php

Función: Es el router principal de la aplicación. Decide qué controlador cargar según la acción (action) recibida por GET o POST.

Lógica:

Si la acción es de administración, carga Admin.php.

Si es de usuario registrado, carga Registrado.php.

Si no, carga noRegistrado.php.

```
3
4 // Router principal: decide qué controlador cargar según la acción
5 $action = $_GET['action'] ?? $_POST['action'] ?? 'catalogo';
6
7 // Acciones de administración
8 $acciones_admin = [
9     'administracion',
10    'admin_dashboard',
11    'gestionar_usuarios',
12    'gestionar_productos',
13    'anadir_producto',
14    'editar_producto',
15    'eliminar_producto',
16    'procesar_crear_producto',
17    'procesar_editar_producto',
18    'gestionar_marcas',
19    'gestionar_pedidos',
20    'cerrar_sesion_admin'
21 ];
22
23 // Acciones de usuarios registrados
24 $acciones_registrado = [
25     'catalogo_registrado',
26     'filtrar_por_marca',
27     'buscar_productos',
28     'ver_detalle_producto',
29     'anadir_a_carrito',
30     'ver_carrito',
31 ];
32
33
34 // Si la acción es de admin, carga el controlador de admin
35 if (in_array($action, $acciones_admin)) {
36     require_once __DIR__ . '/controlador/Admin.php';
37 } elseif (in_array($action, $acciones_registrado)) {
38     require_once __DIR__ . '/controlador/Registrado.php';
39 } else {
40     // Por defecto, controlador de usuarios no registrados/registrados
41     require_once __DIR__ . '/controlador/noRegistrado.php';
42 }
43
44 // Si llegamos aquí, significa que no se ha encontrado una acción válida
45
46 ?>
47
```

## Admin.php

Función: Controlador para acciones de administración (gestión de usuarios, productos, marcas, pedidos).

Funciones principales:

mostrarDashboardAdmin: Muestra el panel principal del admin.

cerrarSesionAdmin: Cierra la sesión del admin.

gestionarUsuarios: Lista y busca usuarios.

gestionarProductos: Lista productos.

gestionarMarcas: Lista marcas.

gestionarPedidos: Lista pedidos.

Actions: Un switch que llama a la función adecuada según la acción.

```
<?php
// controlador/Admin.php

// Incluir los modelos y utilidades necesarios
require_once __DIR__ . '/../modelo/Usuario.php'; // Para gestionar usuarios y obtener datos del admin
require_once __DIR__ . '/../modelo/Producto.php'; // Para gestionar productos
require_once __DIR__ . '/../modelo/Marca.php'; // Para gestionar marcas
require_once __DIR__ . '/../modelo/Pedido.php'; // Para gestionar pedidos
require_once __DIR__ . '/../modelo/cookies_sesiones.php'; // Para manejar sesiones y cookies

// --- Comprobación de Autenticación y Rol (CRITICO) ---
// Asegurarse de que la sesión esté iniciada y que el usuario sea un administrador.
start_session(); // Inicia la sesión si no está iniciada

// Si el ID de usuario NO está en sesión o el tipo de usuario NO es 'administrador', redirigir al login
if (!isset($_SESSION['id_us']) || $_SESSION['tipo_us'] != 'administrador') { // CAMBIO CLAVE: 'admin' a 'administrador'
    // Si no está logueado o no es administrador, redirigir al login con un mensaje de acceso denegado
    header("Location: ../index.php?action=login&mensaje=acceso_denegado_admin");
    exit(); // Detener la ejecución del script
}

// Obtener datos del usuario administrador de la sesión
$id_admin_logueado = $_SESSION['id_us'];
$nombre_admin_logueado = $_SESSION['nom_us'];
$tipo_admin_logueado = $_SESSION['tipo_us']; // Debería ser 'administrador'

// Instancias globales de modelos para acceder a ellos dentro de las funciones
$usuario_model = new Usuario();
$producto_model = new Producto();
$marca_model = new Marca();
$pedido_model = new Pedido();

// --- Funciones del Controlador ---

/**
 * Muestra el dashboard principal para el administrador.
 */
function mostrarDashboardAdmin() {
    global $nombre_admin_logueado, $titulo_pagina;

    $titulo_pagina = "Panel de Administración - " . htmlspecialchars($nombre_admin_logueado);

    // Cargar la vista del dashboard de administración.
    require_once __DIR__ . '/../vista/admin_dashboard.php';
}

/**
 * Cierra la sesión del administrador.
 */
function cerrarSesionAdmin() {
    unset($_SESSION['id_us']); // Elimina las variables de sesión del usuario
    unset($_SESSION['nom_us']);
    unset($_SESSION['tipo_us']);
    session_destroy(); // Destruye completamente la sesión

    unset($_COOKIE['usuario']); // Opcional: eliminar la cookie de "recordarme" si existe

    // Redirigir al catálogo de no registrado
    header("Location: ../index.php?action=login&mensaje=sesion_cerrada_admin"); // Redirige a login
    exit();
}

// --- Funciones de gestión (se desarrollarán más adelante) ---
```

```

// --- Funciones de gestión (se desarrollarán mas adelante) ---

/**
 * Muestra la lista de usuarios para la gestión del administrador.
 */
function gestionarUsuarios() {
    global $titulo_pagina, $usuario_model;
    $titulo_pagina = "Gestión de Usuarios";
    $usuarios = $usuario_model->getAllUsuarios(); // Obtener todos los usuarios no admin/administrador
    require_once __DIR__ . '/../vista/admin_usuarios.php';
}

/**
 * Muestra la lista de productos para la gestión del administrador.
 */
function gestionarProductos() {
    global $titulo_pagina, $producto_model;
    $titulo_pagina = "Gestión de Productos";
    $productos = $producto_model->getAllProductos(); // getAllProductosConMarca() si necesitas la marca aqui
    require_once __DIR__ . '/../vista/admin_productos.php';
}

/**
 * Muestra la lista de marcas para la gestión del administrador.
 */
function gestionarMarcas() {
    global $titulo_pagina, $marca_model;
    $titulo_pagina = "Gestión de Marcas";
    $marcas = $marca_model->getAllMarcas();
    require_once __DIR__ . '/../vista/admin_marcas.php'; // Necesitas crear esta vista si no existe
}

/**
 * Muestra la lista de pedidos para la gestión del administrador.
 */
function gestionarPedidos() {
    global $titulo_pagina, $pedido_model;
    $titulo_pagina = "Gestión de Pedidos";
    $pedidos = $pedido_model->getAllPedidos();
    require_once __DIR__ . '/../vista/admin_pedidos.php'; // Necesitas crear esta vista si no existe
}

// --- Dispatcher (Router Simple) ---
$action = $_GET['action'] ?? $_POST['action'] ?? 'administracion'; // 'administracion' es la acción por defecto para el admin

switch ($action) {
    case 'administracion':
    case 'admin_dashboard':
        mostrarDashboardAdmin();
        break;
    case 'gestionar_usuarios':
        // Mostrar la lista de usuarios o la búsqueda
        if (isset($_GET['busqueda'])) {
            // Buscar usuarios según el término
            $usuarios = $usuario_model->buscarUsuarios($_GET['busqueda']);
        } else {
            $usuarios = $usuario_model->getAllUsuarios();
        }
        $titulo_pagina = "Gestión de Usuarios";
        require_once __DIR__ . '/../vista/admin_usuarios.php';
        break;
    case 'eliminar_usuario':
        if (isset($_GET['id'])) {
            $usuario_model->eliminarUsuario((int)$_GET['id']);
        }
        header("Location: /TP6/index.php?action=gestionar_usuarios");
}

```

```

    }
    header("Location: /TFG/index.php?action=gestionar_usuarios");
    exit();
    case 'editar_usuario':
        // Aquí tu lógica para editar usuario
        // ...
        require_once __DIR__ . '/../vista/admin_editar_usuario.php';
        break;
    case 'anadir_usuario':
        // Aquí tu lógica para añadir usuario
        // ...
        require_once __DIR__ . '/../vista/admin_anadir_usuario.php';
        break;
    case 'gestionar_productos':
        gestionarProductos();
        break;
    case 'gestionar_marcas':
        gestionarMarcas();
        break;
    case 'gestionar_pedidos':
        gestionarPedidos();
        break;
    case 'cerrar_sesion_admin':
        cerrarSesionAdmin();
        break;
    default:
        // Si la acción no es reconocida para el admin, redirigir al dashboard del admin
        header("Location: ../index.php?action=administracion");
        exit();
}
?>

```

## Registrado.php

Función: Controlador para usuarios registrados (catálogo, carrito, perfil, pedidos).

Funciones principales:

filtrarProductosRegistrado: Filtra productos por marca.

buscarProductosRegistrado: Busca productos.

verDetalleProductoRegistrado: Muestra detalles de un producto.

anadirACarritoRegistrado: Añade productos al carrito.

eliminarProductoCarrito: Elimina un producto del carrito.

procesarPedido: Procesa el pedido y vacía el carrito.

confirmacionPedido: Muestra confirmación de pedido.

verDetallePedido: Muestra detalles de un pedido.

cerrarSesion: Cierra la sesión del usuario.

```

function mostrarCatalogoRegistrado() {
    global $producto_model, $marca_model, $nombre_usuario_logueado;

    $productos = $producto_model->getAllProductos();
    $marcas = $marca_model->getAllMarcas();
    $titulo_pagina = "Bienvenido, " . htmlspecialchars($nombre_usuario_logueado) . " - Catálogo";

    // Cargar la vista del catálogo (podría ser una versión ligeramente modificada para registrados)
    require_once 'vista/catalogo.php'; // Reutilizamos la misma vista por ahora
}

/**
 * Filtra los productos por una marca específica y muestra los resultados para usuarios registrados.
 */
function filtrarProductosRegistrado() {
    global $producto_model, $marca_model, $nombre_usuario_logueado;

    if (isset($_GET['id_marca']) && is_numeric($_GET['id_marca'])) {
        $id_marca = (int)$_GET['id_marca'];
        $productos = $producto_model->getProductosByMarca($id_marca);

        $marca_info = $marca_model->getMarcaById($id_marca);
        if ($marca_info) {
            $titulo_pagina = "Productos de " . htmlspecialchars($marca_info[1]) . " - Registrado";
        } else {
            $titulo_pagina = "Productos por Marca Desconocida - Registrado";
        }
    } else {
        header("Location: /TFG/index.php?action=catalogo");
        exit();
    }

    $marcas = $marca_model->getAllMarcas();
    require_once 'vista/catalogo.php';
}

/**
 * Busca productos por un término dado y muestra los resultados para usuarios registrados.
 */
function buscarProductosRegistrado() {
    global $producto_model, $marca_model, $nombre_usuario_logueado;

    if (isset($_GET['busqueda']) && !empty($_GET['busqueda'])) {
        $termino_busqueda = trim($_GET['busqueda']);
        $productos = $producto_model->buscarProductos($termino_busqueda);
        $titulo_pagina = "Resultados para: " . htmlspecialchars($termino_busqueda) . " - Registrado";
    } else {
        header("Location: /TFG/index.php?action=catalogo");
        exit();
    }

    $marcas = $marca_model->getAllMarcas();
    require_once 'vista/catalogo.php';
}

```



```

/**
 * Muestra los detalles de un producto específico para usuarios registrados.
 */
function verDetalleProductoRegistrado() {
    global $producto_model, $marca_model, $nombre_usuario_logueado;

    if (isset($_GET['id_producto']) && is_numeric($_GET['id_producto'])) {
        $id_producto = (int)$_GET['id_producto'];
        $producto_detalle = $producto_model->getProductoById($id_producto);

        if ($producto_detalle) {
            $titulo_pagina = htmlspecialchars($producto_detalle[1]) . " - Detalles";
            require_once 'vista/producto_detalle.php'; // Reutilizamos la misma vista
        } else {
            header("Location: /TFG/index.php?action=catalogo&error=producto_no_encontrado");
            exit();
        }
    } else {
        header("Location: /TFG/index.php?action=catalogo");
        exit();
    }
}

/**
 * Añade un producto al carrito para el usuario logueado.
 * Esta función es llamada desde noRegistrado.php cuando el usuario YA está logueado.
 */
function anadirACarritoRegistrado() {
    global $carrito_model, $id_usuario_logueado;

    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        $id_producto = $_POST['id_producto'] ?? null;
        $cantidad = $_POST['cantidad'] ?? 1;

        // Validación robusta
        if (!$id_producto || !is_numeric($id_producto)) {
            header("Location: /TFG/index.php?action=catalogo&error=producto_invalido");
            exit();
        }

        // Llama al modelo de Carrito para añadir el producto
        $carrito_model->añadirProducto($id_usuario_logueado, $id_producto, $cantidad);

        header("Location: /TFG/index.php?action=ver_carrito&success=producto_anadido");
        exit();
    } else {
        header("Location: /TFG/index.php?action=catalogo");
        exit();
    }
}

```

```

/**
 * Muestra el contenido del carrito de compras del usuario logueado.
 */
function verCarrito() {
    global $carrito_model, $id_usuario_logueado, $nombre_usuario_logueado;

    // CORRECCIÓN: Usar getContenidoCarrito
    $items_carrito = $carrito_model->getContenidoCarrito($id_usuario_logueado);
    $titulo_pagina = "Tu Carrito de Compras, " . htmlspecialchars($nombre_usuario_logueado);

    // Cargar la vista del carrito.
    // **IMPORTANTE:** Necesitarás crear este archivo: vista/carrito.php
    require_once 'vista/carro.php';
}

/**
 * Actualiza la cantidad de un producto en el carrito.
 */
function actualizarCantidadCarrito() {
    global $carrito_model, $id_usuario_logueado;

    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        $id_producto = $_POST['id_producto'] ?? null;
        $nueva_cantidad = $_POST['nueva_cantidad'] ?? null;

        if ($id_producto && is_numeric($id_producto) && is_numeric($nueva_cantidad) && $nueva_cantidad >= 0) {
            if ($nueva_cantidad == 0) {
                $carrito_model->eliminarProducto($id_usuario_logueado, (int)$id_producto);
            } else {
                $carrito_model->actualizarCantidadProducto($id_usuario_logueado, (int)$id_producto, (int)$nueva_cantidad);
            }
            header("Location: /TFG/index.php?action=ver_carrito");
            exit();
        }
    }
    header("Location: /TFG/index.php?action=ver_carrito&error=error_actualizar_cantidad");
    exit();
}

/**
 * Elimina un producto específico del carrito.
 */
function eliminarProductoCarrito() {
    global $carrito_model, $id_usuario_logueado;

    if (isset($_GET['id_producto']) && is_numeric($_GET['id_producto'])) {
        $id_producto = (int)$_GET['id_producto'];
        $carrito_model->eliminarProducto($id_usuario_logueado, $id_producto);
        header("Location: /TFG/index.php?action=ver_carrito&mensaje=producto_eliminado");
        exit();
    }
    header("Location: /TFG/index.php?action=ver_carrito&error=error_eliminar_producto");
    exit();
}

```

NoRegistrado.php

Función: Controlador para usuarios no registrados (catálogo, login, registro).

Funciones principales:

mostrarCatalogo: Muestra el catálogo general.

filtrarProductos: Filtra productos por marca.

buscarProductos: Busca productos.

verDetalleProducto: Muestra detalles de un producto.

mostrarLogin: Muestra el formulario de login.

procesarLogin: Procesa el login.

procesarRegistro: Procesa el registro.

anadirACarritoNoRegistrado: Redirige al login si intenta añadir al carrito.

```

function mostrarCatalogo() {
    global $producto_model, $marca_model;

    $productos = $producto_model->getAllProductos();
    $marcas = $marca_model->getAllMarcas();
    $titulo_pagina = "Nuestro Catálogo de Juguetes";

    require_once __DIR__ . '/../vista/catalogo.php';
}

/**
 * Filtra los productos por una marca específica y muestra los resultados.
 */
function filtrarProductos() {
    global $producto_model, $marca_model;

    if (isset($_GET['id_marca']) && is_numeric($_GET['id_marca'])) {
        $id_marca = (int)$_GET['id_marca'];
        $productos = $producto_model->getProductosByMarca($id_marca);

        $marca_info = $marca_model->getMarcaById($id_marca);
        if ($marca_info) {
            $titulo_pagina = "Productos de " . htmlspecialchars($marca_info[1]);
        } else {
            $titulo_pagina = "Productos por Marca Desconocida";
        }
    } else {
        header("Location: index.php?action=catalogo");
        exit();
    }

    $marcas = $marca_model->getAllMarcas();
    require_once __DIR__ . '/../vista/catalogo.php';
}

/**
 * Busca productos por un término dado y muestra los resultados.
 */
function buscarProductos() {
    global $producto_model, $marca_model;

    if (isset($_GET['busqueda']) && !empty($_GET['busqueda'])) {
        $termino_busqueda = trim($_GET['busqueda']);
        $productos = $producto_model->buscarProductos($termino_busqueda);
        $titulo_pagina = "Resultados para: '" . htmlspecialchars($termino_busqueda) . "'";
    } else {
        header("Location: index.php?action=catalogo");
        exit();
    }

    $marcas = $marca_model->getAllMarcas();
    require_once __DIR__ . '/../vista/catalogo.php';
}

```

```

function mostrarLogin() {
    global $titulo_pagina;
    $titulo_pagina = "Iniciar Sesión";
    require_once __DIR__ . '/../vista/login.php';
}

/**
 * Procesa el formulario de login.
 */
function procesarLogin() {
    global $usuario_model;

    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        $email = $_POST['email'] ?? '';
        $password = $_POST['password'] ?? '';
        $recuerdame = isset($_POST['recuerdame']);

        if (empty($email) || empty($password)) {
            header("Location: index.php?action=login&error=campos_vacios");
            exit();
        }

        $usuario_login = $usuario_model->verificarLogin($email, $password);

        if ($usuario_login) {
            $id_usu = $usuario_login[0];
            $nom_us = $usuario_login[1];
            $tipo_us = $usuario_login[2];

            set_session("id_usu", $id_usu, "nom_us", $nom_us, "tipo_us", $tipo_us);

            if ($recuerdame) {
                set_cookie("usuario", $email);
            } else {
                unset_cookie("usuario");
            }

            // AHORA SE COMPRUEBA EL TIPO DE USUARIO 'administrador'
            if ($tipo_us === 'administrador') {
                header("Location: index.php?action=administracion");
                exit();
            } elseif ($tipo_us === 'registrado') {
                header("Location: index.php?action=catalogo_registrado");
                exit();
            } else {
                // Este caso debería ser raro si los tipos de usuario están controlados.
                header("Location: index.php?action=login&error=tipo_usuario_invalido");
                exit();
            }
        } else {
            header("Location: index.php?action=login&error=credenciales_invalidas");
            exit();
        }
    } else {
        header("Location: index.php?action=login");
        exit();
    }
}

```

```

function procesarRegistro() {
    global $usuario_model;

    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        $nombre = $_POST['nombre'] ?? '';
        $apellidos = $_POST['apellidos'] ?? '';
        $email = $_POST['email'] ?? '';
        $password = $_POST['password'] ?? '';
        $confirm_password = $_POST['confirm_password'] ?? '';

        if (empty($nombre) || empty($apellidos) || empty($email) || empty($password) || empty($confirm_password)) {
            header("Location: index.php?action=registro&error=campos_vacios");
            exit();
        }

        if ($password !== $confirm_password) {
            header("Location: index.php?action=registro&error=passwords_no_coinciden");
            exit();
        }

        // Por defecto, se registra como 'registrado'. No 'admin' desde aquí.
        if ($usuario_model->registrarUsuario($nombre, $apellidos, $email, $password, 'registrado')) {
            header("Location: index.php?action=login&registro_exitoso=true");
            exit();
        } else {
            header("Location: index.php?action=registro&error=registro_fallido");
            exit();
        }
    } else {
        header("Location: index.php?action=registro");
        exit();
    }
}

/**
 * Maneja la acción de añadir un producto al carrito para usuarios NO REGISTRADOS.
 * Redirige al login con un mensaje si el usuario no está logueado.
 */
function anadirACarritoNoRegistrado() {
    // Siempre redirige al login si no está logueado
    header("Location: index.php?action=login&error=necesitas_login_carrito");
    exit();
}

```

Producto.php

Función: Modelo para la gestión de productos.

Funciones principales:

insertarProducto: Inserta un nuevo producto.

getAllProductos: Obtiene todos los productos.

actualizarProducto: Actualiza un producto.

eliminarProducto: Elimina un producto.

actualizarCantidad: Cambia el stock de un producto.

```

public function insertarProducto($titulo, $precio, $imagen_producto, $descripcion, $id_marca, $cantidad) {
    $sentencia = "INSERT INTO producto (titulo, precio, imagen_producto, descripcion, id_marca, cantidad) VALUES (?, ?, ?, ?, ?, ?)";

    $conn = $this->conn->getConnection();
    $consulta = $conn->prepare($sentencia);

    if ($consulta === false) {
        error_log("Error al preparar la consulta de inserción de producto: " . $conn->error);
        return false;
    }

    // 'sdssii' -> s: string (titulo), d: double (precio), s: string (imagen), s: string (descripcion), i: integer (id_marca), i: integer (cantidad)
    $consulta->bind_param("sdssii", $titulo, $precio, $imagen_producto, $descripcion, $id_marca, $cantidad);
    $result = $consulta->execute(); // Capturamos el resultado de execute()

    $insertado = false;
    if ($result && $consulta->affected_rows === 1) { // Comprobamos que la ejecución fue exitosa Y que se insertó 1 fila
        $insertado = true;
    } else if ($result === false) {
        error_log("Error al ejecutar la consulta de inserción de producto: " . $consulta->error);
    }

    $consulta->close();
    return $insertado;
}

```

```

public function getAllProductos() {
    // SQL para seleccionar todos los productos. Se une con la tabla 'marca' para obtener el nombre de la marca.
    $sentencia = "SELECT p.id_producto, p.titulo, p.precio, p.imagen_producto, p.descripcion, p.id_marca, p.cantidad, m.nombre AS nombre_marca
    FROM producto p JOIN marca m ON p.id_marca = m.id_marca"; // Usamos JOIN explícito para claridad

    $conn = $this->conn->getConnection(); // Obtener la conexión
    $consulta = $conn->prepare($sentencia);

    if ($consulta === false) {
        error_log("Error al preparar la consulta de obtener todos los productos: " . $conn->error);
        return [];
    }

    $result_execute = $consulta->execute(); // Capturamos el resultado de execute()

    if ($result_execute === false) {
        error_log("Error al ejecutar la consulta de obtener todos los productos: " . $consulta->error);
        return [];
    }

    $id_producto = "";
    $titulo = "";
    $precio = "";
    $imagen_producto = "";
    $descripcion = "";
    $id_marca = "";
    $cantidad = "";
    $nombre_marca = ""; // Variable para el nombre de la marca

    // El orden de las variables en bind_result DEBE COINCIDIR exactamente con el orden de las columnas en el SELECT.
    $consulta->bind_result($id_producto, $titulo, $precio, $imagen_producto, $descripcion, $id_marca, $cantidad, $nombre_marca);

    $productos = [];
    while ($consulta->fetch()) {
        // Se devuelve un array numérico para cada producto, según tu estilo.
        $productos[$id_producto] = [$id_producto, $titulo, $precio, $imagen_producto, $descripcion, $id_marca, $cantidad, $nombre_marca];
    }

    $consulta->close();
    return $productos;
}

```

```

public function actualizarProducto($id_producto, $titulo, $precio, $imagen_producto, $descripcion, $id_marca, $cantidad) {
    $sentencia = "UPDATE producto SET titulo = ?, precio = ?, imagen_producto = ?, descripcion = ?, id_marca = ?, cantidad = ? WHERE id_producto = ?";

    $conn = $this->conn->getConnection();
    $consulta = $conn->prepare($sentencia);

    if ($consulta === false) {
        error_log("Error al preparar la consulta de actualización de producto: " . $conn->error);
        return false;
    }

    $consulta->bind_param("sdssiii", $titulo, $precio, $imagen_producto, $descripcion, $id_marca, $cantidad, $id_producto);
    $result = $consulta->execute();

    $modificado = false;
    if ($result && $consulta->affected_rows === 1) {
        $modificado = true;
    } else if ($result === false) {
        error_log("Error al ejecutar la consulta de actualización de producto: " . $consulta->error);
    }

    $consulta->close();
    return $modificado;
}

/**
 * Elimina un producto de la base de datos.
 *
 * @param int $id_producto ID del producto a eliminar.
 * @return bool True si el producto fue eliminado con éxito, false en caso contrario.
 */
public function eliminarProducto($id_producto) {
    $sentencia = "DELETE FROM producto WHERE id_producto = ?";

    $conn = $this->conn->getConnection();
    $consulta = $conn->prepare($sentencia);

    if ($consulta === false) {
        error_log("Error al preparar la consulta de eliminación de producto: " . $conn->error);
        return false;
    }

    $consulta->bind_param("i", $id_producto);
    $result = $consulta->execute();

    $eliminado = false;
    if ($result && $consulta->affected_rows === 1) {
        $eliminado = true;
    } else if ($result === false) {
        error_log("Error al ejecutar la consulta de eliminación de producto: " . $consulta->error);
    }

    $consulta->close();
    return $eliminado;
}

```



```

public function actualizarProducto($id_producto, $titulo, $precio, $imagen_producto, $descripcion, $id_marca, $cantidad) {
    $sentencia = "UPDATE producto SET titulo = ?, precio = ?, imagen_producto = ?, descripcion = ?, id_marca = ?, cantidad = ? WHERE id_producto = ?";

    $conn = $this->conn->getConnection();
    $consulta = $conn->prepare($sentencia);

    if ($consulta === false) {
        error_log("Error al preparar la consulta de actualización de producto: " . $conn->error);
        return false;
    }

    $consulta->bind_param("sdssiii", $titulo, $precio, $imagen_producto, $descripcion, $id_marca, $cantidad, $id_producto);
    $result = $consulta->execute();

    $modificado = false;
    if ($result && $consulta->affected_rows === 1) {
        $modificado = true;
    } else if ($result === false) {
        error_log("Error al ejecutar la consulta de actualización de producto: " . $consulta->error);
    }

    $consulta->close();
    return $modificado;
}

/**
 * Elimina un producto de la base de datos.
 *
 * @param int $id_producto ID del producto a eliminar.
 * @return bool True si el producto fue eliminado con éxito, false en caso contrario.
 */
public function eliminarProducto($id_producto) {
    $sentencia = "DELETE FROM producto WHERE id_producto = ?";

    $conn = $this->conn->getConnection();
    $consulta = $conn->prepare($sentencia);

    if ($consulta === false) {
        error_log("Error al preparar la consulta de eliminación de producto: " . $conn->error);
        return false;
    }

    $consulta->bind_param("i", $id_producto);
    $result = $consulta->execute();

    $eliminado = false;
    if ($result && $consulta->affected_rows === 1) {
        $eliminado = true;
    } else if ($result === false) {
        error_log("Error al ejecutar la consulta de eliminación de producto: " . $consulta->error);
    }

    $consulta->close();
    return $eliminado;
}

```

carrito.php

Función: Modelo para la gestión del carrito de compras.

Funciones principales:

getContenidoCarrito: Obtiene todos los productos del carrito.

vaciarCarrito: Vacía el carrito.

```

public function getContenidoCarrito($id_usu) {
    // **CORRECCIÓN:** Cambiado el nombre de la tabla de 'carrito_items' a 'carrito'.
    // **CORRECCIÓN:** Se obtiene p.precio directamente de la tabla 'producto' para el cálculo del subtotal.
    $sentencia = "SELECT c.id_producto, p.titulo, p.imagen_producto, c.cantidad, p.precio precio_actual, (c.cantidad * p.precio) AS subtotal
    FROM carrito c, producto p WHERE c.id_producto = p.id_producto
    AND c.id_usu = ?";

    $consulta = $this->conn->getConnection()->prepare($sentencia);

    $consulta->bind_param("i", $id_usu); // 'i' para integer (id_usu)
    $consulta->execute();

    // Variables locales donde meter los valores al introducirse en el while
    $id_producto_local = "";
    $titulo_local = "";
    $imagen_producto_local = "";
    $cantidad_local = "";
    $precio_actual_local = ""; // **CORRECCIÓN:** Variable para el precio actual del producto
    $subtotal_local = "";

    // **CORRECCIÓN:** Ajustado bind_result para los nuevos campos.
    $consulta->bind_result($id_producto_local, $titulo_local, $imagen_producto_local, $cantidad_local, $precio_actual_local, $subtotal_local);

    $items_carrito = [];
    while ($consulta->fetch()) {
        // Se mantiene el array numérico interno según tu preferencia.
        $items_carrito[$id_producto_local] = [
            $id_producto_local,
            $titulo_local,
            $imagen_producto_local,
            $cantidad_local,
            $precio_actual_local, // **CORRECCIÓN:** Incluir el precio actual del producto
            $subtotal_local
        ];
    }

    $consulta->close();
    return $items_carrito;
}

/**

```

```

public function vaciarCarrito($id_usu) {

    $sentencia = "DELETE FROM carrito WHERE id_usu = ?";

    $consulta = $this->conn->getConnection()->prepare($sentencia);

    $consulta->bind_param("i", $id_usu); // 'i' para entero
    $vaciar = $consulta->execute();

    // No comprobamos affected_rows === 1 porque puede vaciar 0 o N elemento
    // Solo nos importa que la ejecución no haya tenido errores.
    $vaciado = false;
    if ($vaciar) {
        $vaciado = true;
    }

    $consulta->close();
    return $vaciado;
}

>

```

usuario.php

Función: Modelo para la gestión de usuarios.

Funciones principales:

registrarUsuario: Registra un nuevo usuario.

verificarLogin: Verifica las credenciales de login.

```
public function registrarUsuario($nombre, $apellidos, $correo, $password, $tipo_usu = 'registrado') {
    //Hashear la contraseña antes de guardarla en la base de datos por seguridad.**
    $cifrada = password_hash($password, PASSWORD_DEFAULT);

    // Verifico si el correo ya existe antes de insertar.
    // Uso getUsuarioByEmail para la verificación.
    if ($this->getUsuarioByEmail($correo) !== null) {
        error_log("Error: Intento de registrar usuario con correo ya existente: " . $correo);
        return false; // El correo ya existe
    }

    // SQL para insertar un nuevo usuario
    $sentencia = "INSERT INTO usuarios (nombre, apellidos, correo, psw, tipo_usu) VALUES (?, ?, ?, ?, ?)";

    $consulta = $this->conn->getConnection()->prepare($sentencia);

    // Manejo de errores en la preparación de la consulta.**
    if ($consulta === false) {
        error_log("Error al preparar la consulta de registro de usuario" );
        return false;
    }

    // Uso $cifrada para la inserción de la contraseña.
    $consulta->bind_param("sssss", $nombre, $apellidos, $correo, $cifrada, $tipo_usu);
    $result = $consulta->execute();

    $insertado = false;
    if ($result && $consulta->affected_rows === 1) {
        $insertado = true;
    } else if ($result === false) {
        error_log("Error al ejecutar la consulta de registro de usuario: " . $consulta->error);
    }

    $consulta->close();
    return $insertado;
}

public function verificarLogin($correo, $password) {
    // Obtenemos todos los datos del usuario por su correo, incluyendo la contraseña hasheada.
    // getUsuarioByEmail devuelve un array numérico con [id_usu, nombre, apellidos, correo, psw, tipo_usu]
    $usuario_datos = $this->getUsuarioByEmail($correo);

    if ($usuario_datos === null) {
        return null; // Usuario no encontrado por correo
    }

    // Extraer la contraseña hasheada de la base de datos (posición 4 del array)
    $cifrada_bd = $usuario_datos[4];

    //Verifico la contraseña hasheada con password_verify().**
    if (password_verify($password, $cifrada_bd)) {
        // Credenciales correctas, devolvemos los datos necesarios para la sesión
        // [id_usu, nombre, tipo_usu]
        return [$usuario_datos[0], $usuario_datos[1], $usuario_datos[5]]; // id_usu, nombre, tipo_usu
    } else {
        return null; // Contraseña incorrecta
    }
}
```

En cuanto al uso de AJAX lo he usado solo para poder ocultar o ver la contraseña a la hora del registro