

## **Proyecto Integrador Paradigmas y Lenguajes de programación III**

CARRERA: Ingeniería en sistemas de información

MATERIA: Paradigmas y Lenguajes de Programación III

COMISIÓN: "U" (única) "A"

PROFESOR: Mgter. Ing. Agustín Encina

ESTUDIANTES: Alegre Isabella

FECHA: 07-10-2025



## **Proyecto Integrador Paradigmas y Lenguajes de programación III**

### **Nombre del Proyecto**

- Booksy

### **Descripción de la idea**

El proyecto Booksy es una red social y base de datos de libros, diseñada para crear una comunidad vibrante de lectores. El objetivo es superar a las plataformas existentes ofreciendo una interfaz de usuario más intuitiva y atractiva, junto con funcionalidades innovadoras que enriquezcan la experiencia del usuario.

La plataforma permitirá a los usuarios descubrir y explorar libros, acceder a reseñas y valoraciones, así como compartir sus propias opiniones. Cada lector podrá organizar su biblioteca personal en estanterías personalizadas, contar con un perfil propio configurable, y llevar un registro detallado de sus lecturas.

Además, la plataforma permitirá a los usuarios contribuir directamente a la base de datos de libros, añadiendo títulos que no se encuentren en el catálogo. Un equipo de bibliotecarios validará estas nuevas entradas para mantener la calidad y precisión de la información.

Con estas características, Booksy busca convertirse en un espacio dinámico, colaborativo y confiable para lectores, autores y amantes de la literatura, integrando comunidad, descubrimiento y creación en un solo lugar.



## Objetivos Generales

Desarrollar una plataforma web llamada Booksy, concebida como red social y base de datos de libros, que permita a los usuarios descubrir, registrar y reseñar lecturas, gestionar bibliotecas personales y generar estadísticas de hábitos de lectur, asegurando además control de privacidad en sus perfiles y fomentando la interacción social entre lectores.

## Objetivos Específicos

- **Diseñar e implementar** una interfaz moderna, intuitiva y adaptable que optimice la experiencia de usuario en la plataforma.
- **Desarrollar perfiles de usuario personalizables**, con opción de configurarlos como públicos o privados según preferencias de privacidad.
- **Crear y gestionar bibliotecas personales** donde los usuarios organicen sus lecturas en distintas categorías (leído, leyendo, por leer, favoritos).
- **Incorporar un registro detallado de lecturas**, que permita al usuario llevar un historial con fechas de inicio y fin.
- **Facilitar la publicación de reseñas y valoraciones** cada vez que se complete una lectura, enriqueciendo la base de datos colaborativa.
- **Incorporar un sistema de contribución al catálogo**, permitiendo a los usuarios sugerir nuevos libros o publicar sus propias obras.
- **Definir un sistema de roles y permisos** diferenciado para usuarios y administradores.
- **Garantizar escalabilidad, seguridad y protección de datos personales**, incluyendo autenticación robusta y control de acceso a la información.



## Requerimientos Funcionales del Proyecto:

- **Módulo Inicio de Sesión y Registro**

**R1.1:** El sistema debe permitir a los usuarios registrarse con una dirección de correo electrónico, un nombre de usuario y una contraseña

**R1.2:** El sistema debe permitir a los usuarios iniciar y cerrar sesión de manera segura.

**R1.3:** El sistema debe permitir a los usuarios recuperar su contraseña si la olvidan.

Actores: Usuario normal, Administrador del sistema

- **Módulo Perfil Personal**

**R2.1:** El sistema debe permitir a los usuarios editar su perfil. Esto incluye la opción de subir una foto, agregar una biografía corta, y especificar intereses de lectura (géneros, autores favoritos, etc.).

**R2.2:** El perfil personal debe mostrar información principal:

- Foto de perfil.
- Nombre y biografía.
- Intereses de lectura.
- Botones de acceso rápido: Editar Perfil, Recomendaciones y Bibliotecas.

**R2.3:** El sistema debe permitir al usuario gestionar sus bibliotecas personales, pudiendo:

- Listar todas las bibliotecas creadas.

**R2.5:** El sistema debe permitir al usuario agregar o quitar libros en sus bibliotecas.

Actores: Usuario normal



- **Modulo Gestion de Usuarios**

**R3.1:** El sistema permite al administrador gestionar los usuarios registrados en el sistema, asignar roles y permisos adecuados.

Actores: Administrador del Sistema

- **Módulo de Gestión de Libros**

**R4.1:** El sistema debe permitir gestionar el catálogo de libros al administrador. Debe poder listar los libros cargados dentro del sistema, agregar nuevos libros con la información acorde, editar libros y actualizar la información editada, guardar cambios necesarios y borrar libros en el sistema.

**R4.2:** El sistema debe permitir que cada libro tenga su propia ficha con título, autor, portada, descripción y categoría

**R4.3:** El sistema debe permitir al administrador gestionar solicitudes de nuevos libros por parte del usuario.

Actores: Administrador del Sistema

- **Módulo Búsqueda y descubrimientos de Libros**

**R5.1:** El sistema debe permitir a los usuarios acceder al catálogo cargado de libros en el sistema con la información detallada de cada uno (título, autor, sinopsis, ISBN, año de publicación, editorial, número de páginas, portada).

**R5.2:** El sistema debe permitir a los usuarios buscar libros por título, autor, género o ISBN.

Actores: Usuario Normal



- **Modulo Detalle del Libro**

**R6.1:** El sistema debe permitir acceder al perfil detallado para cada libro con todas sus especificaciones y las reseñas de la comunidad.

**R6.2:** El sistema debe permitir a los usuarios añadir un libro a sus bibliotecas personales: "Quiero leer", "Leyendo" y "Leído".

**R6.3:** El sistema debe permitir a los usuarios dejar una reseña escrita.

**R6.4:** El sistema debe mostrar reseñas de otros usuarios en la ficha del libro, con nombre de usuario, comentario y fecha.

- Actores: Usuario Normal

### **Módulo de Seguimiento**

**R7.1:** El sistema debe permitir que los usuarios puedan buscar y visualizar los perfiles de otros usuarios que se encuentren dentro del sistema.

**R7.2:** El sistema debe permitir que los usuarios puedan seguir a otros usuarios que se encuentran registrados dentro del sistema.

**R7.3:** El sistema debe permitir que cada usuario tenga dentro de su perfil un recuento de la cantidad de seguidores y seguidos que tiene dentro del sistema.

**R7.4:** El sistema debe permitir al usuario que al ingresar al sistema, pueda ver la actividad de los usuarios que sigue.

Actores: Usuario Normal



## **Requerimientos No Funcionales del Proyecto**

### **Rendimiento**

- RNF1.1: El sistema debe cargar los perfiles de usuario, las páginas de libros y las búsquedas en menos de 3 segundos, incluso con una alta carga de usuarios concurrentes.
- RNF1.2: La base de datos debe ser capaz de manejar millones de registros de libros y usuarios sin degradar la velocidad de consulta.

### **Usabilidad**

- RNF3.1: La interfaz de usuario debe ser intuitiva, limpia y fácil de navegar, siguiendo los principios de un diseño centrado en el usuario.
- RNF3.2: El diseño debe ser responsivo y funcionar de manera óptima en diferentes dispositivos (computadora de escritorio, tabletas y móviles).
- RNF3.3: La aplicación debe proporcionar mensajes de error claros y útiles a los usuarios cuando algo falle.

### **Escalabilidad**

- RNF5.1: La arquitectura del sistema debe ser modular, permitiendo agregar nuevas funcionalidades (ej. tiendas de libros, venta de libros, etc.) en el futuro sin reestructurar todo el sistema.
- RNF5.2: El sistema debe ser capaz de escalar horizontalmente para manejar un crecimiento exponencial de usuarios y datos.



## **Tecnologías a Utilizar**

### **-Backend:** Typescript con Node.js

Decidí integrar Node.js por su capacidad para manejar un alto volumen de tráfico y peticiones concurrentes de manera eficiente. Una red social de libros, con usuarios publicando, comentando y compartiendo contenido, genera múltiples interacciones al mismo tiempo.

### **-Base de Datos:** PostgreSQL

Ambas bases de datos son robustas y escalables, capaces de manejar grandes volúmenes de datos y un número creciente de usuarios a medida que la red social crece.

### **-ORM:** TypeORM

Su elección se debe a que me permite interactuar con la base de datos utilizando objetos JavaScript en lugar de escribir consultas SQL complejas. Esto hace el código más legible y manejable.

### **-FrontEnd:** React con Vite y Tailwind CSS

- **React:** Es ideal para una aplicación con muchas funcionalidades y componentes dinámicos (como feeds de noticias, perfiles de usuario, listas de libros). Su arquitectura basada en componentes permite reutilizar código y gestionar el estado de la interfaz de manera eficiente.
- **Tailwind CSS:** Integre Tailwind ya que me da mayor velocidad y flexibilidad en el diseño. Me permite diseñar la interfaz directamente en el código HTML, sin tener que escribir CSS personalizado para cada componente.

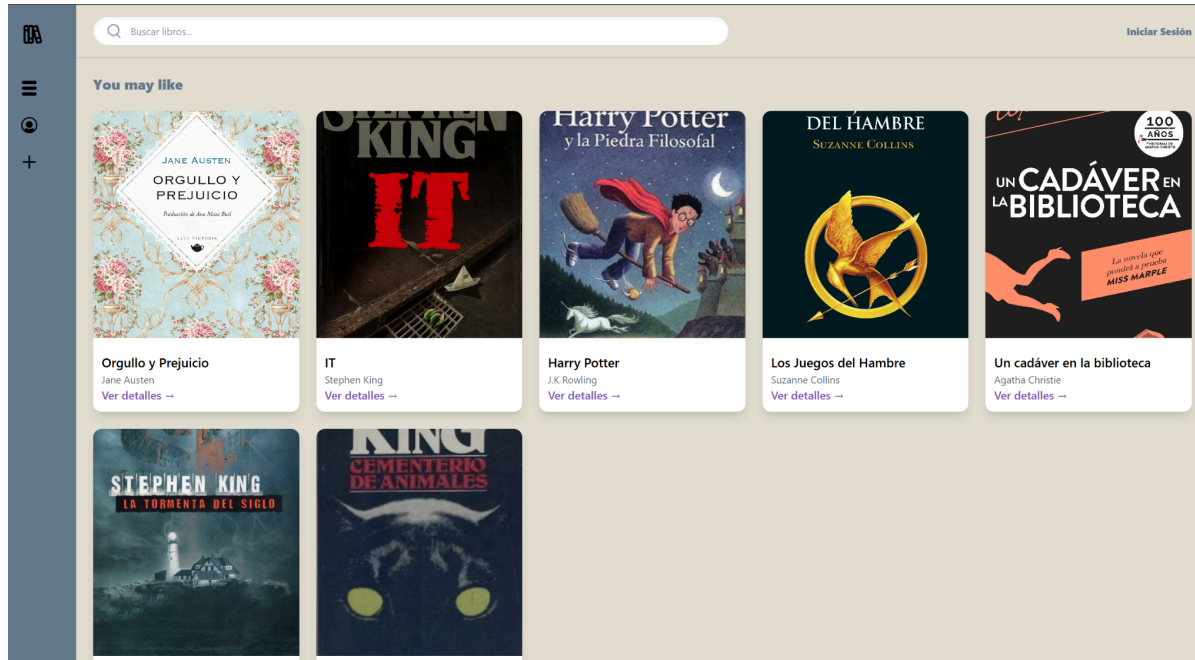




## Descripción de las vistas e implementación del sistema Booksy

### 1. Vista Principal: Exploración de Libros\

- **Ruta:** /
- **Descripción:** Es la página inicial de la aplicación y punto de acceso principal al catálogo de libros.
- **Implementación:**
  - Se obtiene la lista de libros desde el backend mediante peticiones GET /books.
  - La vista utiliza componentes reutilizables de React (como BookCard) y estilos con **Tailwind CSS** para lograr un diseño limpio y adaptable.
  - Si el usuario no está autenticado, igualmente puede explorar el catálogo, aunque no puede guardar libros ni dejar reseñas.
  - Listado en formato de tarjetas (Box): Los libros se presentan en una cuadrícula. Cada libro ocupa su propia tarjeta o "box" individual, lo que permite una visualización limpia y ordenada.
  - Contenido de cada "box": Cada tarjeta incluye la información esencial para que el usuario identifique el libro de un vistazo:
    - Portada del libro
    - Título y Autor
    - Enlace a "Ver detalles": Una llamada a la acción que permite al usuario hacer clic para obtener más información sobre el libro, como la sinopsis, reseñas, etc.



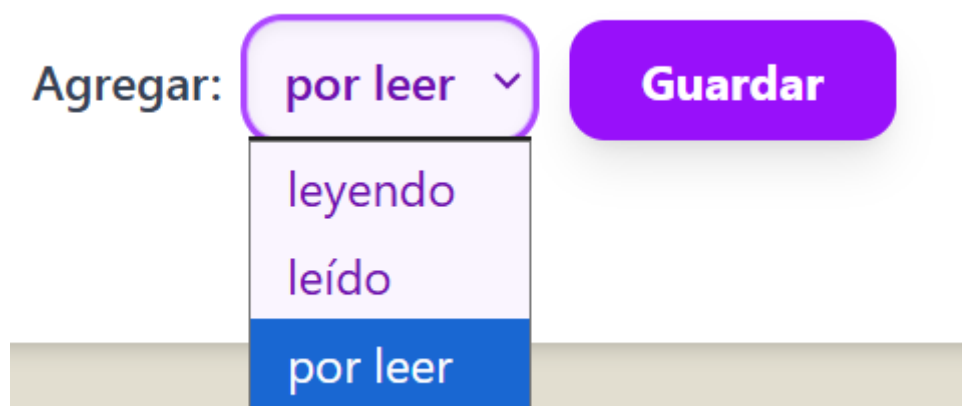
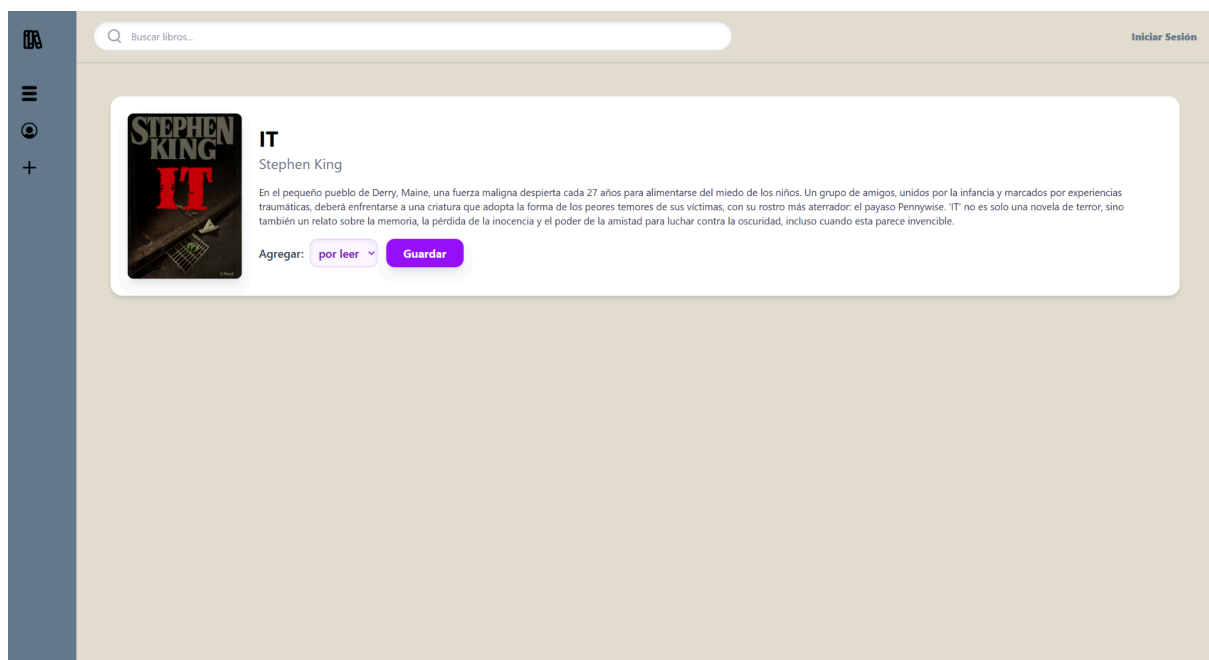
## 2. Vista de Detalles del Libro

Esta es la pantalla a la que el usuario accede al hacer clic en "Ver detalles" desde la página principal. Se centra en proporcionar toda la información relevante sobre un libro específico.

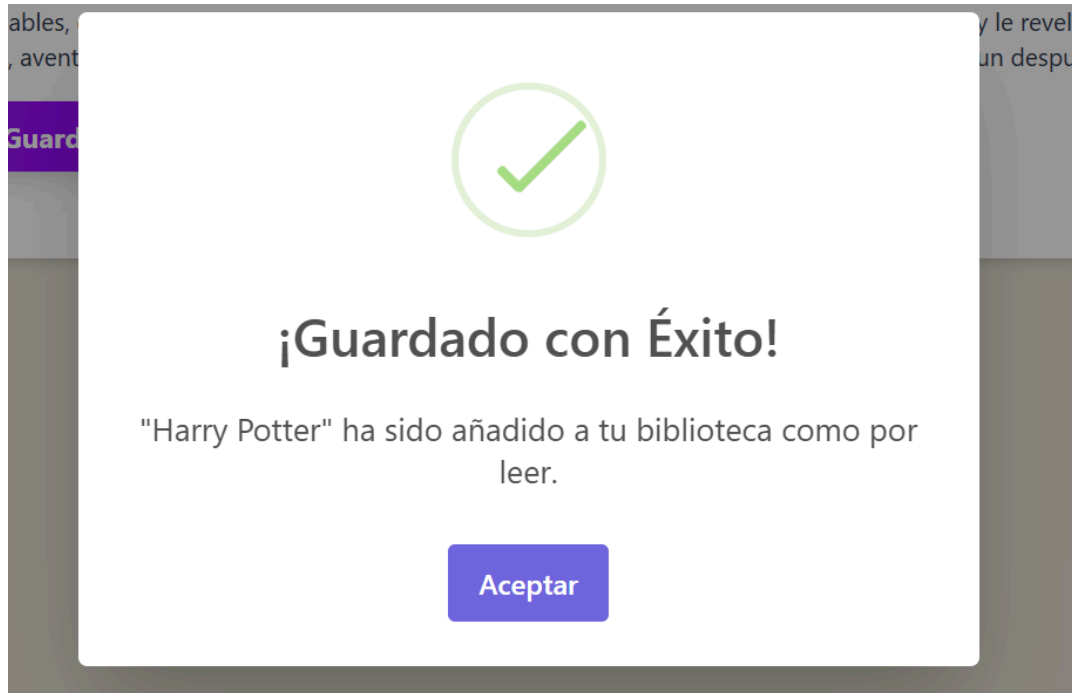
- **Ruta:** /books/:id
- **Implementación:**
  - Se obtiene el detalle del libro desde el backend mediante GET /books/:id.
  - **Diseño:** La información se presenta en una tarjeta grande y bien organizada, dividida en dos columnas.
    - Izquierda: Se muestra la portada del libro en un tamaño considerable.
    - Derecha: Contiene los detalles textuales, incluyendo el título, el autor y una sinopsis o descripción completa de la obra.
  - **Acciones Principales:**



- Incluye un Select con los estados "Por leer", "Leyendo" y "Leído", y un botón Guardar, que envía la selección al backend mediante POST /user/books.
- Botón "Guardar": Una llamada a la acción clara para que el usuario pueda guardar el libro en su colección personal con el estado seleccionado con un solo clic.



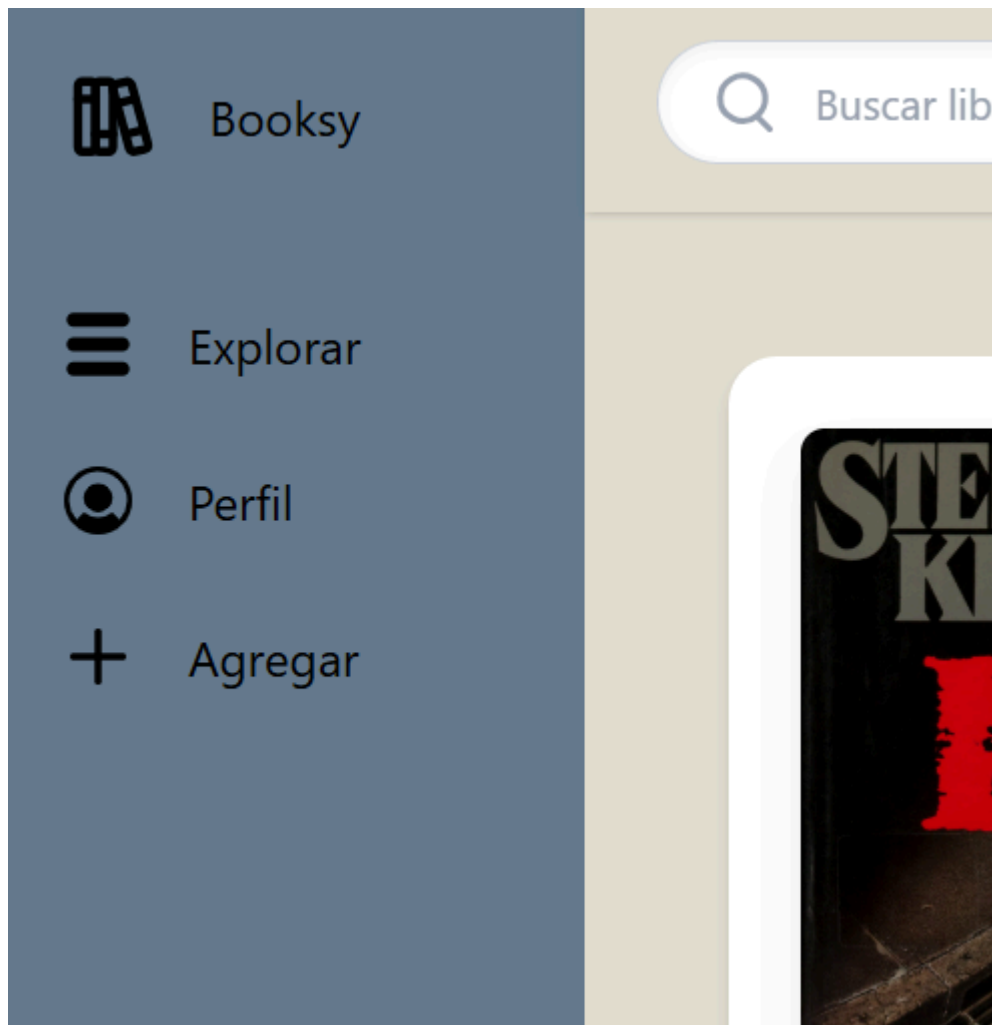
Una vez guardado con el estado seleccionado, el usuario puede visualizar una tarjeta pop up de 'Guardado con Éxito'.



### 3. Barra de Navegación Lateral (Sidebar)

A la izquierda de la pantalla, se encuentra una barra de navegación vertical y minimalista que, al ser desplegada, da acceso a las secciones principales de la plataforma

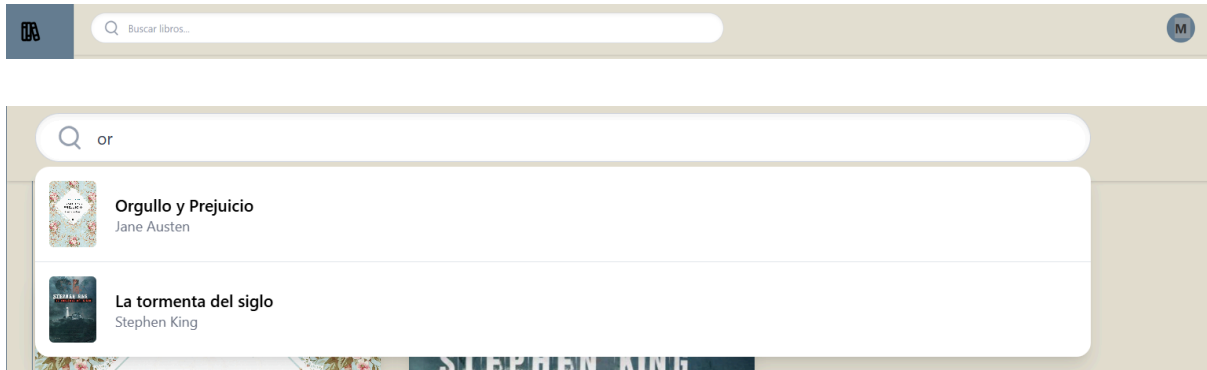
- **Presente en todas las vistas..**
- **Implementación:**
  - Construida como un componente React (Sidebar) que utiliza react-router-dom para la navegación interna.
  - Opciones:
    - **Explorar:** redirige al listado principal.
    - **Perfil:** redirige al perfil del usuario si está autenticado; en caso contrario, al login.
    - **Agregar:** lleva al formulario de sugerencias.
  - Implementada con **Tailwind CSS** para una experiencia responsiva y animaciones suaves al expandir o contraer.



#### 4. Navbar

- **Presente en la parte superior de la interfaz.**
- **Implementación:**
- Contiene el logo de la app, barra de búsqueda y accesos rápidos al perfil o cierre de sesión.
- La búsqueda usa un hook personalizado con *debounce* (**useDebounce**) para optimizar peticiones al backend durante la escritura.



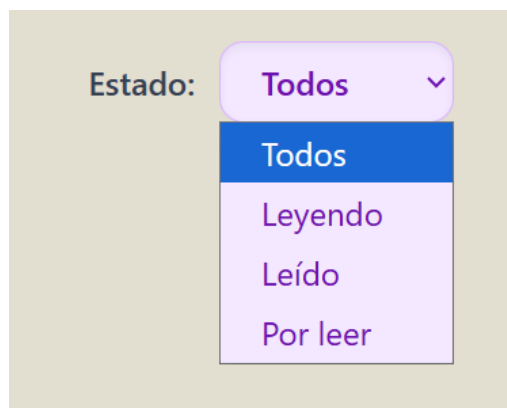
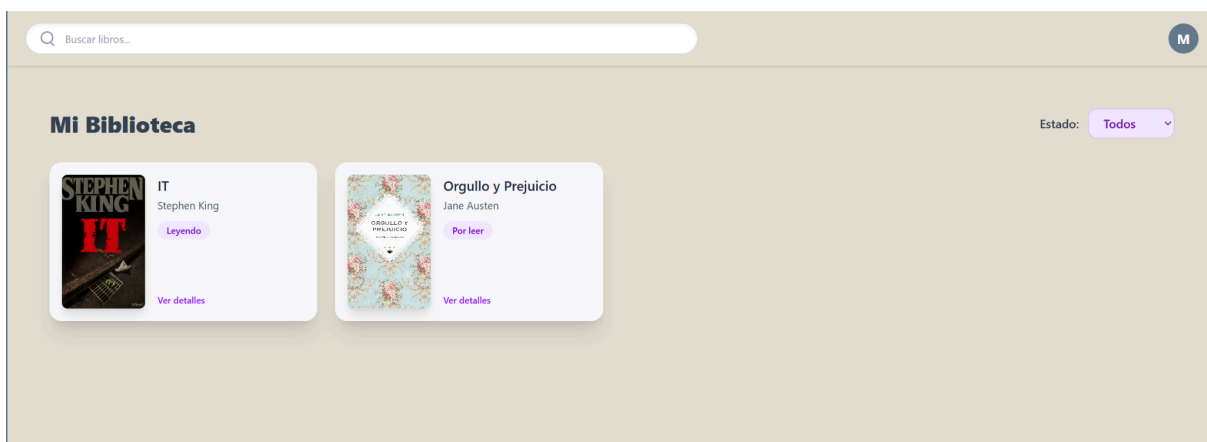


## 5. Biblioteca Personal ("Mis libros guardados")

- Ruta: /api/users/:userId/library
- Esta sección funciona como la estantería virtual del usuario. Aquí puede ver y gestionar todos los libros que ha decidido guardar.

### Implementación:

- Se listan los libros guardados por el usuario autenticado mediante GET /user/books.
- Cada libro muestra su estado actual ("Por leer", "Leyendo" o "Leído").





## 6. Formulario de Sugerencias

Esta es la pantalla a la que el usuario accede al hacer clic en "Agregar" desde la barra lateral.

Esta interfaz ofrece a los usuarios un canal directo para contribuir y ayudar a mejorar el servicio.

- **Ruta:** /suggestions
- **Campos del Formulario:** El formulario solicita información estructurada para facilitar la gestión de las sugerencias:
  - Nombre del libro
  - Autor
  - Género
- **Feedback del Sistema:** Incluye un campo de texto abierto ("¿Qué mejorarías del sistema?") para que los usuarios puedan dar sugerencias generales sobre la plataforma.
- **Acción Final:** El botón "Enviar sugerencia" completa el proceso.
- Al enviar, los datos se envían al backend con POST /suggestions.



## 7. Formulario de Inicio de Sesión

**Ruta:** /Login

**Implementación:**

- Validación de correo y contraseña mediante POST /auth/login.
- Si es exitoso, se almacena el *token JWT* en el almacenamiento local (localStorage) y se redirige al perfil.
- Maneja errores de autenticación con mensajes claros.

## 8. Formulario de Registro

**Ruta:** /register

**Implementación:**

- Permite registrar nuevos usuarios con nombre, email y contraseña.
- Envía los datos al backend con POST /auth/register.
- Incluye validaciones de formato y mensajes de confirmación.





## 9. Loader

Se incorporó un componente “loader” que me permita poder



## Responsividad e Implementación Adaptativa

Para garantizar una experiencia de usuario óptima en distintos dispositivos (computadoras, tablets y móviles), Booksy fue desarrollada bajo el principio de diseño responsive, aprovechando las utilidades de Tailwind CSS y componentes condicionales en React.



## 1. Uso de Clases de Tailwind CSS

Tailwind ofrece prefijos como sm:, md:, lg: y xl: para aplicar estilos específicos según el ancho de la pantalla. Gracias a esto, cada componente puede modificar su diseño de forma automática al detectar un cambio de tamaño, sin necesidad de media queries manuales.

### Ejemplo de adaptación:

- En la página principal (Home.jsx) y en la biblioteca del usuario (UserLibrary.jsx), la cantidad de columnas en la lista de libros se ajusta dinámicamente:
  - Pantallas pequeñas: grid-cols-1
  - Pantallas medianas: sm:grid-cols-2
  - Pantallas grandes: md:grid-cols-3
  - Pantallas extra grandes: lg:grid-cols-4, xl:grid-cols-5

Esto asegura que las tarjetas (BookCard) se muestran de forma organizada y legible en cualquier dispositivo.

## 2. Adaptación de la Barra Lateral (Sidebar)

El componente Sidebar.jsx también presenta un comportamiento adaptativo:

- En pantallas pequeñas, ocupa un ancho fijo (w-20) para optimizar el espacio disponible.
- En pantallas medianas y grandes, la barra lateral puede expandirse al pasar el cursor (md:w-24 md:hover:w-48), mostrando los textos junto a los íconos.

De esta manera, se equilibra la estética minimalista con la funcionalidad.

## 3. Navbar y Comportamiento del Toggle Sidebar



En pantallas de escritorio (tamaño mediano y superior), la Sidebar permanece siempre visible.

En dispositivos móviles, en cambio, se oculta de manera predeterminada para aprovechar al máximo el espacio en pantalla.

Para mantener la accesibilidad a la navegación, se incorporó un botón “toggle” en el componente Navbar.jsx, que permite mostrar u ocultar la barra lateral.

#### **4. Funcionamiento del toggleSidebar**

- **Estado Global (isSidebarOpen) – MainLayout.jsx:**

El estado que controla la visibilidad de la barra lateral se define en el componente MainLayout.jsx, encargado de estructurar toda la aplicación.

- **Función toggleSidebar:**

Es una función que cambia el valor del estado isSidebarOpen. Se declara utilizando useCallback para evitar recreaciones innecesarias en cada renderizado, mejorando el rendimiento general.

- **Paso de Props:**

La función toggleSidebar se pasa como prop tanto a Navbar.jsx como a Sidebar.jsx, permitiendo que ambos componentes interactúen con el mismo estado compartido.

- **Botón de la Navbar:**

Dentro de Navbar.jsx, existe un botón con la clase md:hidden, lo que significa que solo se muestra en pantallas pequeñas. Al hacer clic, invoca la función toggleSidebar, mostrando u ocultando el panel lateral.

- **Transiciones en la Sidebar:**

En Sidebar.jsx, las clases condicionales controlan el desplazamiento del menú:

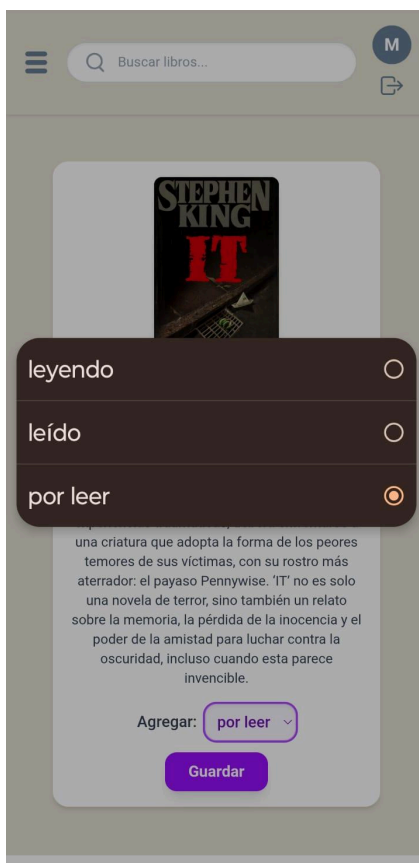


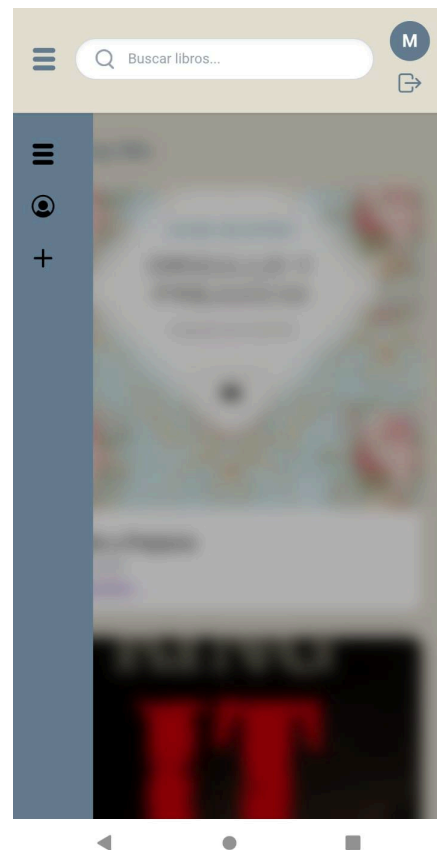
- translate-x-full: oculta la barra lateral fuera de la pantalla.
- translate-x-0: la muestra completamente visible.
- transition-transform: agrega una animación suave durante el cambio de estado.

### La responsividad en Booksy se logra combinando:

- El poder de las clases utilitarias de Tailwind CSS,
- La gestión de estado y reactividad de React,
- Y un sistema condicional de visibilidad que adapta dinámicamente la interfaz.

### Vistas en tamaño móvil





## Contextos y Hooks Utilizados en Booksy

En Booksy, se implementaron dos contextos principales: AuthContext y SearchContext.

- AuthContext: se encarga de gestionar el estado de autenticación de los usuarios en toda la aplicación. Tiene un AuthProvider que envuelve toda la aplicación y provee los valores del contexto a sus componentes hijos.
- El archivo SearchContext.jsx se encarga de gestionar la consulta de búsqueda global utilizada en la aplicación, permitiendo sincronizar el texto de búsqueda entre componentes.

Además, se implementaron tanto hooks nativos como personalizados

Nativos:

- useState utilizado para gestionar:
  - El token, usuario y estado de carga en AuthContext.jsx.
  - Variables locales en componentes como formularios o vistas de detalle.
- useEffect: Ejecuta efectos secundarios en respuesta a cambios de estado o propiedades.



- useContext: se utiliza para implementar los hooks personalizados useAuth y useSearch
- useCallback: Se utiliza en MainLayout.jsx para evitar recrear la función toggleSidebar en cada renderizado, mejorando el rendimiento y evitando renders innecesarios.
- useMemo: teniendo en cuenta que permite memorizar el resultado de un cálculo y utilizarlo siempre que sus dependencias no cambien, evitando así operaciones repetitivas e innecesarias. Se utilizó para filtrar y mapear la lista de los libros.

Personalizado:

- useDebounce: Este hook se diseñó para optimizar las búsquedas en tiempo real.

## Hosting Aplicado

Para poner Booksy en funcionamiento de manera pública y accesible desde cualquier dispositivo, se implementó un proceso de deploy dividido entre el backend y el frontend, utilizando servicios especializados que garantizan rendimiento, escalabilidad y disponibilidad permanente.

### - Backend - Render

El servidor backend desarrollado en Node.js con TypeScript y utilizando Express y PostgreSQL fue desplegado en la plataforma Render.

SERVICE NAME	STATUS	RUNTIME	REGION	DEPLOYED
⊕ Booksy	✓ Deployed	Node	Oregon	1d
⊕ Booksy-API	Canceled deploy	Node	Oregon	4d
☰ proyecto-integrador	✓ Available	PostgreSQL 17	Oregon	4d

Características principales del despliegue:

- Render permite ejecutar aplicaciones Node.js de forma continua en un entorno en la nube, sin necesidad de administrar infraestructura.
- Se configuró un servicio web (Web Service) que ejecuta el comando de inicio definido en el package.json, iniciando el servidor Express.

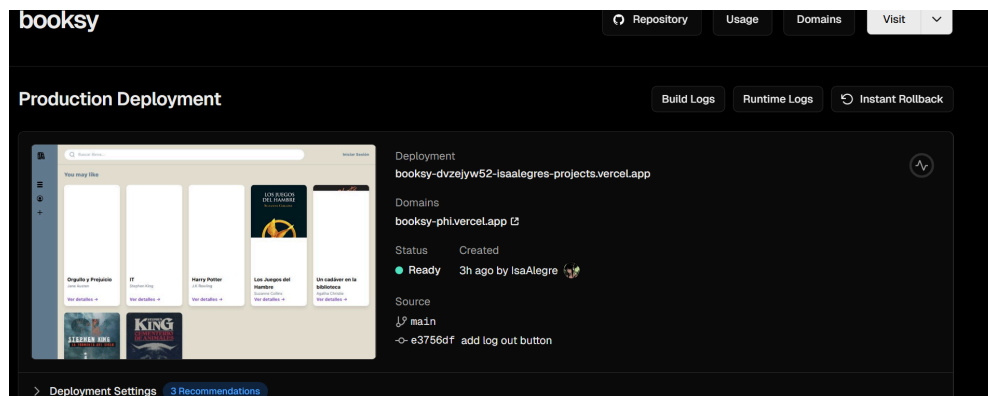


- La conexión con la base de datos PostgreSQL se gestionó a través de las variables de entorno configuradas en el panel de Render, manteniendo las credenciales seguras.

## - Frontend - Vercel

El frontend de Booksy, desarrollado con React, Vite y Tailwind CSS, fue desplegado en la plataforma Vercel, especialmente optimizada para proyectos basados en frameworks JavaScript modernos.

Se conectó el repositorio de GitHub del cliente (/client) directamente con Vercel.



## Funcionalidades Pendientes y Futuras Ampliaciones

### 1. Sistema de Reseñas y Valoraciones

- Se implementará la posibilidad de que los usuarios puedan agregar reseñas escritas y valoraciones con estrellas en la página de detalle de cada libro.
- **Objetivo:** Fomentar la interacción dentro de la comunidad de lectores y enriquecer la base de datos colaborativa.

### 2. Personalización del Perfil de Usuario

- Incorporar la posibilidad de que cada usuario configure su perfil y lo personalice según sus preferencias.
- **Objetivo:** Darle al usuario mayor control sobre su identidad dentro de la plataforma, fortaleciendo el sentido de pertenencia y comunidad.



### 3. Visualización de Perfiles de Otros Usuarios

- Permitir que los usuarios puedan visitar y visualizar los perfiles de otros lectores, ver sus bibliotecas públicas y reseñas compartidas.
- **Objetivo:** Favorecer la interacción social y el descubrimiento de nuevos libros a través de los intereses de otros usuarios.

### 4. Estadísticas de Lectura

- Incluir una sección donde el usuario pueda visualizar gráficos estadísticos sobre su progreso de lectura (por ejemplo, páginas leídas por mes o géneros más leídos).
- **Objetivo:** Motivar la lectura y ofrecer al usuario información personalizada sobre sus hábitos.