

Tienda de libros

08/03/2024

—

Ricardo Barrientos

Felipe Saavedra

Presentación del Proyecto Fullstack

Introducción

¡Bienvenidos a la presentación del proyecto de la tienda de libros Fullstack! En este proyecto, hemos aplicado todas las habilidades y conocimientos adquiridos durante nuestro bootcamp de desarrollo Fullstack. Nuestra meta es crear una plataforma completa que permita a los usuarios explorar y adquirir sus libros favoritos de una manera rápida y segura.

División del Proyecto

Este proyecto se divide en dos partes principales: el frontend y el backend. Cada una de estas partes juega un papel crucial en la creación de la experiencia completa para nuestros usuarios.

1. Frontend

Utilizaremos tecnologías como React, React Router Dom y Bootstrap para desarrollar la interfaz de usuario del lado del cliente. Estas herramientas nos permitirán crear una experiencia de navegación fluida y receptiva, asegurando que nuestros usuarios puedan explorar nuestro catálogo de libros de manera fácil e intuitiva.

2. Backend

Para el desarrollo del backend, nos apoyaremos en Express, una estructura de aplicación web para Node.js, junto con otras dependencias como CORS, Dotenv y PostgreSQL. Además, implementaremos funcionalidades como autenticación de usuarios, validación de datos y seguridad con herramientas como Bcrypt.js y JsonWebToken. Para garantizar la calidad del código, realizaremos pruebas utilizando Jest y Supertest.

Gestión de la Base de Datos

La gestión de la base de datos será fundamental para el funcionamiento de nuestra tienda de libros. Utilizaremos PostgreSQL para almacenar la información de los libros, los usuarios registrados y cualquier otro dato relevante para la aplicación. Con PostgreSQL, podemos garantizar la seguridad y la integridad de nuestros datos, así como realizar consultas eficientes para ofrecer una experiencia óptima a nuestros usuarios.

Conclusiones

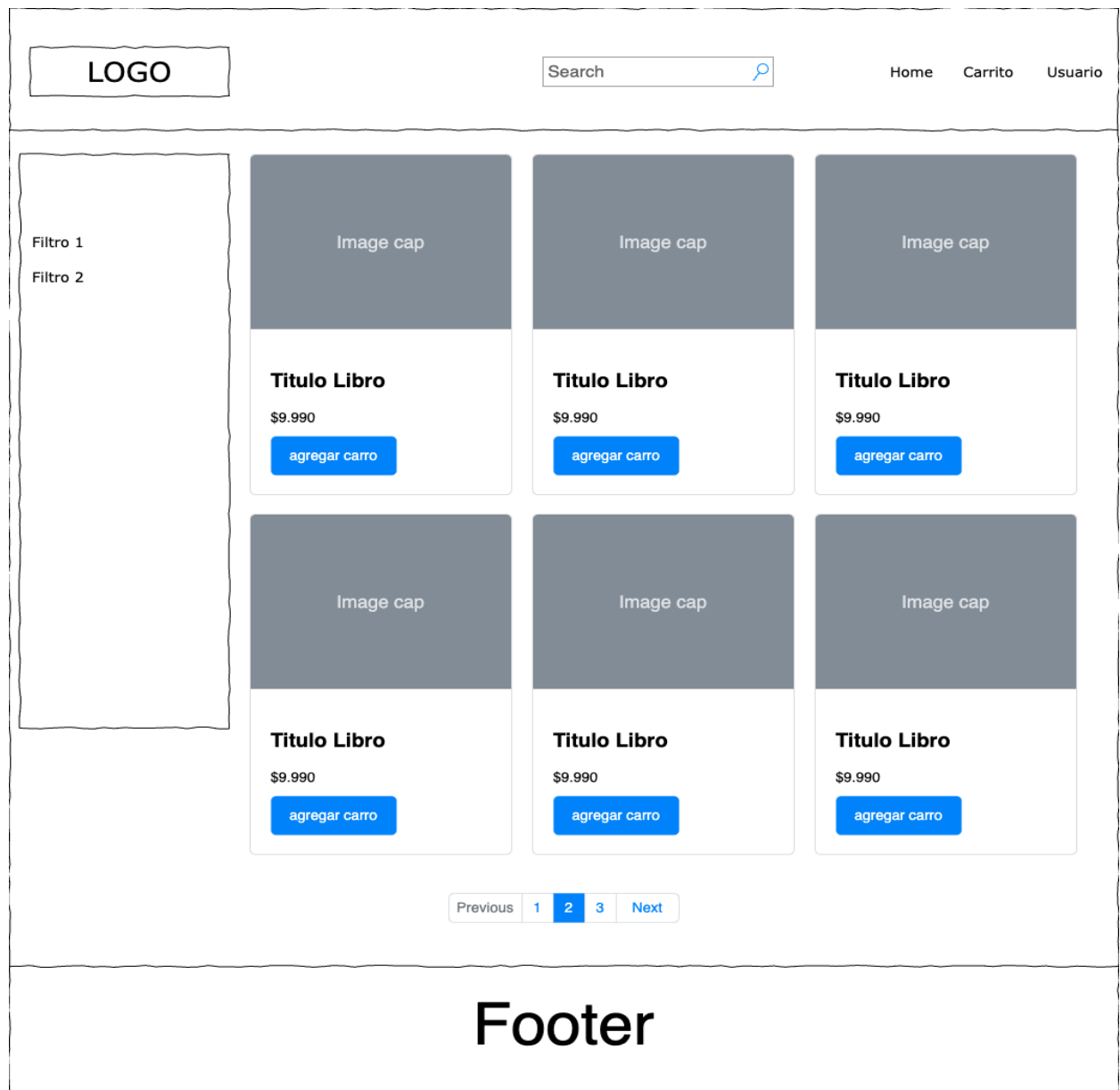
En resumen, este proyecto de tienda de libros Fullstack nos ha permitido aplicar todas las habilidades y conocimientos adquiridos durante nuestro bootcamp de desarrollo. Con un enfoque en el frontend y el backend, junto con una sólida gestión de base de datos.

I. Hito 01

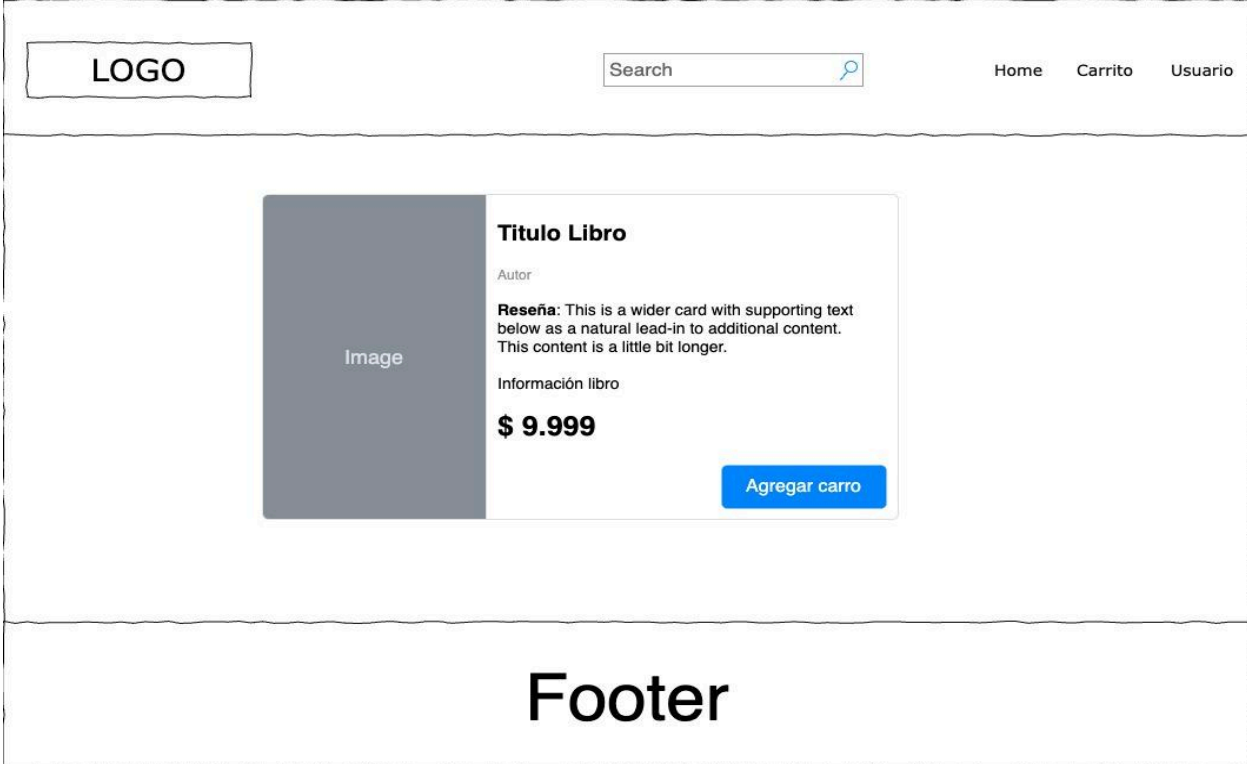
1. Diseño de interfaz gráfica

A continuación entregamos los wireframes de nuestra tienda virtual, al ser diseños conceptuales, estos podrían sufrir cambios durante la etapa de desarrollo del sitio web.

- Home

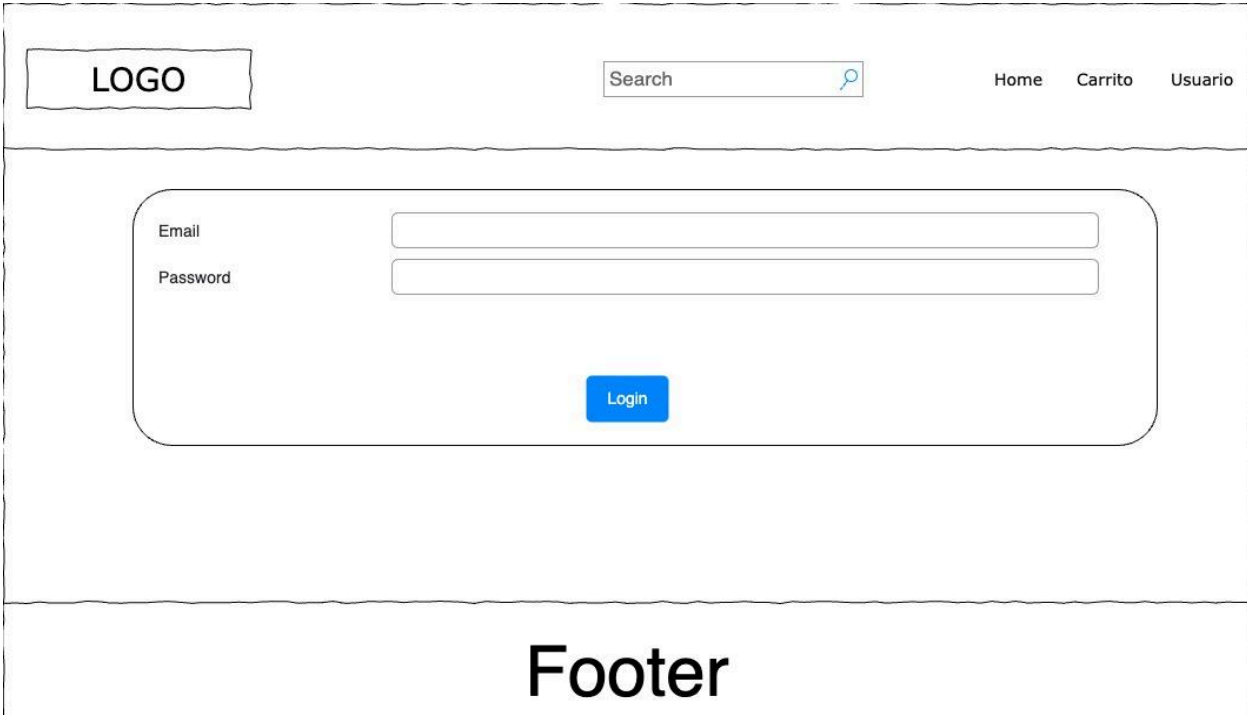


- Detalle Libro



Wireframe of a book detail page. The header contains a 'LOGO' placeholder, a search bar with a magnifying glass icon, and navigation links for 'Home', 'Carrito', and 'Usuario'. The main content area features a book card with a placeholder 'Image' on the left. To the right of the image, the text reads: 'Titulo Libro', 'Autor', 'Reseña: This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.', 'Información libro', and a price of '\$ 9.999'. A blue button labeled 'Agregar carro' is positioned at the bottom right of the card. The footer area is labeled 'Footer'.

- Login



Wireframe of a login page. The header is identical to the book detail page, featuring a 'LOGO' placeholder, a search bar, and navigation links for 'Home', 'Carrito', and 'Usuario'. The main content area contains a login form with two input fields labeled 'Email' and 'Password'. A blue button labeled 'Login' is centered below the input fields. The footer area is labeled 'Footer'.

- Registro

LOGO

Search

Home Carrito Usuario

Email

Password

Dirección

1234 Main St

Ciudad

Registrar

Footer

- Carrito

LOGO

Search

Home Carrito Usuario

64x64

Libro 1
\$ 9.999

- 1 +

64x64

Libro 2
\$ 9.999

- 1 +

64x64

Libro 3
\$ 9.999

- 1 +

Total \$9.999

Pagar

Footer

- Pago

LOGO

Search

Home Carrito Usuario

**Su orden ha sido ingresada
con el codigo ORD 9999999**

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua.

Footer

- Órdenes

LOGO

Search

Home Carrito Usuario

Ordenes

Ordenes	Fecha de compra	Total
Orden 1	dd/mm/aaaa	\$9.999
Orden 2	dd/mm/aaaa	\$9.999
Orden 3	dd/mm/aaaa	\$9.999

Footer

- CRUD

LOGO

Search

Home

Carrito

Usuario

Libros

Titulo

Autor

Reseña

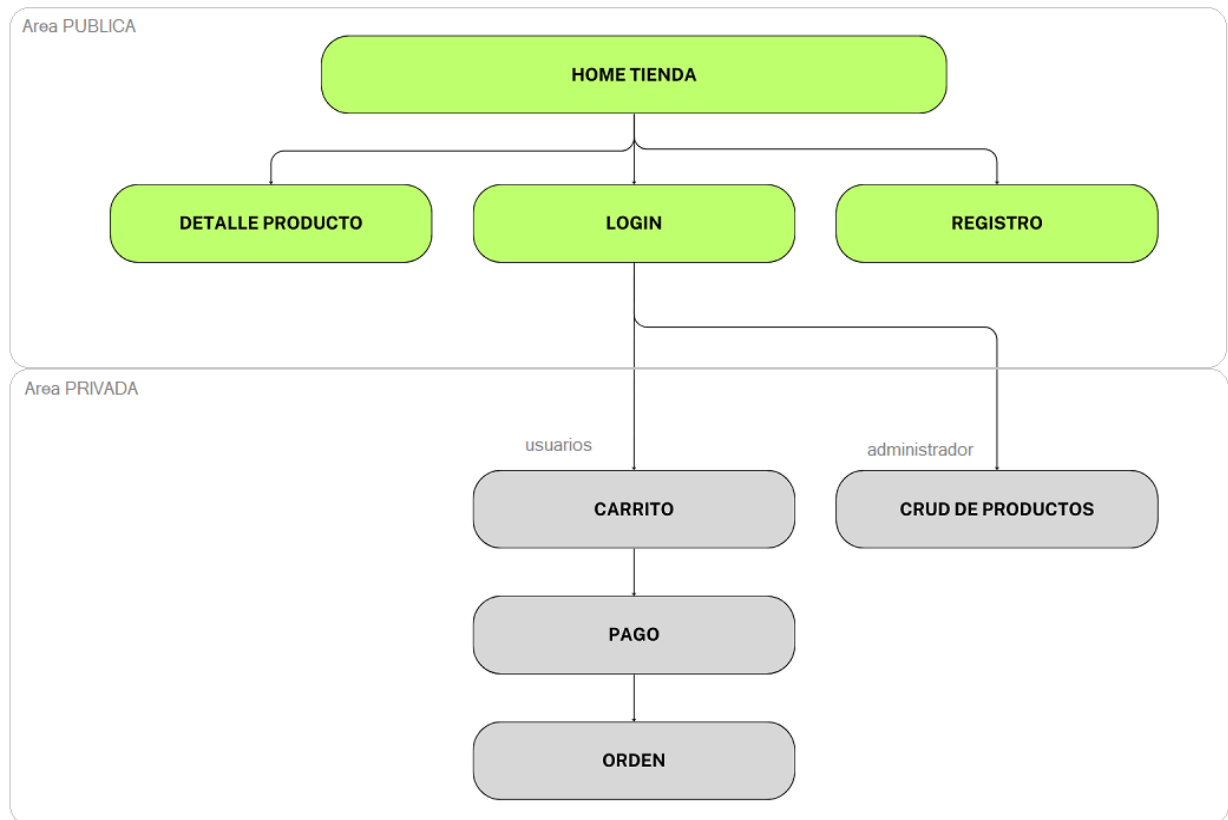
Agregar

Libro	Autor	Editorial	Precio	Acción
Titulo 1	Lorem ipsum	Lorem ipsum	\$9.999	Editar / Eliminar
Titulo 2	Lorem ipsum	Lorem ipsum	\$9.999	Editar / Eliminar
Titulo 3	Lorem ipsum	Lorem ipsum	\$9.999	Editar / Eliminar

Footer

2. La definición de navegación de vistas

Las vistas del proyecto Tienda de Libros se dividen en vistas públicas y privadas, para garantizar una experiencia de usuario fácil y segura.



Vistas Públicas:

Home Tienda: La vista principal del sitio web, donde los usuarios pueden explorar una variedad de libros y servicios ofrecidos.

Detalle Producto: Una vista detallada que proporciona información específica sobre un libro seleccionado.

Registró: Una vista que permite a los usuarios crear una cuenta personal en el sitio web proporcionando la información necesaria.

Login: Una vista donde los usuarios pueden iniciar sesión en sus cuentas existentes para acceder a funciones adicionales y realizar compras.

Vistas Privadas:

Carrito: Una vista accesible para usuarios registrados que les permite gestionar los libros seleccionados para la compra antes de proceder al pago.

Pago: Una vista donde los usuarios pueden completar la transacción seleccionando el método de pago.

Orden: Una vista que permite a los usuarios ver el estado de su compra.

CRUD de Productos: Una vista exclusiva para administradores que les permite gestionar el inventario de libros, incluyendo la creación, edición y eliminación de libros.

3. Listado de dependencias a utilizar en el proyecto Tienda de Libros.

Esta lista proporciona una descripción básica de cada dependencia.

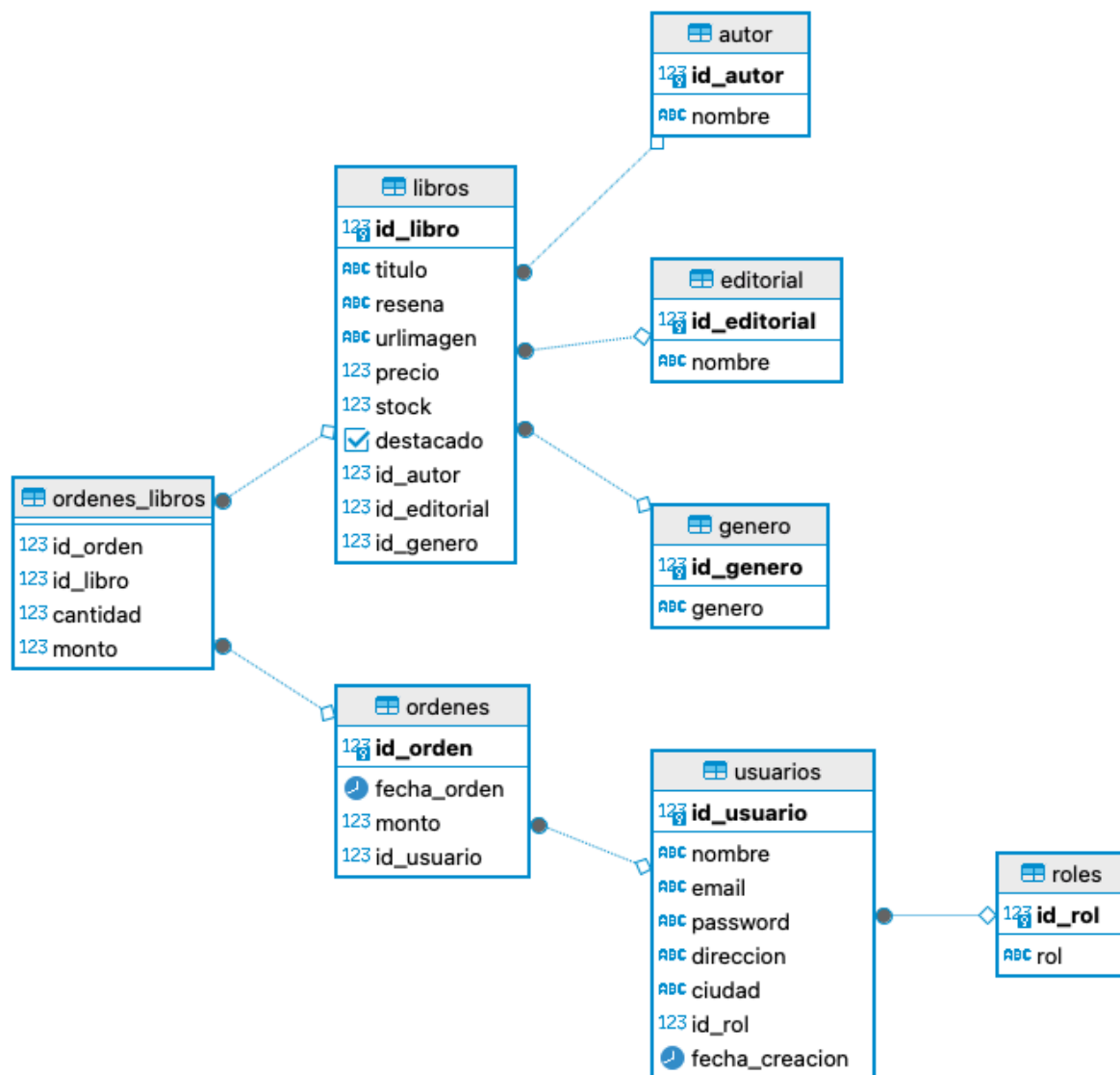
Frontend (Dependencias del lado del cliente):

- **React:** Biblioteca de JavaScript para construir interfaces de usuario.
- **React-router-dom:** Enrutador para aplicaciones de React que permite la navegación entre diferentes componentes.
- **Bootstrap:** Framework de diseño front-end para desarrollar sitios y aplicaciones web.

Backend (Dependencias del lado del servidor):

- **Express:** Framework web de Node.js para construir aplicaciones y APIs.
- **Cors:** Middleware de Express para habilitar el acceso a recursos de otros dominios.
- **Dotenv:** Módulo para cargar variables de entorno desde un archivo `.env`.
- **Pg:** Cliente PostgreSQL para Node.js.
- **Pg-format:** Proporciona una función de formateo seguro para consultas PostgreSQL.
- **Express-validator:** Middleware para la validación de datos en Express.
- **Bcryptjs:** Biblioteca para el hashing de contraseñas.
- **Jsonwebtoken:** Implementación de JSON Web Tokens (JWT) para autenticación.
- **Jest:** Framework de pruebas unitarias para JavaScript.
- **Supertest:** Biblioteca para realizar pruebas HTTP en Node.js.

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



5. Diseño Contrato de Datos de la API REST

Rutas Públicas

- Ruta: ``/libros``

****Home****

- Método: ``GET``
- Descripción: *Obtiene todos los libros disponibles.*
- Respuesta Exitosa:
 - Código: ``200 OK``
 - Contenido: *Array de objetos JSON con todos los libros.*

- Ruta: ``/libros/:id``

****Detalle del Producto****

- Método: ``GET``
- Descripción: *Obtiene los detalles de un libro específico.*
- Parámetros de Ruta:
 - ``id``: *Identificador único del libro.*
- Respuesta Exitosa:
 - Código: ``200 OK``
 - Contenido: *Objeto JSON con el libro solicitado.*
- Respuesta de Error:
 - Código: ``404 Not Found``
 - Contenido: *Mensaje indicando que el libro NO fue encontrado.*

- Ruta: `/autores``
 - **Obtener listado de autores****
 - Método: `GET``
 - Descripción: *Obtiene todos los autores disponibles.*
 - Respuesta Exitosa:
 - Código: `200 OK``
 - Contenido: *Array de objetos JSON con todos los autores*
- Ruta: `/editoriales``
 - Método: `GET``
 - Descripción: *Obtiene todas las editoriales disponibles.*
 - Respuesta Exitosa:
 - Código: `200 OK``
 - Contenido: *Array de objetos JSON con todas las editoriales.*
- Ruta: `/generos``
 - Método: `GET``
 - Descripción: *Obtiene todos los géneros disponibles.*
 - Respuesta Exitosa:
 - Código: `200 OK``
 - Contenido: *Array de objetos JSON con todos los géneros.*

- Ruta: ``/registro``

****Registro de Usuario****

- Método: ``POST``
- Descripción: *Registra un nuevo usuario.*
- Parámetros de Solicitud:
 - *Datos de usuario en formato JSON (nombre, correo, contraseña).*
- Respuesta Exitosa:
 - Código: ``201 Creado``
 - Contenido: *Mensaje indicando que el usuario ha sido registrado exitosamente.*

- Ruta: ``/login/``

****Inicio de Sesión****

- Método: ``POST``
- Descripción: *Inicio sesión de usuario.*
- Parámetros de Solicitud:
 - *Datos de usuario en formato JSON (correo, contraseña)*
- Respuesta Exitosa:
 - Código: ``200 OK``
 - Contenido: *Objeto JSON que contiene correo del usuario y token de autenticación.*
- Respuesta de Error:
 - Código: ``404 Not Found``
 - Contenido: *Mensaje indicando que el usuario no fue encontrado.*

Rutas Privadas

Este diseño asegura que las rutas de carrito y orden sean accesibles sólo para usuarios autenticados mediante el uso de un token de autenticación obtenido a través de la ruta de inicio de sesión.

- Ruta: `/ordenes/`

****Crear orden****

- Método: `POST`

- Descripción: *Crea la compra del usuario.*

- Parámetros de Ruta:

- *Datos la compra en formato JSON (producto, cantidad, monto, total, usuario)*

- Cabecera de Autenticación:

- `Authorization: Bearer <token>`

- Respuesta Exitosa:

- Código: `200 OK`

- Contenido: ID de la orden.

- Respuesta de Error:

- Código: `401 No autorizado`

- Contenido: *Mensaje que el usuario no está autorizado o el token NO es válido.*

- Ruta: `/ordenes/:id`

****Órdenes****

- Método: `GET`
- Descripción: *Obtiene las órdenes realizadas por el usuario.*
- Parámetros de Ruta:
 - `id`: *Identificador único del usuario.*
- Cabecera de Autenticación:
 - `Authorization: Bearer <token>`
- Respuesta Exitosa:
 - Código: `200 OK`
 - Contenido: *Array de objetos JSON con las órdenes realizadas por el usuario.*
- Respuesta de Error:
 - Código: `401 No autorizado`
 - Contenido: *Mensaje indicando que el usuario no está autorizado o el token es inválido.*

CRUD

- Ruta: `/libros`

****Crear un libro****

- Método: `POST`
- Descripción: *Crea un nuevo libro.*
- Parámetros de Ruta:
 - *Información del libro en formato json (título, reseña, precio, stock, id autor, id editorial, id genero, url imagen, destacado).*
- Cabecera de Autenticación:
 - `Authorization: Bearer <token>`
- Respuesta Exitosa:
 - Código: `200 Libro creado`
 - Contenido: *Información del libro creado en formato json.*
- Respuesta de Error:
 - Código: `401 No autorizado`
 - Contenido: *Mensaje indicando que el usuario no está autorizado o el token es inválido.*

- Ruta: `/libros/:id`

****Editar un libro****

- Método: `PUT`

- Descripción: *Actualiza un libro.*

- Parámetros de Ruta:

- *Información del libro en formato json (id libro, título, reseña, precio, stock, id autor, id editorial, id genero, url imagen, destacado).*

- Cabecera de Autenticación:

- `Authorization: Bearer <token>`

- Respuesta Exitosa:

- Código: `200 Libro actualizado`

- Contenido: *Información del libro actualizado en formato json.*

- Respuesta de Error:

- Código: `401 No autorizado`

- Contenido: *Mensaje indicando que el usuario no está autorizado o el token es inválido.*

- Ruta: `/libros/:id`

****Elimina un libro****

- Método: `DELETE`
- Descripción: *Crea un nuevo libro.*
- Parámetros de Ruta:
 - `id`; *identificador del libro que se debe eliminar.*
- Cabecera de Autenticación:
 - `Authorization: Bearer <token>`
- Respuesta Exitosa:
 - Código: `200 Libro eliminado`
- Respuesta de Error:
 - Código: `401 No autorizado`
 - Contenido: *Mensaje indicando que el usuario no está autorizado o el token es inválido.*