IES ALBARREGAS

TÉCNICO SUPERIOR EN DESARROLLO DE APLICACIONES WEB

Departamento de Informática

PROYECTO

LibriX

Manual Técnico

Autor/es: Cristina Polo Sánchez
Curso Académico: 2023 - 2025

Índice

1. Introducción

LibriX es una aplicación web desarrollada con el objetivo de ofrecer una biblioteca digital accesible, intuitiva y segura para cualquier persona interesada en la lectura. Su funcionalidad gira en torno a un sistema de suscripción que permite acceder a una amplia colección de libros digitales, sin necesidad de descargarlos ni usar dispositivos adicionales.

La aplicación se ha desarrollado teniendo en cuenta la usabilidad y accesibilidad, con un diseño responsive adaptado a múltiples dispositivos (PC, móvil, tablet). Entre las funcionalidades principales que se han implementado destacan:

- Visualización del catálogo de libros con filtros de búsqueda, pueden buscar libros por su título.
- Registro e inicio de sesión con autenticación segura.
- Lector digital integrado con opciones de personalización (fuente, fondo, tamaño).
- Gestión de suscripciones (mensual/anual), gestionado desde el perfil del usuario.
- Recomendación de libros basada en el historial de lecturas del usuario, especialmente los géneros que ha consultado o leído con mayor frecuencia.
- Panel de estadísticas para el seguimiento de lecturas populares y suscripciones.
- Sistema de roles: acceso diferenciado entre usuarios registrados y no registrados.
- Exportación de datos en archivo Excel para consultar los usuarios y libros que haya en la plataforma, podrá hacerlo el administrador
- Backend con Supabase para el almacenamiento y la gestión de datos.

Con LibriX se busca no solo ofrecer una solución técnica, sino también fomentar la lectura digital y adaptarse a los nuevos hábitos de consumo cultural.

Este proyecto está pensado especialmente para quienes, como muchos lectores, echamos en falta un espacio cómodo, sencillo y accesible donde poder leer sin distracciones, sin complicaciones técnicas ni barreras de acceso. LibriX nace de esa necesidad real: ofrecer un entorno digital que respete el acto de leer, que lo facilite y que sea posible desde cualquier lugar, sin tener que descargar archivos ni depender de

dispositivos específicos. En definitiva, es una herramienta hecha por y para lectores, pensada para adaptarse a la rutina diaria y fomentar el hábito de la lectura en la era digital.

2. Arquitectura de la aplicación

La arquitectura de LibriX se basa en una separación clara entre el frontend (interfaz del usuario) y el backend (gestión de datos y lógica de negocio), basado en arquitectura *serverless*, lo que reduce la complejidad de la infraestructura. Esta arquitectura permite escalar fácilmente, mantener el código de forma más limpia y reducir los costes de mantenimiento.

A continuación, se detalla tanto el conjunto de tecnologías como la lógica funcional del sistema.

Lógica funcional del sistema

La aplicación funciona en base a flujos diferenciados según el tipo de usuario y su nivel de acceso:

- **Usuario no registrado**:
  - Puede acceder al catálogo completo de libros.
  - No puede leer libros.
- **Usuario registrado sin suscripción activa**:
  - Tiene perfil personal editable.
  - Accede al historial de búsqueda y libros marcados.
  - Puede gestionar métodos de pago y elegir tipo de suscripción.
- **Usuario registrado con suscripción activa**:
  - Puede leer libros directamente en el lector digital.
  - Accede a recomendaciones personalizadas.
  - Su actividad queda registrada para generar estadísticas.
- **Administrador**:
  - Puede subir nuevos libros y autores.
  - Visualiza usuarios y suscripciones.
  - Visualiza estadísticas generales y exporta datos en archivo Excel.

Las interacciones clave se centran en:

- **Lectura de libros en línea**: el sistema carga archivos EPUB desde Supabase para que sean visibles en el lector.

- **Sistema de suscripción**: validación del estado activo/inactivo al acceder al lector.
- **Gestión de filtros y búsqueda**: permite encontrar libros por autor, género, popularidad o novedades.
- **Estadísticas**: mostradas en panel gráfico con opción de exportar.

Esta lógica se implementa en un entorno controlado y seguro, utilizando servicios modernos y tecnologías confiables como las que se describen a continuación:

2.1 Frontend

El desarrollo del frontend de **LibriX** se ha centrado en crear una experiencia de usuario fluida, moderna y accesible desde cualquier dispositivo.

El objetivo principal ha sido ofrecer una interfaz intuitiva, rápida y visualmente atractiva, que facilite la navegación y lectura, incluso para usuarios con poca experiencia digital. Para ello, se ha optado por un enfoque basado en componentes reutilizables, siguiendo buenas prácticas de desarrollo web en cliente.

La arquitectura del frontend está organizada por rutas, páginas y componentes, lo que permite mantener un código limpio, escalable y fácil de mantener.

Cada vista (por ejemplo, el catálogo, el lector, el perfil o el panel de estadísticas) se implementa como un componente principal, que a su vez se compone de subcomponentes.

También se ha tenido en cuenta el diseño responsive, adaptando cada elemento visual al tamaño de pantalla correspondiente (PC, móvil o tablet). Para ello, se ha utilizado un framework de diseño basado en clases utilitarias, lo que ha acelerado considerablemente el proceso de maquetación.

2.1.1 Tecnologías usadas

- **React (v19)**: Biblioteca principal para la construcción del interfaz del usuario. Se eligió por su enfoque declarativo, la reutilización de componentes y su gran comunidad de soporte.
- **Tailwind CSS (v3.4.17)**: Framework de utilidades CSS que permite maquetar de forma rápida y consistente sin escribir CSS personalizado. Facilita la creación de interfaces modernas y adaptativas.
- **Vite (v6.2)**: Herramienta para desarrollo rápido que mejora la velocidad de compilación y recarga en caliente frente a alternativas como Webpack. Es especialmente útil en proyectos React.
- **React Router**: Manejador de rutas que permite dividir la aplicación en múltiples páginas lógicas, manteniendo la navegación fluida sin recarga del navegador.

2.1.2 Entorno de desarrollo

- **Visual Studio Code**: Editor de código utilizado por su ligereza, extensibilidad y compatibilidad con tecnologías web modernas.
- **Node.js (v22.13)**: Entorno de ejecución de JavaScript utilizado para la gestión de dependencias (mediante npm) y ejecución del entorno de desarrollo.
- **Vercel**: Plataforma de despliegue continuo utilizada para el hosting del frontend. Se integra fácilmente con GitHub y permite automatizar la publicación de versiones.
- **GitHub**: Sistema de control de versiones distribuido, utilizado para el seguimiento del código, colaboración y despliegue con Vercel.

2.2 Backend

El backend de **LibriX** se ha desarrollado utilizando **Supabase**, una plataforma que ofrece un conjunto de servicios backend integrados, construidos sobre PostgreSQL. Este enfoque *Backend as a Service* permite reducir la complejidad del servidor, centrándose en la lógica de negocio y en la integración segura con el frontend.

La elección de Supabase responde a varios motivos clave: su facilidad de integración con React, su interfaz gráfica para la gestión de tablas, usuarios y reglas de seguridad, así como su soporte nativo para almacenamiento y funciones personalizadas. Todo ello sin necesidad de desarrollar un servidor intermedio, lo que reduce el tiempo de desarrollo y los posibles errores.

Se han implementado varias tablas en la base de datos: usuario, libro, suscripcion, autor, favorito, lectura_usuario, comentario....

Además, se han definido políticas de seguridad (RLS) para que cada usuario solo acceda a sus propios datos, y funciones SQL personalizadas para operaciones más avanzadas.

2.2.1 Tecnologías usadas

- **Supabase**: Plataforma principal del backend, que cubre:
    - **Base de datos PostgreSQL**: Para almacenamiento relacional estructurado.
    - **Reglas de seguridad (RLS)**: Para proteger los datos según el rol y la identidad del usuario.
    - **Almacenamiento**: Espacio en la nube para portadas de libros y archivos EPUB e imágenes para los autores.
    - **Funciones SQL** : Lógica ejecutada directamente en la base de datos (por ejemplo, para registrar eventos o mantener integridad referencial).

Supabase fue elegida por ser una solución moderna, open-source y muy integrada con el stack de frontend utilizado, lo que ha facilitado el desarrollo.

2.2.2 Entorno de desarrollo

- **Panel web de Supabase**: Herramienta visual integrada en la plataforma que permite la creación y edición de tablas, definición de relaciones y claves, ejecución de consultas SQL, configuración de políticas de seguridad (RLS), y gestión de usuarios y almacenamiento de archivos.

3. Documentación técnica

3.1 Análisis

Durante el proceso de análisis y planificación del proyecto **LibriX**, se han elaborado diversos diagramas que han servido como guía estructural para el desarrollo. Estos elementos han permitido definir claramente la funcionalidad, la organización de los datos y la interacción entre los diferentes actores del sistema. A continuación, se describen los principales documentos y diagramas utilizados:

Diagrama de Casos de Uso

Se ha desarrollado un diagrama de casos de uso para identificar las acciones principales que pueden realizar los distintos tipos de usuarios (usuarios no registrados, usuarios registrados y administrador). Este diagrama ha servido como base para definir los requisitos funcionales de la aplicación.

Casos principales:

1. **Usuario no registrado**

• Ver catálogo y géneros

Los visitantes de la plataforma pueden acceder al catálogo general de libros digitales. Pueden navegar por los géneros disponibles sin necesidad de estar registrados, lo que les permite explorar la oferta antes de crear una cuenta.

• Registrarse / Iniciar sesión

Funcionalidad que permite a nuevos usuarios crear una cuenta en la plataforma proporcionando datos como nombre, correo electrónico y contraseña. Los usuarios existentes pueden iniciar sesión para acceder a sus funciones personalizadas y su perfil, mediante un sistema seguro de autenticación.

---

2. **Usuario registrado**

• **Ver catálogo y géneros**

Igual que los usuarios no registrados, los usuarios con cuenta también pueden explorar todo el catálogo de libros y géneros. Además, en este caso, pueden recibir recomendaciones personalizadas basadas en su actividad.

• **Leer libros (solo con suscripción activa)**

Los usuarios registrados que tengan una suscripción activa pueden acceder al lector digital integrado para leer libros directamente desde la plataforma. Esta función está restringida a suscriptores.

• **Gestionar suscripción**

Desde su perfil, los usuarios pueden gestionar su suscripción: activarla, cancelarla o renovarla. Esta función les permite controlar el acceso a contenido premium como la lectura de libros.

• **Modificar datos del perfil**

Funcionalidad para que el usuario pueda actualizar su información personal en cualquier momento. Esto incluye nombre, apellidos, correo electrónico, usuario, contraseña y otros datos asociados a su cuenta.

• **Ver recomendaciones**

Basado en el historial de lectura, el sistema sugiere libros que podrían ser de interés para el usuario. Estas recomendaciones buscan mejorar la experiencia personalizada de lectura.

---

3. Administrador

• **Consultar estadísticas (globales)**

El administrador tiene acceso a estadísticas globales del sistema: número total de usuarios, actividad mensual, entre otros. Esta información permite tomar decisiones estratégicas.

• Exportar datos

Función exclusiva para administradores que permite descargar información estructurada (en formato Excel) sobre usuarios, libros y suscripciones. Esta opción es útil para informes internos o respaldo de datos.

• Subir / editar / eliminar libros

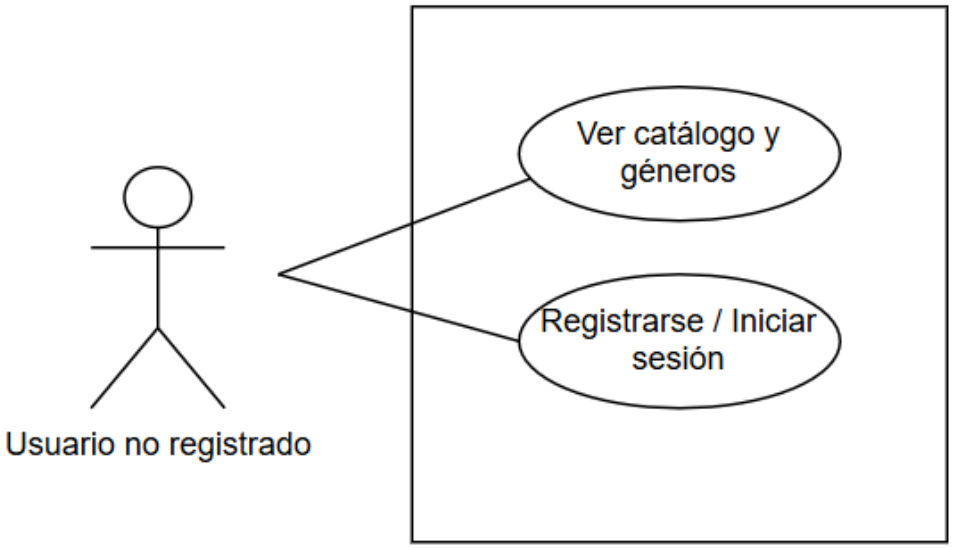
El administrador tiene control total sobre el catálogo de libros. Puede agregar nuevos libros digitales, editar la información de los existentes (título, autor, descripción…) o eliminarlos.

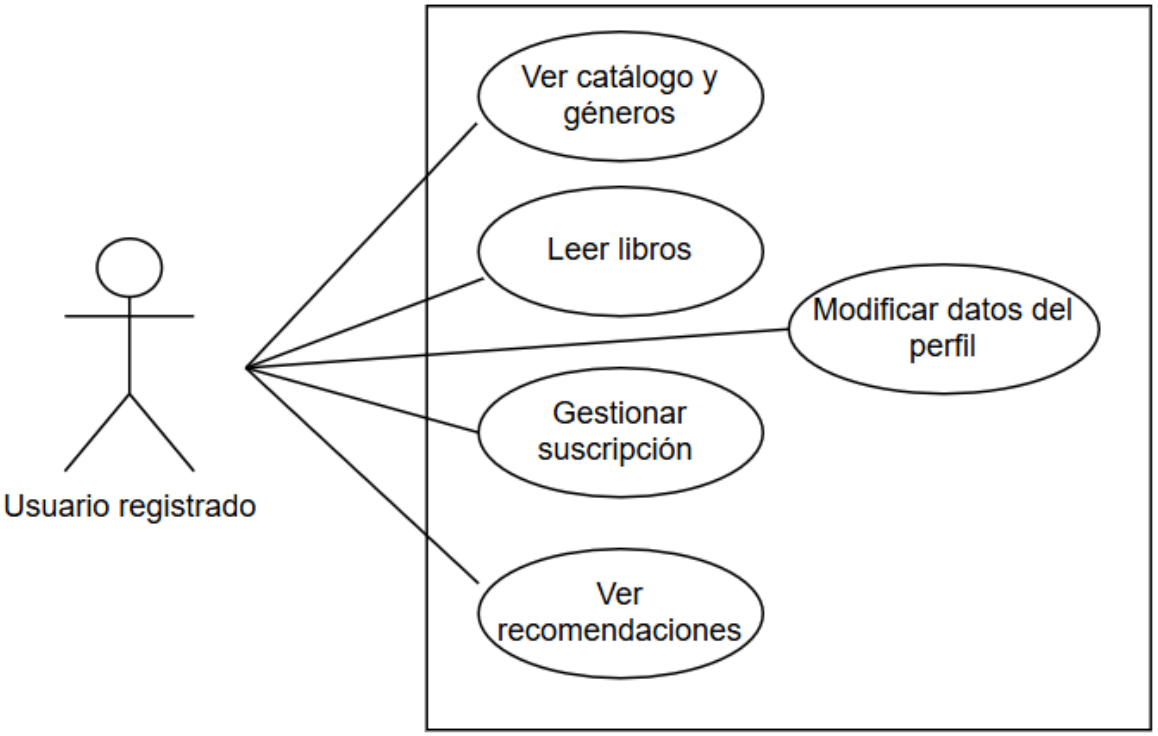
• Subir / editar / eliminar autores

Además del catálogo de libros, también puede mantener la base de datos de autores. Esto incluye agregar autores nuevos, editar su información biográfica o eliminar registros.

• Visualizar suscripciones y usuarios

El administrador puede acceder a una vista completa de todos los usuarios registrados, su estado de suscripción, fechas de registro, y otra información relevante. Esto le permite supervisar y gestionar el funcionamiento general de la plataforma.


Ver catálogo y géneros
Registrarse / Iniciar sesión
Usuario no registrado


Ver catálogo y géneros
Leer libros
Modificar datos del perfil
Gestionar suscripción
Ver recomendaciones
Usuario registrado

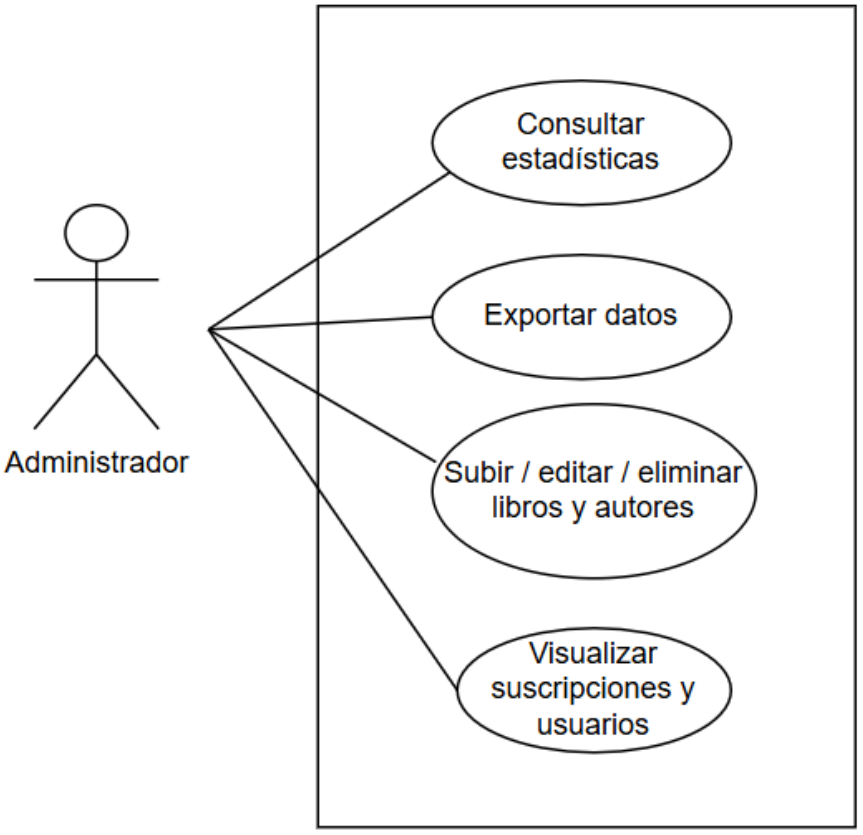

Consultar estadísticas
Exportar datos
Subir / editar / eliminar libros y autores
Visualizar suscripciones y usuarios
Administrador

Diagrama Entidad-Relación (ER)

El modelo ER ha sido clave para estructurar la base de datos relacional en Supabase. Define las principales entidades del sistema y las relaciones entre ellas. Las entidades más relevantes son:

- **Usuario**

  Almacena los datos personales y de acceso de cada usuario, incluyendo su rol (por ejemplo, lector o administrador). También se incluyen campos relacionados con suscripciones y métodos de pago (tarjeta, fecha_tarjeta, cvc).

  Campos clave:

  - **id:** Identificador único del usuario (clave primaria).
  - **rol:** Rol del usuario (por ejemplo, admin o usuario).
  - **nombre / apellidos:** Nombre y apellidos del usuario.
  - **email:** Correo electrónico para autenticación.
  - **contrasena:** Contraseña cifrada del usuario.
  - **usuario:** Alias público o nombre de usuario visible.
  - **subscripcion:** Indicación textual de la suscripción (puede ser redundante).
  - **tarjeta:** Número de tarjeta de crédito.
  - **fecha_tarjeta:** Fecha de expiración de la tarjeta.
  - **cvc:** Código de seguridad de la tarjeta.
  - **fecha_registro:** Fecha de alta del usuario en la plataforma.

  Se relaciona con:

  - **Suscripción**:
    Relación 1:N → Un usuario puede tener varias suscripciones a lo largo del tiempo.
  - **Lectura_Usuario**:
    Relación 1:N → Un usuario puede tener múltiples registros de lectura (uno por libro leído).

- **Comentario**
  Relación 1:N → Un usuario puede escribir muchos comentarios en distintos libros.
- **Favorito**:
  Relación 1:N → Un usuario puede marcar muchos libros como favoritos.

- **Libro**

  Guarda la información clave de cada libro: título, descripción, género, archivo digital (archivo_url) y portada (portada_url).

  Campos clave:

  - **id:** Identificador único del libro (clave primaria).
  - **autor_id:** Clave foránea que enlaza al autor del libro.
  - **titulo:** Título del libro.
  - **descripcion:** Descripción o sinopsis del contenido.
  - **portada_url**: URL de la imagen de la portada.
  - **archivo_url:** Enlace al archivo digital (EPUB).
  - **genero:** Género literario del libro.
  - **fecha_creacion:** Fecha en la que se añadió el libro a la base de datos.

  Se relaciona con:

  - **Autor**:
    Relación N:1 → Cada libro está escrito por un único autor.
  - **Lectura_Usuario**:
    Relación 1:N → Un libro puede ser leído por muchos usuarios.
  - **Comentario**:
    Relación 1:N → Un libro puede recibir muchos comentarios de diferentes usuarios.

- **Favorito**:
  Relación 1:N → Un libro puede estar marcado como favorito por muchos usuarios.

- **Autor**

  Incluye información personal y bibliográfica de los autores: nombre, nacionalidad, fechas y biografía.

  Campos claves:

  - **id:** Identificador único del autor (clave primaria).
  - **nombre:** Nombre completo del autor.
  - **nacionalidad:** País de origen.
  - **fecha_nacimiento:** Fecha de nacimiento.
  - **fecha_fallecimiento:** Fecha de fallecimiento (opcional).
  - **biografia:** Breve reseña o historia del autor.
  - **foto_url:** URL de la foto o retrato del autor.
  - **created_at:** Fecha de creación del registro del autor.

  Se relaciona con:

  - **Libro**:
    Relación 1:N → Un autor puede tener escritos varios libros.

- **Suscripcion**

  Registra los datos de cada suscripción realizada por los usuarios: plan (mensual o anual), fecha de inicio y fin, estado actual y claves foráneas hacia usuarios.

  Campos claves:

  - **id:** Identificador único de la suscripción (clave primaria).
  - **usuario_id:** Clave foránea que enlaza con el usuario suscriptor.
  - **plan:** Tipo de plan (mensual, anual...).
  - **estado:** Estado actual (activo, cancelado...).

- **fecha_inicio:** Fecha de inicio de la suscripción.
- **fecha_fin:** Fecha de expiración de la suscripción.
- **fecha_creacion:** Fecha de creación del registro de suscripción.

Se relaciona con:

- **Usuario**:
  Relación 1:N → Cada suscripción pertenece a un único usuario.

- **Lectura_Usuario**

Tabla intermedia que vincula usuarios y libros. Registra el avance del usuario (progreso), la última posición de lectura (cfi) y la fecha de última lectura.
Es esencial para generar recomendaciones y estadísticas personalizadas.

Campos claves:

- **id:** Identificador único del registro de lectura (clave primaria).
- **usuario_id:** Clave foránea que enlaza con el usuario lector.
- **libro_id:** Clave foránea que enlaza con el libro leído.
- **cfi:** Posición exacta dentro del EPUB (para reanudar lectura).
- **ultima_lectura:** Fecha del último acceso al libro.

Se relaciona con:

- **Usuario**:
  Relación 1:N → Cada entrada de lectura pertenece a un único usuario.
- **Libro**:
  Relación 1:N → Cada entrada de lectura corresponde a un único libro.

- **Comentario**

Tabla que almacena los comentarios que los usuarios dejan en los libros. Contiene el texto del comentario, la puntuación (estrellas) y referencias al libro_id y al usuario_id.

Campos clave:

- **id:** Identificador único del comentario (clave primaria).
- **usuario_id:** Clave foránea que enlaza con el autor del comentario.
- **libro_id:** Clave foránea que enlaza con el libro comentado.
- **nombre_usuario:** Alias del usuario (puede ser redundante).
- **comentario:** Texto completo del comentario.
- **estrellas:** Valoración en estrellas (de 1 a 5).
- **fecha:** Fecha en la que se publicó el comentario.

Se relaciona con:

- **Usuario**:
  Relación 1:N → Cada comentario es escrito por un único usuario.
- **Libro**:
  Relación 1:N → Cada comentario está asociado a un único libro.

- **Favorito**

La tabla favorito almacena los libros marcados como favoritos por los usuarios. Esta funcionalidad permite que los usuarios puedan guardar libros de su interés para acceder rápidamente a ellos en el futuro.

Campos clave:

- **usuario_id:** Clave foránea que referencia a usuario.
- **libro_id:** Clave foránea que referencia a libro.
- **fecha_creacion:** Fecha en que se marcó el libro como favorito.

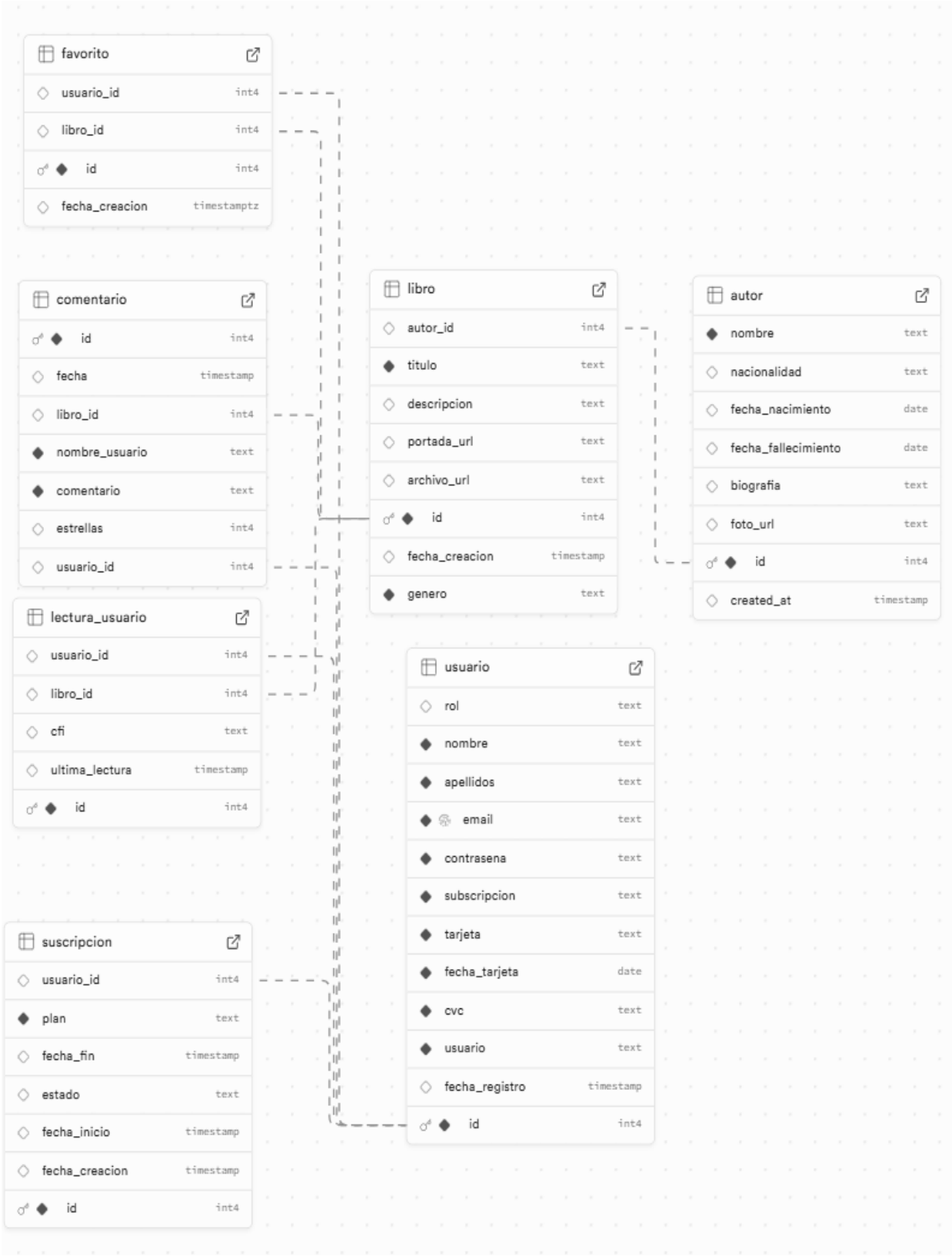
- **id:** Identificador único del favorito (innecesario si se usa una clave compuesta, pero puede usarse como simplificación técnica).

Se relaciona con:

- **Usuario**:
  Relación 1:N → Cada favorito pertenece a un único usuario.
- **Libro**:
  Relación 1:N → Cada favorito está asociado a un único libro.

Se han definido claves primarias, claves foráneas y restricciones que garantizan la integridad referencial de los datos. Este diagrama fue especialmente útil al crear las tablas y definir las relaciones entre ellas en el panel de Supabase.

A continuación se muestra el esquema del modelo E/R.


favorito
usuario_id int4
libro_id int4
id int4
fecha_creacion timestamptz
comentario
id int4
fecha timestamp
libro_id int4
nombre_usuario text
comentario text
estrellas int4
usuario_id int4
libro
autor_id int4
titulo text
descripcion text
portada_url text
archivo_url text
id int4
fecha_creacion timestamp
genero text
autor
nombre text
nacionalidad text
fecha_nacimiento date
fecha_fallecimiento date
biografia text
foto_url text
id int4
created_at timestamp
lectura_usuario
usuario_id int4
libro_id int4
cfi text
ultima_lectura timestamp
id int4
usuario
rol text
nombre text
apellidos text
email text
contrasena text
subscripcion text
tarjeta text
fecha_tarjeta date
cvc text
usuario text
fecha_registro timestamp
id int4
suscripcion
usuario_id int4
plan text
fecha_fin timestamp
estado text
fecha_inicio timestamp
fecha_creacion timestamp
id int4

Requisitos funcionales y no funcionales.

Requisitos funcionales (RF)

Gestión de Libros

- Creación, edición y eliminación de libros.
- Clasificación por género, autor, novedades, popularidad, etc.
- Visualización del catálogo.
- Visualización de la ficha detallada de cada libro.
- Posibilidad de subir libros en formato EPUB con su portada correspondiente. (solo admin)
- Asignación de autores existentes o nuevos al subir un libro. (solo admin)

Usuarios y Suscripciones

- Registro e inicio de sesión con autenticación segura.
- Gestión de suscripciones (mensual/anual).
- Perfiles de usuario con posibilidad de modificación de datos personales (nombre, correo...).
- Bloqueo de contenido de lectura a usuarios no registrados.
- Seguimiento de lecturas para usuarios registrados.
- Permitir que el usuario retome la lectura desde donde la dejó, guardando automáticamente su progreso por libro.
- Visualización de recomendaciones personalizadas en base a géneros leídos.

Lectura Online

- Lector digital de libros EPUB integrado.
- Ajustes de visualización (tamaño de letra, tipo de fuente, cambio de fondo).

- Paginación con teclas del teclado y botones.
- Vista previa de capítulos o índice del libro si está disponible.

Panel de Administración

- Gestión de libros y autores: añadir, modificar, eliminar o listar.
- Gestión de usuarios y suscripciones.
- Consulta de actividad de usuarios.
- Panel de estadísticas
- Análisis de suscripciones activas.
- Exportación de datos en formato Excel (libros, usuarios, autores).

Requisitos No Funcionales (RNF)

- Disponibilidad 24/7: Accesible en cualquier momento desde cualquier dispositivo conectado.
- Seguridad: Cifrado de datos sensibles. Control de accesos por roles (usuario, administrador).
- Escalabilidad: El sistema debe estar preparado para un catálogo con cientos/miles de libros sin degradar el rendimiento.
- Usabilidad: Interfaz clara, accesible, responsive y coherente en móviles, tablets y escritorio.
- Compatibilidad: Funcionalidad completa en los navegadores (Chrome principalmente).

3.2 Pruebas realizadas

Prueba 1: Registro de usuario

Componente: Register.jsx

Tabla implicada: usuario (Supabase)

Juegos de datos de entrada:

- Nombre: "Laura"
- Apellido: "Gómez"
- Email: lauragomez@gmail.com
- Suscripción: "Mensual"
- Número de tarjeta: "4357287625429873"
- Fecha de vencimiento: "12/2027"
- CVC: "698"
- Usuario: "laura_g"
- Contraseña: "Laurita2025@"

Resultado de salida:

- El sistema registra al usuario correctamente.
- Se guarda la contraseña en formato cifrado.
- Se crea la suscripción y la tarjeta asociada.
- El usuario es redirigido al dashboard inicial.

Prueba 2: Inicio de sesión de un usuario

Componente: Login.jsx

Tabla implicada: usuario (Supabase)

Juegos de datos de entrada:

- Email: "lauragomez@gmail.com"
- Contraseña: "Laurita2025@"

Resultado de salida:

- El sistema autentica al usuario correctamente.
- Se valida el hash de la contraseña.
- Se inicia sesión y se almacena el token en localStorage.

Prueba 3: Comentario y puntuación de libro

Componente: Comentarios.jsx

Tabla implicada: comentario (Supabase)

Juegos de datos de entrada:

- Usuario: "laura_g"
- Libro: "El retrato de Dorian Gray"
- Comentario: "Me ha encantado la narrativa, muy fluida."
- Puntuación: 5

Resultado de salida:

- El sistema guarda el comentario y la puntuación correctamente.
- El libro actualiza su media de estrellas.
- El comentario se renderiza instantáneamente con avatar y nombre.

Prueba 4: Edición de perfil

Componente: PerfilUsuario.jsx

Tablas implicadas: usuario (Supabase)

Juegos de datos de entrada:

- Cambio de email a "laura.gomez@hotmail.com"
- Cambio de fecha de vencimiento a "01/2030"

Resultado de salida:

- Los datos se actualizan correctamente en Supabase.
- Se muestra notificación visual de éxito (toast).

Prueba 5: Añadir libro a favoritos

Componente: DetalleLibro.jsx

Tablas implicadas: favoritos (Supabase)

Juegos de datos de entrada:

- Usuario laura_g
- ID del libro: 15

Resultado de salida:

- El sistema añade el libro a la lista de favoritos del usuario.
- El ícono de "corazón" cambia de color para indicar el guardado.
- La lista de favoritos se actualiza en tiempo real en el perfil.

Prueba 6: Subida de nuevo libro (admin)

Componente: AñadirLibro.jsx

Tablas implicadas: libro (Supabase)

Juegos de datos de entrada:

- Título: "Cien años de soledad"
- Autor: "Gabriel García Márquez"
- Género: "Realismo mágico"
- Descripción: "Una obra maestra que mezcla lo real y lo fantástico."
- Archivo EPUB: cien_anos.epub
- Portada: portada.jpg

Resultado de salida:

- Se sube correctamente el archivo EPUB y la imagen a Supabase Storage.
- Se crea un nuevo registro en la tabla libro.
- El libro aparece disponible en el catálogo para los usuarios.

4. Proceso de despliegue

Para desplegar mi aplicación React he utilizado **Vercel** y el repositorio está alojado en **GitHub**. A continuación, explico brevemente el proceso seguido tanto en local como en la plataforma de despliegue.

4.1 Desarrollo en local

Comencé creando una aplicación en React utilizando **Vite** (v.6.2.0) mediante el comando **npm create vite@latest**, eligiendo la plantilla correspondiente y asignándole un nombre al proyecto. Luego accedí a la carpeta generada (cd nombre-del-proyecto) e instalé todas las dependencias necesarias con **npm install (v.10.9.2)**.

Una vez instalada la base del proyecto, probé la aplicación localmente para asegurarme de que funcionaba correctamente ejecutando **npm run dev**.

Esto permitió visualizarla en el navegador a través de un servidor local, en este caso, el puerto es **localhost:5173.**

Una vez verificado su funcionamiento, inicialicé un repositorio Git en la carpeta del proyecto. Añadí todos los archivos al control de versiones, realicé el primer commit con los cambios iniciales.

Luego, creé un repositorio en GitHub y lo conecté con el repositorio local. Finalmente, subí el proyecto a la plataforma para tenerlo disponible online y poder enlazarlo con Vercel.

4.2 Despliegue en Vercel

Con el proyecto ya subido a GitHub, accedí a la plataforma de **Vercel** y me identifiqué con mi cuenta de GitHub para permitir la integración entre ambas plataformas.

Una vez dentro, seleccioné la opción para crear un nuevo proyecto y elegí el repositorio que había subido previamente. Vercel detectó automáticamente que se trataba de una aplicación desarrollada con React y Vite, por lo que no fue necesario configurar nada manualmente.

Tras confirmar los ajustes por defecto, inicié el proceso de despliegue. En pocos segundos, Vercel compiló el proyecto y generó un enlace público desde el cual ya era posible acceder a la aplicación en línea.

Además, gracias a la integración con GitHub, cualquier cambio que realice en el código y suba al repositorio se desplegará automáticamente en Vercel, manteniendo siempre la versión más reciente de la aplicación publicada.

5. Propuestas de mejoras

1. **Modo oscuro inteligente:** Adaptar el tema visual según la hora del día o la configuración del sistema operativo del usuario.
2. **Gamificación:** Implementar logros, insignias y niveles por lectura completada, comentarios publicados o retos mensuales.
3. **Verificación 2FA (doble factor):** Añadir un paso adicional de seguridad al iniciar sesión.
4. **Revisión de dispositivos conectados:** Opción para cerrar sesiones remotas desde el perfil.
5. **Pasarela de pagos extendida:** Integración con más métodos de pago (Bizum, Google Pay, etc.).
6. **Marcadores y anotaciones:** Permitir al usuario marcar páginas clave, subrayar fragmentos y dejar notas personales sincronizadas con su cuenta.
7. **Soporte multilenguaje:** Traducción completa de la interfaz, e incluso metadatos de libros si están disponibles.
8. **Clubes de lectura virtuales:** Espacios sociales donde los usuarios pueden leer el mismo libro, compartir comentarios por capítulos y debatir en tiempo real o por foros internos.
9. **Comentarios con reacciones:** Los usuarios pueden reaccionar a los comentarios de otros con emojis o puntuaciones.

6. Bibliografía

Supabase

Plataforma de backend como servicio que proporciona base de datos PostgreSQL y almacenamiento. Se utilizó como backend principal para la creación de las tablas y almacenamiento de los epubs y fotos.

Sitio web: https://supabase.com

EmailJS

Servicio que permite enviar correos electrónicos directamente desde el frontend. Utilizado en este proyecto para el envío automático de confirmación en el formulario de contacto.

Sitio web: https://www.emailjs.com

Frase del día

Repositorio de GitHub que expone una API con frases célebres de autores. Se utilizó para mostrar una frase diaria en la pantalla de inicio de la biblioteca digital.

Repositorio: https://github.com/LuisJorgeLozano/FraseDelDia

EPUBReader (react-reader)

Componente React de código abierto para renderizar archivos EPUB en el navegador. Integrado en el proyecto para ofrecer una experiencia de lectura digital dentro de la plataforma LibriX.

Repositorio: https://github.com/gerhardsletten/react-reader

bcryptjs

Librería utilizada para cifrar contraseñas antes de almacenarlas en la base de datos, mejorando la seguridad del sistema.

Repositorio: https://github.com/dcodeIO/bcrypt.js

React

Librería de JavaScript para construir interfaces de usuario. Utilizada como frontend de la aplicación LibriX.

Sitio web: https://react.dev

Tailwind CSS

Framework de CSS utilitario que permite diseñar interfaces modernas y responsivas mediante clases directamente en el JSX. Utilizado para el diseño de la plataforma. Sitio web: https://tailwindcss.com