Tabla de contenido

[1. Introducción 1](#_Toc198795101)

[1.1. Objetivo del reto 1](#_Toc198795102)

[1.2. Temática elegida 1](#_Toc198795103)

[2. Estrategia de implementación 2](#_Toc198795104)

[2.1. Diagrama de arquitectura 2](#_Toc198795105)

[2.2. Tecnologías utilizadas 3](#_Toc198795106)

[2.3. Estructura de carpetas del proyecto 5](#_Toc198795107)

[3. Front-end 6](#_Toc198795108)

[3.1. Paleta de colores 6](#_Toc198795109)

[3.2. Tipografía 8](#_Toc198795110)

[3.3. Mockups 8](#_Toc198795111)

[4. Back-end 9](#_Toc198795112)

[4.1. CRUD y manejo de rutas 9](#_Toc198795113)

[4.2. Autenticación 9](#_Toc198795114)

[4.3. Validación de datos 9](#_Toc198795115)

[5. Base de datos 10](#_Toc198795116)

[5.1. Diagrama entidad-relación 10](#_Toc198795117)

[5.2. Script de la BD 11](#_Toc198795118)

[6. API 13](#_Toc198795119)

[6.1. Descripción de los endpoints 13](#_Toc198795120)

[6.2. Métodos, parámetros y ejemplos 13](#_Toc198795121)

[7. Pruebas 14](#_Toc198795122)

[7.1. Create 14](#_Toc198795123)

[7.2. Read 14](#_Toc198795124)

[7.3. Update 14](#_Toc198795125)

[7.4. Delete 14](#_Toc198795126)

[8. Anexos 15](#_Toc198795127)

[8.1. Repositorio GitHub 15](#_Toc198795128)

[8.2. Demo 15](#_Toc198795129)

# 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

## Diagrama de arquitectura

## 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 |
|  | 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

## Mockups

# Backend

## CRUD y manejo de rutas

## Autenticación

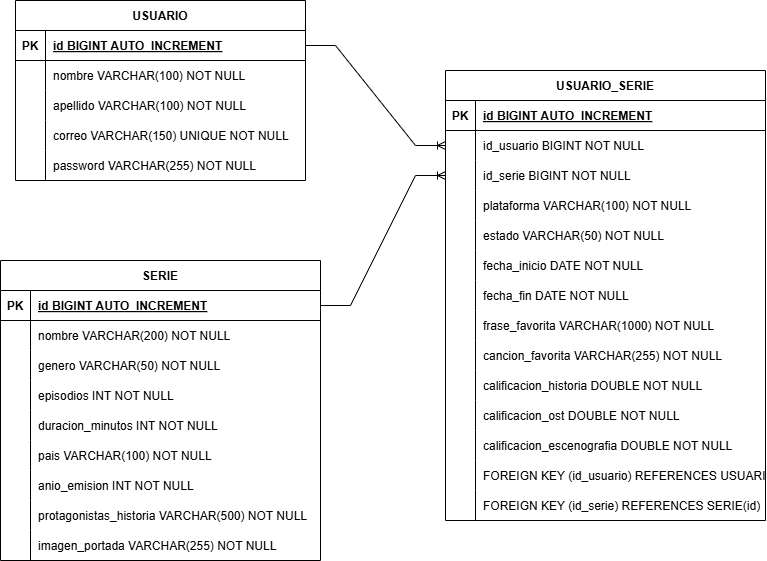
## Validación de datos

# 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 );  -- 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),  anio\_emision INT,  protagonistas\_historia VARCHAR(500),  imagen\_portada VARCHAR(255) );  -- 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),  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,  FOREIGN KEY (id\_usuario) REFERENCES USUARIO(id),  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. |
| day-3 | Contiene los avances correspondientes al día 26 de mayo, como parte del periodo de prueba. |

## Demo