Tabla de contenido

[1. Introducción 1](#_Toc198899016)

[1.1. Objetivo del reto 1](#_Toc198899017)

[1.2. Temática elegida 1](#_Toc198899018)

[2. Estrategia de implementación 2](#_Toc198899019)

[2.1. Tecnologías utilizadas 2](#_Toc198899020)

[2.2. Estructura de carpetas del proyecto 4](#_Toc198899021)

[3. Frontend 5](#_Toc198899022)

[3.1. Paleta de colores 5](#_Toc198899023)

[3.2. Tipografía 7](#_Toc198899024)

[3.3. Mockups 8](#_Toc198899025)

[4. Backend 17](#_Toc198899026)

[4.1. CRUD y manejo de rutas 17](#_Toc198899027)

[4.2. Autenticación 17](#_Toc198899028)

[4.3. Validación de datos 18](#_Toc198899029)

[5. Base de datos 19](#_Toc198899030)

[5.1. Diagrama entidad-relación 19](#_Toc198899031)

[5.2. Script de la Base de Datos 20](#_Toc198899032)

[6. API 22](#_Toc198899033)

[6.1. Descripción de los endpoints 22](#_Toc198899034)

[6.2. Métodos, parámetros y ejemplos 22](#_Toc198899035)

[7. Pruebas 23](#_Toc198899036)

[7.1. Create 23](#_Toc198899037)

[7.2. Read 23](#_Toc198899038)

[7.3. Update 23](#_Toc198899039)

[7.4. Delete 23](#_Toc198899040)

[8. Anexos 24](#_Toc198899041)

[8.1. Repositorio GitHub 24](#_Toc198899042)

[8.2. Demo 25](#_Toc198899043)

# Introducción

## Objetivo del reto

Convertir un wireframe en una aplicación funcional, demostrando habilidades en HTML y CSS para construir la estructura y el estilo de la interfaz, así como en JavaScript para manipular el DOM y consumir API’s.

Además, desarrollar un backend capaz de interactuar con una base de datos y exponer la información al frontend a través de endpoints definidos.

## Temática elegida

La temática elegida para el desarrollo del reto es una aplicación web que permita a los usuarios llevar un registro de las series que han visto. Cada usuario puede agregar series a su historial personal, registrar su progreso y asignar calificaciones a las series vistas.

Para acceder al sistema, se implementa un mecanismo de autenticación mediante JSON Web Token (JWT), lo que garantiza que cada usuario solo pueda ver y modificar su propia información.

Esta temática permite trabajar con relaciones entre entidades como usuarios y series, implementar seguridad en el acceso, consumir API’s, y gestionar datos dinámicos tanto en el frontend como en el backend.

# Estrategia de implementación

## Tecnologías utilizadas

Para el desarrollo de la aplicación se seleccionaron tecnologías modernas que permiten implementar una solución robusta, segura y escalable. A continuación, se listan dichas tecnologías.

|  |  |
| --- | --- |
| Diseño | |
| Tecnología | Descripción |
| Canva | Para la elaboración de los mockups. |
| Diagrams.net (draw.io) | Para el diagrama de arquitectura y el diagrama de entidad-relación. |

|  |  |
| --- | --- |
| Frontend | |
| Tecnología | Descripción |
| HTML y CSS | Lenguajes base para estructura y estilo |
| JavaScript | Lógica del cliente y consumo de API’s. |
| Vue.js | Framework JavaScript para construir interfaces reactivas. |

|  |  |
| --- | --- |
| Backend | |
| Tecnología | Descripción |
| Spring Boot 2.5.2 | Framework de Java para construir la API REST. |
| Java 11 | Lenguaje base del backend. |
| Spring Data JPA | Acceso y persistencia de datos en MySQL. |
| Spring Web | Controladores REST. |
| Lombok | Reducción de boilerplate (getters y setters). |
| Argon2 | Hash seguro de contraseñas. |
| JJWT | Generación y validación de JSON Web Tokens. |

|  |  |
| --- | --- |
| Base de Datos | |
| Tecnología | Descripción |
| MySQL | Sistema de gestión de bases de datos relacional. |
| Laragon | Entorno local para ejecutar MySQL y herramientas auxiliares. |

|  |  |
| --- | --- |
| Pruebas | |
| Tecnología | Descripción |
| Postman | Para pruebas manuales de endpoints de la API. |
| Spring Boot Starter Test | Pruebas unitarias del backend. |

|  |  |
| --- | --- |
| Control de Versiones | |
| Tecnología | Descripción |
| Git/GitHub | Control de cambios y repositorio del proyecto. |

## Estructura de carpetas del proyecto

# Frontend

En esta sección se detallan los aspectos clave que definen el diseño visual y la experiencia de usuario de la aplicación. Se cubren los siguientes tres aspectos: 1. Paleta de colores para crear un ambiente visual coherente, 2. Las fuentes elegidas para una buena legibilidad y 3. Los mockups que sirven como guía para construir la interfaz.

## Paleta de colores

Para el proyecto se utilizará la siguiente paleta de cinco colores en donde cada color cumple un rol específico dentro de la interfaz. Cada color está especificado en formatos HEX y RGB para facilitar su implementación tanto en diseño como en desarrollo.

|  |  |  |
| --- | --- | --- |
| Color de fondo |  | Para la base visual de la interfaz y proporcionar un entorno neutro que permita destacar los elementos interactivos y jerárquicos del diseño.  HEX: #F9F9F9 | RGB(249, 249, 249) |

|  |  |  |
| --- | --- | --- |
| Color primario |  | Para los elementos de acción principal, como botones primarios, enlaces destacados, encabezados o llamadas a la acción.  HEX: #D9ABFF | RGB(217, 171, 255) |

|  |  |  |
| --- | --- | --- |
| Color secundario |  | Para complementar al color primario y aplicarlo en botones secundarios, fondos alternos, tarjetas, iconos o elementos menos destacados.  HEX: #ABE4FF | RGB(171, 228, 255) |

Se incluyen además colores de apoyo destinados a representar estados del sistema como éxito, error o advertencia.

|  |  |  |
| --- | --- | --- |
| Estado “exitoso” |  | Para indicar acciones completadas correctamente o estados positivos como notificaciones de confirmación, etiquetas de éxito o validaciones de formularios.  HEX: #DDFFAB | RGB(221, 255, 171) |

|  |  |  |
| --- | --- | --- |
| Estado “advertencia” |  | Para alertar al usuario sobre acciones que requieren atención como mensajes preventivos o validaciones incompletas.  HEX: #FFDAAB | RGB(255, 218, 171) |

|  |  |  |
| --- | --- | --- |
| Estado “error” |  | Para representar estados negativos o fallos como formularios no válidos o alertas críticas.  HEX: #FFABAB | RGB(255, 171, 171) |

## Tipografía

Para el proyecto se seleccionaron fuentes de Google Fonts para asegurar compatibilidad web, carga rápida y facilidad de integración sin necesidad de importar archivos .ttf manualmente.

La elección tipográfica busca transmitir una interfaz moderna y clara, adecuada para una experiencia de usuario cómoda en distintos dispositivos. A continuación, se listan las fuentes seleccionadas y su uso.

|  |  |
| --- | --- |
| Fuente | Descripción |
| Quicksand | Para el texto de los títulos y encabezados.  Es una tipografía geométrica con bordes ligeramente redondeados. Su diseño aporta un estilo moderno y mantiene una estructura sólida que facilita la lectura. |
| Open Sans | Para el texto del cuerpo.  Es una tipografía altamente utilizada en entornos web debido a su neutralidad, legibilidad y adaptabilidad. Su diseño permite que el texto sea fácilmente leído por el usuario, incluso en pantallas pequeñas. |

## Mockups

Los mockups presentados en esta sección representan una visualización previa de la interfaz final, sirviendo como guía para el desarrollo de los componentes del frontend. Estos diseños ayudan a planificar la distribución de elementos, jerarquía visual y navegación entre pantallas. A partir de ellos se definieron los estilos, la estructura general y la interacción esperada por parte del usuario.

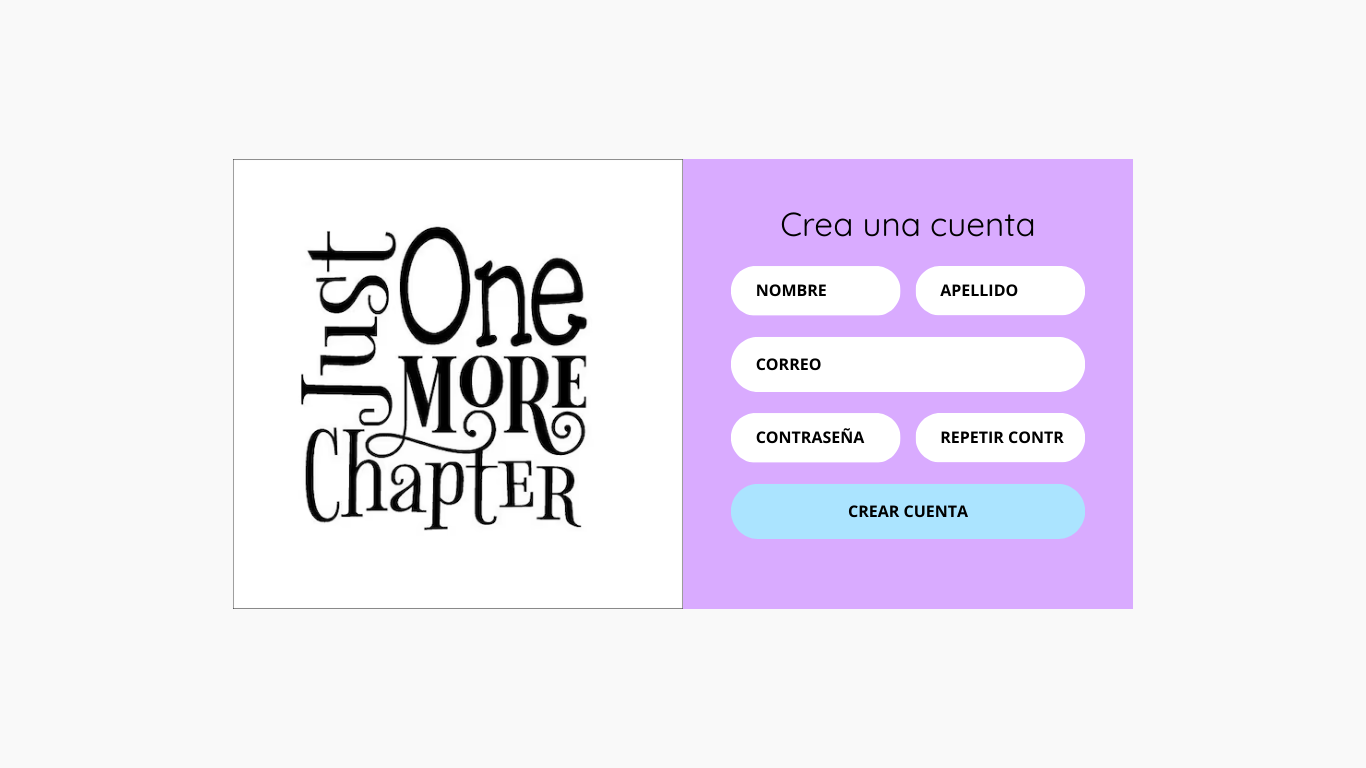
**Pantallas generales**

Login

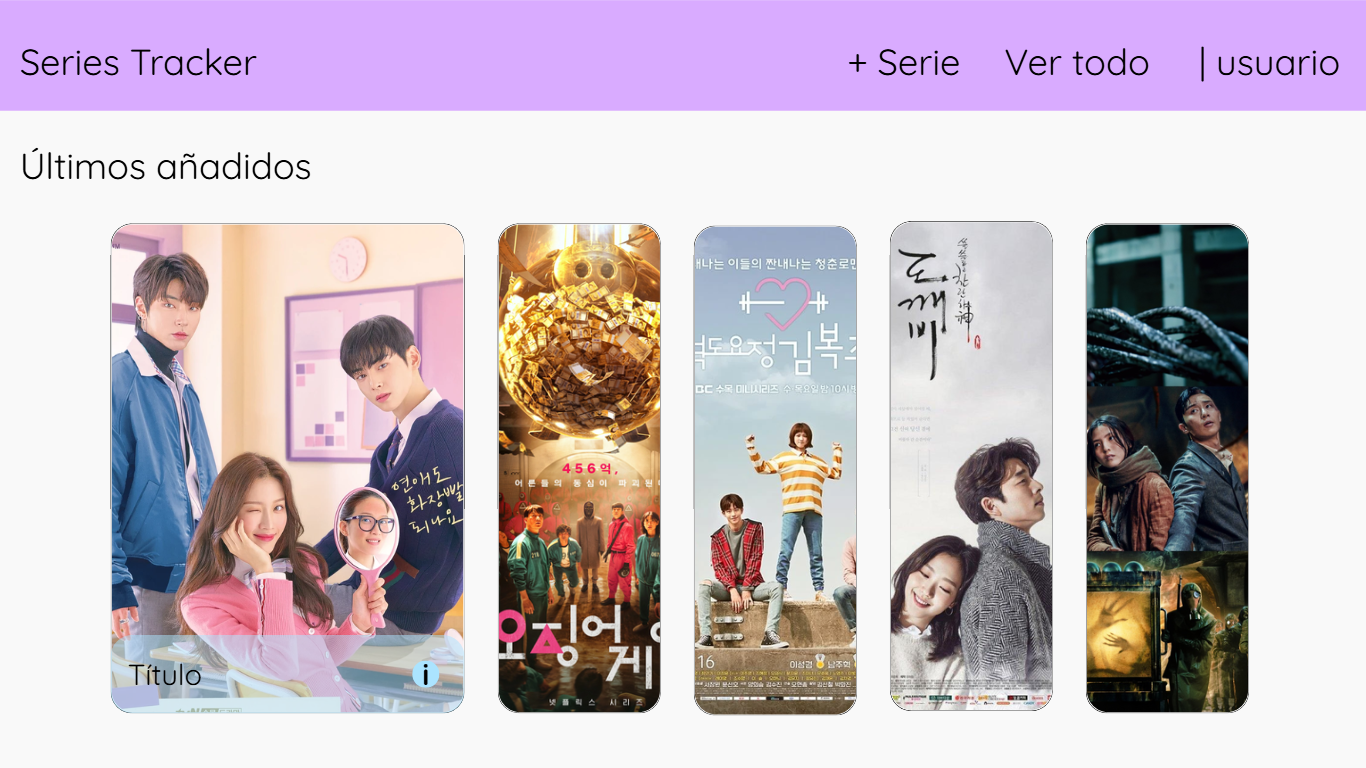
****

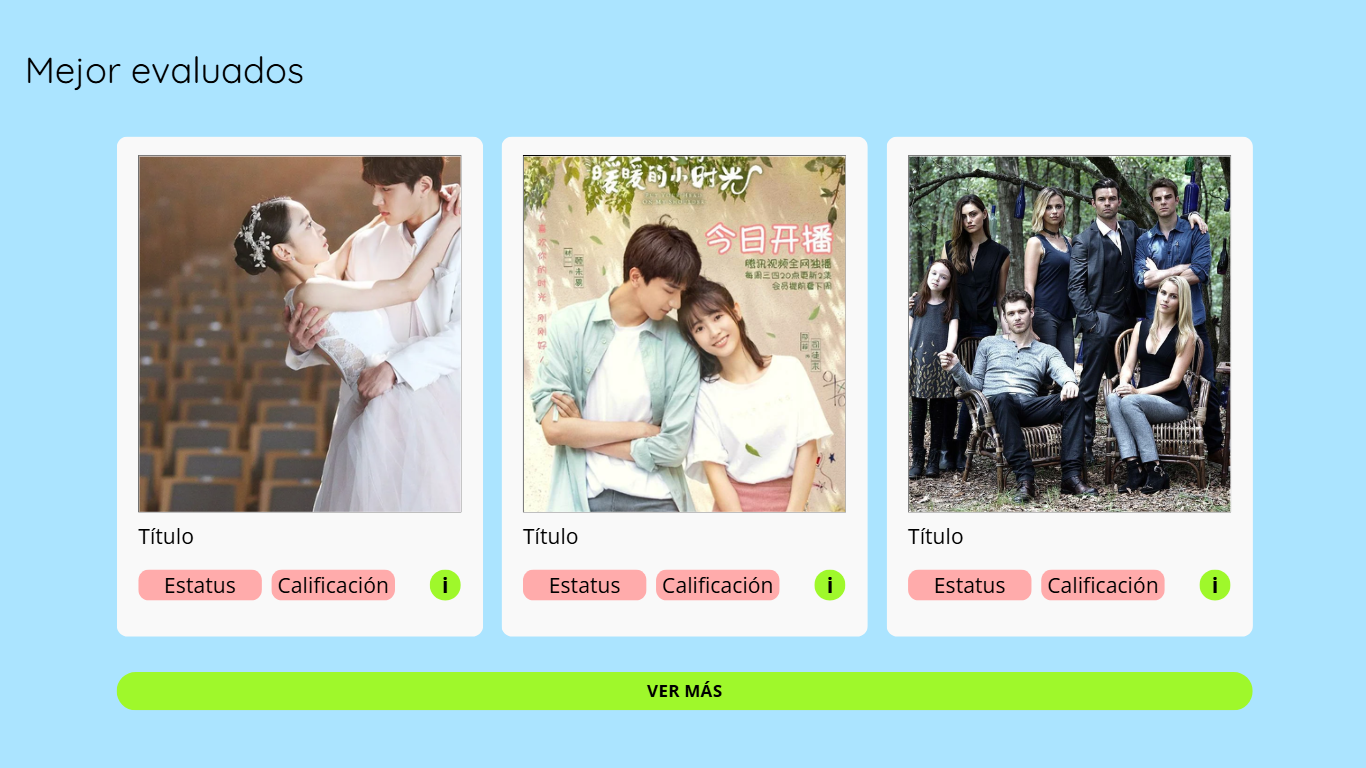
**Pantallas para el usuario**

Crear cuenta

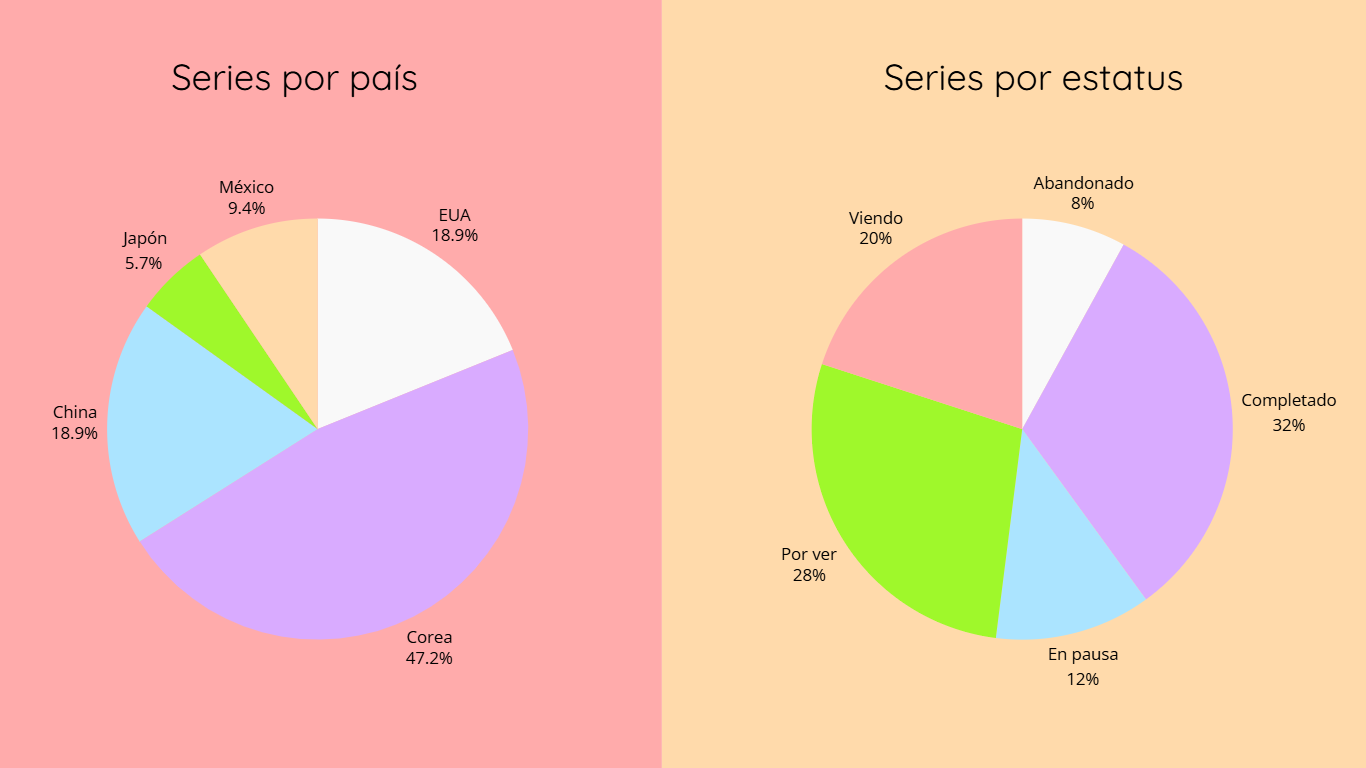
****

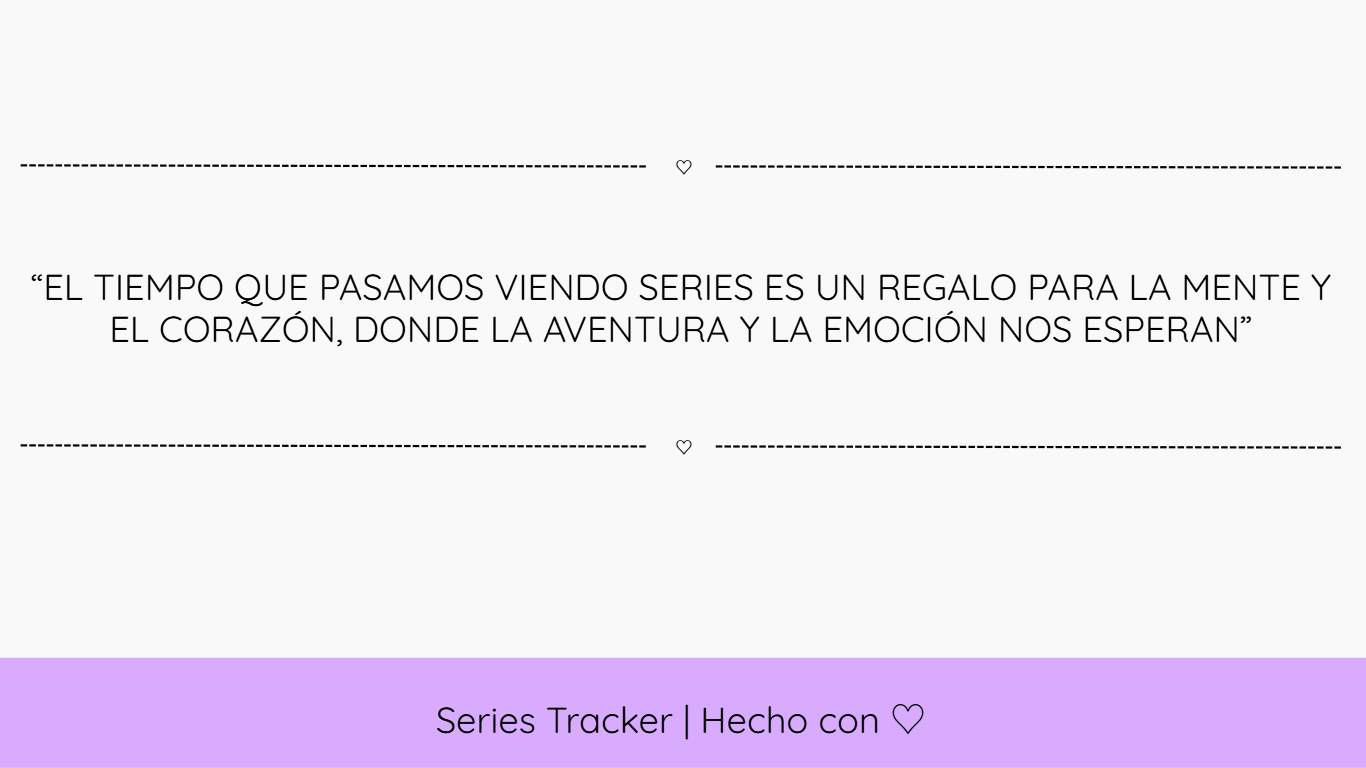
Inicio



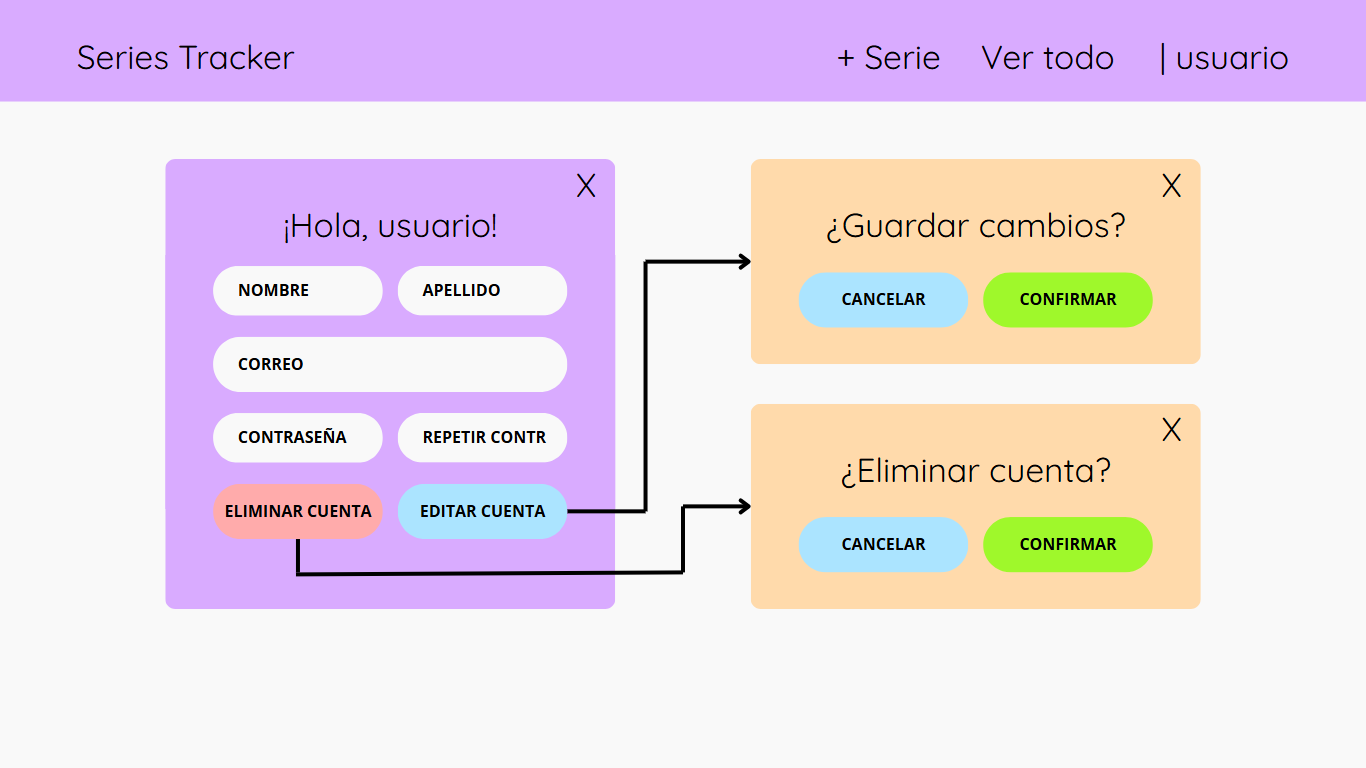








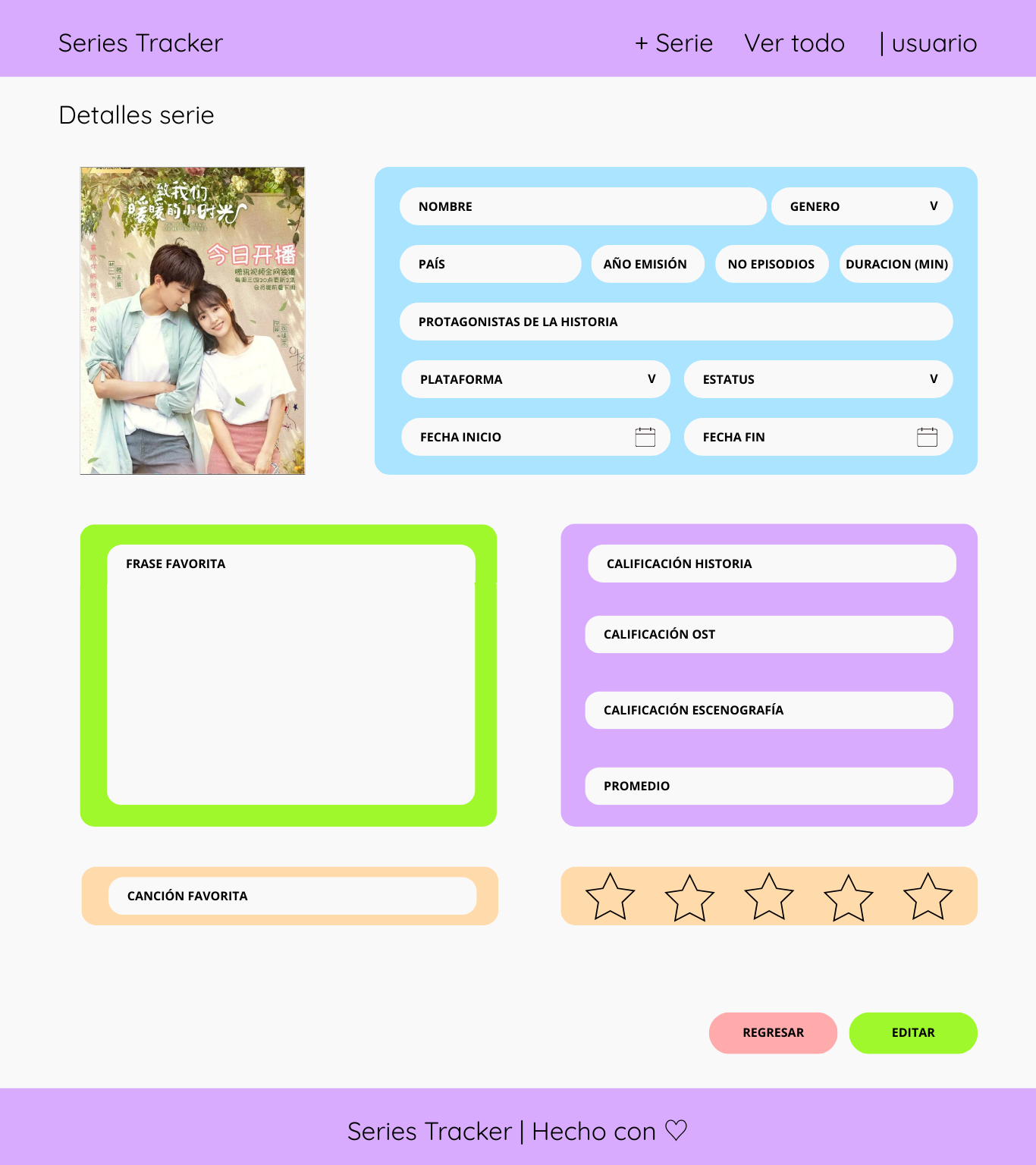
Perfil de usuario (modals)



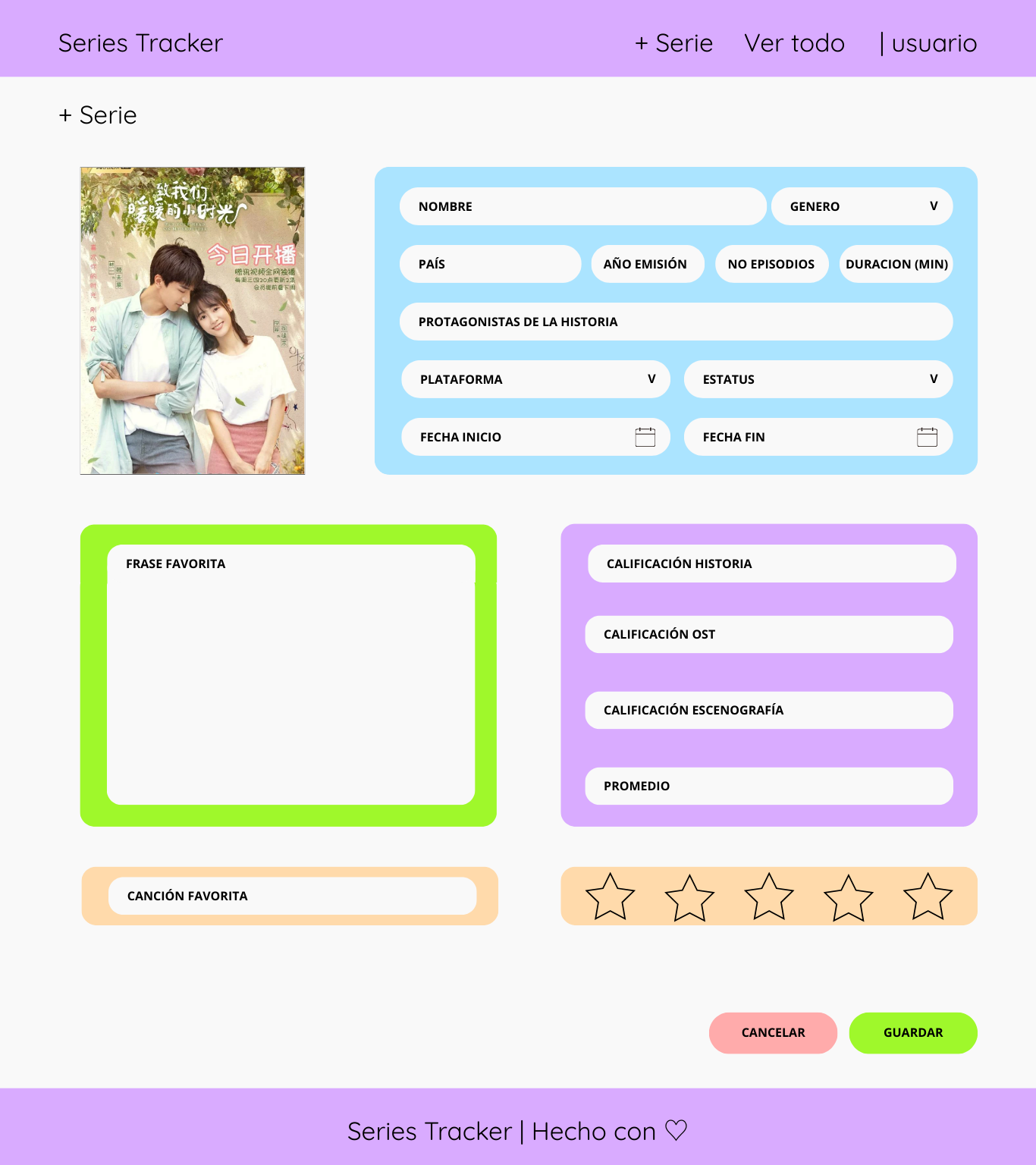
Información general serie (modal)



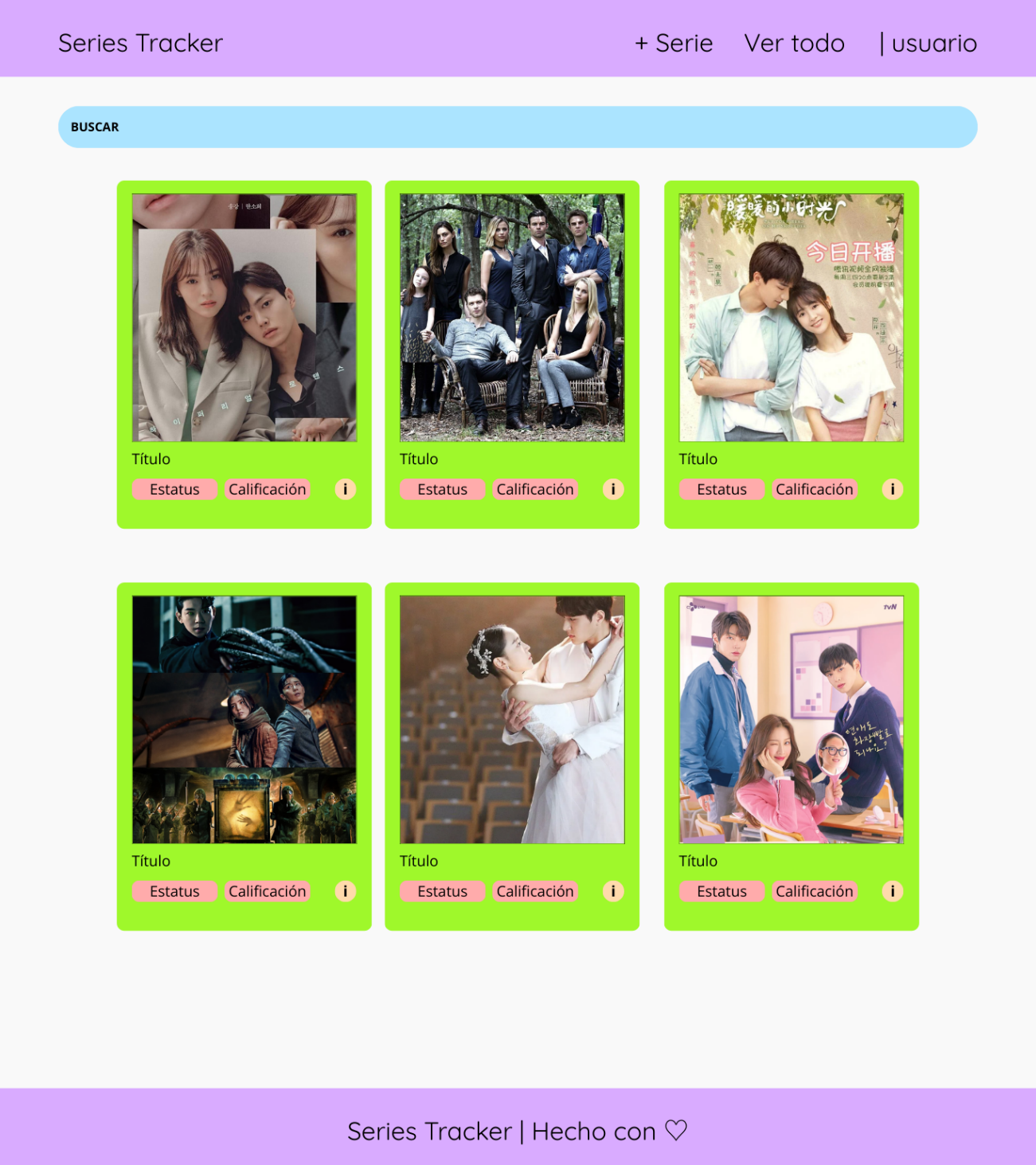
Detalles serie



Agregar serie



Listar series



Editar serie



# Backend

En esta sección se describen los componentes fundamentales que conforman la lógica y estructura del backend de la aplicación. Se cubren los siguientes tres aspectos: 1. La implementación del CRUD y el manejo de rutas para interactuar con los datos, 2. El sistema de autenticación con JWT que garantiza el acceso seguro de los usuarios y 3. La validación de datos para asegurar la integridad y consistencia de la información procesada por el sistema.

## CRUD y manejo de rutas

El backend implementado con Spring Boot expone una API REST que permite realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar) sobre las entidades Usuario, Serie y UsuarioSerie. Cada ruta está asociada a un controlador, y se accede mediante métodos HTTP estándar (GET, POST, PUT, DELETE). Estas rutas están organizadas de forma clara y separadas por recurso, lo que facilita su mantenimiento y consumo desde el frontend.

## Autenticación

Para controlar el acceso a la aplicación, se implementó un sistema de autenticación basado en JSON Web Tokens (JWT). Los usuarios deben iniciar sesión para recibir un token que será enviado en cada solicitud posterior mediante el encabezado Authorization. Este mecanismo permite proteger rutas sensibles y diferenciar entre usuarios autenticados y no autenticados, garantizando que solo el dueño de los datos pueda acceder a ellos o modificarlos.

## Validación de datos

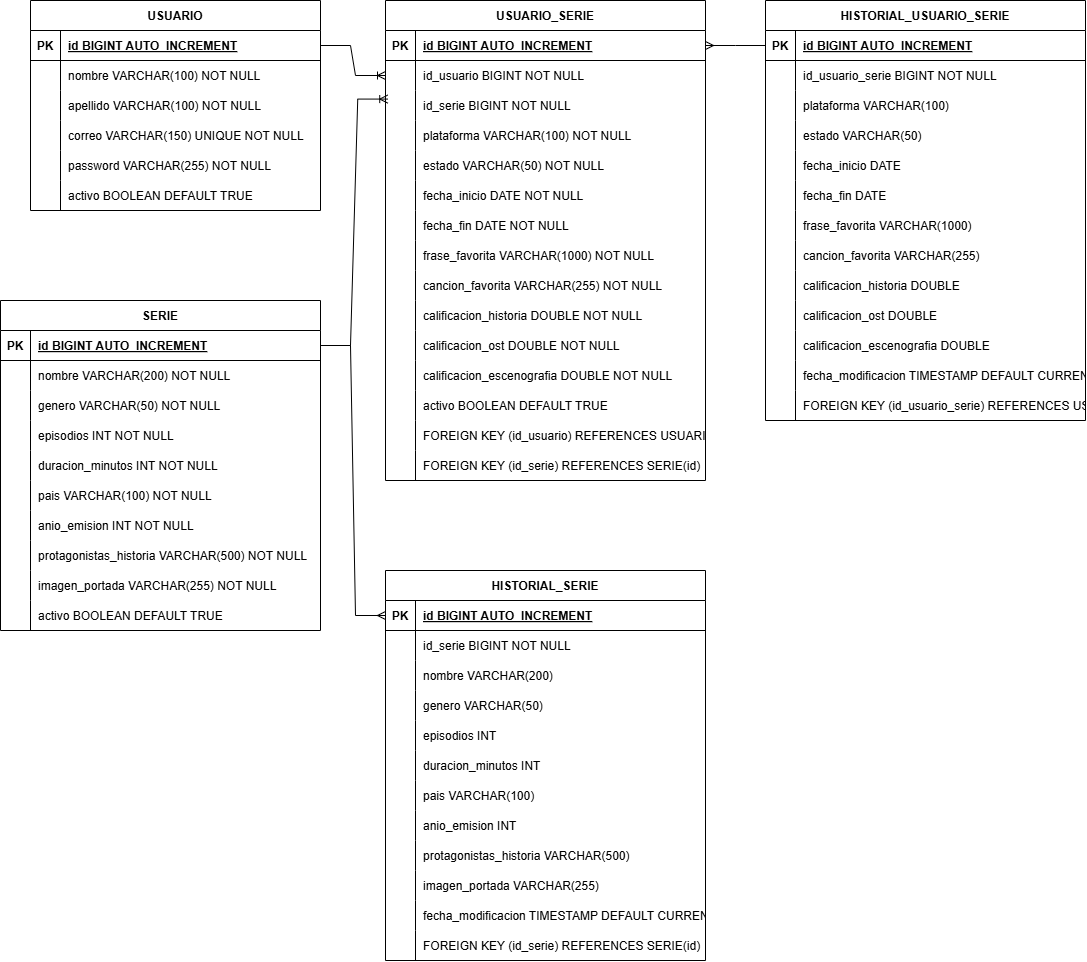
Antes de procesar cualquier solicitud, el backend realiza validaciones para asegurar la integridad y seguridad de los datos. Se verifica, por ejemplo, que los campos obligatorios estén presentes, que los tipos de datos sean correctos, y que no se repitan registros únicos como correos electrónicos. Estas validaciones se aplican tanto a nivel de entidad como en los controladores, ayudando a prevenir errores y ataques comunes.

# Base de datos

Para el funcionamiento de la aplicación es necesario contar con una base de datos que almacene la información relacionada con los usuarios, las series y la interacción entre ellos.

## Diagrama entidad-relación

A continuación, se presenta el diagrama entidad-relación que visualiza la estructura de la base de datos y cómo se interrelacionan las diferentes entidades.



## Script de la Base de Datos

A continuación, se presenta el script SQL que crea la base de datos SERIES\_TRACKER y las tablas correspondientes para almacenar los datos relevantes. Este script está diseñado para MySQL y utiliza convenciones estándar para definir las tablas, claves primarias, claves foráneas y las relaciones entre las entidades.

|  |
| --- |
| -- Crear base de datos CREATE DATABASE SERIES\_TRACKER; USE SERIES\_TRACKER;  -- Tabla USUARIO CREATE TABLE USUARIO (  id BIGINT AUTO\_INCREMENT PRIMARY KEY,  nombre VARCHAR(100) NOT NULL,  apellido VARCHAR(100),  correo VARCHAR(150) UNIQUE NOT NULL,  password VARCHAR(255) NOT NULL  activo BOOLEAN DEFAULT TRUE );  -- Tabla SERIE CREATE TABLE SERIE (  id BIGINT AUTO\_INCREMENT PRIMARY KEY,  nombre VARCHAR(200) NOT NULL,  genero VARCHAR(50) NOT NULL,  episodios INT NOT NULL,  duracion\_minutos INT NOT NULL,  pais VARCHAR(100) NOT NULL,  anio\_emision INT NOT NULL,  protagonistas\_historia VARCHAR(500) NOT NULL,  imagen\_portada VARCHAR(255) NOT NULL,  activo BOOLEAN DEFAULT TRUE );  -- Tabla USUARIO\_SERIE CREATE TABLE IF NOT EXISTS USUARIO\_SERIE (  id BIGINT AUTO\_INCREMENT PRIMARY KEY,  id\_usuario BIGINT NOT NULL,  id\_serie BIGINT NOT NULL,  plataforma VARCHAR(100) NOT NULL,  estado VARCHAR(50) NOT NULL,  fecha\_inicio DATE NOT NULL,  fecha\_fin DATE NOT NULL,  frase\_favorita VARCHAR(1000) NOT NULL,  cancion\_favorita VARCHAR(255) NOT NULL,  calificacion\_historia DOUBLE NOT NULL,  calificacion\_ost DOUBLE NOT NULL,  calificacion\_escenografia DOUBLE NOT NULL,  activo BOOLEAN DEFAULT TRUE,  FOREIGN KEY (id\_usuario) REFERENCES USUARIO(id),  FOREIGN KEY (id\_serie) REFERENCES SERIE(id) );  -- Tabla HISTORIAL\_USUARIO\_SERIE CREATE TABLE IF NOT EXISTS HISTORIAL\_USUARIO\_SERIE (  id BIGINT AUTO\_INCREMENT PRIMARY KEY,  id\_usuario\_serie BIGINT NOT NULL,  plataforma VARCHAR(100),  estado VARCHAR(50),  fecha\_inicio DATE,  fecha\_fin DATE,  frase\_favorita VARCHAR(1000),  cancion\_favorita VARCHAR(255),  calificacion\_historia DOUBLE,  calificacion\_ost DOUBLE,  calificacion\_escenografia DOUBLE,  fecha\_modificacion TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  FOREIGN KEY (id\_usuario\_serie) REFERENCES USUARIO\_SERIE(id) );  -- Tabla HISTORIAL\_SERIE CREATE TABLE IF NOT EXISTS HISTORIAL\_SERIE (  id BIGINT AUTO\_INCREMENT PRIMARY KEY,  id\_serie BIGINT NOT NULL,  nombre VARCHAR(200),  genero VARCHAR(50),  episodios INT,  duracion\_minutos INT,  pais VARCHAR(100),  anio\_emision INT,  protagonistas\_historia VARCHAR(500),  imagen\_portada VARCHAR(255),  fecha\_modificacion TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  FOREIGN KEY (id\_serie) REFERENCES SERIE(id) ); |

# API

## Descripción de los endpoints

## Métodos, parámetros y ejemplos

# Pruebas

## Create

## Read

## Update

## Delete

# Anexos

## Repositorio GitHub

El código fuente del proyecto, incluyendo el frontend y el backend, se encuentra disponible en el siguiente repositorio de GitHub. En él se puede consultar la estructura del proyecto, los controladores, servicios, entidades, así como los archivos de configuración y documentación técnica adicional.

|  |  |
| --- | --- |
| https://github.com/SalmaCzSz/Series-Tracker.git | |
| Rama | Descripción |
| master | Contiene la versión estable y final del proyecto. Incluye tanto el frontend como el backend integrados, funcionando de forma correcta y lista para producción. |
| develop | Contiene las últimas funcionalidades en desarrollo. Aquí se reciben los avances diarios al final del día. |
| day-1 | Contiene los avances correspondientes al día 22 de mayo, como parte del periodo de prueba. Incluye la base del proyecto y la documentación inicial. |
| day-2 | Contiene los avances correspondientes al día 23 de mayo, como parte del periodo de prueba. Incluye el diseño de las vistas y ajustes en la base de datos. |
| day-3 | Contiene los avances correspondientes al día 26 de mayo, como parte del periodo de prueba. |

## Demo