

PRÁCTICA FINAL

Programación de Aplicaciones Telemáticas



PAULA BONET, ALEJANDRA BOTÍN, JOEL ALAYÓN
MARIO VIDAL DOMÍNGUEZ
3ºB GITT-BA
20 DE MAYO DE 2025

ÍNDICE

1. INTRODUCCIÓN	2
1.1 Descripción del proyecto.....	2
2. FUNCIONALIDADES DEL SISTEMA	2
2.1 Funcionalidades para usuarios estándar.....	2
2.2 Funcionalidades para administradores	3
2.3 Navegación y experiencia de usuario.....	3
3. MODELO VISTA CONTROLADOR (MVC)	3
4. TESTING.....	4
4.1 Pruebas unitarias.....	4
4.2 Pruebas de integración.....	4
4.3 Pruebas end-to-end (E2E).....	4
4.4 Herramientas utilizadas.....	4
4.5 Organización de las pruebas	4
4.6 Cobertura	5
5. DESARROLLO DEL PROYECTO.....	5
5.1 Estructura	5
5.2 Funcionalidades Clave	5
5.3 Endpoints	5
6. FUNCIONAMIENTO (USER/ADMIN)	7
6.1 Registro de Usuario (Register)	7
6.2 Inicio de Sesión (Login).....	8
6.3 Vista de Catálogo.....	8
6.4 Detalle de Contenido	8
6.5 Wishlist (Lista de Deseos)	9
6.6 Cerrar Sesión (Logout)	9
6.7 Vista de Administrador.....	9
7. DESPLIEGUE Y EJECUCIÓN	10
8. REPARTO DE TAREAS	10
8.1 Metodología de trabajo.....	10
9. CONCLUSIONES.....	11

1. INTRODUCCIÓN

1.1 Descripción del proyecto

Este proyecto consiste en el desarrollo de una **aplicación web** que permite a los usuarios gestionar contenido multimedia, como **películas y libros**, mediante funcionalidades clave como:

- Registro e inicio de sesión.
- Visualización del catálogo de contenido.
- Añadir contenido a una lista de deseos (wishlist).
- Crear y consultar reseñas (reviews).
- Cerrar sesión y mantener la autenticación mediante cookies.

Está construido sobre una arquitectura cliente-servidor utilizando **Spring Boot** en el backend, y se comunica a través de una **API RESTful**. Los datos se almacenan en una base de datos embebida **H2** y el sistema está preparado para **despliegue en la nube** usando Render.

1.1.1 Objetivos del proyecto

- Desarrollar un backend funcional con Spring Boot que gestione usuarios, contenido y reviews.
- Aplicar buenas prácticas de diseño de software, incluyendo una arquitectura en capas.
- Implementar seguridad mediante sesiones y cookies.
- Realizar pruebas unitarias y de integración que verifiquen el correcto funcionamiento del sistema.
- Organizar el desarrollo en **sprints**, simulando un entorno profesional ágil.

1.1.2 Justificación

El desarrollo de este sistema permite aplicar en un proyecto real múltiples conocimientos adquiridos en el curso, incluyendo:

- Desarrollo web con Java y Spring Boot.
- Acceso a bases de datos con JPA.
- Seguridad web básica (cookies de sesión).
- Testing (JUnit 5, TestRestTemplate).
- Metodologías ágiles y control de versiones con Git.

Además, se ha elegido este tema porque permite trabajar con modelos de datos interesantes, aplicar lógicas de negocio reales (wishlist, reviews) y tener un caso de uso atractivo para cualquier usuario final.

Está destinado a usuarios interesados en explorar y comentar sobre obras audiovisuales y literarias, con una experiencia sencilla e intuitiva desde cualquier navegador. Creando así una comunidad entre usuarios, y permitiéndoles interactuar entre ellos.

2. FUNCIONALIDADES DEL SISTEMA

2.1 Funcionalidades para usuarios estándar

- ✓ Registro e inicio de sesión (autenticación).
- ✓ Acceso a un catálogo de películas y libros.
- ✓ Visualización del detalle de cada elemento del catálogo.

- ✓ Publicación de comentarios.
- ✓ Gestión de listas de deseos (wishlist)

2.2 Funcionalidades para administradores

- ✓ Gestión de catalogo (creación de libros y películas)

2.3 Navegación y experiencia de usuario

- ✓ Página inicial con acceso a login y registro.
- ✓ Vista principal del catálogo una vez autenticado.
- ✓ Página de detalle de cada obra con comentarios y botón para añadir nuevo contenido (si se es admin).

3. MODELO VISTA CONTROLADOR (MVC)

El proyecto sigue una arquitectura basada en el patrón Modelo-Vista-Controlador (MVC), que permite una separación clara de responsabilidades, facilitando el mantenimiento, escalabilidad y pruebas del sistema. En esta estructura, los **Modelos** representan las entidades del dominio como usuarios, contenidos o reviews, y se encargan de mapear la información de la base de datos. Los **Controladores** gestionan las peticiones HTTP, actuando como intermediarios entre la lógica de negocio y las respuestas al cliente. Por último, aunque este proyecto se centra en el backend, las **Vistas** estarían representadas en el frontend que consume esta API REST. Esta división permite trabajar de forma modular y colaborativa, permitiendo que distintas partes del equipo desarrollen componentes de forma simultánea y ordenada.

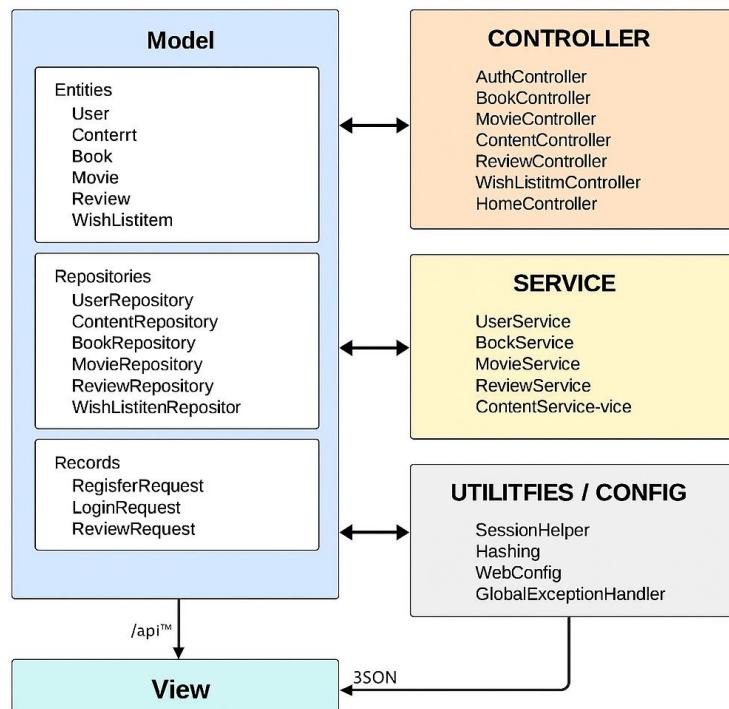


Figura 1: Diagrama MVC del Proyecto

4. TESTING

Durante el desarrollo del proyecto se ha aplicado una estrategia de pruebas completa y estructurada para garantizar la calidad del software, la estabilidad del sistema y la correcta implementación de la lógica de negocio. Se han aplicado tres tipos principales de pruebas:

4.1 Pruebas unitarias

Validan el comportamiento de clases individuales como los modelos (por ejemplo, User, Book, Movie, etc.) y servicios con lógica sencilla. Estas pruebas comprueban:

- Creación de objetos.
- Correcto funcionamiento de getters/setters.
- Comportamiento esperado de registros (RegisterRequest, LoginRequest, etc.).

4.2 Pruebas de integración

Evalúan que los distintos componentes (repositorios, servicios y controladores) interactúan correctamente dentro del contexto de Spring Boot. Aquí se trabaja con la base de datos real (H2 en memoria) y sin mocks.

Estas pruebas confirman que:

- El flujo entre repositorio, servicio y controlador es funcional.
- Los datos se guardan, actualizan o eliminan correctamente.
- Se accede a los endpoints con los permisos adecuados.

4.3 Pruebas end-to-end (E2E)

Simulan escenarios completos desde el punto de vista del usuario final. Se usan para verificar que el sistema se comporta correctamente ante flujos reales, como:

- Registro → Login → Obtener perfil.
- Añadir a wishlist → Ver wishlist.
- Crear review → Obtener reviews.
- Logout → Acceso denegado.
-

Estas pruebas se ejecutan contra la aplicación en puerto real (DEFINED_PORT) usando TestRestTemplate.

4.4 Herramientas utilizadas

- **JUnit 5**: framework principal para todas las pruebas.
- **Spring Boot Test**: contexto completo para integración y E2E.
- **TestRestTemplate**: para lanzar peticiones HTTP en pruebas E2E.
- **H2**: base de datos en memoria utilizada durante los tests.

4.5 Organización de las pruebas

El código de pruebas está dividido en tres paquetes principales:

- com.paulabonets.peliculas.unit → pruebas unitarias de modelos y estructuras simples.
- com.paulabonets.peliculas.integration → pruebas de integración de servicios, lógica y repositorios.
- com.paulabonets.peliculas.e2e → pruebas de extremo a extremo simulando escenarios reales.

4.6 Cobertura

Las pruebas cubren:

- Modelos: atributos y lógica básica.
- Servicios: lógica interna sin necesidad de mocks.
- Endpoints principales: registro, login, logout, wishlist, reviews, perfil.
- Comportamientos de error: acceso sin sesión, eliminar review de otro usuario, etc.

5. DESARROLLO DEL PROYECTO

5.1 Estructura

El backend de la aplicación se ha desarrollado en **Java 17 con Spring Boot** y está organizado en paquetes según la **responsabilidad funcional**, siguiendo una arquitectura tipo Modelo-Vista-Controlador (MVC). A continuación, se describen las principales capas del sistema:

5.1.1 Backend (src/main/java/com/paulabonets/peliculas)

- **controller**: contiene los controladores REST que exponen los distintos endpoints de la API.
- **model**: define las entidades JPA del sistema como User, Content, Book, Movie, Review, WishListItem, etc.
- **repository**: incluye interfaces que extienden JpaRepository para interactuar con la base de datos.
- **service**: implementa la lógica de negocio de cada entidad.
- **util**: clases auxiliares como SessionHelper para gestionar autenticaciones.
- **Seeder**: inicializa los datos al arrancar la aplicación (por ejemplo, cargando libros y películas).
- **records**: estructuras DTO para transportar datos, como RegisterRequest, LoginRequest o ReviewRequest.

5.1.2 Frontend (src/main/resources/static)

El frontend se implementa con **HTML, CSS y JavaScript**, y está ubicado dentro del directorio resources/static. Las principales vistas son:

- index.html: muestra el catálogo principal (contenido público).
- login.html: formulario de inicio de sesión.
- register.html: formulario de registro de usuario.
- Archivos JavaScript y hojas de estilo (.js y .css) que soportan la lógica del cliente.

5.2 Funcionalidades Clave

- **Autenticación personalizada** con sesiones gestionadas por cookies, validación de credenciales, y roles (USER, ADMIN).
- **Sistema de reviews**, wishlist, registro y login de usuarios.
- **Filtros y búsqueda** por tipo de contenido y título.
- **Endpoints RESTful** para toda la lógica de negocio.

5.3 Endpoints

A continuación, se detalla la API REST organizada por funcionalidad:

Autenticación – /api/auth

Método	Endpoint	Descripción	Requiere sesión	Código esperado
POST	/register	Registra un nuevo usuario	✗ No	200 OK
POST	/login	Inicia sesión y devuelve una cookie	✗ No	200 OK
GET	/me	Devuelve datos del usuario autenticado	✓ Sí	200 OK
POST	/logout	Cierra sesión y elimina la cookie	✓ Sí	200 OK

Wishlist – /api/wishlist

Método	Endpoint	Descripción	Requiere sesión	Código esperado
GET	/	Devuelve la lista de deseos del usuario	✓ Sí	200 OK
POST	/{contentId}	Añade un contenido a la wishlist	✓ Sí	200 OK
DELETE	/{contentId}	Elimina un contenido de la wishlist	✓ Sí	200 OK

Reviews – /api/reviews

Método	Endpoint	Descripción	Requiere sesión	Código esperado
POST	/	Crea una nueva review para un contenido	✓ Sí	201 Created
GET	/content/{contentId}	Devuelve todas las reviews asociadas a un contenido	✓ Sí	200 OK
GET	/{id}	Devuelve los detalles de una review específica	✓ Sí	200 OK
DELETE	/{id}	Elimina una review del usuario actual	✓ Sí	200 OK

Libros – /api/books

Método	Endpoint	Descripción	Requiere sesión	Código esperado
POST	/	Crea un nuevo libro	✗ No (o depende del control de acceso)	201 Created
PUT	/{id}	Actualiza los datos de un libro	✗ No (o depende del control de acceso)	200 OK

Películas – /api/movies

Método	Endpoint	Descripción	Requiere sesión	Código esperado
POST	/	Crea una nueva película	✗ No (o depende del control de acceso)	201 Created
PUT	/{id}	Actualiza los datos de una película	✗ No (o depende del control de acceso)	200 OK

Contenido General – /api/content

Método	Endpoint	Descripción	Requiere sesión	Código esperado
GET	/	Lista todos los contenidos con filtros opcionales (type, title)	✗ No	200 OK
GET	/{id}	Devuelve un contenido por su ID	✗ No	200 OK
DELETE	/{id}	Elimina un contenido por su ID	✗ No (o depende del control de acceso)	204 No Content

Página principal – /

Método	Endpoint	Descripción	Requiere sesión	Código esperado
GET	/	Redirige a la página de login (/login.html)	✗ No	302 Redirect

6. FUNCIONAMIENTO (USER/ADMIN)

Seguidamente se detallan las distintas pantallas de la aplicación con una breve explicación de cada una. Puedes incluir una imagen debajo de cada punto para ilustrar la funcionalidad:

6.1 Registro de Usuario (Register)

Pantalla con un formulario para registrar un nuevo usuario introduciendo:

- Nombre
 - Correo electrónico
 - Contraseña
- Una vez registrado, el usuario puede iniciar sesión en el sistema.

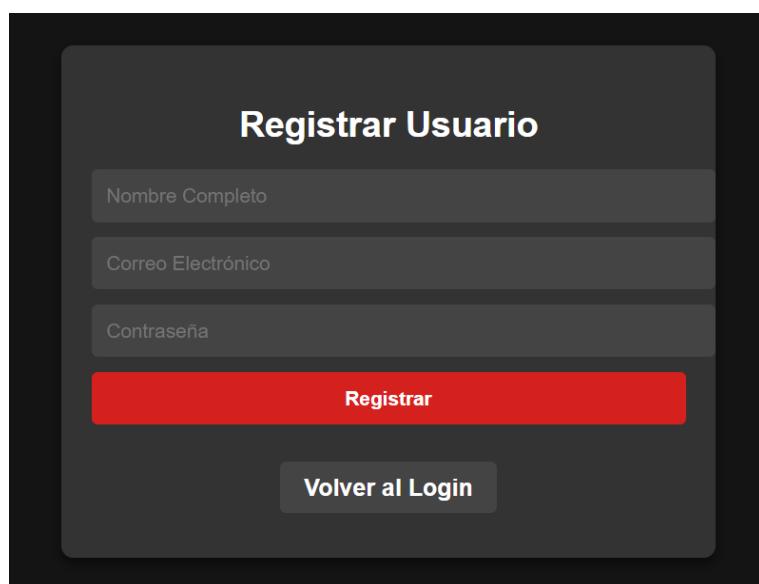


Figura 2: Página de Registro

6.2 Inicio de Sesión (Login)

Formulario de autenticación donde el usuario introduce su email y contraseña. Al iniciar sesión correctamente, es redirigido a la vista principal del catálogo.

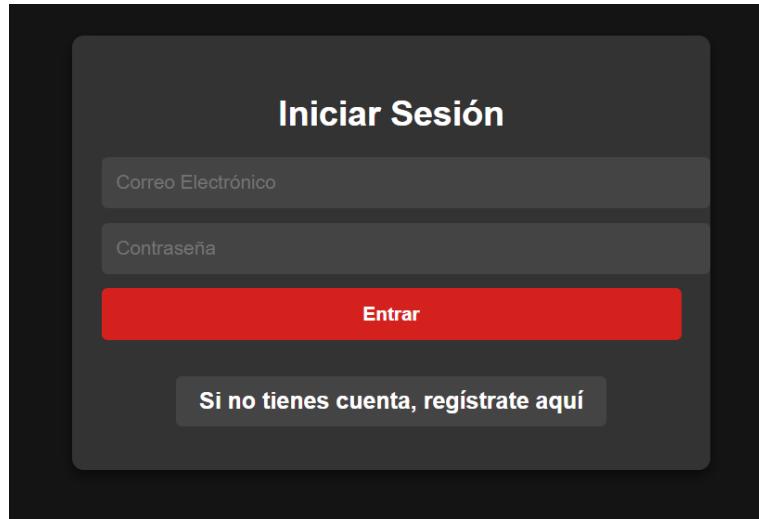


Figura 2: Página de Login

6.3 Vista de Catálogo

Pantalla principal tras el login.

Muestra todos los contenidos (libros y películas) disponibles en el sistema.

Incluye filtros por tipo (BOOK, MOVIE)

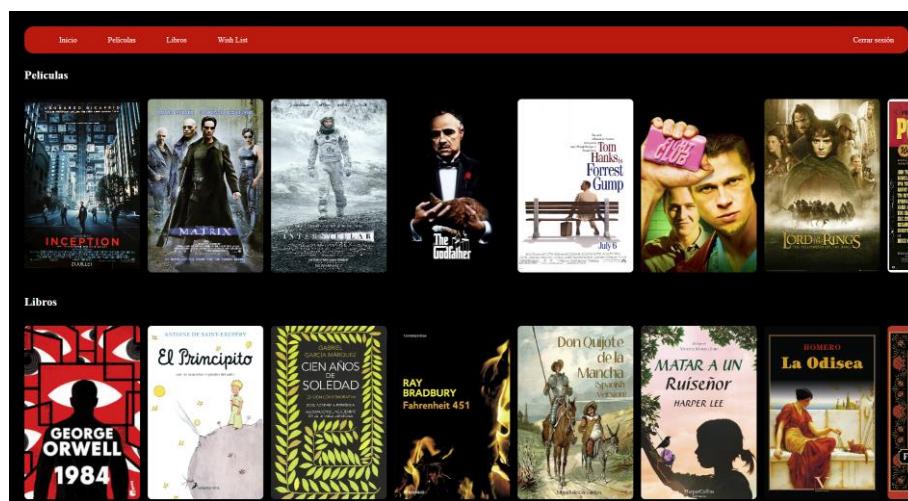


Figura 3: Página de Inicio

6.4 Detalle de Contenido

Al hacer clic en un contenido del catálogo, se accede a su vista detallada.

Se muestra información ampliada como:

- Descripción
- Imagen
- Autor/director
- Reviews de otros usuarios

Añadir Reseña (Review)

Formulario para dejar una review sobre un contenido.

Permite introducir:

- Puntuación (1 a 5 estrellas)
 - Comentario
- La review queda asociada al contenido y al usuario autenticado.

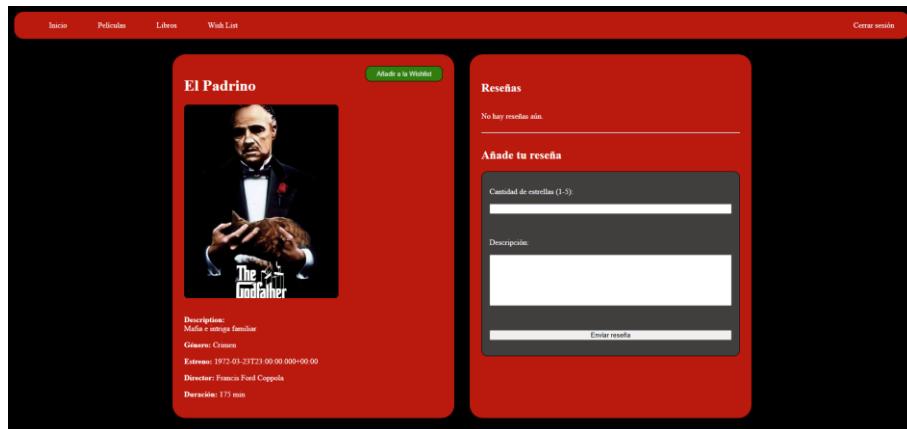


Figura 4: Página de Detalle de Película

6.5 Wishlist (Lista de Deseos)

Pantalla donde el usuario visualiza los contenidos que ha guardado como favoritos.

Permite añadir o eliminar contenidos desde la vista de detalle.

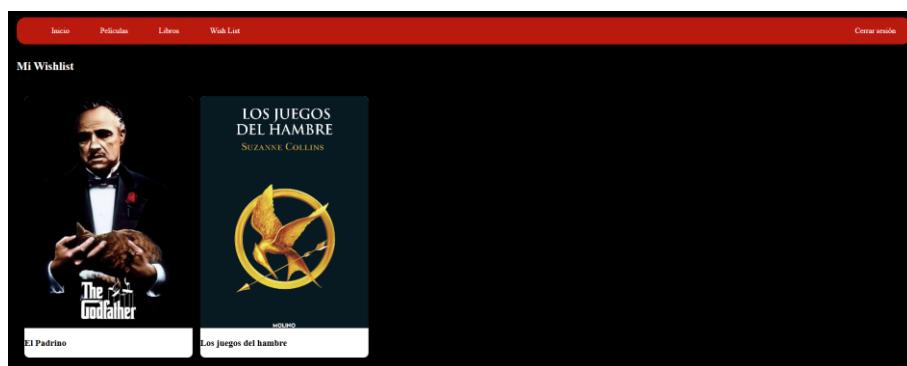


Figura 5: Página de Wishlist

6.6 Cerrar Sesión (Logout)

Botón para finalizar la sesión del usuario.

Elimina la cookie de sesión y redirige al login.

6.7 Vista de Administrador

Si el sistema incluye un usuario con rol ADMIN, esta pantalla permite:

- Crear nuevos contenidos
- Editar libros/películas existentes
- Eliminar contenidos
(Solo visible si el rol del usuario lo permite)

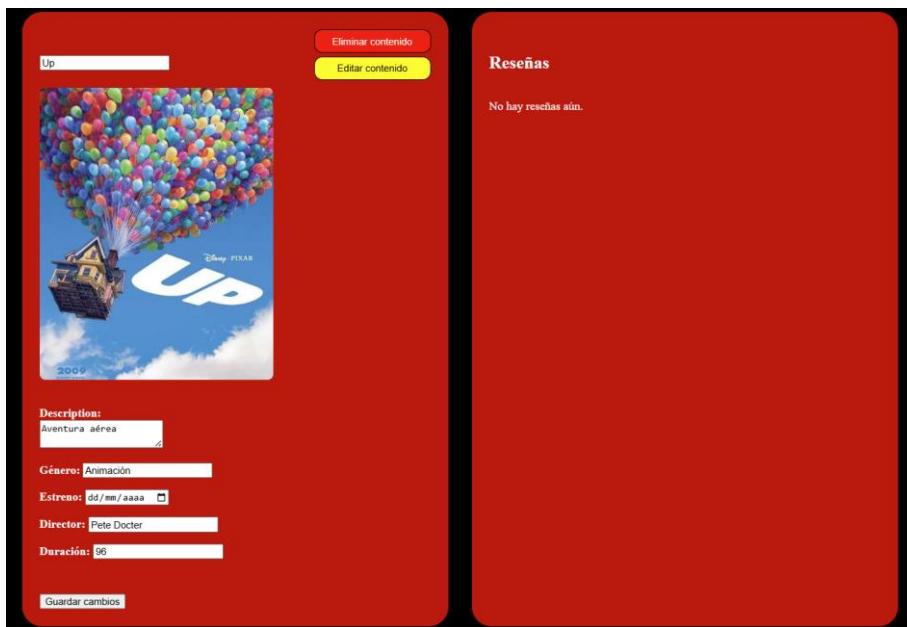


Figura 6: Página de Administrador

7. DESPLIEGUE Y EJECUCIÓN

La aplicación está desplegada en Render usando un contenedor de Docker.

8. REPARTO DE TAREAS

El desarrollo del proyecto se ha realizado en equipo, distribuyendo las tareas según las fortalezas e intereses de cada miembro:

Nombre	Rol principal	Tareas realizadas
Joel	Backend developer	Desarrollo de gran parte de la lógica de backend, controladores, servicios y repositorios.
Alejandra	Full-stack developer & testing lead	Parte del backend, integración con el frontend, y desarrollo completo de tests (unitarios, integración y E2E).
Paula	Frontend developer & documentación	Frontend: Maquetación de interfaces, pruebas de usuario y redacción completa de la documentación del proyecto.

8.1 Metodología de trabajo

El equipo ha seguido un enfoque colaborativo, inspirado en prácticas ágiles:

- **Distribución clara de responsabilidades.**
- **Revisión mutua de código**, especialmente en puntos críticos como la autenticación o la gestión de datos.
- **Control de versiones con Git**, usando ramas separadas para funcionalidades y realizando merges tras validación y pruebas.
- **Comunicación constante** por medio de herramientas de mensajería y gestión de tareas.

9. CONCLUSIONES

El desarrollo de este proyecto ha supuesto una experiencia enriquecedora tanto a nivel técnico como organizativo. A lo largo de su realización, hemos profundizado en conceptos clave del desarrollo web full-stack, poniendo en práctica las buenas prácticas de arquitectura, diseño, pruebas y colaboración en equipo.

Desde el lado del **backend**, el uso de **Spring Boot** nos ha permitido estructurar una API REST limpia, segura y robusta, implementando funcionalidades como autenticación con sesiones, control de acceso por roles, persistencia con JPA, y controladores especializados para cada tipo de recurso (usuarios, contenido, reviews, listas de deseos...).

Además, se ha integrado una lógica de negocio coherente mediante **servicios desacoplados**, reforzada con pruebas **unitarias, de integración** y **end-to-end**, que garantizan el correcto funcionamiento de las funcionalidades críticas. Esta base sólida ha facilitado iterar y probar con confianza, validando todo el flujo de usuario, desde el registro hasta la interacción con los contenidos.

En el **frontend**, se ha optado por una solución ligera en **HTML, CSS y JavaScript**, con páginas funcionales y fáciles de entender para el usuario. Las pantallas de login, registro, catálogo y detalle de contenido permiten una navegación sencilla y fluida, cubriendo los principales casos de uso. La personalización visual y la organización del código han buscado mantener la claridad, sin sobrecargar el desarrollo con frameworks pesados.

Uno de los principales **retos** ha sido garantizar la comunicación fluida entre cliente y servidor, sobre todo en lo referente a la gestión de cookies de sesión y el manejo de errores HTTP. También hemos trabajado de forma continua en la mejora de la experiencia de usuario, añadiendo validaciones, mensajes informativos y control de estados.

En cuanto al trabajo en equipo, la división de responsabilidades entre desarrollo de frontend, backend, testing y documentación nos ha permitido avanzar de forma ágil, aprendiendo a coordinarnos, versionar el proyecto y revisar el trabajo de forma colaborativa.

En resumen, el proyecto no solo cumple los objetivos funcionales establecidos, sino que también refleja el aprendizaje de metodologías de desarrollo modernas, buenas prácticas de codificación y herramientas del ecosistema Java. Es una base excelente para seguir creciendo como desarrolladores y ampliar el sistema con nuevas funcionalidades en el futuro.