TRABAJO DE FIN DE CURSO

FILME

2º DESARROLLO DE APLICACIONES WEB

IES LAGUNA DE JOATZEL

REALIZADO POR LOS ALUMNOS:

Jesús Alejandro Campos Laiton

Javier Roldán Gallego

ÍNDICE DEL PROYECTO

Introducción

FilMe es un proyecto dedicado al mundo del cine. Con todos los avances tecnológicos y las plataformas de streaming, las películas y series se han vuelto mucho más accesibles. Este crecimiento, ha causado que cada vez más personas busquen herramientas que faciliten el intercambio de opiniones y recomendaciones, dando lugar a un incremento notable en la cantidad de críticas, valoraciones y sugerencias entre los usuarios.

También, vivimos en una era en la cuál todo se transmite por las redes sociales, por lo tanto, FilMe está orientado a este ámbito, buscando adaptarse y aprovechar el poder de las redes sociales para crear una comunidad digital en torno al cine.

El objetivo de FilMe es poder unificar todas las necesidades de los usuarios en el mundo del cine desde poder ver una lista de películas con su valoraciones y críticas o ver información actualizada sobre la disponibilidad de las películas en las principales plataformas de streaming, hasta poder hacer tu propia lista de películas, recomendar al resto de usuarios, hacer críticas, ver los perfiles de otros usuarios, reuniendo todas estas funciones en una misma aplicación.

En este documento, vamos a hablar de cómo desarrollamos FilMe, planteando en este todas las etapas del desarrollo del sistema. En primer lugar, se analizarán los requisitos necesarios para el funcionamiento adecuado de la aplicación. Posteriormente, se describirán el proceso de diseño y desarrollo de la aplicación, detallando las decisiones tomadas en cuanto a la arquitectura, las tecnologías empleadas y la implementación de las funcionalidades clave, así como el despliegue y la conclusión del proyecto final.

Justificación y Objetivos

¿Por qué hago este proyecto? ¿Qué problemas resuelve o qué servicios ofrece?

El proyecto Filme surge debido a la necesidad de centralizar en una sola plataforma la información e interacción social relacionada con el mundo del cine. Actualmente, los amantes del cine deben navegar por múltiples aplicaciones y plataformas para descubrir nuevas películas o series, ver sus tráileres y valoraciones, compartir o recibir recomendaciones y críticas e incluso ver en qué plataformas está disponible dicha película.

Esto genera dispersión de la información y hace mucho menos accesible todo ello y los usuarios acaban en una plataforma de streaming buscando por buscar en lugar de ver las mejores películas del momento.

Para ello creamos FilMe, una aplicación que unifica todas estas necesidades, ya que permite a los usuarios acceder a una amplia base de datos de películas, donde no solo pueden consultar tráilers y valoraciones, sino también conocer las opiniones de otros usuarios y ver recomendaciones personalizadas. Además FilMe no solo proporciona información sobre películas si no que se adapta a la era de las redes sociales proporcionando un nuevo mundo a los cinéfilos.

FilMe permite tener tu propio perfil personalizado en el que tener tu lista de películas favoritas o realizar críticas y recomendaciones, además de ver los perfiles de los otros usuarios permitiendo crear una comunidad de cine dentro de la aplicación. Esto permite que los usuarios interactúen, compartan sus listas de películas favoritas, recomienden títulos a otros y participen en debates sobre el cine.
De esta manera, FilMe no solo facilita el descubrimiento de películas, sino que crea una comunidad de cinéfilos conectados por su pasión por el cine, ofreciendo una experiencia más enriquecedora y personalizada.

Problemas que resuelve:

1. Dispersión de información: Unifica en una sola plataforma la información sobre películas, críticas y recomendaciones.
2. Interacción social limitada: Permite crear una red de seguidores del cine para compartir y recomendar películas.
3. Gestión de favoritos: Facilita el almacenamiento de películas favoritas en un perfil personal.

Servicios que ofrece:

1. Perfil personalizado: Los usuarios pueden gestionar su perfil, sus películas favoritas, críticas y recomendaciones.
2. Interacción social: Interactuar con todos los perfiles de la aplicación, recomendar películas, hacer críticas.
3. Exploración de cine: Acceso a las películas mejor valoradas y categorías populares.
4. Información actualizada: información sobre la disponibilidad de las películas en las principales plataformas de streaming. Información actualizada con nuevos estrenos, etc.

¿Qué hacemos en este proyecto?

Desarrollamos una aplicación web con una interfaz intuitiva y moderna que conecta a los usuarios con sus películas favoritas y con el resto de usuarios de la aplicación.

FilMe se divide en dos partes principales;:

1. **Front-End (React):**
    - **Visualizar películas**: La aplicación obtiene los datos de películas de diversas fuentes a través de APIs externas.
    - **Gestionar el perfil de usuario**: Los usuarios pueden crear y editar su perfil personal, añadiendo información personal, preferencias de cine, listas de películas favoritas y otras configuraciones.
    - **Explorar otros perfiles**: Los usuarios pueden explorar los perfiles de otros usuarios, ver sus listas de recomendaciones, críticas y valoraciones, y conectar con personas con intereses similares.

2. **Back-End (Spring Boot y Java):**
    - **API para la gestión de usuarios**: El back-end maneja la creación, autenticación y autorización de los usuarios. Se emplea un sistema de autenticación basado en **JWT** (JSON Web Tokens) para garantizar la seguridad de las sesiones.
    - **Gestión de películas, favoritos y críticas**: El sistema permite a los usuarios añadir películas a su lista de favoritos, dejar críticas sobre los títulos que han visto y puntuar las películas. La API gestiona todas estas interacciones y las almacena en la base de datos, de manera que los usuarios pueden consultar sus críticas y las de otros.

- **Conexión con la base de datos**: Toda la información relativa se almacena en una base de datos MySQL.

Metodología de trabajo

- Planificación de las tareas del proyecto

En cuanto a la planificación de las tareas, como iremos viendo en los distintos diagramas, lo primero que empezaremos será el inicio de la documentación, planteamiento de la aplicación y estructura básica del proyecto.

Primero hemos planteado que queremos crear y cómo lo vamos a hacer, cómo ya hemos visto en la explicación del proyecto y las tecnologías que usaremos. Hemos comenzado creando la base de datos del proyecto en MySQL y planteando las necesidades de ella para poder implementar todo lo que queremos hacer en la aplicación.

Más adelante comenzamos con el diseño básico del frontend, diseño de la página principal, paleta de colores y estructura principal de menú y páginas esenciales. Esto lo hemos utilizado para ir dándole forma visual al proyecto y comenzar las bases de diseño, además de poder ver como irán quedando las páginas principales y de desarrollar ideas a partir del diseño básico e inconvenientes o oportunidades que van saliendo con ello.

Después crearemos la estructura básica de Spring Boot a partir de la base de datos con JPA creando los modelos principales, servicios y controladores, poniendo en práctica las bases de Spring y dándole forma inicial al backend. Implementar los métodos básicos de insert y update, además de crear el login y el registro de usuarios que ya conectará con nuestra aplicación. Con todo ello crearemos la API de la aplicación con todos los endpoints para ir dándole la funcionalidad que buscamos.

Más adelante juntaremos ambas partes, empezando por la parte de registro de usuarios, utilizando el formulario del frontend para mandar los datos a los métodos creados en el backend y así crear usuarios en la base de datos desde la aplicación y no desde el gestor de BD. Continuando con ello se hará funcional el login accediendo a la base de datos y enviando datos al backend para comprobar si el usuario existe. Una vez todo es comprobado se inicia sesión y utilizamos la configuración del back para crear un JWT y manejarlo como gestión de sesiones. El token se guarda en el local storage y se guarda la sesión del usuario y así todos sus datos.

Continuaremos con la funcionalidad de la aplicación accediendo a una API de películas y así mostrando un grid de películas con la página de detalles de cada una. El perfil de usuario contendrá un apartado con sus datos (y el update de los mismos) y luego el perfil público que será lo que vean todos los usuarios con sus favoritos, críticas y recomendaciones.

Habrá que implementar la subida de imágenes en la base de datos para el portal del usuario, que podrá tener una imagen personalizada. Con esto crearemos un buscador público de todos los usuarios de la aplicación, y con ello se accede al perfil público de cada usuario creado anteriormente.

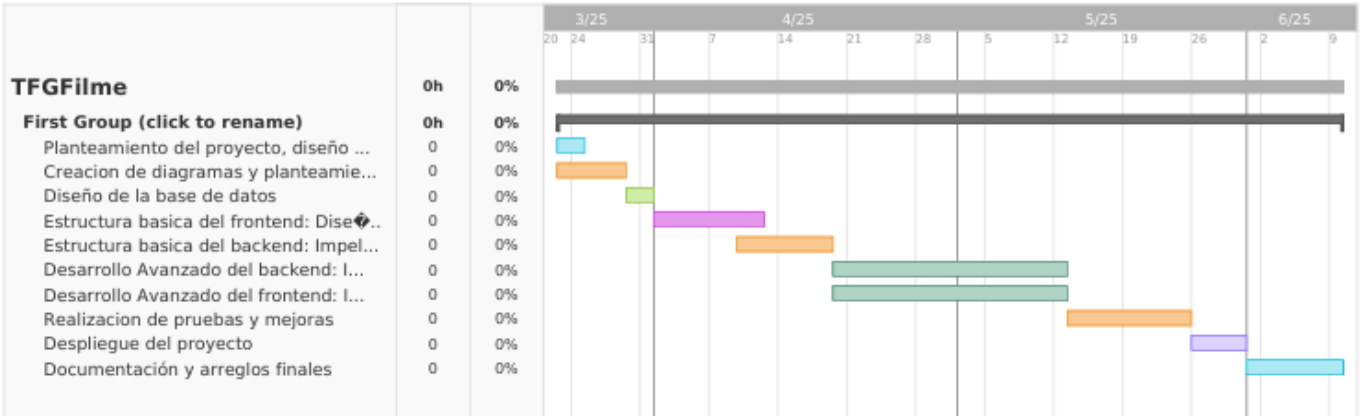
Implementaremos también la posibilidad de añadir una película a favoritos desde los detalles de la película y que así se añada automáticamente a “Mi lista” y a la lista de favoritos del perfil público.
Finalmente, iremos puliendo los fallos y mejoras de la aplicación para dejarlo completo, y por último el despliegue de la aplicación completa.

¿Quién realiza cada tarea?

La idea principal del proyecto es dividir equitativamente las tareas e ir ayudando con inconvenientes. Ambas partes haremos frontend y backend aunque Javier se centrará más en la parte de backend de Spring Boot y Jesús más en la parte de diseño. Aun así tocaremos ambas partes y los dos integrantes participaremos en toda la implementación de métodos, funcionalidades y juntado el backend y el frontend.

- Tiempo estimado para cada tarea. Dependencias entre tareas. Diagrama de Gantt de planificación de proyectos.

En este diagrama representamos la estructura y organización de nuestro proyecto a lo largo del tiempo de desarrollo (Marzo a Junio)



- Comunicación del equipo:

En cuanto a las herramientas de trabajo para comunicarnos y poder hacer el proyecto en conjunto, hemos optado por realizar un grupo de discord en el cual anotar ideas y apuntes, y en general tener esa fuente de comunicación exclusiva sobre el proyecto.

Además de utilizar GitHub para tener nuestro repositorio en común para ir actualizando los cambios tanto del frontend como del backend.
El github lo hemos organizado de la siguiente forma: 2 ramas (1 para cada integrante del proyecto) ambas ramas están actualizadas y tienen el mismo contenido, pero cuando trabajamos a la vez en el proyecto evitamos conflictos, de esta forma cada uno va subiendo los cambios a su rama.

Así siempre antes de empezar a programar las ramas están actualizadas y ambos partimos del mismo proyecto pero evitando conflictos por cambios simultáneos. Cuando realizamos cambios simultáneos al terminar hacemos un merge para juntarlo todo. Sino, con un pull es suficiente.

La comunicación y reuniones se realizan por el grupo de discord. Al inicio del proyecto tuvimos reuniones en las cuales formamos las ideas principales del proyecto y la base de datos. Después de organizar todo hemos establecido los sábados por la mañana cada semana (máximo 2) como día específico de reunión para ver que llevamos y organizar la semana.

A pesar de ello, solemos comunicar cambios e ideas siempre por el grupo de discord y cualquier otro día de la semana que coincidimos en el avance del proyecto podemos tener alguna reunión para dudas, avances, solución de errores o ayudas con el proyecto desde ambas partes.

Desarrollo del proyecto

- Tecnologías y herramientas

El desarrollo de FilMe ha requerido una selección de tecnologías y herramientas tanto para el front-end como para el back-end, así como para la gestión de la base de datos, pruebas y despliegue. A continuación, se describen las más relevantes

Front-End (React): Interfaz de usuario y gestión de estados, diseño principal de la aplicación. **React Router** para gestionar la navegación entre páginas y rutas protegidas según el estado de autenticación del usuario.

Back-End (Spring Boot): Desarrollo del backend, controladores , servicios y API REST.

Base de Datos (MySQL): Almacenamiento de usuarios, películas y relaciones. Se ha gestionado mediante **JPA e Hibernate**, permitiendo el mapeo de entidades Java a tablas relacionales.

RapidApi (API externa): Obtención de datos de películas. La comunicación se ha realizado a través de peticiones **fetch** en el front-end, estructurando y limpiando los datos para su presentación.

JSON Web Tokens(JWT): Sistema de autenticación y autorización de usuarios, gestión de sesiones. El token generado al iniciar sesión se guarda en el localStorage del navegador y se usa para acceder a rutas protegidas y recursos restringidos.

IntelliJ Visual Studio Code: Entornos de desarrollo para backend y frontend respectivamente

PostMan: Pruebas de la API REST.

Chrome DevTools: Pruebas de rendimiento, accesibilidad y análisis de calidad del frontend

Control de Versiones (GitHub): Gestión colaborativa y control de cambios.

- Diseño o esquemas de las tecnologías del proyecto

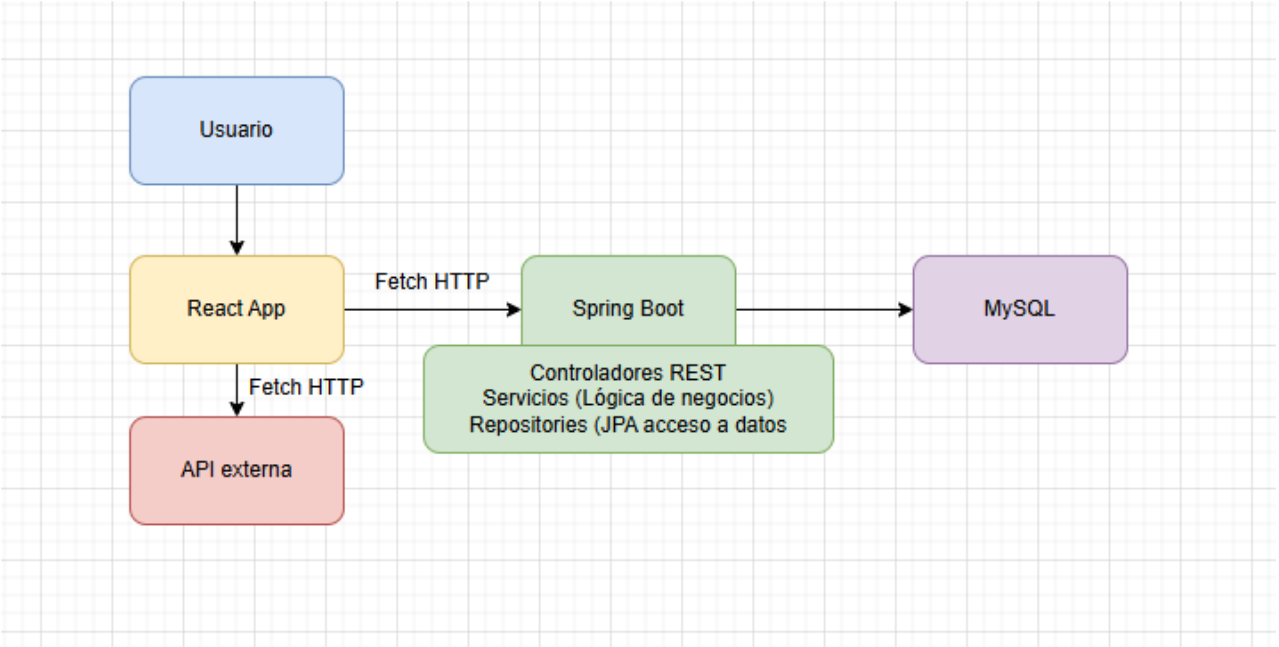
Este apartado describe la arquitectura general del sistema y las tecnologías empleadas en el desarrollo de FilMe, a través de diagramas estructurado, se presenta cómo se interrelacionan los diferentes componentes de la aplicación, qué funciones cumple cada uno, y cómo se asegura la comunicación eficiente entre el cliente y el servidor.

El diseño sigue una arquitectura full-stack con una separación clara de responsabilidades entre front-end, back-end, base de datos y servicios externos.

Modelo de desarrollo en cascada

- Análisis de Requisitos: Recopilación de funcionalidades y servicios.
- Diseño del Sistema: Diagramas UML y arquitectura de tecnologías.
- Implementación: Desarrollo de back-end y front-end.
- Pruebas: Verificación de funcionalidades y pruebas unitarias.
- Despliegue: Publicación y documentación del proyecto.
- Mantenimiento: Actualización y soporte.

- Arquitectura de la aplicación:



- Comunicación Frontend–Backend

La arquitectura de FilMe se basa en una separación clara entre el frontend (React) y el backend (Spring Boot), permitiendo una comunicación fluida a través de peticiones HTTP. A continuación se detalla cómo se estructura y comunica cada capa del sistema.

Backend – Spring Boot

Dividido principalmente en Repositorios, Servicios, Controladores y Modelos.

Repositories
Usan Spring Data JPA para acceder a la base de datos MySQL. Cada entidad (Usuario, Película, Crítica, etc.) cuenta con su propio repositorio para realizar operaciones como findById, findAll, save, etc., sin necesidad de implementar consultas SQL manuales.

Services
Contienen la lógica de negocio. Son los encargados de realizar las operaciones requeridas por los controladores. Validan datos, interactúan con la base de datos a través de los repositorios, y pueden generar tokens o lanzar excepciones si es necesario.

Controllers:
Son los puntos de entrada al backend desde el frontend. Implementan endpoints RESTful que permiten operaciones como login, registro, búsqueda de usuarios, actualización de perfil, etc.

Por ejemplo, en el UsuarioController se encuentran rutas como:

POST /usuarios/register → registro de usuarios

POST /usuarios/login → autenticación y generación de JWT

GET /usuarios/auth/perfil → devuelve el perfil asociado al token enviado

Estos controladores reciben las solicitudes HTTP y delegan el procesamiento al nivel de servicios.

Ejemplo: UsuarioService maneja el login de usuario, el guardado y actualización de perfiles, o la lógica para renombrar el nickname y generar un nuevo JWT.

Models
Las clases modelo representan las entidades de base de datos(Usuario, Película). Están anotadas con @Entity y forman parte del sistema de persistencia de JPA (Java Persistence API).

Cada clase modela una tabla de la base de datos e incluye atributos como campos (@Id, @Column, @OneToMany, etc.), validaciones (@NotBlank, @Size, etc.) y relaciones entre entidades.
Estas clases son fundamentales para mapear objetos de Java a registros en la base de datos de forma automática.

DTOs (Data Transfer Objects)
Se utilizan para transportar datos entre capas sin exponer directamente las entidades de base de datos, aumentando la seguridad y el control sobre los datos intercambiados.

Seguridad con JWT
Se emplea autenticación mediante JSON Web Tokens. Cuando el usuario inicia sesión, se genera un JWT con su nickname. Este token debe enviarse en cada petición protegida, dentro del encabezado Authorization: Bearer <token>, y es validado en el backend.

Frontend – React
Dividido principalmente en Pages, Components y Services.

Pages
Páginas principales de la aplicación, cada una incluye la lógica para renderizar contenido, gestionar estado y coordinar la comunicación con los servicios.

Components
Son los componentes reutilizables como:
RegisterCard.jsx; MovieCard.jsx; PublicProfileCard.jsx

Estos encapsulan partes específicas de la interfaz (formularios, tarjetas, listas, etc.) y permiten reutilización y mantenimiento más eficiente.

Services
Los archivos de services son módulos JS que contienen funciones asincrónicas que realizan llamadas HTTP (con fetch) a los endpoints del backend y a las API’s externas.

Por ejemplo:

registerUser(user) → hace POST a /usuarios/register
login(nickname, password) → hace POST a /usuarios/login

Estos servicios permiten centralizar la lógica de comunicación con el backend y mantener el código desacoplado y limpio.

Proceso de Comunicación
La aplicación comunica los services del frontend con los controladores del backend mediante HTTP. El usuario interactúa con la página y ciertas interacciones (clicks, submits de formularios, redirecciones a páginas) tienen llamadas a los services en react. Los servicios utilizan los fetch HTTP para hacer llamadas a los End Points configurados en los controladores del backend

Ejemplo concreto de flujo (registro de usuario)

- En React, se ejecuta registerUser(user) cuando el usuario envía el formulario.
- Esta función hace un POST a /usuarios/register.
- Spring Boot lo recibe en UsuarioController.save(), valida el cuerpo (@Valid), y lo pasa a UsuarioService.save().
- El servicio guarda el usuario en la base de datos y devuelve el objeto persistido.
- La respuesta llega al frontend, que muestra un mensaje de éxito o errores de validación, si los hay.

- Código del programa:

Acabamos de ver la estructura del programa por lo que ahora, en este apartado, se detallan las partes del desarrollo que presentaron una mayor complejidad a nivel técnico y de implementación. Estas áreas destacan tanto por el grado de dificultad en su lógica como por los retos que supusieron en cuanto a diseño, rendimiento o integración con otras partes del sistema. Los detalles del código se mencionarán en la presentación, pero vamos a destacar los componentes más importantes:

Backend: La estructura principal del backend es sencilla ya que hemos optado por el mismo diseño visto en clase con modelos, repositories, services y controllers similares. Lo más destacable son algunas funciones para obtener datos y la configuración del JWT.

Por ejemplo:

- **JWTUtil.java**: Tiene toda la configuración del token, en el método de crear el token obtiene el nickname y lo introduce dentro del payload(una de las partes del JWT) por lo que así el token del usuario va relacionado con su nickname. En el método de validarToken obtiene el cuerpo(body) del token y el subject (el nickname) para validar si ese token que se pasa es válido.

- **Usuario controller**: Tenemos los métodos de login y authPerfil a destacar en login utilizamos un nickname y una password de un usuario, comprobamos que existan y se usa el método de crear el token para asignarlo a ese usuario. El authPerfil obtiene un token, lo valida y con el nickname (que está en el payload del token) obtiene ese usuario.

- **Favorito, crítica y recomendación controllers:** Tenemos algunos métodos a destacar como los save() o getByUsuarioId()

Frontend: La parte de react es mucho más extensa ya que contiene diseño, implementación del backend en el frontend y toda la funcionalidad de la aplicación. Las clases más complejas y destacables son:

- **Services**: Contienen todos los fetch para acceder a los métodos del backend y poder usarlos para:
  agregar funcionalidad en el front, para acceder a las APIs externas (movieService) o para la subida de imágenes.

- **Header**.jsx: Contiene useState para modificar los estados de variables como user o isMenuOpen para hacer un menú responsive, useEffect como hook

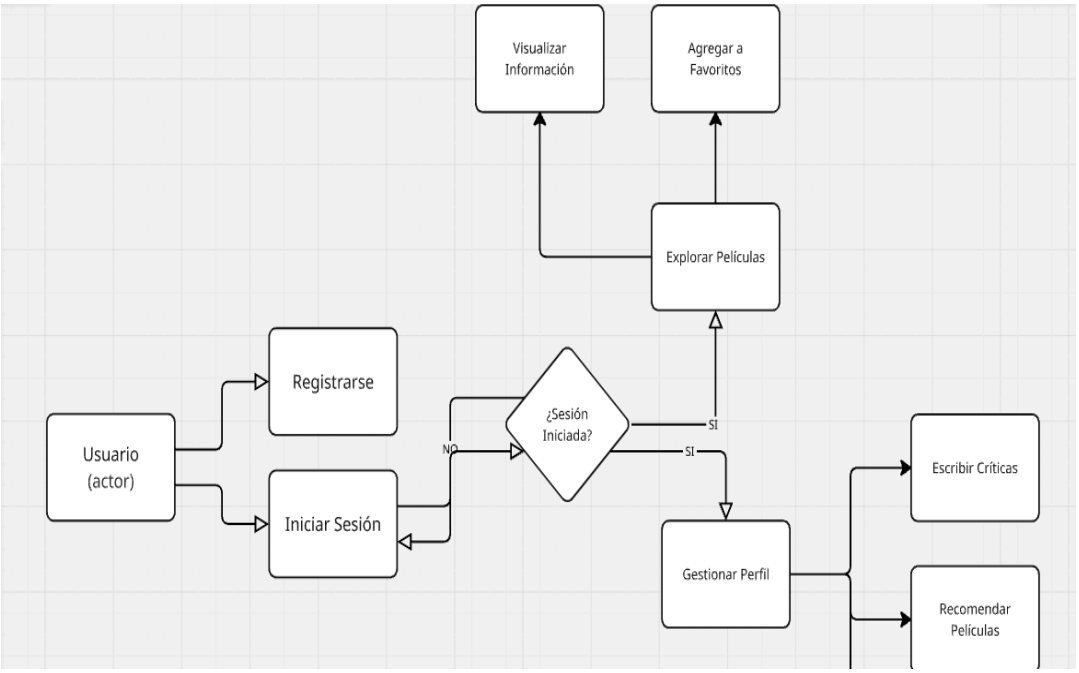
para usar otros métodos como por ejemplo obtener el usuario o handleClick para funcionalidad al clickar. Además de tener múltiples condicionales para mostrar o no distintas funcionalidades.

- **MovieDetailLayout.jsx**: Diseño complejo para mostrar la información de las películas. Contiene los svgs para todos los iconos usados para el diseño del componente, handleClick para ver si la estrella está seleccionada (llamar addFavoritos) o si la estrella está vacía (deleteFavoritos).
  Se usan otros handlerClick para llamar a métodos y un useNaviagate para dirigir a otra página. Cabe destacar fetchMovie para hacer la request a la API y obtener los datos de la película.

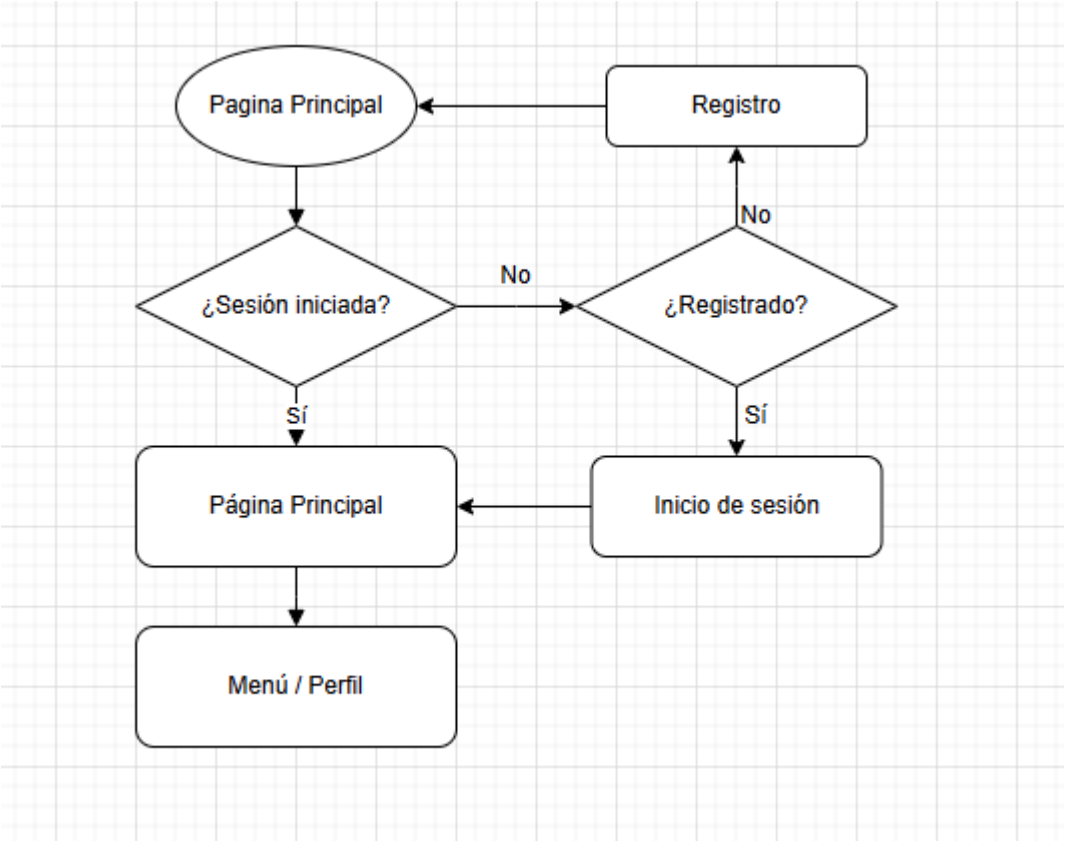
- **PublicProfileCard.jsx:** PortalFilme con diseño complejo, obtención de recomendaciones favoritos y críticas, llamadas a múltiples métodos…
  Obtención del usuario con la sesión iniciada a partir del token (como en la mayoría de componentes)
  GetRecomendaciones, favoritos y críticas, obteniéndose a través del user.id.
  Gestión del cambio de imagen con useUploadImage.

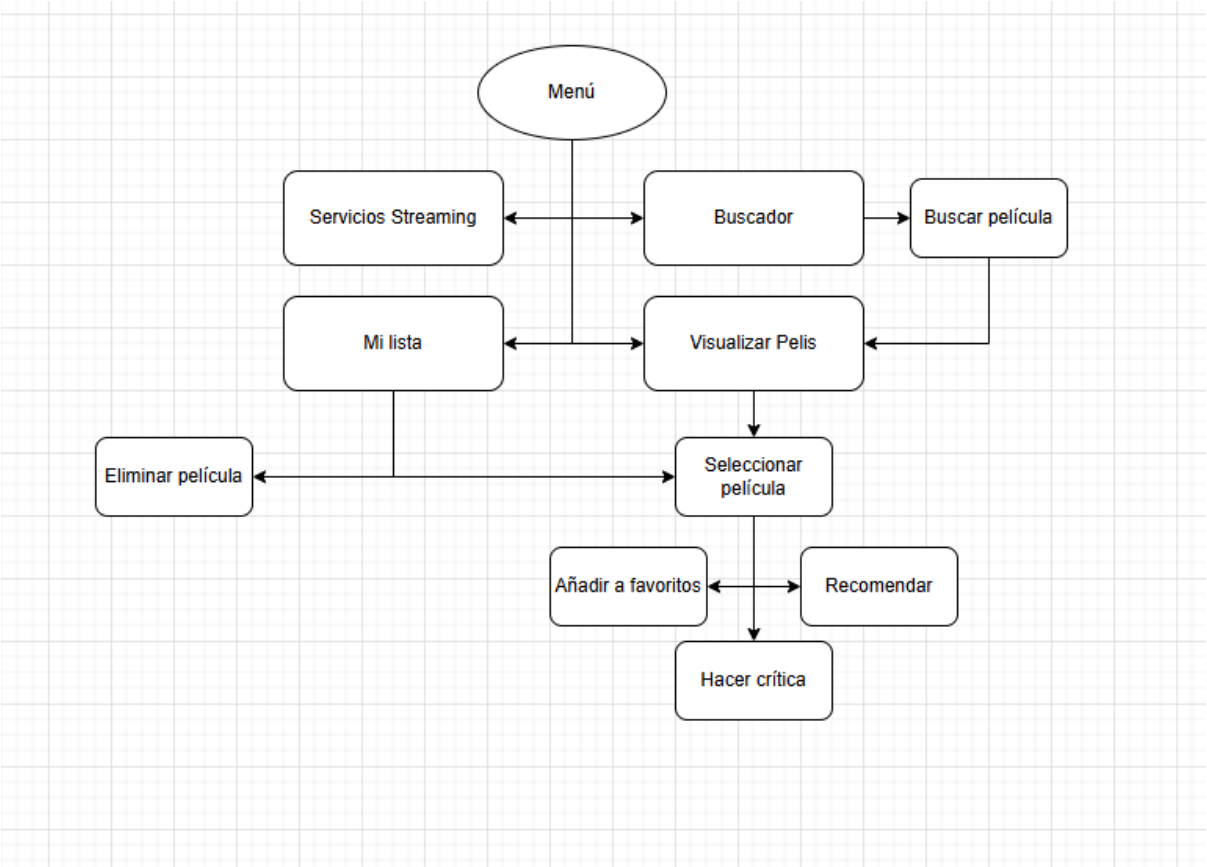
Otros componentes a destacar: Critica.jsx, Login.jsx, Register,jsx, Populares,jsx, ProfileCard.jsx App.jsx

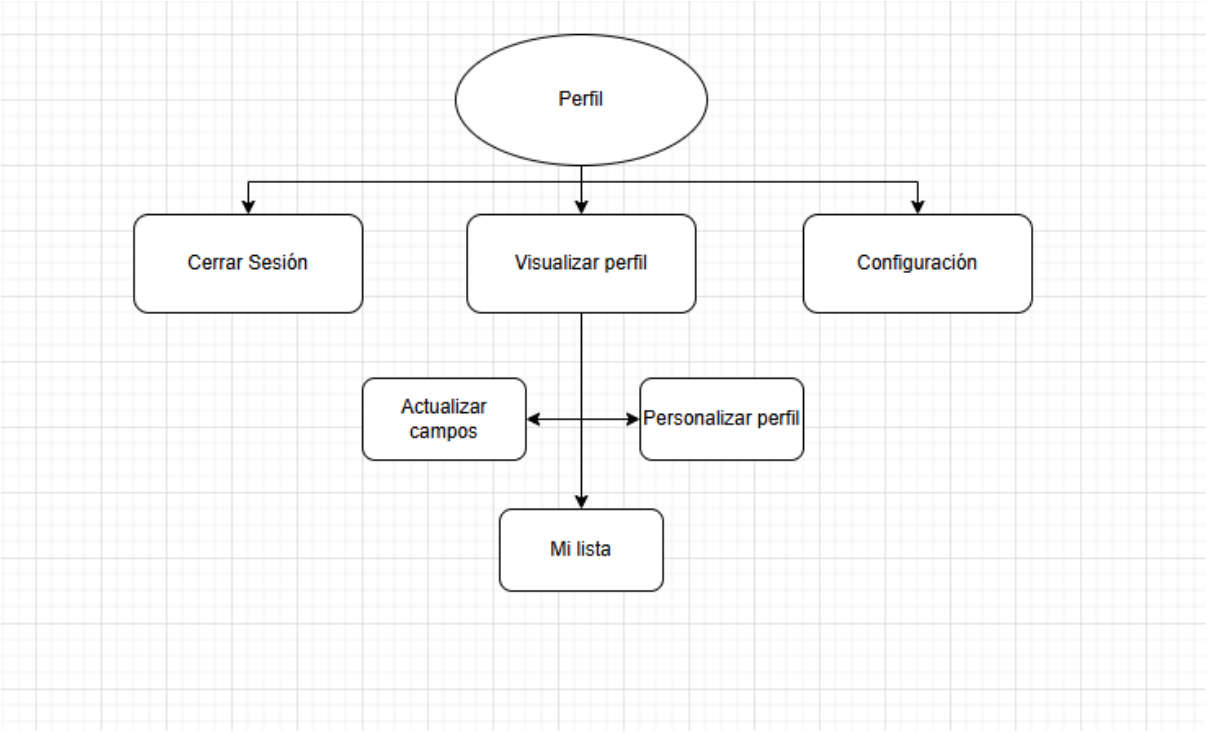
- Diagrama de Casos de Uso:



- Diagrama de Flujo de la Aplicación:

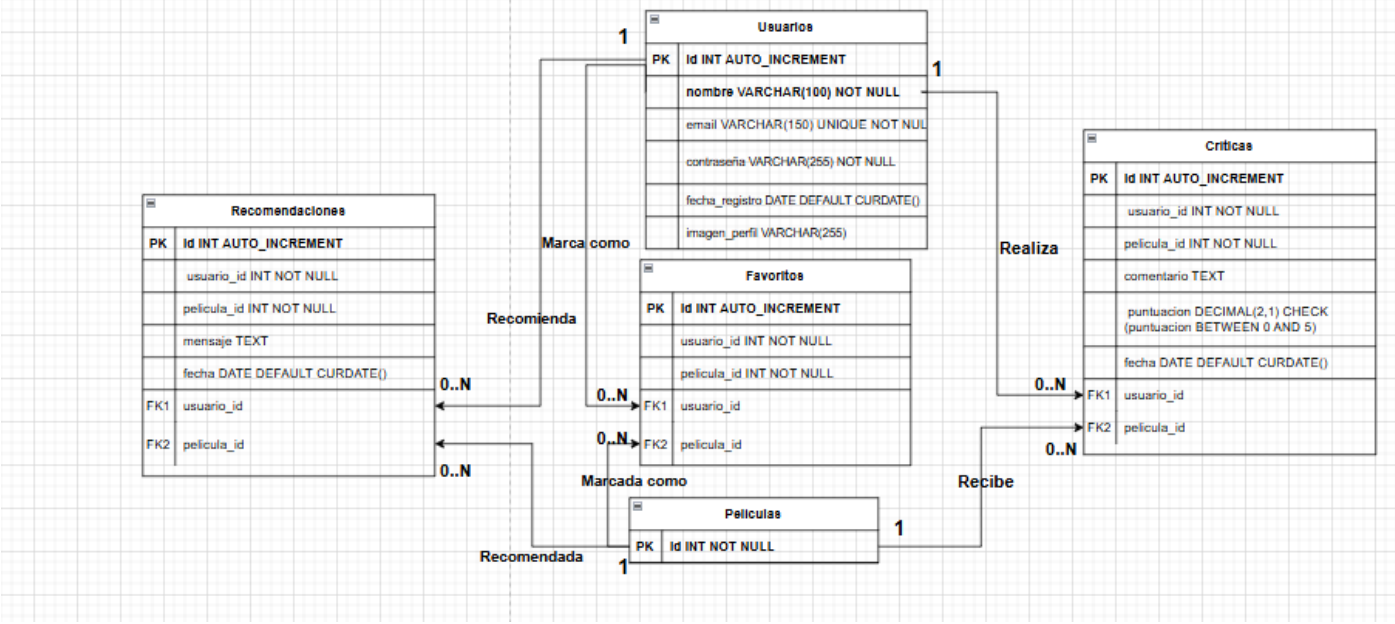



Menú
Servicios Streaming
Buscador
Buscar película
Mi lista
Visualizar Pelis
Eliminar película
Seleccionar película
Añadir a favoritos
Recomendar
Hacer crítica


Perfil
Cerrar Sesión
Visualizar perfil
Configuración
Actualizar campos
Personalizar perfil
Mi lista

- Diagrama Entidad-Relación (ER):

Este diagrama refleja la estructura de la base de datos, incluyendo tablas de usuarios, películas, críticas, recomendaciones, favoritos.



Usuarios:

Contiene la información básica de cada usuario registrado en la plataforma.Los nickname y email son únicos para evitar duplicados. Se añade imagen_perfil para personalización. Tiene relaciones 1 a muchos con recomendaciones, críticas y favoritos.

Películas:
Contiene los identificadores únicos de películas, usando el formato tt de IMDb.Se usa solo el ID como clave, ya que los detalles de la película se obtienen de la API externa.

Favoritos:
Relación entre usuarios y películas marcadas como favoritas. Relación muchos a muchos entre usuarios y películas. Se impone una restricción única por (usuario_id, pelicula_id) para evitar duplicados.

Críticas:
Almacena las reseñas que los usuarios hacen de películas .Permite valorar películas con una puntuación del 0 al 10 y añadir un comentario libre.

Recomendaciones:
Guarda las películas recomendadas por usuarios a otros usuarios.

Relaciones:

Las relaciones de la aplicación están creadas para juntar películas y usuarios, es decir, críticas, favoritos y recomendaciones pueden ser consideradas tablas intermedias entre las relaciones de usuario a película

- Usuarios ↔ Películas (Favoritos)

Tipo: Relación muchos a muchos (N:N)
Tabla intermedia: favoritos

Cada usuario puede marcar varias películas como favoritas, y cada película puede ser favorita de muchos usuarios.
Esto se implementa mediante una tabla intermedia favoritos con claves foráneas a usuarios y películas.

Se usa un índice único compuesto para evitar duplicados (usuario_id, pelicula_id).

- Usuarios ↔ Películas (Críticas)

Tipo: Relación muchos a muchos, extendida con atributos
Tabla intermedia: críticas

Los usuarios pueden realizar críticas de películas. Esta relación no solo enlaza ambas tablas, sino que además almacena atributos adicionales:

Comentario, puntuación, fecha de la crítica

Esto convierte a críticas en una relación con información propia (relación con entidad asociativa).

- Usuarios ↔ Películas (Recomendaciones)

Tipo: Relación muchos a muchos, extendida
Tabla intermedia: recomendaciones

Un usuario puede recomendar múltiples películas, y cada película puede ser recomendada por muchos usuarios.
Además, esta recomendación puede incluir un mensaje personalizado y una fecha.
La tabla recomendaciones funciona como una relación enriquecida, similar a criticas.

- Películas ↔ Favoritos / Críticas / Recomendaciones

En todos estos casos, pelicula_id actúa como clave foránea hacia la tabla peliculas.
Esto permite:

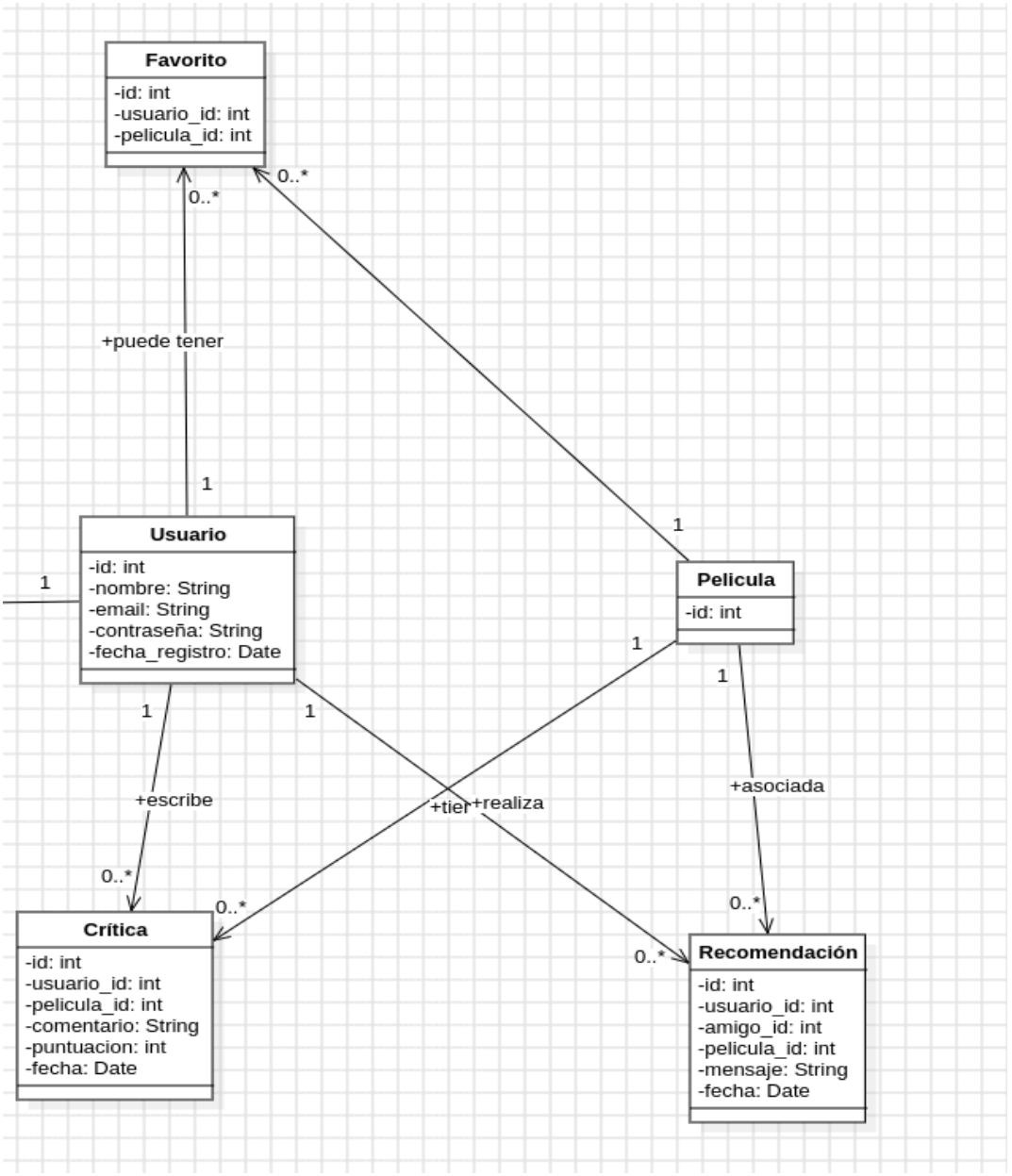
Controlar la integridad referencial (si se borra una película, se eliminan sus relaciones).
Reutilizar el ID de IMDb como clave primaria (sin necesidad de duplicar información como títulos, descripciones, etc., ya que se obtienen desde la API externa).

- Integridad referencial

Todas las relaciones están protegidas con FOREIGN KEYS que usan ON DELETE CASCADE, lo que asegura que si un usuario o película es eliminada, sus relaciones también se eliminan automáticamente.

- Diagrama de Clases UML

Este diagrama refleja las distintas clases y atributos de cada una de estas de nuestro proyecto. También, las relaciones entre las diferentes clases.

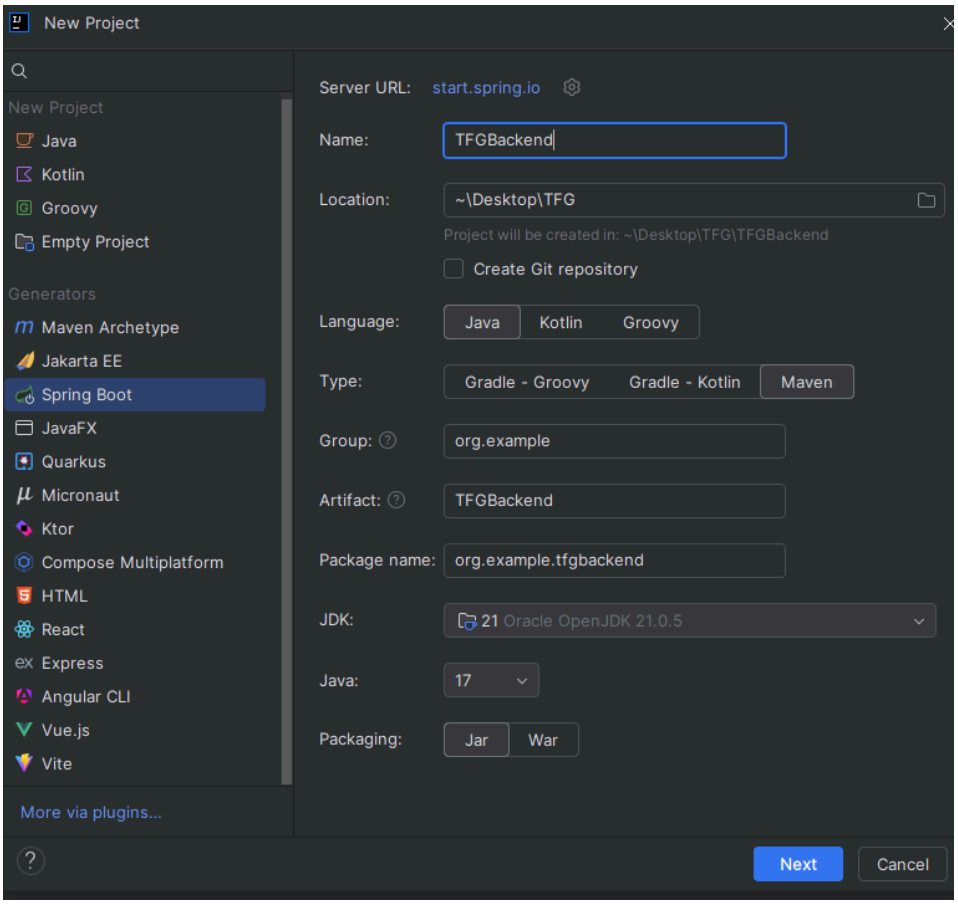


- Configuración y gestión de implementación del proyecto:

Requisitos previos del sistema
Describe qué se necesita instalar para que el proyecto funcione correctamente en local.

Herramientas	Versión mínima	Uso en el proyecto
Java JDK	17	Backend con SpringBoot
NodeJS + npm	18 / 9	Frontend con React
MySQL	8.0+	Base de datos
Maven	3.8+	Gestión de dependencias en backend
Git	Cualquiera	Control de versiones
IntelliJ / VS Code	Libre	Entornos de desarrollo

Configuración del entorno - backend
Crear base de datos FilMe en MySQL * explicar BD *
Crear el proyecto en Intelllj:
Nuevo proyecto Spring Boot con Java y Maven JDK-21, versión de java 17



La aplicación se inicia desde la clase TfgBackendApplication, ubicada en el paquete raíz, que utiliza la anotación @SpringBootApplication para cargar automáticamente todos los componentes, servicios y configuraciones
A continuación se detallan las propiedades más relevantes del archivo application.properties, que controla la conexión a la base de datos, comportamiento de JPA, tamaño máximo de archivos y otros ajustes:

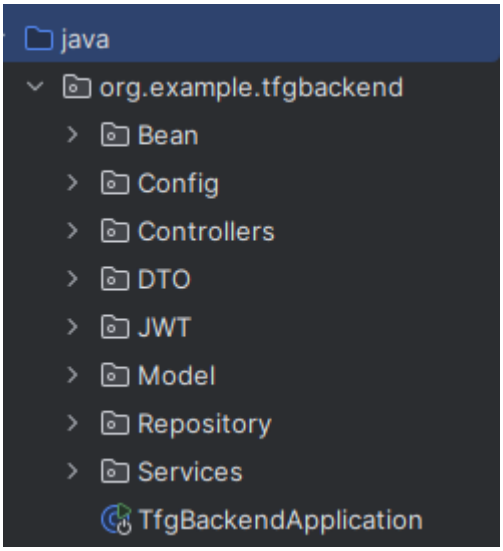
Clave	**Propósito**
server.port=8080	Puerto en el que se ejecuta el backend
spring.datasource.url	URL de conexión a la base de datos MariaDB (cineapp)
spring.datasource.username	Usuario de acceso a la base de datos
spring.jpa.hibernate.ddl-auto=update	Actualiza automáticamente las tablas en base al modelo Java
spring.jpa.database-platform	Dialecto MySQL para Hibernate
spring.servlet.multipart.max-file-size =1MB	Límite de tamaño para subir imágenes de perfil

Dependencias principales (pom.xml)
El archivo pom.xml define las librerías necesarias para el backend. Aquí están las más destacadas y su función:

Dependencia	**Función principal**
spring-boot-starter-web	Construir APIs REST
spring-boot-starter-data-jpa	ORM con Hibernate y conexión con MySQL/MariaDB
mariadb-java-client	Driver JDBC para conectar con la base de datos MariaDB

spring-boot-starter-validation	Validaciones con anotaciones como @NotBlank, @Size, etc.
spring-boot-starter-cache	Soporte de caché (usado con @Cacheable en controladores)
jjwt (API, Impl, Jackson)	Generación y validación de tokens JWT
spring-boot-starter-security	Configuración de seguridad básica
hibernate-core	Motor ORM para persistencia de entidades

Estructura:

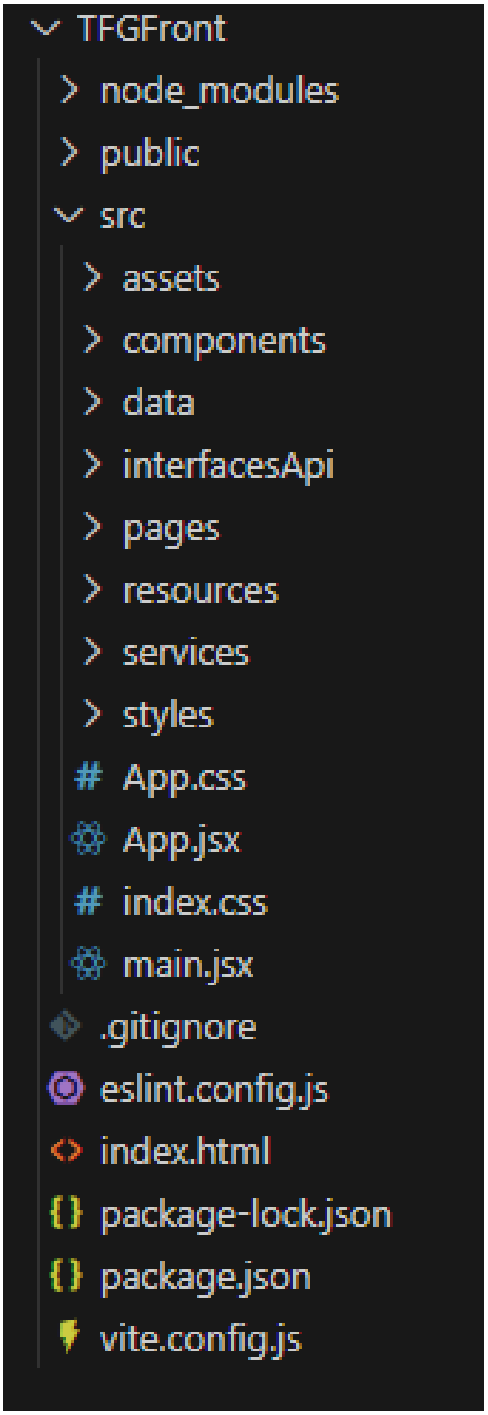


Configuración del entorno - frontend:

El frontend de *FilMe* está desarrollado con React instalado junto con Vite en Visual Studio Code

npm install para instalar las dependencias y npm run dev para cargar el proyecto, este corre en el servidor localhost:5173

Estructura:


TFGFront
node_modules
public
src
assets
components
data
interfacesApi
pages
resources
services
styles
App.css
App.jsx
index.css
main.jsx
.gitignore
eslint.config.js
index.html
package-lock.json
package.json
vite.config.js

- Pruebas:

Para llevar a cabo las pruebas del proyecto, ha sido necesario configurar previamente el entorno local para realizar un entorno de pruebas.

Inicio del servidor de base de datos:
Se inicia XAMPP activando el módulo de MySQL, que contiene la base de datos cineapp utilizada por la aplicación.

Pruebas del backend en Postman:
Antes de lanzar el frontend, se probaron los endpoints del backend utilizando Postman, verificando las respuestas de login, registro, perfil de usuario, subida de imágenes, etc.

Arranque del backend:
Se ejecuta el backend con Spring Boot desde el IDE (IntelliJ)

Arranque del frontend y pruebas:
npm run dev que abre el puerto 5173

Esto lanza la interfaz de usuario, que permite probar las funcionalidades conectadas con el backend. Para hacer debugging, hemos utilizado console.log, lógica de programación y el debugger.

Funcionalidad	**Descripción**
Registro de usuario	Al rellenar el formulario de registro desde el front, se agrega un usuario en la BD. Te redirige a /login y te muestra un mensaje de éxito
Registro de usuario	Comprobar las validaciones del backend en el formulario de registro. Número de caracteres, nickname o email repetidos, mensajes de error mostrados correctamente
Inicio de sesión	En el postman comprobamos que la respuesta del método en un JWT. En el front mandamos un formulario con los datos (correspondientes de la BD) y comprobamos el JWT se guarde en localStorage. Mensaje de error por credenciales incorrectas o redirección a página inicial si todo es correcto
Cerrar sesión	Al clicar el botón se cierra sesión, te redirige a login y borra el

	token del localStorage
Obtener perfil	En el postman te devuelve el usuario con el JWT. En el front recogemos el JWT del localStorage y comprobamos que se carguen bien los datos.
Insertar imagen en el perfil	Se actualiza en tiempo real la subida de la imagen y también al cambiar tu imagen de perfil. Al buscar al usuario, se muestra correctamente su imagen de perfil.
Actualizar datos del usuario	Se actualizan los datos desde el input del front a la base de datos y sale el mensaje correctamente.
Listado de películas	Se muestran todas las películas con imagen y título recogidas de la API externa.
Movie detail	Al clicar en una película en el listado de películas, esta te redirige correctamente a los detalles de la película pulsada.
Añadir película a favoritos	Cuando pulsas el icono de estrella, se añade a favoritos en la BD (se hace la relación correcta de el ID de esa pelicula y el ID del usuario logueado). Cuando pulsas el icono de la estrella rellena se elimina el favorito de la BD
Mi lista	Los favoritos se añaden y borran dinámicamente a mi lista.
Recomendar película	Cuando pulsas el icono de recomendación, se añade a recomendados en la BD. Muestra un mensaje de éxito o un mensaje de que ya está recomendada en caso de que lo esté
Buscador de usuarios	Busca bien y cualquier cadena, coincide con la base de datos y te redirige bien al perfil correcto.
Críticas y valoraciones	Cuando se inserta una crítica correctamente se añade bien al perfil recogiendo bien la puntuación y redirige al perfil con mensaje de éxito
Portal FilMe	Muestra bien las películas favoritas, recomendadas y críticas en función del usuario. Si es el usuario que está en sesión, muestra la opción de cambiar imagen, si es un usuario público no la muestra.
Accesibilidad y responsive	Comprobando el responsive en cada página desde las devTools del navegador

- Errores:

Nos hemos encontrado con muchos errores a lo largo del desarrollo de la aplicación, la mayoría de ellos se daban al implementar métodos del backend en el frontend y a la hora de dar funcionalidad en la aplicación. Todos los errores han sido menores y con prueba y error, console.log y lógica de programación se han podido solucionar fácilmente. Alguno de ellos son:

Errores en las respuestas: Al obtener los métodos del backend en los services del frontend nos hemos encontrado con errores, ya que hay que estar atentos de si estos métodos responden con json o con string.

Carga de perfil: En nuestra aplicación se obtiene un JWT al hacer el login y con este obtenemos los datos del perfil. Al principio estaba establecido que el perfil se cargue con el token pasado por prop, pero al actualizar el perfil, se rompía el método getUser. Esto venía debido a que al actualizar el nickname el token cambia, por lo que tuvimos que generar un token nuevo en el método del back en el cual se actualiza el nickname y cogerlo siempre de localStorage en getUser() y de esa manera siempre se establece el JWT correcto.

Obtención de la API: Desde el postman hemos tenido que realizar pruebas para obtener los datos de la API, al ser API públicas daban errores por claves erróneas, falta de parámetro etc.

Obtención de datos de la API: Al cargar las APIs externas hemos tenido problemas para obtener los datos. La forma que hemos encontrado de saber detallada y fácilmente qué datos devuelve la API es creando una interfaz en TypeScript copiando la respuesta de la API(en el postman) gracias a Visual Studio Code

Carga de imágenes: Al cargar la imagen y hacer el upload, tuvimos diversos problemas en cuanto a la sincronización de las funciones pertinentes del frontend y la escritura del archivo en el backend y la base de datos, y es que el funcionamiento tenía un comportamiento anómalo y no cargaba la imagen en tiempo real o cargaba la anterior seleccionada. Fue resuelto de manera que, hasta que no terminase el flujo de subida de la imagen, primero, ejecución de la función upload del front, llega a un endpoint del backend y este escribe dicho archivo en una carpeta de uploads y se guarda en el usuario, pues hasta que este proceso no acaba, no se ejecuta refreshUser() que actualiza los datos del usuario (ya que se ha cambiado la imagen) y así tomamos la imagen guardada y ya podríamos mostrarla adecuadamente.

Carga del portalFilme: El error que nos hemos encontrado era que PortalFilMe al entrar por primera vez (antes que entrar en Populares que ahí se cargan todas las películas), en esta pestaña las películas que haya en favoritos, recomendaciones y críticas se cargan de una en una con getMovieById(), ya que no hay algún endpoint en la API que nos permita otras solicitudes, entonces al haber muchas peticiones simultáneamente nos dio un error HTTP 429 “Throttling Exception”, la manera de solucionar esto fue hacer dos estados que nos permitieran comprobar primero que cargasen todos los favoritos, hasta que no

terminara, que no cargaran las recomendaciones y hasta que no terminaran estas, que no cargaran las críticas, y sin bajar rendimiento y evitando ese error.

No EntityManager: Error de Spring JPA(Hibernate). Cuando se ejecuta una operación de escritura (en este caso un delete) pero no hay una transacción activa para esa operación. El EntityManager (el objeto interno que gestiona las operaciones en base de datos) necesita estar dentro de una transacción para realizar un delete.
Se soluciona utilizando la anotación @Transactional de Spring
Spring - No EntityManager with actual transaction available for current thread - cannot reliably process 'persist' call - Stack Overflow

Errores en las validaciones: Hemos tenido problemas al mostrar los errores de validación del backend en el frontend, finalmente usando el `@RestControllerAdvice` y `@ExceptionHandler` se han podido mandar mensajes al frontend recogidos desde el service como objetos con claves {success, data} y mostrados en el jsx(html) como `<p className="error">{errors.nombre || ' '}</p>`

Error al detectar duplicados: En el backend nos aseguramos que no se repitan nicknames o emails en los registros de usuarios utilizando el método existsBy del repository, lo que muestra el error correctamente en el backend, pero al enviar los datos desde el formulario del frontend, ignoraba el error. Esto se debía a la diferencia entre mayúsculas y minúsculas, lo solucionamos normalizando todo con toLowerCase() en los nicknames ya que daba problemas.

- Mejoras futuras

Aunque FilMe ya cuenta con una base sólida y funcionalidades completas que permiten al usuario gestionar su experiencia cinematográfica, el proyecto está planteado para poder crecer y evolucionar a medida que aumenten sus necesidades o usuarios.

A continuación, se describen algunas posibles mejoras y ampliaciones que podrían implementarse en el futuro, tanto a nivel funcional como técnico, con el objetivo de enriquecer la experiencia del usuario, optimizar el rendimiento y aumentar la escalabilidad del sistema.

Sistema de seguidores

- Implementar la opción de seguir a otros usuarios y no solo acceder a perfiles desde el buscador.
- Cuentas públicas o privadas

Sistema de notificaciones

- Notificar a los usuarios cuando reciben una recomendación, crítica o seguimiento.
- Uso de WebSockets o notificaciones push.

Sistema de amigos y chat privado

- Permitir enviar mensajes dentro de la plataforma.
- Añadir funciones sociales que favorezcan la interacción entre usuarios.

Chatbot

- Chatbot con Inteligencia artificial
- Chat personal para comunicación e información sobre la aplicación

Buscador avanzado de películas

- Añadir filtros por género, año, duración o valoración media.
- Autocompletado con sugerencias.

Api mejorada

- API externa premium con todas las películas y series posibles
- Datos avanzados y miles de películas y series

Despliegue en entorno cloud

- Implementar despliegue automático en plataformas como AWS y dockers
- Uso de Docker para empaquetado.

- Conclusión

FilMe ha sido un proyecto que nos ha permitido aplicar de manera práctica los conocimientos adquiridos a lo largo del ciclo formativo de Desarrollo de Aplicaciones Web. Desde la idea inicial hasta el final de la aplicación, hemos atravesado todas las fases del desarrollo de software: análisis, diseño, implementación, pruebas y mejora continua.

La aplicación logra su principal objetivo: ofrecer una plataforma que centraliza toda la experiencia cinéfila en un solo lugar, combinando la consulta de información actualizada sobre películas con funciones sociales como recomendaciones, críticas, favoritos y perfiles personalizados. Esta integración proporciona al usuario una experiencia más rica, dinámica y social en torno al mundo del cine.

Durante el desarrollo, hemos trabajado con tecnologías modernas como React para el frontend, Spring Boot y Java para el backend, y MySQL como sistema de gestión de bases de datos. Además, hemos implementado mecanismos de seguridad como JWT para proteger las sesiones y facilitar una navegación segura. Todo esto ha contribuido a construir una aplicación robusta, escalable y fácilmente ampliable en el futuro.

Uno de los mayores aprendizajes ha sido saber integrar correctamente el frontend con el backend, consiguiendo que ambas partes trabajen en conjunto de forma eficiente. Gracias a esto, hemos comprendido cómo se comunican los distintos componentes de una aplicación web, cómo estructurar una API y cómo consumirla desde el cliente, gestionando estados, sesiones y datos de forma segura y coherente y con un gran rendimiento.

Además, trabajar con APIs externas nos ha enfrentado a nuevos retos técnicos, desde el tratamiento y limpieza de datos hasta la gestión de errores o la adaptación de formatos de respuesta. Esta experiencia nos ha enseñado a enfrentarnos a problemas reales y a buscar soluciones prácticas, mejorando tanto nuestras habilidades técnicas como nuestra capacidad para resolver errores y adaptar el código a cada situación.

También hemos aprendido a trabajar en equipo, organizarnos, dividir tareas y colaborar a través de herramientas como GitHub y Discord, lo que ha sido clave para mantener un flujo de trabajo constante y resolver los inconvenientes que han ido surgiendo.

En resumen, FilMe ha sido una experiencia formativa que nos ha enfrentado a los retos reales del desarrollo web. Este proyecto nos deja preparados para afrontar nuevos desafíos en el mundo laboral y marca el cierre de una etapa académica con un producto funcional, atractivo y con potencial de crecimiento.

Bibliografía:

[1] W3Schools " Tutorial de React" React Tutorial

[2] React “ Biblería de JavaScript para construir interfaces de usuario" React

[3] Spring.io " Documentación de referencia de Spring Boot" Spring Boot :: Spring Boot

[4] MySQL " Manual de referencia de MySQL" MySQL :: MySQL 8.0 Reference Manual

[5] IMDb API " Página de referencia para películas" IMDb: calificaciones, opiniones y dónde ver las mejores películas y programas de TV

[6] Iconify " Iconify – Framework unificado de iconos" Iconify Design: All popular icon sets, one framework.

[7] MDN Web Docs " Guía de JavaScript" Guía de JavaScript - JavaScript | MDN

[10] JetBrains " IntelliJ IDEA – Herramienta de desarrollo para Spring Boot" IntelliJ IDEA – the IDE for Pro Java and Kotlin Development

[11] Postman " Plataforma para construir y utilizar APIs" Postman: The World's Leading API Platform | Sign Up for Free

[12] RapidAPI " Plataforma obtener APIs" Rapid API Marketplace & Management Tools

[13] JWT " JSON Web Token para gestionar las sesiones" JSON Web Token Introduction - jwt.io

[14] GitHub "Control de versiones" GitHub

[14] StackOverflow "Errores de springBoot" Stack Overflow