Tabla de contenido

[1. Introducción 1](#_Toc199351429)

[1.1. Objetivo del reto 1](#_Toc199351430)

[1.2. Temática elegida 1](#_Toc199351431)

[2. Estrategia de implementación 2](#_Toc199351432)

[2.1. Tecnologías utilizadas 2](#_Toc199351433)

[2.2. Estructura de carpetas del proyecto 4](#_Toc199351434)

[3. Frontend 6](#_Toc199351435)

[3.1. Paleta de colores 6](#_Toc199351436)

[3.2. Tipografía 8](#_Toc199351437)

[3.3. Mockups 9](#_Toc199351438)

[4. Backend 18](#_Toc199351439)

[4.1. CRUD y manejo de rutas 18](#_Toc199351440)

[4.2. Autenticación 18](#_Toc199351441)

[4.3. Validación de datos 19](#_Toc199351442)

[5. Base de datos 20](#_Toc199351443)

[5.1. Diagrama entidad-relación 20](#_Toc199351444)

[5.2. Script de la Base de Datos 21](#_Toc199351445)

[6. API 23](#_Toc199351446)

[6.1. Auth 24](#_Toc199351447)

[6.2. Serie 25](#_Toc199351448)

[6.3. Usuario 27](#_Toc199351449)

[6.4. UsuarioSerie 31](#_Toc199351450)

[7. Pruebas 33](#_Toc199351451)

[7.1. Inicio de sesión 33](#_Toc199351452)

[7.2. Registro de usuario 35](#_Toc199351453)

[7.3. Navegación entre pantallas 36](#_Toc199351454)

[7.4. Gestión de series 39](#_Toc199351455)

[7.5. Visualización de detalles 40](#_Toc199351456)

[7.6. Diseño responsivo 41](#_Toc199351457)

[7.7. Manejo de estados vacíos 42](#_Toc199351458)

[8. Anexos 43](#_Toc199351459)

[8.1. Repositorio GitHub 43](#_Toc199351460)

[8.2. Demo 43](#_Toc199351461)

# 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

El proyecto está dividido en tres partes principales: el frontend, desarrollado con Vue.js; el backend, construido con Spring Boot; y una sección dedicada a la documentación.

Esta estructura modular permite separar las responsabilidades de cada parte del sistema, facilitando el desarrollo, mantenimiento y colaboración en equipo. A continuación, se describen las carpetas principales y su función dentro del proyecto.

|  |  |  |
| --- | --- | --- |
| Series-Tracker | | |
|  | Subcarpeta | Descripción |
| Documents | Ninguna | Sección para los diagramas y documentación técnica del proyecto. |
| Frontend | public | Sección para archivos estáticos que no pasan por el proceso de build. |
| src | |
| assets | Sección para archivos estáticos que sí se procesan (CSS globales y fuentes personalizadas). |
| components | Sección para componentes que se reutilizan en las views (nabvar y footer). |
| router | Sección para definir la vista que se muestra según la URL. |
| views | Sección para las páginas completas o rutas principales del sitio. |
| src | main>java>com.series.tracker | |
| controllers | Sección para las clases que manejan las peticiones HTTP para conectar el frontend con el backend. |
| dao | Sección para las clases o interfaces que interactúan con la base de datos. |
| enums | Sección para guardar enumeraciones que representan valores fijos. |
| models | Sección para las clases que representan las entidades de la base de datos. |
| utils | Sección para clases auxiliares (JWT). |
| main>resources  Incluye el application.properties para definir la configuración de la aplicación. | |
| db | Sección para archivos SQL. |

# 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.

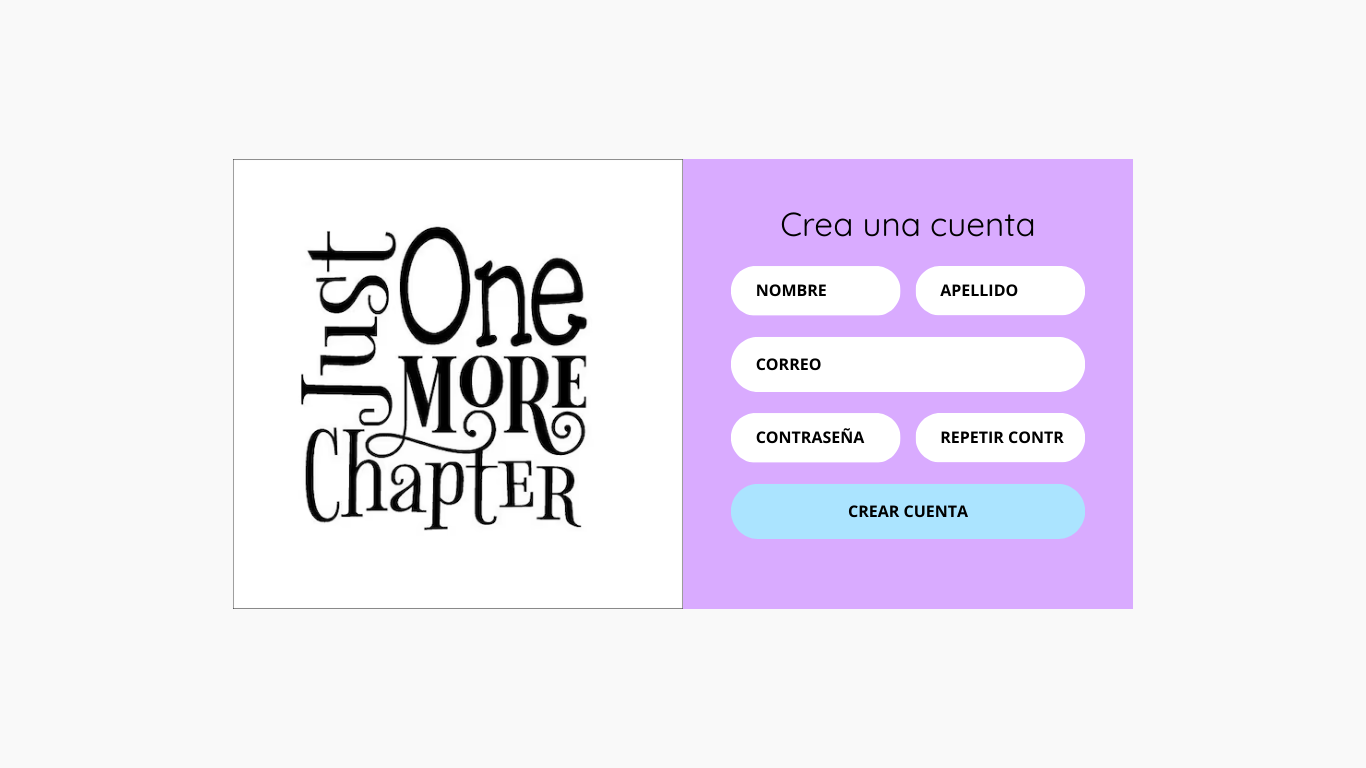
**Pantallas generales**

Login

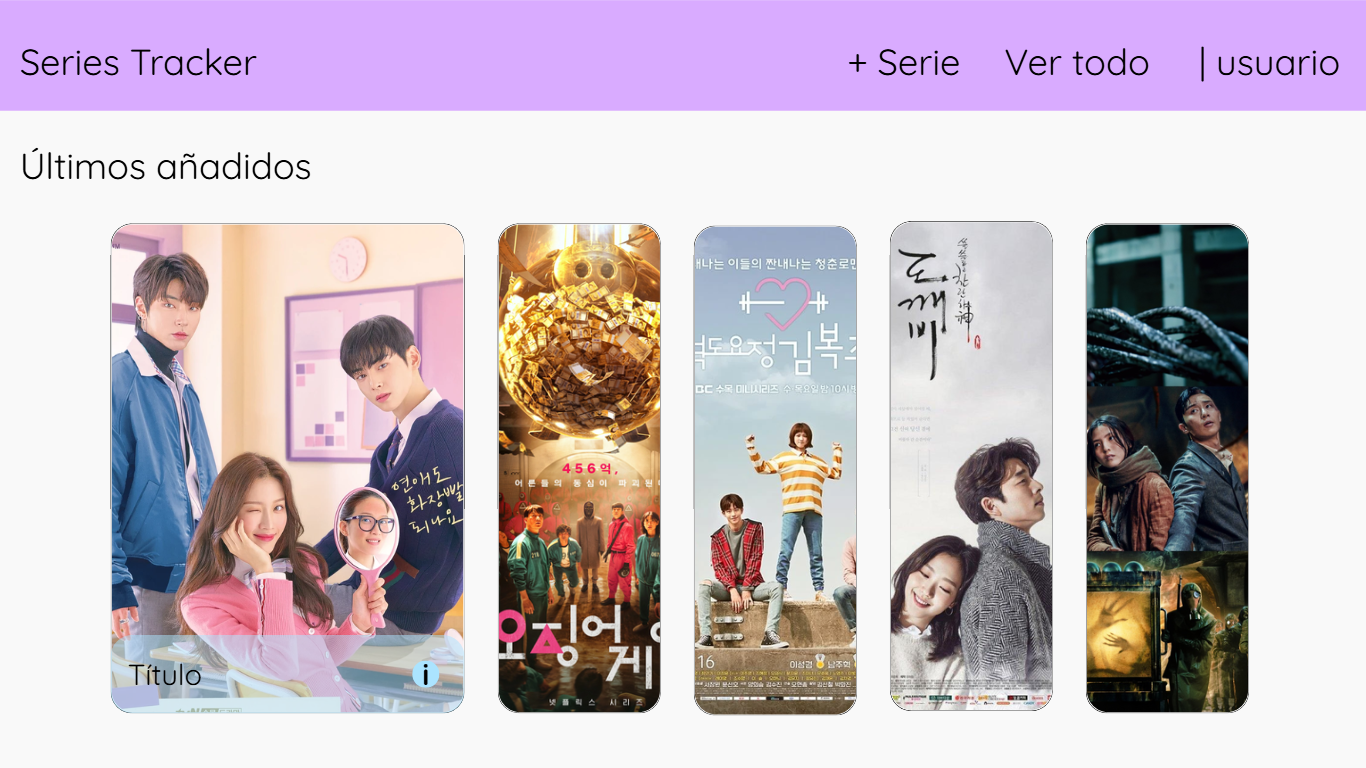
****

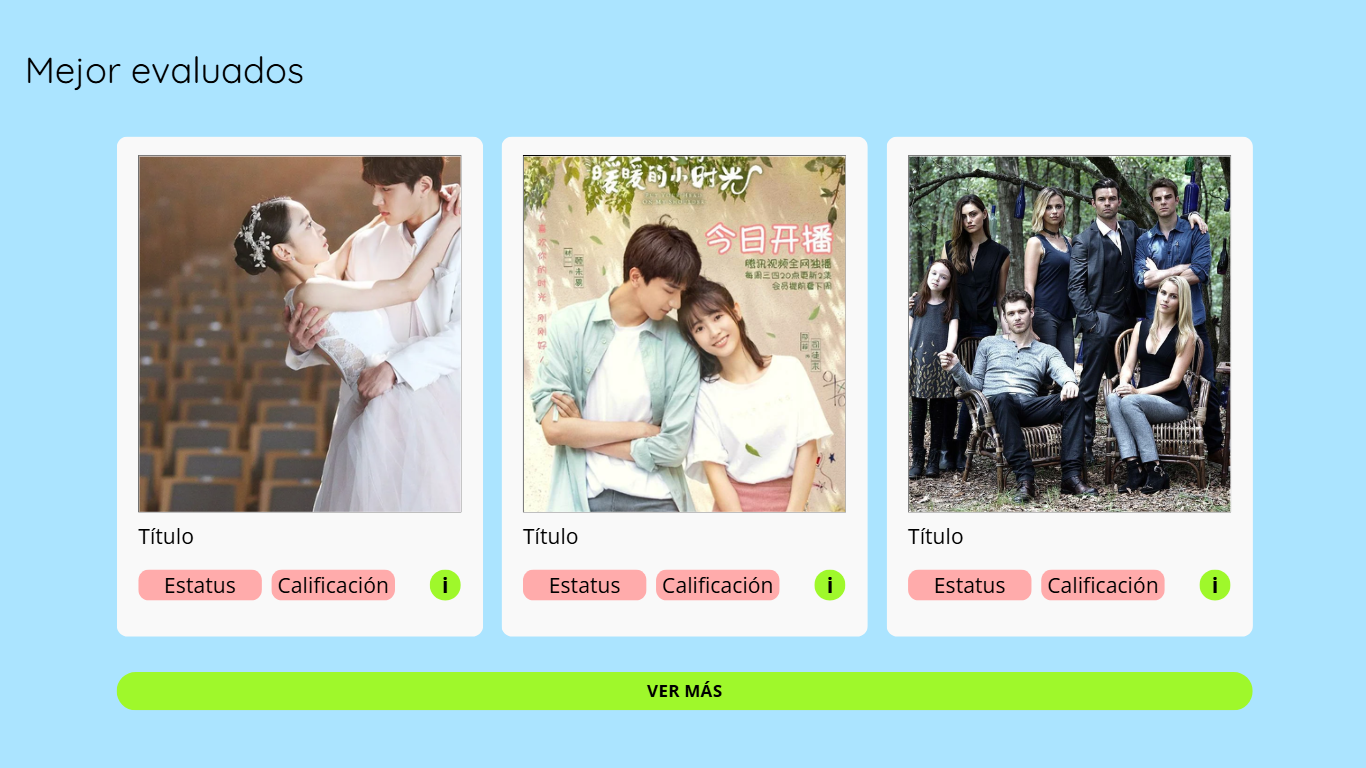
**Pantallas para el usuario**

Crear cuenta

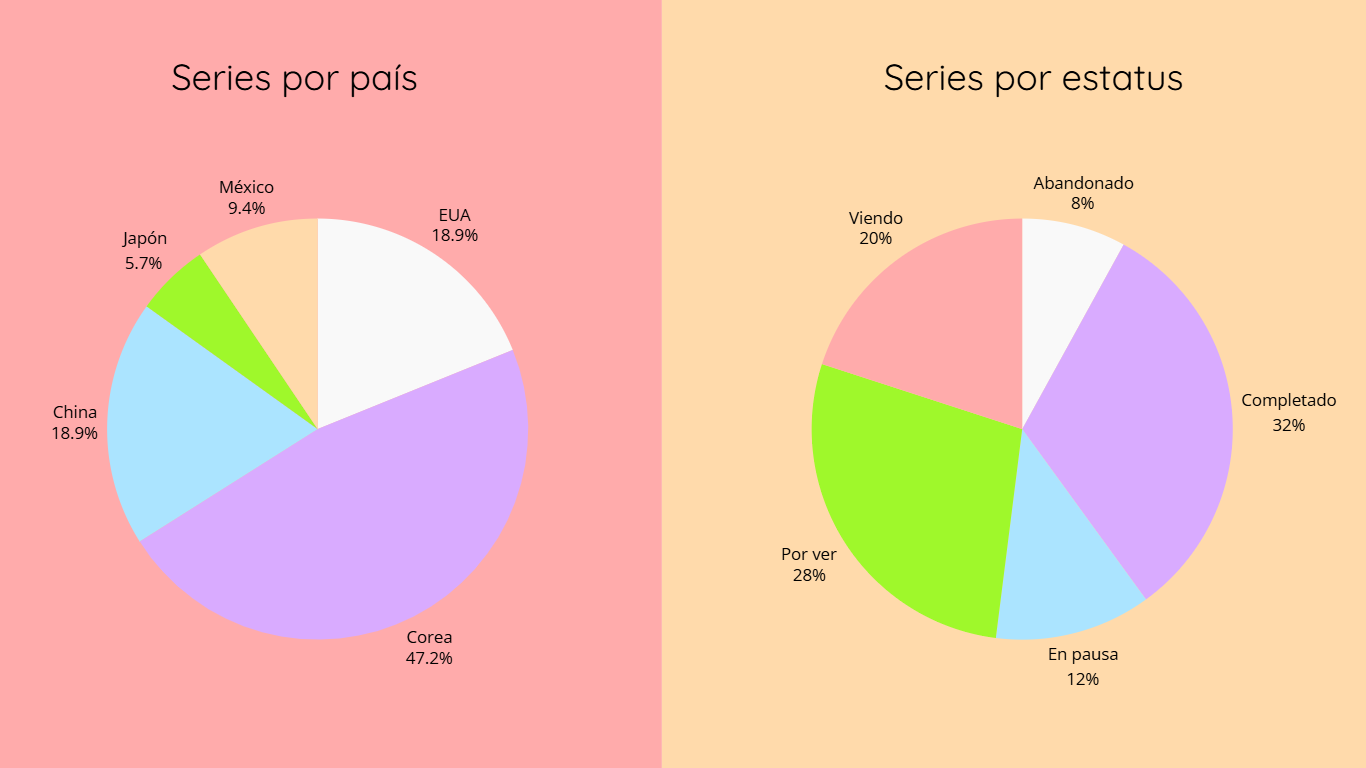
****

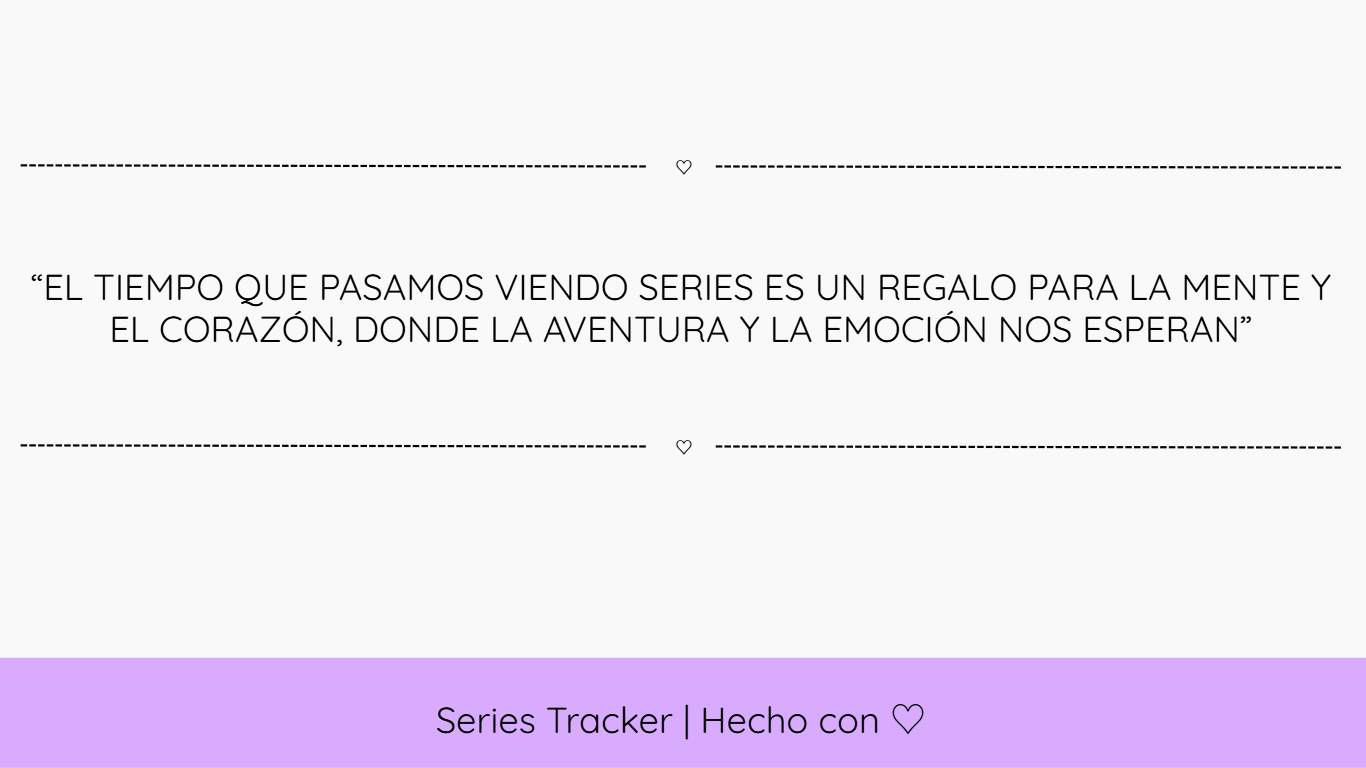
Inicio



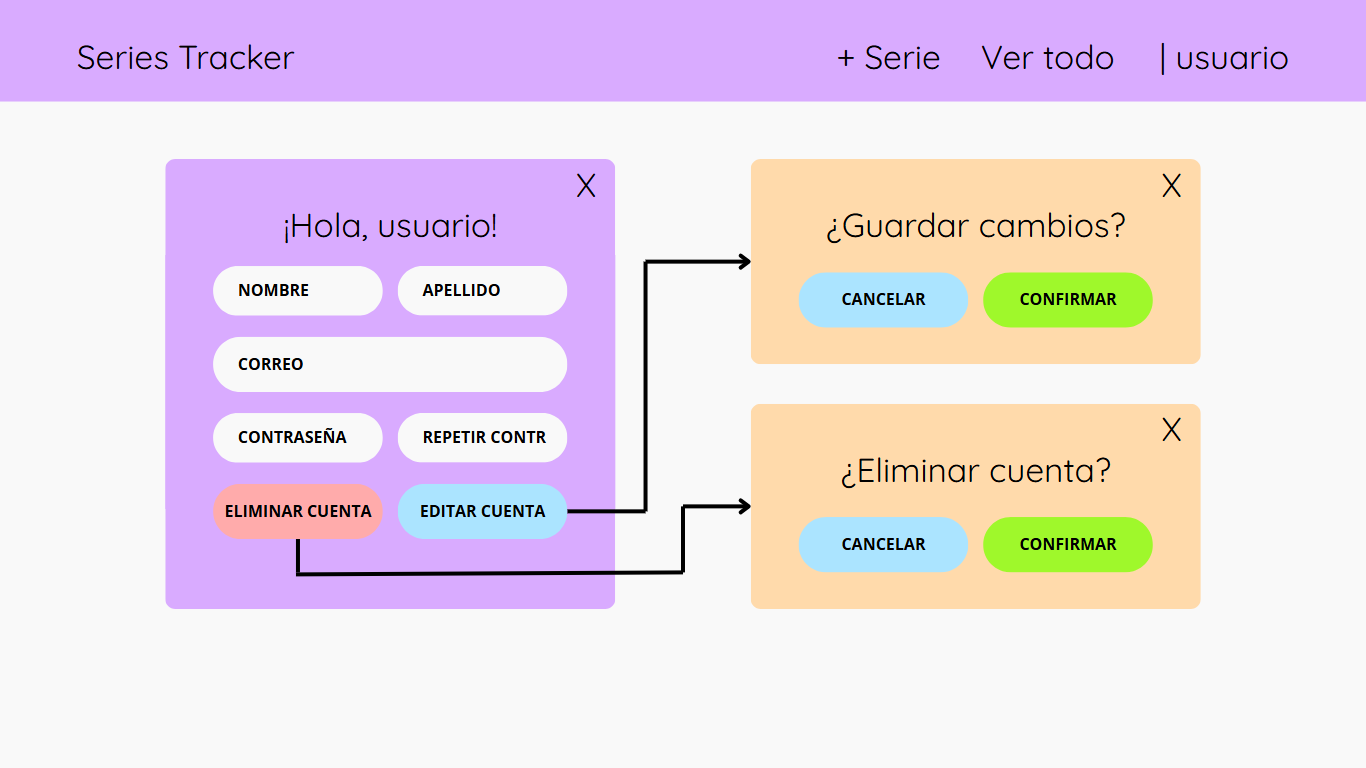








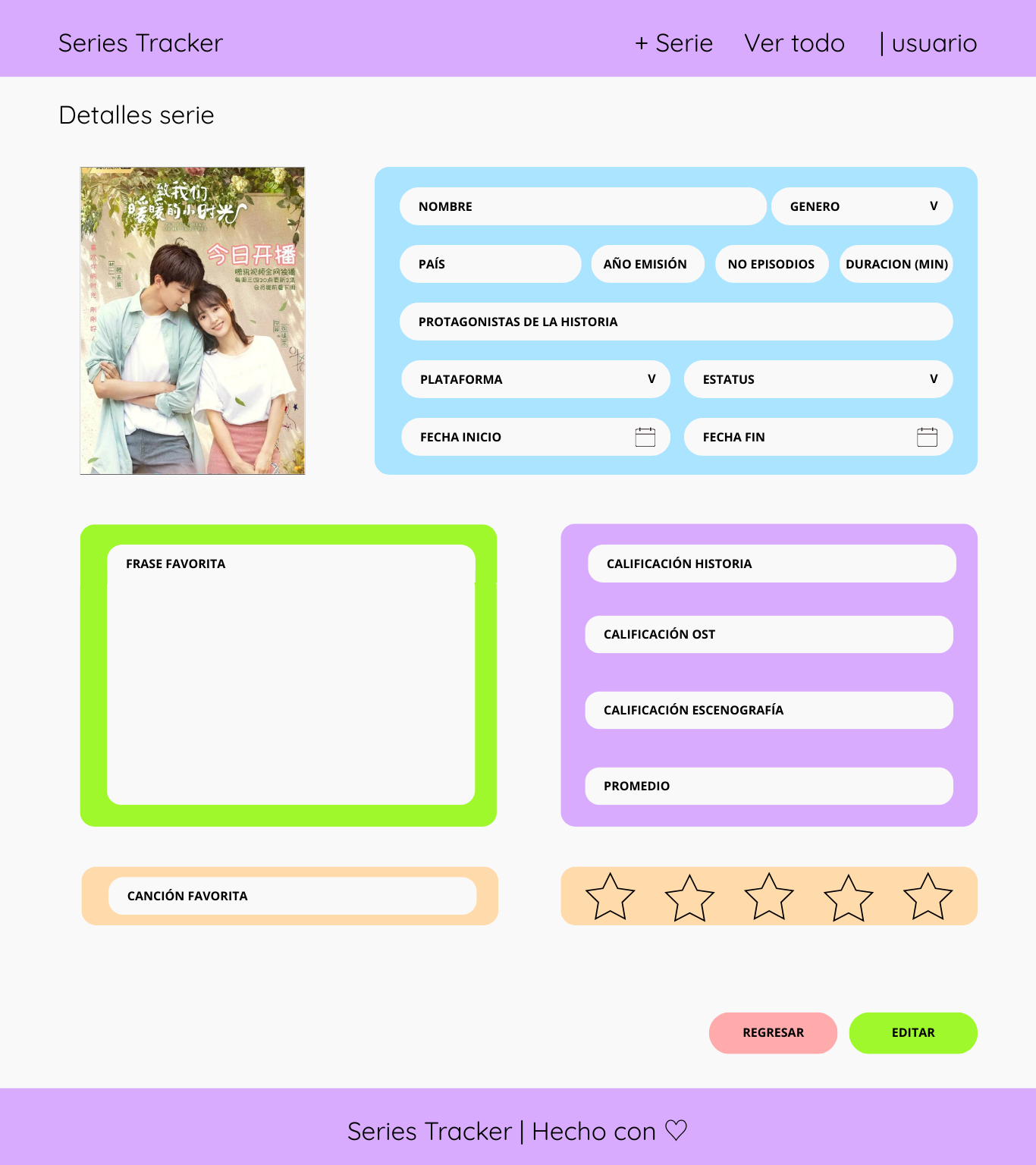
Perfil de usuario (modals)



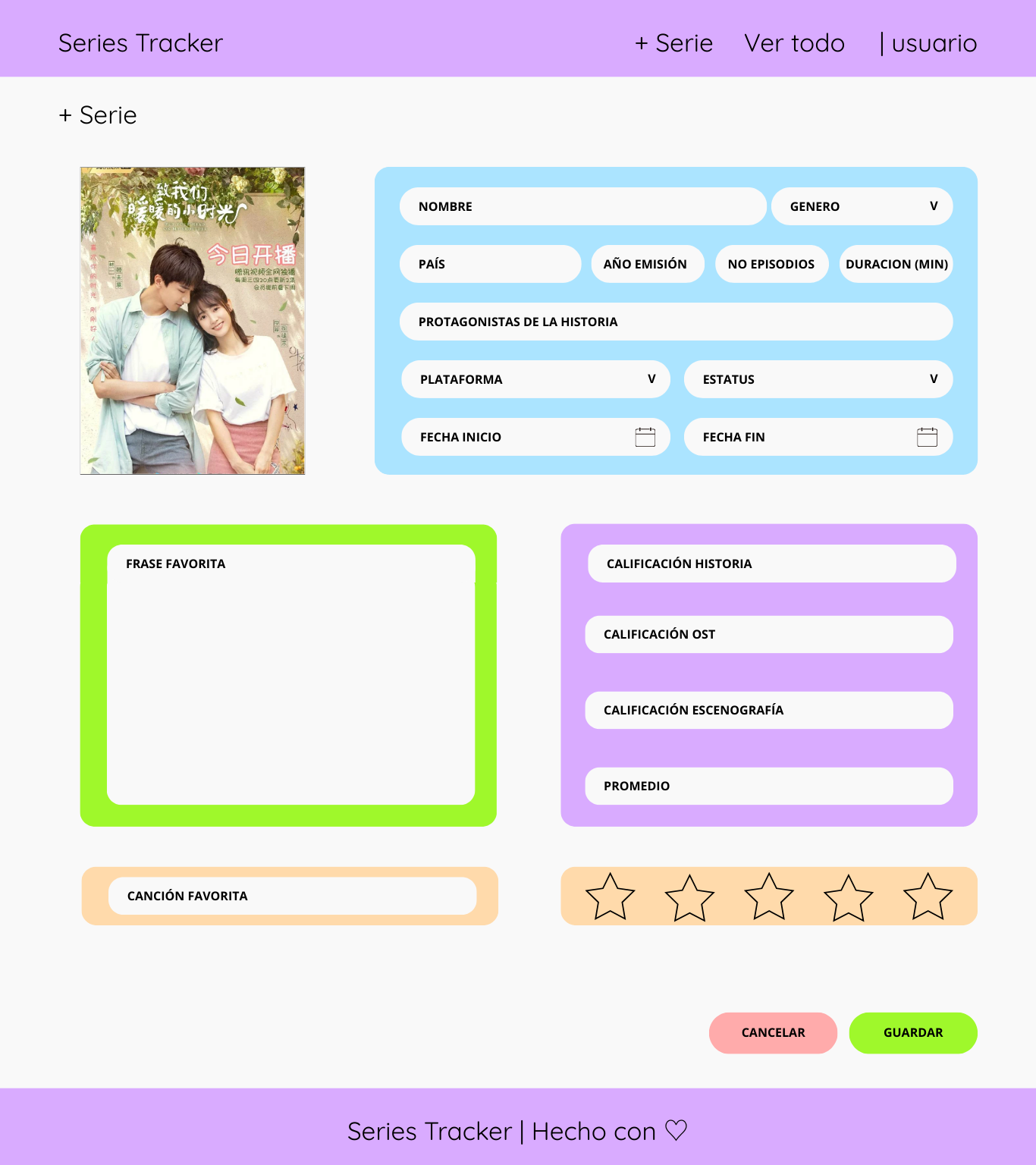
Información general serie (modal)



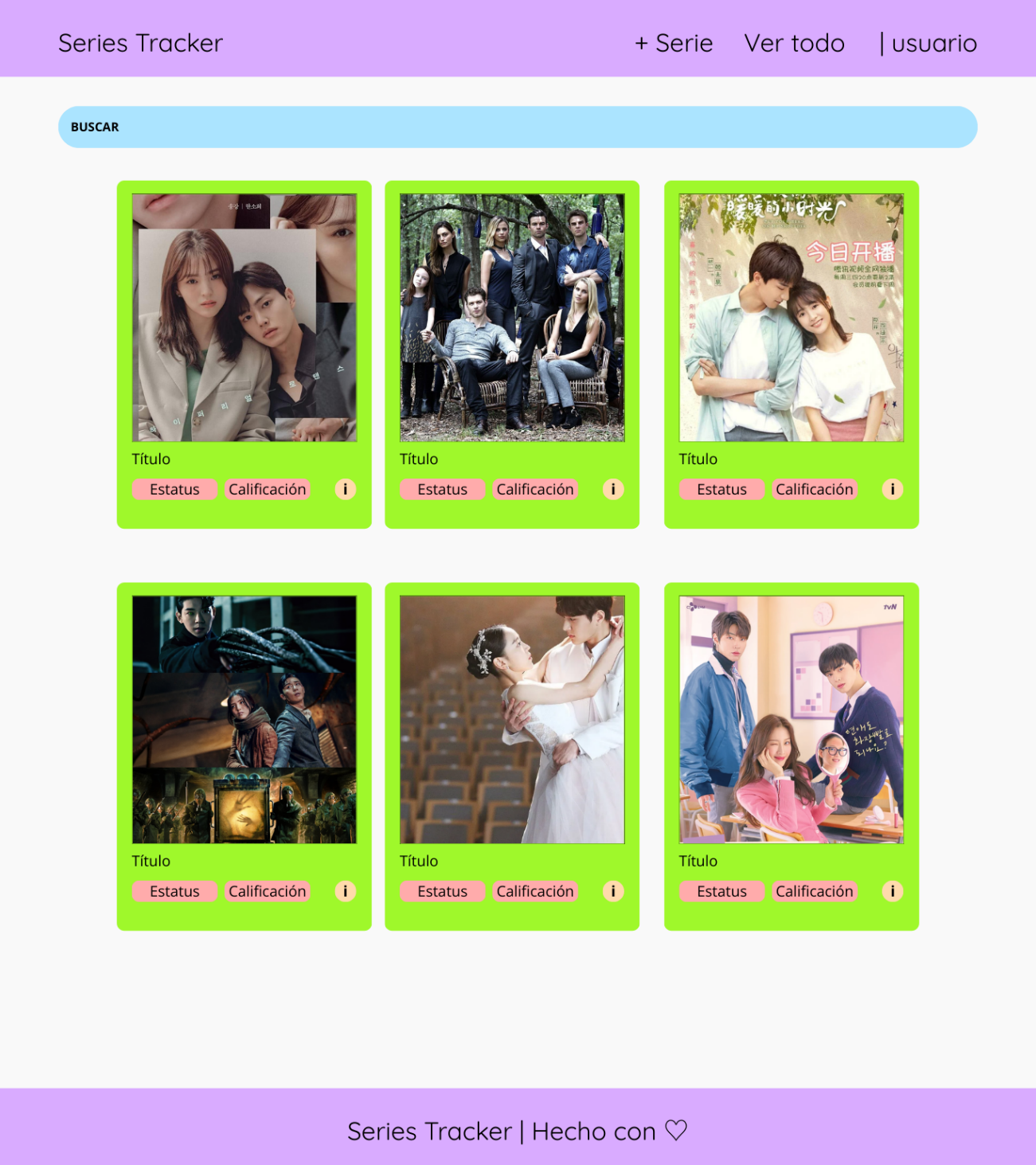
Detalles serie



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

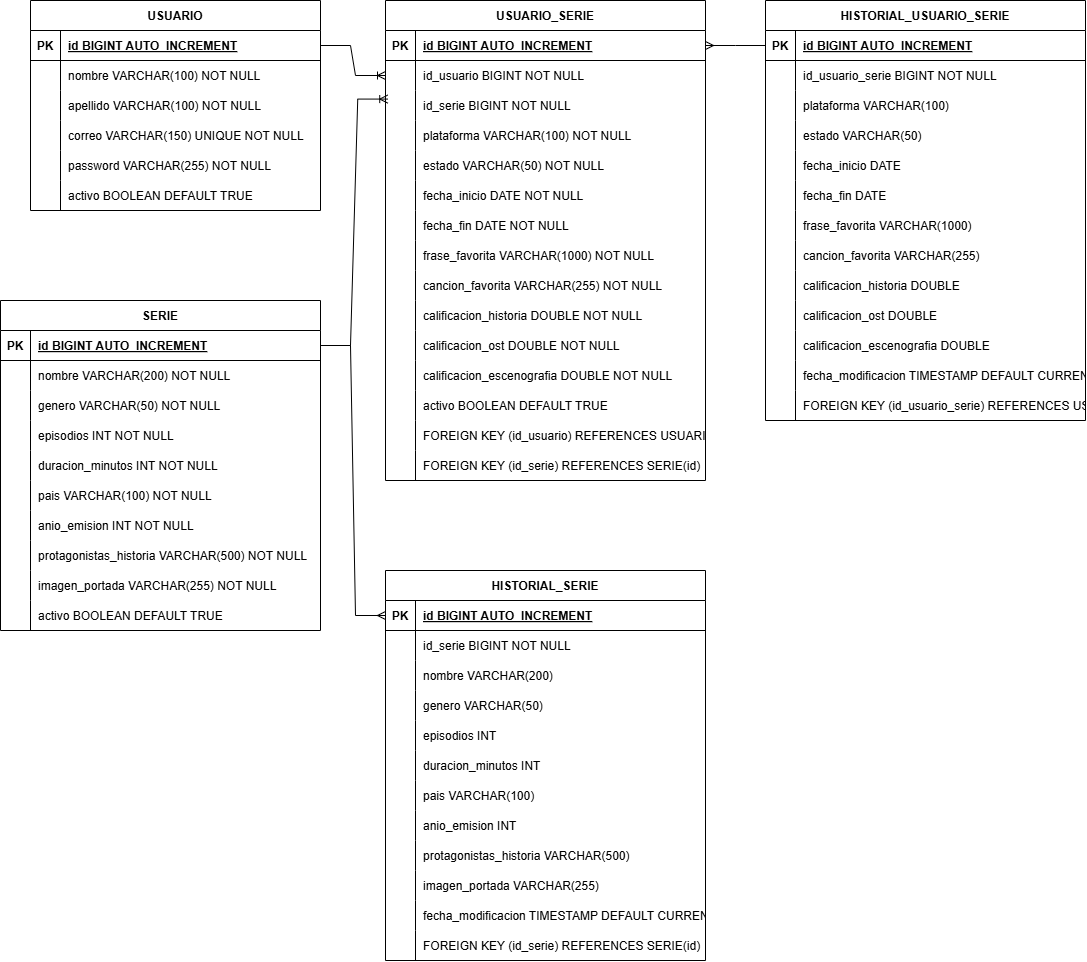
Antes de procesar cualquier solicitud, el backend realiza validaciones para asegurar la integridad y seguridad de los datos. Se verifica, por ejemplo, que los campos obligatorios estén presentes, que los tipos de datos sean correctos, y que no se repitan registros únicos como correos electrónicos. Estas 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

Aquí se detallan los endpoints que permiten la interacción entre los modulos del sistema (frontend, backend y base de datos), facilitando la comunicación y el flujo de información necesario para el correcto funcionamiento del proyecto.

Para cada uno, se describe:

* El método HTTP utilizado (GET, POST, PUT, DELETE, etc.).
* Los parámetros que espera recibir, ya sea en la URL, en el cuerpo de la solicitud o en los encabezados.
* La estructura de la respuesta, incluyendo ejemplos de datos devueltos.
* Captura de pantalla de las pruebas realizadas en Postman, que ilustran la interacción real con el API y validan su correcto funcionamiento.

## Auth

**Iniciar sesión**

|  |
| --- |
| POST - http://localhost:8080/api/login |
| Datos de entrada |
| {    "correo": "pruebas@seriestracker.com",    "password": "123456"  } |
| Datos de salida |
| {     "userId": "2",  "token": " eyJhbGciOiJIUzI1NiJ9.eyJqdGkiOiIyIiwiaWF0IjoxNzQ4NDkzMjQ4LCJzdWIiOiJwcnVlYmFzQHNlcmllc3RyYWNrZXIuY29tIiwiaXNzIjoiTWFpbiIsImV4cCI6MTc0OTA5ODA0OH0.kN9R4lJJcL6BAojeADNyXUqd15yDa1MVCG3dx3KGVVA  "  } |
| Evidencia |
|  |

## Serie

**Listar series**

|  |
| --- |
| GET - http://localhost:8080/api/series |
| Datos de entrada |
|  |
| Datos de salida |
|  |
| Evidencia |
|  |

**Registrar serie**

|  |
| --- |
| POST - http://localhost:8080/api/series |
| Datos de entrada |
|  |
| Datos de salida |
|  |
| Evidencia |
|  |

**Obtener serie por ID**

|  |
| --- |
| GET - http://localhost:8080/api/series/{id} |
| Datos de entrada |
|  |
| Datos de salida |
|  |
| Evidencia |
|  |

**Modificar serie**

|  |
| --- |
| PUT - http://localhost:8080/api/series/{id} |
| Datos de entrada |
|  |
| Datos de salida |
|  |
| Evidencia |
|  |

**Eliminar serie**

|  |
| --- |
| DELETE - http://localhost:8080/api/series/{id} |
| Datos de entrada |
|  |
| Datos de salida |
|  |
| Evidencia |
|  |

## Usuario

**Listar usuarios**

|  |
| --- |
| GET - http://localhost:8080/api/usuarios |
| Datos de entrada |
| Headers:   * Key: Authorization | Value: Bearer + token generado en el login |
| Datos de salida |
| Lista de usuarios activos. |
| Evidencia |
|  |

**Registrar usuario**

|  |
| --- |
| POST - http://localhost:8080/api/usuarios |
| Datos de entrada |
| {    "nombre": "Salma",    "apellido": "Cruz",    "correo": "pruebas@seriestracker.com",    "password": "123456"  } |
| Datos de salida |
| Objeto json con los datos ingresados del nuevo usuario. |
| Evidencia |
|  |

**Obtener usuario por ID**

|  |
| --- |
| GET - http://localhost:8080/api/usuarios/{id} |
| Datos de entrada |
| Headers:   * Key: Authorization | Value: Bearer + token generado en el login   Id = 1 |
| Datos de salida |
| Objeto json con los datos del usuario con id = 1. |
| Evidencia |
|  |

**Modificar usuario**

|  |
| --- |
| PUT - http://localhost:8080/api/usuarios/{id} |
| Datos de entrada |
| Headers:   * Key: Authorization | Value: Bearer + token generado en el login * Key: Content-Type | Value: application/json   Id = 1  {    "nombre": "Salma",    "apellido": "Cruz Sánchez",    "correo": "ssanchez@seriestracker.com",    "password": "123456"  } |
| Datos de salida |
| Objeto json con los datos actualizados del usuario. |
| Evidencia |
|  |

**Eliminar usuario**

|  |
| --- |
| DELETE - http://localhost:8080/api/usuarios/{id} |
| Datos de entrada |
| Headers:   * Key: Authorization | Value: Bearer + token generado en el login   Id = 1 |
| Datos de salida |
| Se espera que se cambie el valor del campo “activo” de 1 (true) a 0 (false). |
| Evidencia |
|  |

## UsuarioSerie

**Listar visualización de series por usuario**

|  |
| --- |
| GET - http://localhost:8080/api/usuarios/series/{usuarioId} |
| Datos de entrada |
|  |
| Datos de salida |
|  |
| Evidencia |
|  |

**Registrar visualización serie**

|  |
| --- |
| POST - http://localhost:8080/api/usarios/series |
| Datos de entrada |
|  |
| Datos de salida |
|  |
| Evidencia |
|  |

**Obtener visualización serie por ID usuario y ID serie**

|  |
| --- |
| GET - http://localhost:8080/api/usuarios/series/{usuarioId}/{serieId} |
| Datos de entrada |
|  |
| Datos de salida |
|  |
| Evidencia |
|  |

**Modificar visualización serie**

|  |
| --- |
| PUT - http://localhost:8080/api/usuarios/series/{usuarioId}/{serieId} |
| Datos de entrada |
|  |
| Datos de salida |
|  |
| Evidencia |
|  |

**Eliminar visualización serie**

|  |
| --- |
| DELETE - http://localhost:8080/api/ usuarios/series/{usuarioId}/{serieId} |
| Datos de entrada |
|  |
| Datos de salida |
|  |
| Evidencia |
|  |

## Enums

**Listar estatus**

|  |
| --- |
| GET - http://localhost:8080/api/enums/estatus |
| Datos de entrada |
| Headers:   * Key: Authorization | Value: Bearer + token generado en el login |
| Datos de salida |
| Lista de estatus de la visualización de las series. |
| Evidencia |
|  |

**Listar genero**

|  |
| --- |
| GET - http://localhost:8080/api/enums/genero |
| Datos de entrada |
| Headers:   * Key: Authorization | Value: Bearer + token generado en el login |
| Datos de salida |
| Lista de géneros para las series. |
| Evidencia |
|  |

**Listar plataforma**

|  |
| --- |
| GET - http://localhost:8080/api/enums/estatus |
| Datos de entrada |
| Headers:   * Key: Authorization | Value: Bearer + token generado en el login |
| Datos de salida |
| Lista de plataformas en las que se puede ver una serie. |
| Evidencia |
|  |

# Pruebas

En esta sección se detallan las pruebas centradas en la interfaz de usuario del proyecto para asegurar que la experiencia sea intuitiva, funcional y libre de errores visuales o de integración.

Se evalúa que los elementos gráficos respondan correctamente a las acciones del usuario, que la navegación sea fluida y que la información mostrada sea clara y precisa.

## Inicio de sesión

|  |
| --- |
| **Caso 1: Verificar que el formulario acepte credenciales válidas** |
| Datos de entrada |
| Usuario existente:   * Correo: pruebas@seriestracker.com * Password: 123456 |
| Resultado esperado |
| Ver en la vista “login” el mensaje “Inicio de sesión exitoso 😊” y ser redireccionado a la vista “inicio”. |
| Resultado obtenido |
|  |

|  |
| --- |
| **Caso 2: Verificar que el formulario rechace credenciales inválidas** |
| Datos de entrada |
| Usuario inexistente:   * Correo: pruebasqa@seriestracker.com * Password: 123456 |
| Resultado esperado |
| Ver en la vista “login” el mensaje “Las credenciales son incorrectas”. |
| Resultado obtenido |
|  |

## Registro de usuario

|  |
| --- |
| **Caso 3: Verificar que se muestren errores si los campos están vacíos o mal formateados** |
| Datos de entrada |
|  |
| Resultado esperado |
|  |
| Resultado obtenido |
|  |

|  |
| --- |
| **Caso 4: Verificar que se muestren errores si los campos contraseña y repetir contraseña no coinciden** |
| Datos de entrada |
|  |
| Resultado esperado |
|  |
| Resultado obtenido |
|  |

|  |
| --- |
| **Caso 5: Verificar que se puede registrar un usuario nuevo e iniciar sesión** |
| Datos de entrada |
|  |
| Resultado esperado |
|  |
| Resultado obtenido |
|  |

## Navegación entre pantallas

|  |
| --- |
| **Caso 6: Comprobar que los enlaces del navbar redirijan correctamente** |
| Datos de entrada |
|  |
| Resultado esperado |
|  |
| Resultado obtenido |
|  |

|  |
| --- |
| **Caso 7: Verificar que el botón “Salir” cierre la sesión y redirija al login** |
| Datos de entrada |
| Se da clic en la opción “salir” del menú desplegable del usuario en el navbar. |
| Resultado esperado |
| Se redirige al usuario a la vista “login”. |
| Resultado obtenido |
|  |

|  |
| --- |
| **Caso 8: Verificar que el enlace al registro de usuario redirija correctamente** |
| Datos de entrada |
| En la vista “login” se da clic en el enlace “¿No tienes una cuenta? ¡Registrate!” |
| Resultado esperado |
| La vista “login” redirige a la vista “registrar usuario”. |
| Resultado obtenido |
|  |

## Gestión de series

|  |
| --- |
| **Caso 9: Verificar que se pueden añadir series y se visualizan bien** |
| Datos de entrada |
|  |
| Resultado esperado |
|  |
| Resultado obtenido |
|  |

|  |
| --- |
| **Caso 10: Verificar que se pueden modificar detalles de las visualizaciones** |
| Datos de entrada |
|  |
| Resultado esperado |
|  |
| Resultado obtenido |
|  |

|  |
| --- |
| **Caso 11: Verificar que la eliminación de una serie actualiza la vista correctamente** |
| Datos de entrada |
|  |
| Resultado esperado |
|  |
| Resultado obtenido |
|  |

## Visualización de detalles

|  |
| --- |
| **Caso 12: Verificar que al seleccionar una serie se muestren sus detalles** |
| Datos de entrada |
|  |
| Resultado esperado |
|  |
| Resultado obtenido |
|  |

|  |
| --- |
| **Caso 13: Verificar que las imágenes carguen correctamente** |
| Datos de entrada |
|  |
| Resultado esperado |
|  |
| Resultado obtenido |
|  |

## Diseño responsivo

|  |
| --- |
| **Caso 14: Verificar que la interfaz se adapta a distintas resoluciones** |
| Datos de entrada |
|  |
| Resultado esperado |
|  |
| Resultado obtenido |
|  |

|  |
| --- |
| **Caso 15: Verificar que no haya elementos superpuestos o mal posicionados en pantallas pequeñas** |
| Datos de entrada |
|  |
| Resultado esperado |
|  |
| Resultado obtenido |
|  |

## Manejo de estados vacíos

|  |
| --- |
| **Caso 16: Verificar que la interfaz muestre mensajes claros cuando no hay información registrada** |
| Datos de entrada |
| Vista “inicio” de un nuevo usuario (sin información en la base de datos sobre las series y sus visualizaciones). |
| Resultado esperado |
| Mensaje “Sin información disponible” en todas las secciones de la vista “inicio”. |
| Resultado obtenido |
|  |

# 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. Incluye primeras vistas del proyecto y avances en la documentación. |

## Demo