PROYECTO EXPRESS

SAYARA APARICIO
SANTIAGO AGUILAR

S1

PEDRO FELIPE GÓMEZ BONILLA

CAMPUSLANDS
CAJASAN
RUTA NODEJS
BUCARAMANGA
2025

APLICACIÓN KARENFLIX

1. SITUACIÓN PROBLEMA

En la actualidad, los fanáticos de películas, animes y series geek no cuentan con una herramienta sencilla y centralizada que les permita **registrar, calificar y rankear contenido** de manera colaborativa.

Existen múltiples plataformas de streaming o bases de datos (IMDb, MyAnimeList, Rotten Tomatoes), pero suelen estar saturadas, no siempre enfocadas en la cultura geek, o presentan limitaciones para **gestionar reseñas, categorías personalizadas y rankings ponderados** de manera transparente.

Esto genera problemas como:

- Dificultad para encontrar recomendaciones confiables de la comunidad geek.
- Ausencia de un sistema de calificaciones justo que combine ratings, likes/dislikes y frescura de reseñas.
- Falta de diferenciación entre usuarios y administradores, lo que complica la gestión de categorías y validación de contenido.
- Dispersión de herramientas: foros para discutir, apps para rankear y páginas para reseñar, en lugar de tener todo en un solo sistema.

2. LEVANTAMIENTO DE REQUERIMIENTOS

Para el desarrollo del proyecto KarenFlix, se realizó un levantamiento de requerimientos enfocado en identificar las necesidades principales de los usuarios al momento de registrar, calificar y rankear películas, animes y series geek, así como la correcta gestión de reseñas, categorías y rankings en una plataforma unificada.

El proceso incluyó:

- Análisis del problema

Se revisaron las dificultades más comunes de los usuarios aficionados a la cultura geek en la gestión de contenido audiovisual:

- Ausencia de una herramienta centralizada para registrar, calificar y rankear.
- Problemas de confiabilidad en los sistemas de calificación tradicionales.
- Falta de diferenciación de roles (usuarios y administradores) para validar contenido y categorías.
- Dificultad para filtrar películas por popularidad, ranking o categoría.

Estas dificultades fueron descritas previamente en la situación problema.

- Definición de funcionalidades clave

Se establecieron las operaciones mínimas necesarias (crear, actualizar, eliminar, consultar) para cada entidad del sistema:

- Usuarios: registro, inicio de sesión, asignación de roles, autenticación JWT.
- Películas/Series: CRUD con aprobación de administradores y validación de duplicados.
- Categorías: CRUD gestionado solo por administradores.
- Reseñas y ratings: creación, edición, eliminación, likes/dislikes y cálculo de ranking ponderado.
- Listados: visualización de películas ordenadas por popularidad o ranking, con filtrado por categoría.

- Priorización

Se diferenciaron dos tipos de requerimientos:

- Requerimientos Funcionales (RF): definen lo que el sistema debe hacer (ej. CRUD de usuarios, películas, reseñas, rankings).

- Requerimientos No Funcionales (RNF): definen cómo debe hacerlo en términos de seguridad, arquitectura, rendimiento, escalabilidad y usabilidad (ej. uso de Node.js + Express, persistencia en MongoDB, transacciones, validaciones, autenticación segura, frontend responsive).

- Referentes

Se tomaron como base buenas prácticas de desarrollo de software web, aplicando:

- Arquitectura modular en el backend con Node.js + Express.

- Principios SOLID para garantizar mantenibilidad y escalabilidad.

- Uso de MongoDB con operaciones transaccionales para consistencia de datos.

- Implementación de patrones de diseño en la organización del código.

- Documentación de endpoints con Swagger y versionamiento semver.

3. REQUERIMIENTOS

3.1. Requerimientos Funcionales

- **RF1:** El sistema debe permitir registro e inicio de sesión de usuarios con autenticación JWT.
- **RF2:** El sistema debe diferenciar roles (usuario y administrador).
- **RF3:** Los administradores podrán aprobar y gestionar películas/series.
- **RF4:** CRUD de películas/series con atributos: título, descripción, categoría, año e imagen opcional.
- **RF5:** Validación para evitar títulos repetidos.
- **RF6:** CRUD de categorías (ejemplo: Anime, Ciencia Ficción, Superhéroes, Fantasía).
- **RF7:** Los usuarios podrán crear, editar y eliminar reseñas.
- **RF8:** Cada reseña debe incluir título, comentario y calificación (escala de 1.0 a 5.0 con incrementos de 0.1).
- **RF9:** Los usuarios podrán dar like/dislike a reseñas de otros (no a las propias).
- **RF10:** El sistema debe calcular un ranking ponderado de películas combinando calificación, likes/dislikes y fecha de reseña.
- **RF11:** Listado de películas con ordenamiento por popularidad o ranking.
- **RF12:** Filtrado de películas por categoría.
- **RF13:** Vista de detalle de la película con información y reseñas asociadas.
- **RF14:** Frontend debe permitir registro/login, listado, detalle de películas y panel admin.
- **RF15:** El sistema debe mostrar mensajes de validación o error desde el backend hacia el frontend.

3.2. Requerimientos No Funcionales

- **RNF1:** El backend debe desarrollarse en Node.js con Express bajo una arquitectura modular (/models, /controllers, /routes, /middlewares, etc.).
- **RNF2:** La persistencia debe realizarse en MongoDB usando el driver oficial (no Mongoose).
- **RNF3:** Deben implementarse transacciones en operaciones críticas (ejemplo: creación de reseñas con rating inicial, likes/dislikes).
- **RNF4:** La aplicación debe aplicar validaciones en endpoints usando `express-validator`.
- **RNF5:** Debe garantizar autenticación y seguridad usando JWT, bcrypt y passport-jwt.
- **RNF6:** Las contraseñas deben almacenarse cifradas.
- **RNF7:** Configuración de variables de entorno con dotenv.
- **RNF8:** El sistema debe limitar peticiones con `express-rate-limit` para prevenir abusos.
- **RNF9:** Documentar todos los endpoints con Swagger.
- **RNF10:** Versionado del API siguiendo **semver**.
- **RNF11:** El frontend debe desarrollarse con HTML, CSS y JS puro, en un repositorio independiente.
- **RNF12:** La interfaz debe ser responsive y amigable.
- **RNF13:** El sistema debe manejar errores centralizados y devolver códigos HTTP correctos.
- **RNF14:** La documentación debe incluir README con descripción, instalación, variables de entorno y ejemplos de uso.
- **RNF15:** La entrega debe incluir un video demo (máx 10 minutos) mostrando backend y frontend funcionando

4. HISTORIAS DE USUARIO CON CRITERIOS DE ACEPTACIÓN

1.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF01	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	Registro de usuarios		
Descripción			
Como usuario quiero registrarme en la plataforma para poder acceder a todas las funcionalidades de KarenFlix.			
Funcionalidad			
El sistema permitirá registrar usuarios guardando sus datos en la base de datos			
Criterios de aceptación	1. El sistema debe permitir registrar un usuario con nombre, correo, contraseña y rol. 2. El sistema debe validar que el correo no esté repetido. 3. El sistema debe guardar la contraseña cifrada con bcrypt.		
Restricciones			
El correo y la contraseña son obligatorios.			

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF02	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	Inicio de sesión		
Descripción			
Como usuario quiero iniciar sesión para acceder a mi cuenta.			
Funcionalidad			
El sistema permitirá autenticar usuarios con JWT.			
Criterios de aceptación	• El sistema debe generar un token JWT al iniciar sesión correctamente. • El sistema debe validar credenciales incorrectas. • El token tendrá tiempo de expiración configurable.		
Restricciones			
El usuario debe estar registrado previamente.			

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF03	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	Gestión de roles		
Descripción			
Como administrador quiero gestionar roles para diferenciar accesos entre usuarios normales y administradores			
Funcionalidad			
El sistema permitirá asignar y validar roles.			
Criterios de aceptación	Solo administradores pueden aprobar películas y gestionar categorías. Los usuarios normales solo pueden crear reseñas.		
Restricciones			
El rol se asigna en el registro y no se puede modificar por el usuario.			

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF04	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	Crear películas		
Descripción			
Como administrador quiero registrar nuevas películas/series para que los usuarios puedan reseñarlas			
Funcionalidad			
El sistema permitirá crear películas/series con atributos básicos			
Criterios de aceptación	El sistema debe registrar título, descripción, categoría, año e imagen opcional. No se permitirá registrar títulos duplicados.		
Restricciones			
Solo los administradores pueden crear películas			

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF05	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	Eliminar películas		
Descripción			
Como administrador quiero eliminar películas/series que no sean válidas.			
Funcionalidad			
El sistema permitirá borrar películas existentes.			
Criterios de aceptación	El sistema debe verificar que la película existe antes de eliminarla. El sistema debe eliminar también las reseñas asociadas.		
Restricciones			
Solo administradores pueden eliminar			

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF06	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	Crear reseñas		
Descripción			
Como usuario quiero crear reseñas para compartir mi opinión sobre películas.			
Funcionalidad			
El sistema permitirá a los usuarios escribir reseñas y asignar calificaciones.			
Criterios de aceptación	La reseña debe incluir título, comentario y calificación (1.0 a 5.0 en incrementos de 0.1). El sistema debe guardar la reseña en la base de datos.		
Restricciones			
El usuario debe estar autenticado.			

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF07	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	Editar películas		
Descripción			
Como administrador quiero editar los datos de películas para mantener la información actualizada.			
Funcionalidad			
El sistema permitirá modificar películas ya registradas..			
Criterios de aceptación	El sistema debe validar que la película exista antes de editarla. El sistema debe permitir cambiar título, descripción, año y categoría.		
Restricciones			
• Solo administradores pueden modificar películas.			

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF08	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	Editar reseñas		
Descripción			
Como usuario quiero editar mis reseñas para corregir o actualizar información.			
Funcionalidad			
El sistema permitirá modificar reseñas creadas por el mismo usuario.			
Criterios de aceptación	Solo el autor puede editar su reseña. El sistema debe registrar la fecha de última modificación.		
Restricciones			
El usuario debe estar autenticado.			

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF09	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	Eliminar reseñas		
Descripción			
Como usuario quiero eliminar mis reseñas cuando ya no quiera que aparezcan.			
Funcionalidad			
El sistema permitirá eliminar reseñas propias.			
Criterios de aceptación	Solo el autor puede eliminar su reseña. El sistema debe actualizar el ranking de la película.		
Restricciones			
El usuario debe estar autenticado.			

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF10	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	Likes y dislikes en reseñas		
Descripción			
Como usuario quiero dar like o dislike a reseñas para valorar la utilidad de las opiniones.			
Funcionalidad			
El sistema permitirá registrar likes y dislikes en reseñas.			
Criterios de aceptación	El sistema no permitirá que el usuario vote en sus propias reseñas. El sistema debe permitir un solo voto por usuario en cada reseña.		
Restricciones			
El usuario debe estar autenticado.			

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF11	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	Crear categorías		
Descripción			
Como administrador quiero crear categorías para organizar las películas y series.			
Funcionalidad			
El sistema permitirá registrar categorías en la base de datos.			
Criterios de aceptación	El sistema debe permitir registrar categorías con nombre único. El sistema debe validar que no existan categorías duplicadas.		
Restricciones			
Solo administradores pueden crear categorías.			

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF12	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	Editar categorías		
Descripción			
Como administrador quiero editar categorías para mantener actualizada la clasificación.			
Funcionalidad			
El sistema permitirá modificar nombres de categorías existentes.			
Criterios de aceptación	El sistema debe validar que la categoría exista antes de editarla. No se permitirán nombres duplicados.		
Restricciones			
Solo administradores pueden editar categorías.			

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF14	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	Eliminar categorías		
Descripción			
Como administrador quiero eliminar categorías que ya no sean necesarias.			
Funcionalidad			
El sistema permitirá borrar categorías existentes.			
Criterios de aceptación	El sistema debe verificar que la categoría exista antes de eliminarla. El sistema debe reasignar o impedir eliminar si hay películas asociadas.		
Restricciones			
Solo administradores pueden eliminar.			

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF15	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	Listado de películas		
Descripción			
Como usuario quiero ver un listado de películas disponibles para elegir cuál reseñar o consultar.			
Funcionalidad			
El sistema mostrará una lista de películas ordenadas por popularidad o ranking.			
Criterios de aceptación	El sistema debe permitir ordenar por ranking o fecha. El sistema debe permitir filtrar por categoría.		
Restricciones			
El usuario debe estar autenticado.			

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF16	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	Detalle de película		
Descripción			
Como usuario quiero ver el detalle de una película para conocer su información y reseñas.			
Funcionalidad			
El sistema permitirá consultar la información completa de una película y sus reseñas asociadas			
Criterios de aceptación	El sistema debe mostrar título, descripción, categoría, año e imagen. El sistema debe mostrar todas las reseñas asociadas con sus likes/dislikes.		
Restricciones			
El usuario debe estar autenticado.			

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF17	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	Ranking ponderado		
Descripción			
Como usuario quiero que las películas tengan un ranking ponderado justo basado en calificaciones, likes y fecha de reseña.			
Funcionalidad			
El sistema calculará el ranking dinámicamente.			
Criterios de aceptación	El sistema debe actualizar el ranking cada vez que se agrega, edita o elimina una reseña. El cálculo debe considerar calificación, likes/dislikes y antigüedad de la reseña.		
Restricciones			
Algoritmo definido en el backend.			

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF18	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	Validación de datos		
Descripción			
Como sistema quiero validar entradas de datos para evitar información incorrecta o insegura.			
Funcionalidad			
El sistema usará `express-validator` en todos los endpoints.			
Criterios de aceptación	• El sistema debe devolver mensajes de error claros al usuario. • El sistema debe rechazar entradas vacías o con formato inválido.		
Restricciones			
• Validaciones en backend obligatorias.			

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF19	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	Limitación de peticiones		
Descripción			
Como sistema quiero limitar las peticiones para evitar abusos.			
Funcionalidad			
El sistema usará `express-rate-limit`.			
Criterios de aceptación	• El sistema debe devolver un error si se excede el número de peticiones permitido. • Los límites deben ser configurables.		
Restricciones			
• Configuración definida en `.env`.			

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF20	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	Frontend responsive		
Descripción			
Como usuario quiero una interfaz responsive para poder usar la aplicación desde cualquier dispositivo.			
Funcionalidad			
El frontend en HTML, CSS y JS será adaptable a escritorio y móvil.			
Criterios de aceptación	Las pantallas deben ajustarse a diferentes resoluciones. No debe perderse ninguna funcionalidad en móviles.		
Restricciones			
● Desarrollo en HTML, CSS y JS puro.			

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF21	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	Registro y login desde frontend		
Descripción			
Como usuario quiero registrarme e iniciar sesión desde el frontend para acceder a la aplicación.			
Funcionalidad			
El frontend permitirá al usuario consumir los endpoints de autenticación.			
Criterios de aceptación	El sistema debe mostrar mensajes de validación o error desde el backend. El token JWT debe guardarse en el navegador para futuras peticiones.		
Restricciones			
• Solo HTML, CSS y JS.			

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF22	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	Panel de administración		
Descripción			
Como administrador quiero tener un panel para gestionar películas y categorías fácilmente.			
Funcionalidad			
El frontend permitirá a los administradores acceder a CRUD de películas y categorías.			
Criterios de aceptación	1. El panel debe mostrar todas las categorías y películas registradas. 2. El panel debe permitir agregar, editar y eliminar.		
Restricciones			
● Solo accesible con rol administrador.			

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF23	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	Listado de reseñas propias		
Descripción			
Como usuario quiero ver una lista de mis reseñas para poder gestionarlas.			
Funcionalidad			
El sistema mostrará reseñas creadas por el usuario autenticado.			
Criterios de aceptación	El sistema debe listar reseñas con fecha de creación. El usuario podrá editar o eliminar desde la lista.		
Restricciones			
• Usuario autenticado.			

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF24	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	Documentación de la API		
Descripción			
Como desarrollador quiero contar con documentación de la API para usar los endpoints fácilmente.			
Funcionalidad			
El sistema expondrá la documentación con `swagger-ui-express`.			
Criterios de aceptación	1. Todos los endpoints deben estar documentados. 2. La documentación debe estar accesible desde un endpoint `/api-docs`.		
Restricciones			
• Debe mantenerse actualizada.			

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF25	**Actor**	Equipo
NOMBRE DEL REQUERIMIENTO	Planeación SCRUM		
Descripción			
Como equipo quiero definir roles y sprints para organizar el trabajo del proyecto.			
Funcionalidad			
El equipo documentará la planeación en un PDF y lo adjuntará al repositorio.			
Criterios de aceptación	1. El documento debe incluir roles, sprints e historias de usuario. 2. El documento debe estar en el repositorio backend.		
Restricciones			
● Uso de plantilla proporcionada.			

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF26	**Actor**	Equipo
NOMBRE DEL REQUERIMIENTO	Video de entrega		
Descripción			
Como equipo quiero grabar un video demo para mostrar el funcionamiento de la aplicación.			
Funcionalidad			
El equipo grabará un video de máximo 10 minutos mostrando código y demo del sistema.			
Criterios de aceptación	1. El video debe mostrar a todos los integrantes. 2. Debe explicar el backend y frontend. 3. El enlace al video debe estar en el README.		
Restricciones			
● Duración máxima de 10 minutos.			

5. METODOLOGÍA

El desarrollo del proyecto KarenFlix se lleva a cabo siguiendo la metodología ágil SCRUM, la cual permitió organizar las tareas en sprints definidos con duración de 1 semana cada uno.

SCRUM se seleccionó por su enfoque iterativo e incremental, lo que permitió priorizar los requerimientos esenciales, realizar revisiones periódicas y asegurar un producto funcional en los tiempos establecidos.

- Roles:

- Product Owner: Encargado de definir y priorizar los requerimientos de la plataforma (usuarios, películas, reseñas, rankings).

- Scrum Master: Facilitador de la metodología y responsable del seguimiento al cumplimiento de los sprints.

- Development Team: Equipo de desarrollo encargado de implementar el backend (Node.js + Express + MongoDB) y el frontend (HTML, CSS y JS puro).

- Actividades:

- Sprint Planning: Planificación de tareas, definición de prioridades y asignación de responsables.

- Daily Stand Up: Reuniones breves de seguimiento del avance, identificación de bloqueos y próximos pasos.

- Sprint Review: Presentación de avances, validación de historias de usuario y demostración funcional del sistema.

- Sprint Retrospective: Reflexión sobre el sprint, análisis de mejoras y aprendizajes para iteraciones siguientes.

---

6. Evidencias

- **Repositorio GitHub:** Proyecto KarenFlix dividido en backend y frontend en repositorios separados. BackEnd FrontEnd

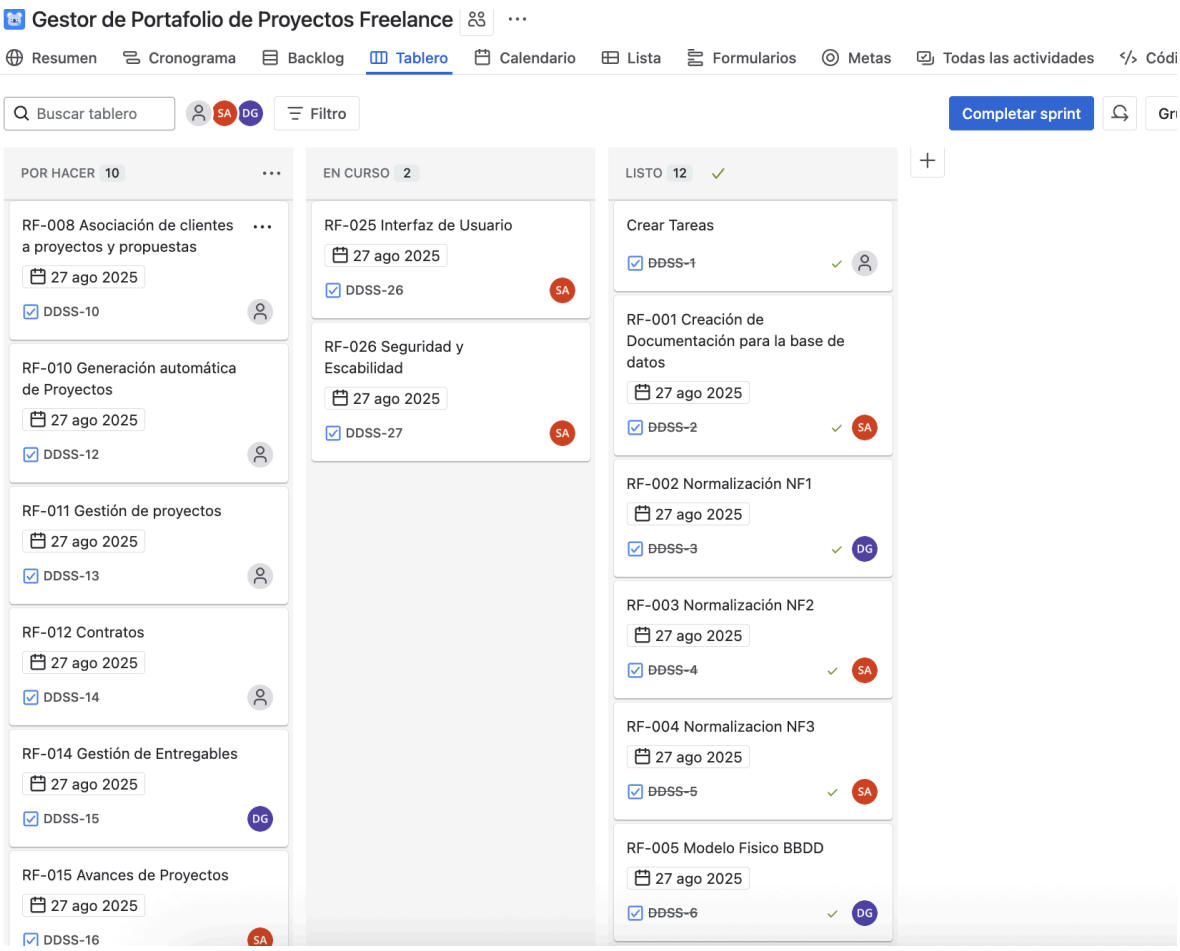
- Uso de conventional commits: Para control de versiones y trazabilidad de cambios.

- Tablero Scrum en Jira: Con las historias de usuario, tareas técnicas y responsables.

- Video grabado

7. Resultados:

- **Plataforma funcional que permite:**
    - Registro e inicio de sesión de usuarios con autenticación JWT.
    - Gestión de películas, categorías y reseñas.
    - Sistema de calificaciones con ranking ponderado.
    - Interfaz frontend responsive que consume el backend.
- **Cumplimiento de los requerimientos planteados.**
- **Uso de MongoDB con transacciones, principios SOLID y una arquitectura modular en el backend.**

- 

7. CONCLUSIONES

- El proyecto permitió desarrollar una solución integral que cubre las necesidades principales de la comunidad geek: gestión de usuarios, películas, reseñas, categorías y rankings en una sola plataforma.

- El uso de MongoDB con transacciones aportó robustez y seguridad en operaciones críticas como reseñas y likes/dislikes.

- La aplicación de principios SOLID y patrones de diseño garantizó un software modular, escalable y mantenible.

- La metodología SCRUM y el uso de Jira mejoraron la organización del equipo, asegurando la entrega en el tiempo definido.

- En la retrospectiva, se identificó como área de mejora la estimación de tiempos de desarrollo y la necesidad de incrementar la automatización en pruebas.