{desafío} latam_ Academia de talentos digitales

Hito I
Diseño y prototipo

Integrantes: Abel Rivas.
Constanza Pérez.
Juan Pablo Mendez.

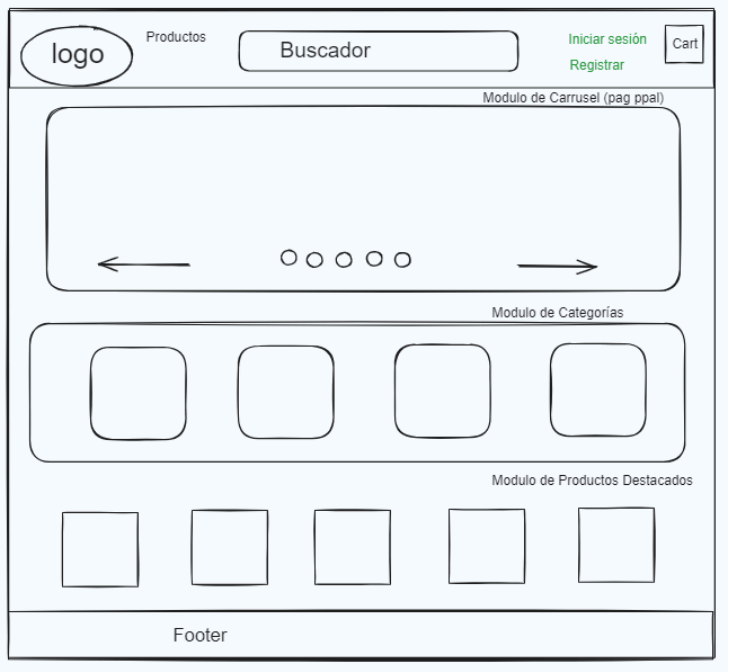
Curso: Desarrollo Full Stack. G63

Tabla de contenido

1. Diseño de la interfaz gráfica.

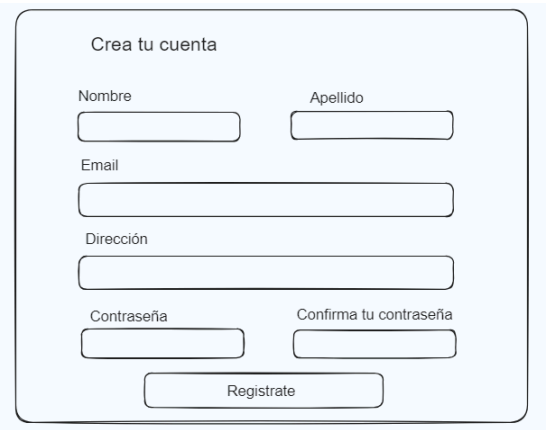
Landing page (Inicio):

- Barra de navegación con accesos directos y barra de búsqueda.
- Vista automática tipo carrusel con productos seleccionados.
- Cuadrícula de productos por categoría y destacados.
- Footer con detalle de empresa y derechos de autor.



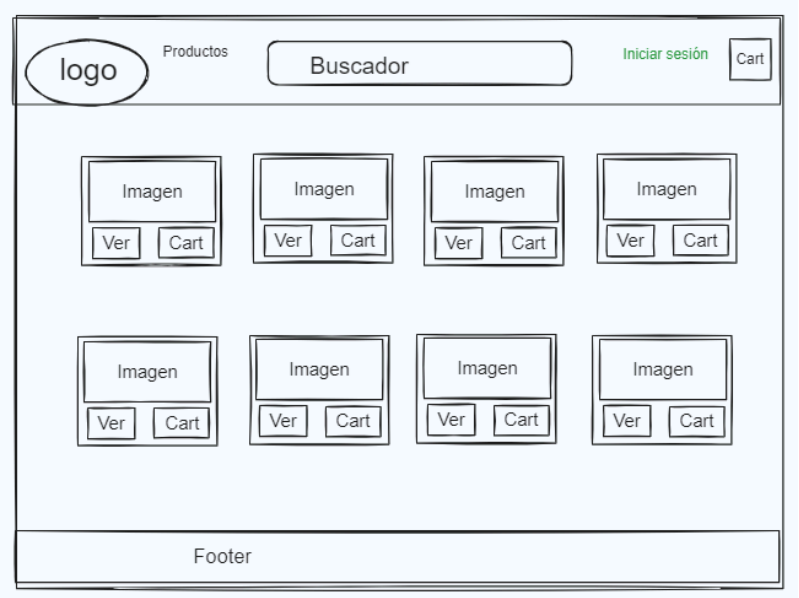
Registro e inicio de sesión (modal):

- Formulario tipo modal con campos necesarios para registrar.
- Selección de casillas de términos y condiciones.
- Botones de acción.



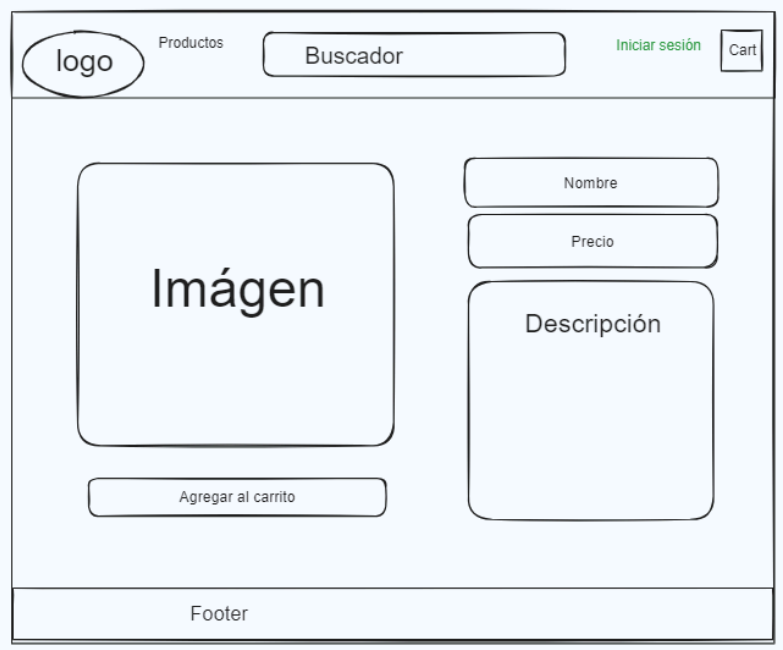
Productos y detalle de productos:

- Cuadrícula de productos, ordenados según definición
- Productos se muestran en cards con botones de opción de ver detalle y agregar al carro de compras
- Barra de búsqueda tiene la funcionalidad de filtrar los productos que coincidan con el valor ingresado



Vista detalle:

- Muestra mayor información del producto consultado, con sus imágenes correspondientes y su descripción.
- También contiene el botón a
- Barra de búsqueda tiene la funcionalidad de filtrar los productos que coincidan con el valor ingresado



Vista de mi perfil (privada):

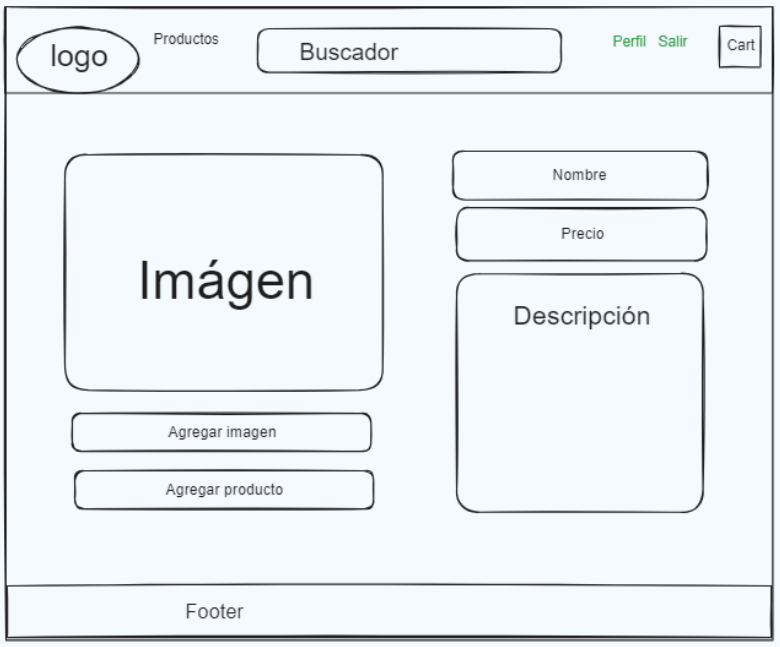
Ofrece la posibilidad de mostrar y actualizar los datos del usuario con sesión iniciada.

- Formulario con datos del usuario.
- Botones con funcionalidad de actualizar datos.

Vista crear producto (privada)

Permite agregar productos a la base de datos para ser utilizado por toda la aplicación.

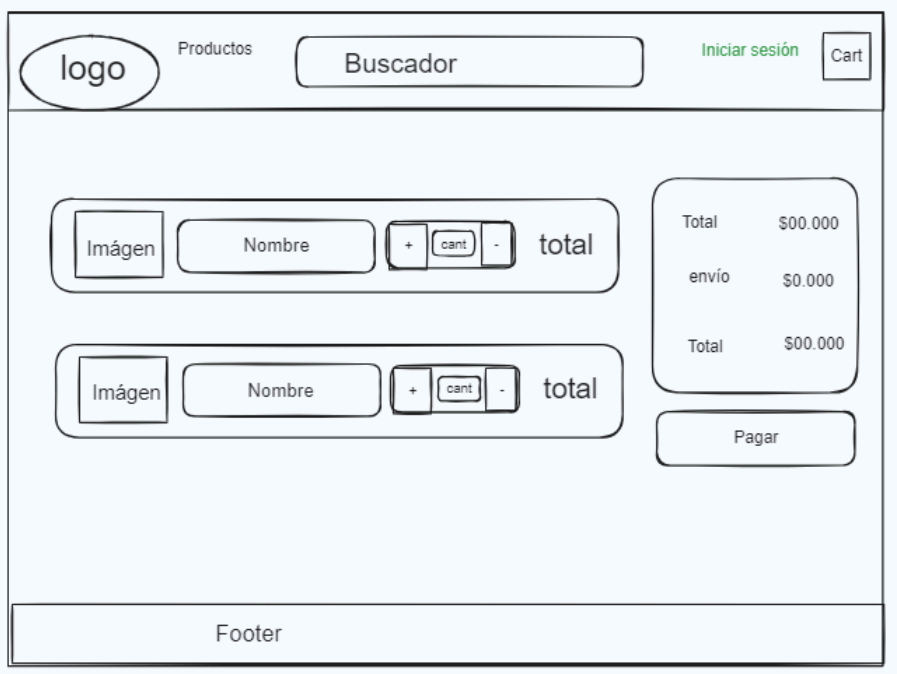
- Formulario con datos de producto
- Botón guardar para confirmar creación.



Carro de compras:

Permite ver los productos seleccionados para la compra, además agregar o quitar unidades del productos.

- Lista de productos agregados.
- Botones de modificación de cantidades.
- Botón de confirmar compra.

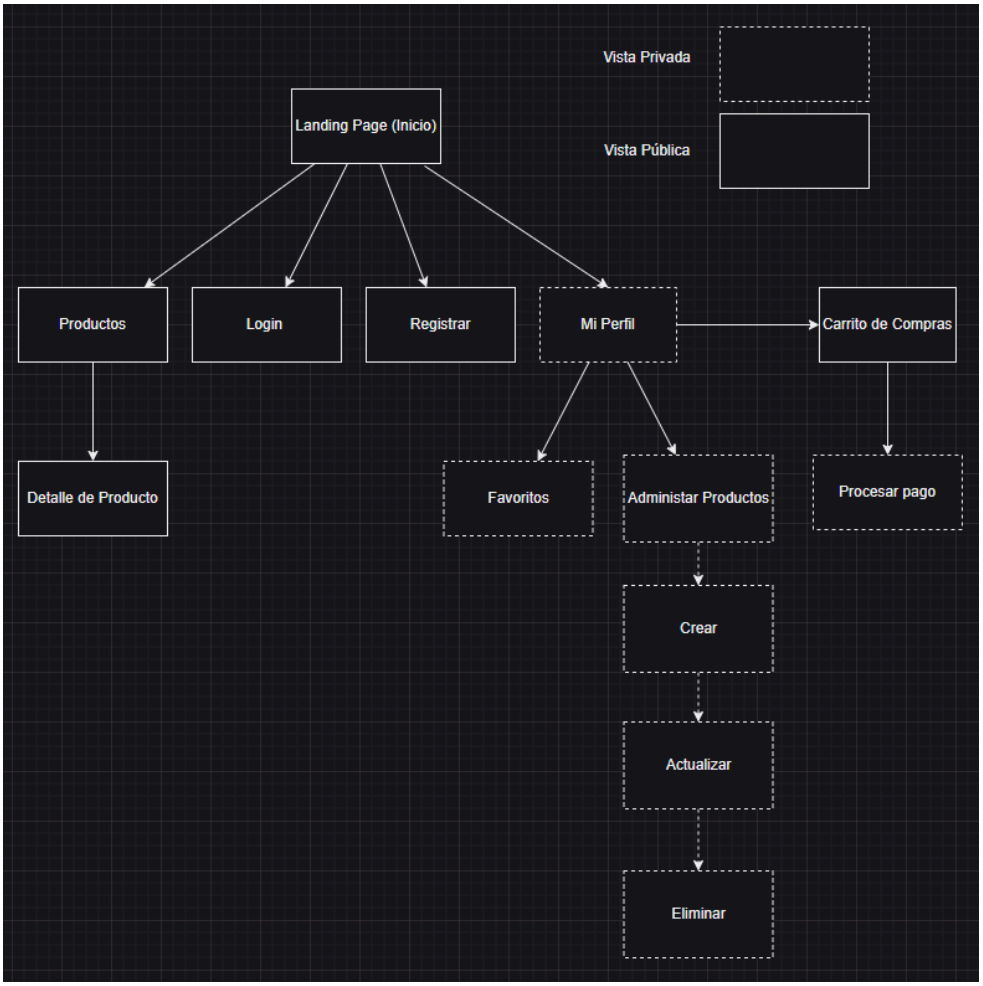


2. Definición de navegación entre vistas.

A continuación, una breve descripción de la navegación entre vistas para el desarrollo de nuestro e-commerce:

- **Landing Page**: Página de inicio que sirve como punto de entrada al e-commerce.
- **Productos**: Vista pública donde los usuarios pueden ver el catálogo de productos. Desde aquí, pueden acceder al **Detalle de Producto** para ver información específica de un producto.
- **Login**: Vista pública para que los usuarios inicien sesión.
- **Registrar**: Vista pública donde los nuevos usuarios pueden crear una cuenta.
- **Mi Perfil**: Vista privada disponible para usuarios autenticados. Aquí pueden gestionar su perfil, acceder a sus productos **favoritos** y **Administrar Productos** (si es administrador), que incluye opciones para **Crear**, **Actualizar** y **Eliminar** productos.
- **Carrito de Compras**: Vista pública que muestra los productos que el usuario ha añadido para la compra. Desde aquí, pueden **procesar Pago** para completar la transacción.

Cada una de estas vistas se conecta de manera lógica para facilitar la experiencia del usuario, permitiendo la navegación fluida entre los productos, gestión de perfil, y el proceso de compra.



3. Listado de dependencias.

Fronted:

```
"dependencies": {
  "@fortawesome/fontawesome-free": "^6.6.0",
  "axios": "^1.3.1",
  "bootstrap": "^5.3.3",
  "mdb-react-ui-kit": "^8.0.0",
  "react": "^18.2.0",
  "react-bootstrap": "^2.10.4",
  "react-dom": "^18.2.0",
  "react-router-dom": "^6.26.0"
},
"devDependencies": {
  "@types/react": "^18.2.56",
  "@types/react-dom": "^18.2.19",
  "@vitejs/plugin-react": "^4.2.1",
  "eslint": "^8.56.0",
  "eslint-plugin-react": "^7.33.2",
  "eslint-plugin-react-hooks": "^4.6.0",
  "eslint-plugin-react-refresh": "^0.4.5",
  "standard": "^17.1.0",
  "vite": "^5.1.4"
}
```

Backend:

```
"dependencies": {
  "bcrypt": "^5.1.1",
  "cors": "^2.8.5",
  "dotenv": "^16.4.5",
  "express": "^4.19.2",
  "jsonwebtoken": "^9.0.2",
  "morgan": "^1.10.0",
  "pg": "^8.12.0",
  "pg-format": "^1.0.4"
},
"devDependencies": {
  "nodemon": "^3.1.4",
  "standard": "^17.1.0"
}
```

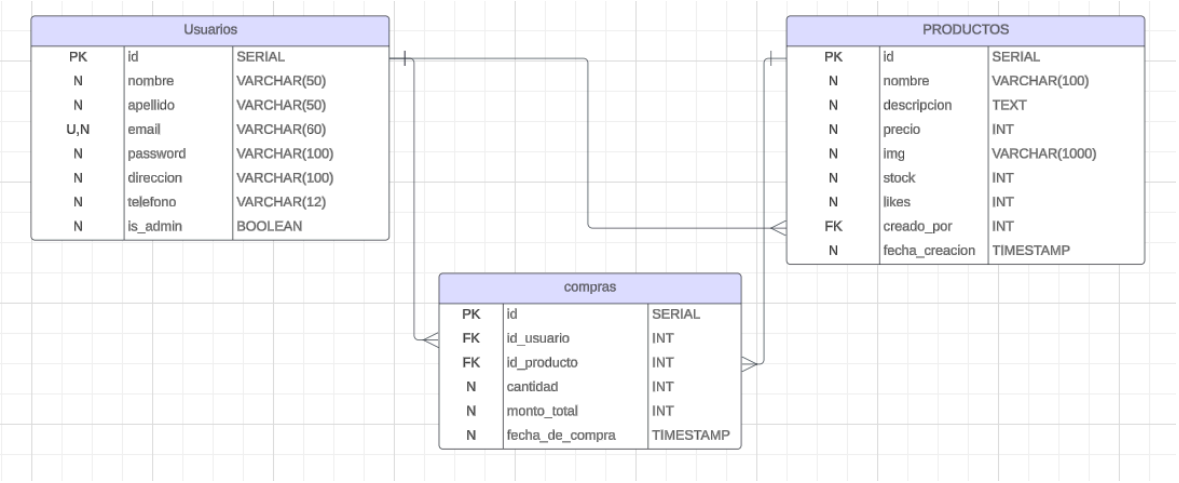
4. Diseño de la base de datos y sus relaciones.

A continuación, se presenta la base de datos que ocuparemos para la gestión de los datos de usuarios, productos y compras.

La tabla usuarios, almacena toda la información de registro de los usuarios de la tienda. Además se hace la diferenciación de un usuario normal con un usuario de tipo administrador con la fila is_admin siendo por defecto un booleano “false”, para que solo los administradores puedan ingresar los productos.

En la tabla productos, se almacena toda la información de todos los productos disponibles en la tienda con relación a el usuario que lo creó.

En la tabla compras, se registran todas las compras realizadas por los usuarios, incluyendo la cantidad de productos y el monto de la compra.



Relaciones

- Usuarios y Productos: Un usuario puede crear múltiples productos, lo que se refleja en la columna creado_por de la tabla productos.
- Usuarios y Compras: Un usuario puede realizar múltiples compras, lo que se refleja en la columna id_usuario de la tabla compras.
- Productos y Compras: Un producto puede ser comprado múltiples veces, lo que se refleja en la columna id_producto de la tabla compras.

5. Diseño del contrato de datos de la API rest.

1. Landing Page

Descripción: Devuelve información general de la tienda, incluyendo promociones o productos destacados.

- **Método HTTP:** GET
- **Endpoint:** /
- **Autenticación:** No requerida

Respuesta Exitosa (200 OK):

```
// GET /
response:
  payload: {
    "nombre": "Producto 1", // String
    "img": "url_imagen_1" // String
}
```

Respuestas de Error:

- 500 Internal Server Error: "Error interno del servidor."

2. Productos

2.1 Obtener todos los productos

Descripción: Devuelve una lista de todos los productos disponibles.

- **Método HTTP:** GET
- **Endpoint:** /productos
- **Autenticación:** No requerida

Respuesta Exitosa (200 OK):

```
// GET /productos
response:
  payload: {
    "id": 1, // Int
    "nombre": "Producto 1", // String
    "descripcion": "Descripción del producto 1", // String
    "precio": 100, // Int
    "img": "url_imagen_1", // String
    "stock": 20, // Int
    "likes": "Lídes del producto 1" // Int
}
```

Respuestas de Error:

- 500 Internal Server Error: "Error interno del servidor."

2.2 Obtener detalle de producto

Descripción: Devuelve la información detallada de un producto específico.

- **Método HTTP:** GET
- **Endpoint:** /productos/{id}
- **Autenticación:** No requerida

Cuerpo de la Solicitud (Request Body):

```
// GET /producto/{id}
request:
  payload:{
    "id": "1" // Int
  }
```

Respuesta Exitosa (200 OK):

```
// GET /productos/{id}
response:
  payload: {
    "id": 1, // Int
    "nombre": "Producto 1", // String
    "descripcion": "Descripción del producto 1", // String
    "precio": 100, // Int
    "img": "url_imagen_1", // String
    "stock": 20, // Int
    "likes": "Lídes del producto 1" // Int
}
```

Respuestas de Error:

- 404 Not Found: "Producto no encontrado."
- 500 Internal Server Error: "Error interno del servidor."

3. Login /Iniciar sesión

Descripción: Permite a los usuarios autenticarse y recibir un token de acceso.

- **Método HTTP:** POST
- **Endpoint:** /login
- **Autenticación:** No requerida

Cuerpo de la Solicitud (Request Body):

```
// POST /login
request:
  payload:{
    "email": "usuario@example.com", // String
    "password": "contraseña" // String
  }
```

Respuesta Exitosa (200 OK):

```
// POST /login
response:
  payload: {
    "token": "jwt_token_aqui", // String
    "user": {
      "id": 1, // Number
      "email": "usuario@example.com", // String
    }
}
```

Respuestas de Error:

- 401 Unauthorized: "Credenciales incorrectas."
- 500 Internal Server Error: "Error interno del servidor."

4. Registrar /Crear nuevo usuario

Descripción: Permite a los nuevos usuarios crear una cuenta.

- **Método HTTP:** POST
- **Endpoint:** /registrar
- **Autenticación:** No requerida

Cuerpo de la Solicitud (Request Body):

```
// POST /registrar
request:
  payload:{
    "nombre": "Juan", // String
    "apellido": "Pérez", // String
    "email": "juan@example.com", // String
    "password": "contraseña", // String
    "direccion": "Dirección del usuario", // String
    "telefono": "123456789" // Int
  }
```

Respuesta Exitosa (201 Created):

```
// POST /registrar
response:
  payload: {
    "mensaje": "Usuario registrado con éxito." // String
  }
```

Respuestas de Error:

- 400 Bad Request: "Datos incompletos o inválidos."
- 500 Internal Server Error: "Error interno del servidor."

5. Mi Perfil

5.1 Obtener perfil de usuario

Descripción: Devuelve la información del perfil del usuario autenticado.

- **Método HTTP:** GET
- **Endpoint:** /mi-perfil
- **Autenticación:** Requerida

Cuerpo de la Solicitud (Request Body):

```
// GET /mi-perfil
request:
  Authorization: Bearer <token>
```

Respuesta Exitosa (200 OK):

```
// GET /mi-perfil
response:
  payload: {
    "id": 1, // Int
    "nombre": "Juan", // String
    "apellido": "Pérez", // String
    "email": "juan@example.com", // String
    "direccion": "Dirección del usuario", // String
    "telefono": "123456789", // Int
    "is_admin": false //Boolean
  }
```

Respuestas de Error:

- 401 Unauthorized: "Token de autenticación no válido o expirado."
- 500 Internal Server Error: "Error interno del servidor."

5.2 Actualizar perfil de usuario

Descripción: Permite al usuario actualizar su información personal.

- **Método HTTP:** PUT
- **Endpoint:** /mi-perfil
- **Autenticación:** Requerida

Cuerpo de la Solicitud (Request Body):

```
// PUT /mi-perfil
request:
  Authorization: Bearer <token>
  payload: {
    "nombre": "Juan", // String
    "apellido": "Pérez", // String
    "direccion": "Nueva dirección", // String
    "telefono": "987654321" // Int
  }
```

Respuesta Exitosa (200 OK):

```
// PUT /mi-perfil
response:
  payload: {
    "mensaje": "Perfil actualizado con éxito." // String
  }
```

Respuestas de Error:

- 400 Bad Request: "Datos incompletos o inválidos."
- 401 Unauthorized: "Token de autenticación no válido o expirado."
- 500 Internal Server Error: "Error interno del servidor."

5.3 Obtener productos favoritos

Descripción: Devuelve la lista de productos favoritos del usuario autenticado.

- **Método HTTP:** GET
- **Endpoint:** /mi-perfil/favoritos
- **Autenticación:** Requerida

Cuerpo de la Solicitud (Request Body):

```
// GET /mi-perfil/favoritos
request:
  Authorization: Bearer <token>
```

Respuesta Exitosa (200 OK):

```
// GET /mi-perfil/favoritos
response:
  payload: {
    "id": 1, // Int
    "nombre": "Producto 1", // String
    "descripcion": "Descripción del producto 1", // String
    "precio": 100, // Int
    "img": "url_imagen_1", // String
    "stock": 20, // Int
    "likes": "Lídes del producto 1" // Int
  }
```

Respuestas de Error:

- 401 Unauthorized: "Token de autenticación no válido o expirado."
- 500 Internal Server Error: "Error interno del servidor."

6. Administrar Productos (solo para administradores)

6.1 Crear producto

Descripción: Permite a los administradores agregar un nuevo producto.

- **Método HTTP:** POST
- **Endpoint:** /productos
- **Autenticación:** Requerida (Rol de administrador)

Cuerpo de la Solicitud (Request Body):

```
// POST /productos
request:
  Authorization: Bearer <token>
  payload:{
    "nombre": "Producto nuevo", // String
    "descripcion": "Descripción del nuevo producto", // String
    "precio": 150, // Int
    "img": "url_imagen_nueva", // String
    "stock": 30, // Int
    "likes": "Líkes del nuevo producto" // Int
  }
```

Respuesta Exitosa (201 Created):

```
// POST /productos
response:
  payload: {
    "mensaje": "Producto creado con éxito."
  }
```

Respuestas de Error:

- 400 Bad Request: "Datos de solicitud inválidos."
- 401 Unauthorized: "Token de autenticación no válido o expirado."
- 500 Internal Server Error: "Error interno del servidor."

6.2 Actualizar producto

Descripción: Permite a los administradores actualizar un producto existente.

- **Método HTTP**: PUT
- **Endpoint**: /productos/{id}
- **Autenticación**: Requerida

Cuerpo de la Solicitud (Request Body):

```
// PUT /productos/{id}
request:
  Authorization: Bearer <token>
  payload: {
    "nombre": "Producto actualizado", // String
    "descripcion": "Descripción actualizada", // String
    "precio": 200, // Int
    "img": "url_imagen_actualizada", // String
    "stock": 40, // Int
    "likes": "Líkes actualizados" // Int
  }
```

Respuesta Exitosa (200 OK):

```
// PUT /productos/{id}
response:
  payload: {
    "mensaje": "Producto actualizado con éxito."
  }
```

Respuestas de Error:

- 400 Bad Request: "Datos de solicitud inválidos."
- 401 Unauthorized: "Token de autenticación no válido o expirado."
- 404 Not Found: "Producto no encontrado."
- 500 Internal Server Error: "Error interno del servidor."

6.3 Eliminar producto

Descripción: Permite a los administradores eliminar un producto.

- **Método HTTP**: DELETE
- **Endpoint**: /productos/{id}
- **Autenticación**: Requerida

Cuerpo de la Solicitud (Request Body):

```
// DELETE /productos/{id}
request:
  Authorization: Bearer <token>
  payload:{
    "id": "1" // Int
  }
```

Respuesta Exitosa (200 OK):

```
// DELETE /productos/{id}
response:
  payload: {
    "mensaje": "Producto eliminado con éxito." // String
  }
```

Respuestas de Error:

- 401 Unauthorized: "Token de autenticación no válido o expirado."
- 404 Not Found: "Producto no encontrado."
- 500 Internal Server Error: "Error interno del servidor."

7. Carrito de Compras

7.1 Obtener carrito de compras

Descripción: Devuelve los productos añadidos al carrito de compras.

- **Método HTTP**: GET
- **Endpoint**: /carrito
- **Autenticación**: Requerida

Cuerpo de la Solicitud (Request Body):

```
// GET /carrito
request:
  Authorization: Bearer <token>
```

Respuesta Exitosa (200 OK):

```
// GET /carrito
response:
  payload: {
    "id": 1, // Int
    "nombre": "Producto 1", // String
    "precio": 100, // Int
    "cantidad": 2 // Int
  }
```

Respuestas de Error:

- 401 Unauthorized: "Token de autenticación no válido o expirado."
- 500 Internal Server Error: "Error interno del servidor."

7.2 Añadir producto al carrito

Descripción: Permite añadir un producto al carrito de compras.

- **Método HTTP**: POST
- **Endpoint**: /carrito
- **Autenticación**: Requerida

Cuerpo de la Solicitud (Request Body):

```
// POST /carrito
request:
  Authorization: Bearer <token>
  payload:{
    "producto_id": 1, // Int
    "cantidad": 2 // Int
  }
```

Respuesta Exitosa (201 Created):

```
// POST /carrito
response:
  payload: {
    "mensaje": "Producto añadido al carrito con éxito." // String
  }
```

Respuestas de Error:

- 400 Bad Request: "Datos de solicitud inválidos."
- 401 Unauthorized: "Token de autenticación no válido o expirado."
- 500 Internal Server Error: "Error interno del servidor."

7.3 Eliminar producto del carrito

Descripción: Permite eliminar un producto del carrito de compras.

- **Método HTTP**: DELETE
- **Endpoint**: /carrito/{producto_id}
- **Autenticación**: Requerida

Cuerpo de la Solicitud (Request Body):

```
// DELETE /carrito/{producto_id}
request:
  Authorization: Bearer <token>
  payload:{
    "id": "1" // Int
  }
```

Respuesta Exitosa (200 OK):

```
// DELETE /carrito/{producto_id}
response:
  payload: {
    "mensaje": "Producto eliminado del carrito con éxito." // String
  }
```

Respuestas de Error:

- 401 Unauthorized: "Token de autenticación no válido o expirado."
- 404 Not Found: "Producto no encontrado en el carrito."
- 500 Internal Server Error: "Error interno del servidor."

7.4 Procesar pago

Descripción: Permite procesar el pago de los productos en el carrito.

- **Método HTTP**: POST
- **Endpoint**: /pago
- **Autenticación**: Requerida

Cuerpo de la Solicitud (Request Body):

```
// POST /pago
request:
  Authorization: Bearer <token>
  payload:{
    "direccion": "Dirección de envío", // String
  }
```

Respuesta Exitosa (200 OK):

```
// POST /pago
response:
  payload: {
    "mensaje": "Pago procesado con éxito." // String
  }
```

Respuestas de Error:

- 400 Bad Request: "Datos de solicitud inválidos."
- 401 Unauthorized: "Token de autenticación no válido o expirado."
- 500 Internal Server Error: "Error interno del servidor."