Movie Website KarenFlix

SIMÓN DANTE SALAMANCA GALVIS

JUAN DAVID SAAVEDRA JAIMEZ

S1

PEDRO FELIPE GÓMEZ BONILLA

CAMPUSLANDS
RUTA NODE.JS
BUCARAMANGA
2025

Movie Website KarenFlix

1. SITUACIÓN PROBLEMA

El consumo de contenidos audiovisuales de cultura geek o cineastas ha crecido de forma exponencial, impulsado principalmente por las audiencias de la última década. Esta expansión ha creado una demanda insatisfecha de plataformas especializadas que cubren las necesidades particulares de comunidades de nicho.

Los consumidores de este tipo de contenido se enfrentan a un problema central: la información fiable está muy dispersa. Para encontrar reseñas expertas sobre cine, series o anime, los usuarios se ven obligados a consultar múltiples fuentes. Estos incluyen la fidelidad al material original, la calidad de la animación, que las plataformas mainstream usualmente no priorizan.

Además de la carencia de sistemas de valoración adaptados. Las plataformas existentes usan métodos de puntuación genéricos que no captan los matices de este contenido. Esta limitación también se encuentra en la falta de comunidades de curadores especializados que elaboran rankings basados en criterios propios de la comunidad.

Frente a estos desafíos, a los integrantes de este proyecto surge la oportunidad de crear KarenFlix, una plataforma especializada que permita a la comunidad cineasta registrar, calificar y clasificar películas, series y anime de manera colaborativa. La solución integraría un sistema de calificación, rankings ponderados, herramientas administrativas avanzadas y mecanismos de interacción social para fomentar la participación y el conocimiento colectivo.

2. LEVANTAMIENTO DE REQUERIMIENTOS

Se realizó una reunión inicial con el cliente para ver de manera clara las diferentes funcionalidades que debe tener la aplicación, además de las tecnologías a implementar. A partir de esta primera reunión se lograron identificar unos requerimientos principales y primordiales para el funcionamiento del programa.

Tiempo después se realizó una segunda reunión con el cliente para confirmar los diferentes requerimientos del programa, a partir de esta reunión se logró la clarificación de ciertas dudas que surgieron y requerimientos extras que son importantes para la funcionalidad del proyecto.

3. REQUERIMIENTOS

Gestión de usuarios

- Registro, inicio de sesión y autenticación mediante JWT.
- Roles: usuario y administrador.
- Los administradores pueden gestionar categorías y aprobar películas.

Gestión de películas y series

- CRUD de películas/series (solo administradores aprueban nuevas entradas).
- Validación para evitar títulos repetidos.
- Atributos mínimos: título, descripción, categoría, año, imagen opcional.

Gestión de reseñas y ratings

- Los usuarios pueden crear, editar y eliminar reseñas.
- Cada reseña incluye: título, comentario, calificación numérica (1-5 con decimales).
- Los usuarios pueden dar like/dislike a reseñas de otros (no a las propias).
- El sistema debe calcular un ranking ponderado de películas basado en calificaciones, likes/dislikes y fecha de reseña.

Categorías

- CRUD de categorías (ejemplo: Anime, Ciencia Ficción, Superhéroes, Fantasía) (Min 4).
- Solo administradores pueden gestionarlas.

Ranking y listados

- Listado de películas con ordenamiento por popularidad y ranking.
- Filtrado por categoría.
- Vista de detalle con información y reseñas asociadas.

Seguridad y autenticación

- Implementación de JWT con passport-jwt y bcrypt.
- Rate limiting para prevenir abusos.
- Validaciones robustas con express-validator.

Gestión de base de datos

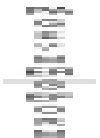
- MongoDB con transacciones para operaciones críticas.
- Consistencia en operaciones de reseñas con likes/dislikes.
- Validación de duplicados y integridad referencial.

3.1. Requerimientos Funcionales

RF-01. El sistema permitirá el registro de usuarios con roles diferenciados de usuario y administrador.

RF-02. El sistema permitirá el inicio de sesión mediante autenticación con JWT usando correo electrónico y contraseña.

RF-03. El sistema permitirá a los usuarios actualizar su información de perfil.



RF-04. El sistema permitirá a los administradores gestionar las categorías de contenido disponibles en la plataforma.

RF-05. El sistema permitirá la creación, lectura, actualización y eliminación de películas, series y animes por parte de los administradores.

RF-06. El sistema implementará un proceso de aprobación para nuevo contenido agregado por usuarios antes de su publicación.

RF-07. El sistema validará que no existan títulos duplicados en la base de datos.

RF-08. El sistema permitirá a los usuarios crear reseñas con título, comentario y calificación numérica del 1 al 5 con decimales.

RF-09. El sistema permitirá a los usuarios editar y eliminar sus propias reseñas.

RF-10. El sistema permitirá a los usuarios dar like o dislike a reseñas de otros usuarios, pero no a las propias.

RF-11. El sistema calculará automáticamente rankings ponderados de contenido basado en calificaciones, likes/dislikes y fecha de las reseñas.

RF-12. El sistema proporcionará listados de contenido ordenados por popularidad y ranking.

RF-13. El sistema permitirá filtrar contenido por categorías específicas.

RF-14. El sistema mostrará vistas detalladas de cada película, serie o anime con toda su información y reseñas asociadas.

RF-15. El sistema permitirá búsquedas de contenido por título y otros criterios relevantes.

RF-16. El sistema registrará y mostrará estadísticas de likes y dislikes por reseña.

RF-17. El sistema mantendrá un historial de cambios en las reseñas para auditoría.

RF-18. El sistema implementará paginación en los listados de contenido y reseñas.

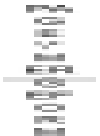
RF-19. El sistema permitirá a los administradores moderar y eliminar reseñas inapropiadas.

RF-20. El sistema generará reportes de actividad y estadísticas de uso para administradores.

3.2. Requerimientos No Funcionales

RNF-01. El sistema utilizará bcrypt para el hasheo seguro de contraseñas antes de almacenarlas en la base de datos.

RNF-02. El sistema implementará autenticación mediante JWT con tokens que expiren automáticamente.



RNF-03. El sistema utilizará express-rate-limit para prevenir ataques de fuerza bruta y limitar peticiones por IP.

RNF-04. El sistema implementará validaciones robustas en todos los endpoints usando express-validator.

RNF-05. El sistema utilizará MongoDB como base de datos principal con el driver oficial de Node.js.

RNF-06. El sistema implementará transacciones MongoDB para operaciones críticas como creación de reseñas y gestión de likes.

RNF-07. El sistema mantendrá consistencia de datos en operaciones concurrentes mediante el uso de transacciones atómicas.

RNF-08. El sistema seguirá una arquitectura modular separando responsabilidades en carpetas como models, controllers, routes, middlewares, services, config y utils.

RNF-09. El sistema documentará todos los endpoints mediante swagger-ui-express con especificaciones OpenAPI.

RNF-10. El sistema versionará la API siguiendo las convenciones de Semantic Versioning.

RNF-11. El sistema configurará CORS adecuadamente para permitir comunicación segura con el frontend.

RNF-12. El sistema utilizará variables de entorno mediante dotenv para toda la configuración sensible.

RNF-13. El sistema implementará manejo centralizado de errores con códigos HTTP apropiados.

RNF-14. El sistema será responsive y funcionará correctamente en dispositivos móviles y desktop.

RNF-15. El sistema mantendrá tiempos de respuesta inferiores a 2 segundos para operaciones de consulta.

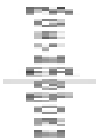
RNF-16. El sistema soportará al menos 100 usuarios concurrentes sin degradación significativa del rendimiento.

RNF-17. El sistema mantendrá disponibilidad del 99% durante horas de operación normal.

RNF-18. El sistema implementará logging completo de errores y actividades críticas para facilitar debugging y auditoría.

RNF-19. El sistema validará todos los datos de entrada contra esquemas predefinidos para prevenir inyecciones.

RNF-20. El sistema implementará índices apropiados en MongoDB para optimizar consultas frecuentes.



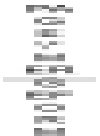
4. HISTORIAS DE USUARIO CON CRITERIOS DE ACEPTACIÓN

1.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF01	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	**Registro de usuarios con roles diferenciados**		
Descripción			
Como usuario puedo registrarme en KarenFlix seleccionando mi rol para acceder a las funcionalidades correspondientes.			
Funcionalidad			
Registro de usuarios con asignación de roles y permisos diferenciados.			
Criterios de aceptación	1. El sistema debe permitir seleccionar el rol durante el registro. 2. La contraseña debe ser hasheada antes del almacenamiento. 3. El correo electrónico debe ser único en el sistema.		
Restricciones			
La contraseña debe tener mínimo 8 caracteres y no se permite registrar el mismo correo dos veces.			

2.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF02	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	**Inicio de sesión con JWT**		
Descripción			
Como usuario puedo iniciar sesión con correo y contraseña para acceder a mi cuenta.			
Funcionalidad			
Autenticación segura mediante JWT con validación de credenciales.			
Criterios de aceptación	1. El sistema debe validar credenciales contra la base de datos. 2. Se debe generar un token JWT tras autenticación exitosa. 3. Se debe mostrar error para credenciales incorrectas.		



Restricciones
Máximo 3 intentos fallidos antes de bloqueo temporal.

3.

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF03	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	**Actualización de perfil**		
Descripción			
Como usuario puedo actualizar mi información de perfil para mantener mis datos actualizados.			
Funcionalidad			
Modificación de datos del perfil manteniendo integridad de información.			
Criterios de aceptación	1. El usuario debe poder editar nombre y descripción. 2. El correo debe mantenerse único tras actualización. 3. Los cambios deben persistir en la base de datos.		
Restricciones			
No se puede cambiar a un correo ya existente.			

4.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF04	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	**Gestión de categorías**		
Descripción			
Como administrador puedo crear, editar y eliminar categorías para organizar el catálogo.			
Funcionalidad			

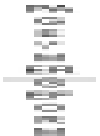
Herramientas administrativas para gestión de categorías de contenido.	
Criterios de aceptación	1. Debe existir mínimo 4 categorías predefinidas. 2. Solo administradores pueden gestionar categorías. 3. Los nombres de categorías deben ser únicos.
Restricciones	
Las categorías con contenido asociado no pueden eliminarse.	

5.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF05	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	**CRUD de películas y series**		
Descripción			
Como administrador puedo gestionar el catálogo de películas, series y animes.			
Funcionalidad			
Gestión completa del catálogo con validaciones de integridad.			
Criterios de aceptación	1. Todos los campos obligatorios deben completarse. 2. La categoría debe existir en el sistema. 3. El año debe ser un valor numérico válido.		
Restricciones			
Solo administradores pueden realizar estas operaciones.			

6.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF06	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Aprobación de contenido**		
Descripción			



Como sistema debo implementar aprobación administrativa antes de publicar contenido nuevo.
Funcionalidad
Control de calidad mediante aprobación de contenido.

Criterios de aceptación	1. El contenido debe permanecer "pendiente" hasta aprobación. 2. Solo administradores pueden aprobar o rechazar. 3. Se debe notificar el estado al usuario.

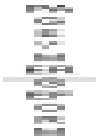
Restricciones
El contenido no aprobado no aparece en búsquedas públicas.

7.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF07	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Validación de títulos únicos**		
Descripción			
Como sistema debo validar que no existan títulos duplicados para mantener integridad del catálogo.			
Funcionalidad			
Validación automática de unicidad de títulos.			
Criterios de aceptación	1. La validación debe ser case-insensitive. 2. Se debe considerar año para títulos similares. 3. La validación debe realizarse en tiempo real.		
Restricciones			
No se permiten títulos idénticos independientemente del tipo de contenido.			

8.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF08	**Actor**	Usuario



NOMBRE DEL REQUERIMIENTO	**Creación de reseñas**
Descripción	
Como usuario puedo crear reseñas con título, comentario y calificación del 1 al 5.	
Funcionalidad	
Sistema de reseñas con calificación granular y contenido textual.	
Criterios de aceptación	1. La calificación debe estar entre 1.0 y 5.0 con decimales. 2. El título es obligatorio con máximo 100 caracteres. 3. Solo una reseña por usuario por contenido.
Restricciones	
Los usuarios deben estar autenticados para crear reseñas.	

9.

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF09	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	**Edición de reseñas propias**		
Descripción			
Como usuario puedo editar o eliminar mis reseñas para corregir o actualizar información.			
Funcionalidad			
Gestión de reseñas propias con historial de cambios.			
Criterios de aceptación	1. Solo el autor puede editar sus reseñas. 2. Se debe mantener historial de cambios. 3. La eliminación requiere confirmación.		
Restricciones			
No se pueden editar reseñas después de 24 horas si tienen más de 10 likes.			

10.

HISTORIA DE USUARIO

Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF01	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	**Sistema de likes y dislikes**		
Descripción			
Como usuario puedo dar like o dislike a reseñas de otros usuarios pero no a las mías.			
Funcionalidad			
Sistema de votación social para reseñas.			
Criterios de aceptación	1. Solo un voto por reseña por usuario. 2. No se permite votar reseñas propias. 3. Se puede cambiar el voto de like a dislike.		
Restricciones			
Los usuarios deben estar autenticados para votar.			

11.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF011	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Cálculo de rankings ponderados**		
Descripción			
Como sistema debo calcular rankings basados en calificaciones, likes y fechas de reseñas.			
Funcionalidad			
Algoritmo de ranking considerando múltiples factores.			
Criterios de aceptación	1. El algoritmo debe ponderar calificación, volumen y fecha. 2. Los likes/dislikes deben influir en el peso de reseñas. 3. El ranking debe actualizarse automáticamente.		
Restricciones			

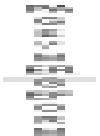
El cálculo debe ejecutarse máximo cada 30 minutos.

12.

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF012	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	**Listados ordenados**		
Descripción			
Como usuario puedo ver listados ordenados por popularidad y ranking.			
Funcionalidad			
Múltiples vistas de contenido con diferentes ordenamientos.			
Criterios de aceptación	1. Debe ofrecer ordenamiento por ranking y popularidad. 2. Los listados deben implementar paginación. 3. Se debe mostrar información resumida.		
Restricciones			
Máximo 20 elementos por página.			

13.

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF013	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	**Registro de usuarios con roles diferenciados**		
Descripción			
Como usuario puedo registrarme en KarenFlix seleccionando mi rol para acceder a las funcionalidades correspondientes.			
Funcionalidad			
Sistema de filtros por categorías con interfaz intuitiva.			



Criterios de aceptación	1. Se debe poder seleccionar múltiples categorías. 2. Los filtros deben aplicarse instantáneamente. 3. Se debe mostrar número de resultados por categoría.
Restricciones	
Máximo 5 categorías seleccionables simultáneamente.	

14.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF014	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	**Vista detallada de contenido**		
Descripción			
Como usuario puedo acceder a vista detallada con toda la información y reseñas.			
Funcionalidad			
Página de detalles completa del contenido y reseñas.			
Criterios de aceptación	1. Debe mostrar toda la información del contenido. 2. Las reseñas deben ordenarse por relevancia. 3. Se debe mostrar calificación promedio.		
Restricciones			
Solo contenido aprobado tiene página de detalles pública.			

15.

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF015	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	**Búsqueda de contenido**		
Descripción			
Como usuario puedo buscar películas, series y animes por título y otros criterios para encontrar			

contenido específico.
Funcionalidad
Motor de búsqueda con múltiples criterios y filtros avanzados.

Criterios de aceptación	1. Debe permitir búsqueda por título con autocompletado. 2. Los resultados deben mostrarse ordenados por relevancia. 3. Se debe incluir filtros por categoría, año y calificación.

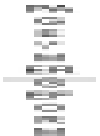
Restricciones
Solo se muestran resultados de contenido aprobado.

16.

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RF016	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	**Estadísticas de reseñas**		
Descripción			
Como usuario puedo ver estadísticas de likes y dislikes por reseña para evaluar su utilidad.			
Funcionalidad			
Visualización de métricas sociales de reseñas.			
Criterios de aceptación	1. Se debe mostrar contador de likes y dislikes. 2. Las estadísticas deben actualizarse en tiempo real. 3. Se debe mostrar ratio de utilidad de la reseña.		
Restricciones			
Solo usuarios autenticados pueden ver estadísticas detalladas.			

17.

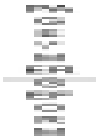
HISTORIA DE USUARIO
Prioridad: Media



CÓDIGO DEL REQUERIMIENTO:	RF017	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Historial de cambios en reseñas**		
Descripción			
Como sistema debo mantener historial de cambios en reseñas para auditoría y trazabilidad.			
Funcionalidad			
Sistema de versionado de reseñas con registro de cambios.			
Criterios de aceptación	1. Se debe registrar fecha y usuario de cada cambio. 2. Se debe mantener versión anterior de la reseña. 3. Los administradores pueden consultar el historial.		
Restricciones			
El historial solo es visible para administradores y autor de la reseña.			

18.

HISTORIA DE USUARIO			
Prioridad: Baja			
CÓDIGO DEL REQUERIMIENTO:	RF018	**Actor**	Usuario
NOMBRE DEL REQUERIMIENTO	**Paginación en listados**		
Descripción			
Como usuario puedo navegar por listados largos usando paginación para mejorar rendimiento.			
Funcionalidad			
Sistema de paginación eficiente con navegación intuitiva.			
Criterios de aceptación	1. Se debe mostrar máximo 20 elementos por página. 2. Se debe incluir navegación numérica y siguiente/anterior. 3. Se debe mostrar total de resultados.		
Restricciones			
La paginación debe cargar en menos de 2 segundos.			



19.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF019	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	**Moderación de reseñas**		
Descripción			
Como administrador puedo moderar y eliminar reseñas inapropiadas para mantener calidad del contenido.			
Funcionalidad			
Herramientas de moderación para control de contenido.			
Criterios de aceptación	1. Se debe poder marcar reseñas como inapropiadas. 2. Se debe notificar al usuario sobre moderación. 3. Se debe mantener registro de acciones de moderación.		
Restricciones			
Solo administradores pueden moderar contenido.			

20.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RF020	**Actor**	Administrador
NOMBRE DEL REQUERIMIENTO	**Reportes de actividad**		
Descripción			
Como administrador puedo generar reportes de actividad y estadísticas para monitorear la plataforma.			
Funcionalidad			
Dashboard administrativo con métricas y reportes.			

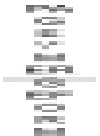
Criterios de aceptación	1. Se debe mostrar estadísticas de usuarios activos. 2. Se debe incluir métricas de contenido más popular. 3. Se debe generar reportes por períodos de tiempo.
Restricciones	
Los reportes solo son accesibles para administradores.	

21.

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RNF21	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Seguridad de contraseñas**		
Descripción			
Como sistema debo usar bcrypt para hashear contraseñas antes de almacenarlas.			
Funcionalidad			
Implementación de seguridad en almacenamiento de credenciales.			
Criterios de aceptación	1. Las contraseñas deben hashearse con bcrypt y salt. 2. Las contraseñas en texto plano no deben almacenarse. 3. Se debe validar fortaleza de contraseña al registro.		
Restricciones			
Las contraseñas deben tener mínimo 8 caracteres con mayúsculas, minúsculas y números.			

22.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RNF22	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Registro de usuarios con roles diferenciados**		
Descripción			
Como sistema debo implementar autenticación JWT con tokens que expiren automáticamente.			
Funcionalidad			



Sistema de tokens seguros con expiración automática.

Criterios de aceptación	1. Los tokens deben expirar en 24 horas. 2. Se debe incluir información de rol en el token. 3. Se debe validar integridad del token en cada petición.

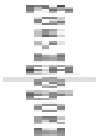
Restricciones
Los tokens expirados deben rechazarse automáticamente.

23.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RNF23	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Limitación de peticiones**		
Descripción			
Como sistema debo usar rate limiting para prevenir ataques de fuerza bruta.			
Funcionalidad			
Control de velocidad de peticiones por IP y usuario.			
Criterios de aceptación	1. Se debe limitar a 100 peticiones por minuto por IP. 2. Se debe implementar bloqueo temporal tras intentos fallidos. 3. Se debe excluir peticiones estáticas del rate limiting.		
Restricciones			
Las APIs críticas deben tener límites más restrictivos.			

24.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RNF24	**Actor**	Sistema



NOMBRE DEL REQUERIMIENTO	Validación de datos
Descripción	
Como sistema debo implementar validaciones robustas usando express-validator.	
Funcionalidad	
Validación de entrada de datos en todos los endpoints.	
Criterios de aceptación	1. Se debe validar tipos de datos y formatos. 2. Se debe sanitizar entrada para prevenir inyecciones. 3. Se debe retornar errores descriptivos de validación.
Restricciones	
Todas las rutas deben implementar validación obligatoria.	

25.

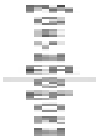
HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RNF25	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Base de datos MongoDB**		
Descripción			
Como sistema debo utilizar MongoDB con driver oficial para persistencia de datos.			
Funcionalidad			
Implementación de base de datos NoSQL con driver nativo.			
Criterios de aceptación	1. Se debe usar driver oficial de MongoDB para Node.js. 2. Se debe implementar conexión con pool de conexiones. 3. Se debe manejar errores de conexión adecuadamente.		
Restricciones			
No se debe usar Mongoose u otros ODMs.			

26.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RNF26	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Transacciones MongoDB**		
Descripción			
Como sistema debo implementar transacciones para operaciones críticas de reseñas y likes.			
Funcionalidad			
Transacciones atómicas para mantener consistencia de datos.			
Criterios de aceptación	1. Las operaciones de reseñas deben ser transaccionales. 2. Se debe hacer rollback automático en caso de error. 3. Las transacciones deben completarse o fallar completamente.		
Restricciones			
Solo operaciones críticas deben usar transacciones por rendimiento.			

27.

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RNF27	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Documentación API con Swagger**		
Descripción			
Como sistema debo documentar todos los endpoints usando swagger-ui-express.			
Funcionalidad			
Documentación automática e interactiva de la API.			
Criterios de aceptación	1. Todos los endpoints deben estar documentados. 2. Se debe incluir ejemplos de peticiones y respuestas. 3. La documentación debe ser accesible vía web.		
Restricciones			



La documentación debe actualizarse automáticamente con cambios de código.

28.

HISTORIA DE USUARIO			
Prioridad: Media			
CÓDIGO DEL REQUERIMIENTO:	RNF28	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Arquitectura modular**		
Descripción			
Como sistema debo seguir arquitectura modular separando responsabilidades en carpetas específicas.			
Funcionalidad			
Organización del código en estructura escalable y mantenible.			
Criterios de aceptación	1. Se debe separar en models, controllers, routes, middlewares, services. 2. Cada módulo debe tener responsabilidad única. 3. Se debe implementar inyección de dependencias donde sea apropiado.		
Restricciones			
La estructura debe seguir principios SOLID y patrones de diseño establecidos.			

29.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RNF29	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Versionado de API**		
Descripción			
Como sistema debo versionar la API siguiendo convenciones de Semantic Versioning.			
Funcionalidad			

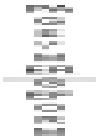
Control de versiones para mantener compatibilidad y evolución de API.	
Criterios de aceptación	1. Se debe implementar versionado en URLs (/api/v1/). 2. Se debe mantener retrocompatibilidad entre versiones menores. 3. Se debe documentar cambios breaking en changelog.
Restricciones	
Los cambios breaking solo pueden introducirse en versiones mayores.	

30.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RNF30	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Configuración CORS**		
Descripción			
Como sistema debo configurar CORS para permitir comunicación segura con frontend.			
Funcionalidad			
Control de acceso entre dominios para seguridad web.			
Criterios de aceptación	1. Se debe permitir solo dominios autorizados. 2. Se debe configurar métodos HTTP permitidos. 3. Se debe manejar preflight requests correctamente.		
Restricciones			
No se debe permitir origen wildcard (*) en producción.			

31.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RNF31	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Variables de entorno**		



Descripción	
Como sistema debo usar variables de entorno para toda configuración sensible.	
Funcionalidad	
Gestión segura de configuración y secretos.	
Criterios de aceptación	1. Se debe usar dotenv para cargar variables. 2. Todas las credenciales deben estar en .env. 3. Se debe proveer .env.example con variables necesarias.
Restricciones	
El archivo .env no debe versionarse en Git.	

32.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RNF32	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Manejo de errores centralizado**		
Descripción			
Como sistema debo implementar manejo centralizado de errores con códigos HTTP apropiados.			
Funcionalidad			
Sistema uniforme de manejo y respuesta de errores.			
Criterios de aceptación	1. Se debe usar middleware de manejo de errores global. 2. Los errores deben incluir código HTTP correcto. 3. Se debe registrar errores para debugging.		
Restricciones			
No se deben exponer detalles internos del sistema en producción.			

33.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL	RNF33	**Actor**	Usuario

REQUERIMIENTO:			
NOMBRE DEL REQUERIMIENTO	**Diseño responsive**		
Descripción			
Como usuario puedo acceder a la plataforma desde dispositivos móviles y desktop correctamente.			
Funcionalidad			
Interfaz adaptable a diferentes tamaños de pantalla.			
Criterios de aceptación	1. Se debe adaptar a pantallas de 320px a 1920px. 2. Los elementos deben ser táctiles en dispositivos móviles. 3. Se debe mantener usabilidad en todas las resoluciones.		
Restricciones			
El diseño debe ser mobile-first y progresivamente mejorado.			

34.

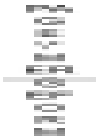
HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RNF34	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Rendimiento de consultas**		
Descripción			
Como sistema debo mantener tiempos de respuesta inferiores a 2 segundos.			
Funcionalidad			
Optimización de rendimiento para experiencia fluida.			
Criterios de aceptación	1. Las consultas de lectura deben responder en menos de 2 segundos. 2. Se debe implementar paginación para listados grandes. 3. Se debe usar índices apropiados en base de datos.		
Restricciones			
Las operaciones complejas pueden tomar hasta 5 segundos máximo.			

35.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RNF34	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Rendimiento de consultas**		
Descripción			
Como sistema debo mantener tiempos de respuesta inferiores a 2 segundos.			
Funcionalidad			
Optimización de rendimiento para experiencia fluida.			
Criterios de aceptación	4. Las consultas de lectura deben responder en menos de 2 segundos. 5. Se debe implementar paginación para listados grandes. 6. Se debe usar índices apropiados en base de datos.		
Restricciones			
Las operaciones complejas pueden tomar hasta 5 segundos máximo.			

36.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RNF36	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Disponibilidad del sistema**		
Descripción			
Como sistema debo mantener disponibilidad del 99% durante operación normal.			
Funcionalidad			
Alta disponibilidad y tolerancia a fallos.			
Criterios de aceptación	1. Se debe implementar reinicio automático en caso de fallo. 2. Se debe monitorear salud del sistema. 3. Se debe mantener logs de tiempo de inactividad.		
Restricciones			



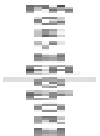
El mantenimiento programado debe notificarse con 24 horas de anticipación.

37.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RNF37	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Logging y auditoría**		
Descripción			
Como sistema debo implementar logging completo para debugging y auditoría.			
Funcionalidad			
Sistema de registro de eventos y errores para monitoreo.			
Criterios de aceptación	1. Se debe registrar todas las operaciones críticas. 2. Los logs deben incluir timestamp y nivel de severidad. 3. Se debe rotar logs automáticamente por tamaño.		
Restricciones			
Los logs no deben contener información sensible como contraseñas.			

38.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RNF38	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Validación contra inyecciones**		
Descripción			
Como sistema debo validar entrada de datos para prevenir ataques de inyección.			
Funcionalidad			



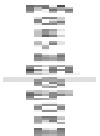
Sanitización y validación de datos para seguridad.	
Criterios de aceptación	1. Se debe escapar caracteres especiales en consultas. 2. Se debe validar tipos de datos antes de procesamiento. 3. Se debe implementar whitelist de caracteres permitidos.
Restricciones	
Toda entrada de usuario debe considerarse potencialmente maliciosa.	

39.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RNF39	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Optimización de consultas**		
Descripción			
Como sistema debo implementar índices apropiados para optimizar consultas frecuentes.			
Funcionalidad			
Optimización de base de datos para mejorar rendimiento.			
Criterios de aceptación	1. Se debe crear índices en campos de búsqueda frecuente. 2. Se debe optimizar consultas de ranking y filtrado. 3. Se debe monitorear rendimiento de consultas.		
Restricciones			
Los índices no deben impactar significativamente las operaciones de escritura.			

40.

HISTORIA DE USUARIO			
Prioridad: Alta			
CÓDIGO DEL REQUERIMIENTO:	RNF40	**Actor**	Sistema
NOMBRE DEL REQUERIMIENTO	**Backup y recuperación**		



Descripción	
Como usuario puedo registrarme en KarenFlix Como sistema debo implementar respaldo automático y procedimientos de recuperación. mi rol para acceder a las funcionalidades correspondientes.	
Funcionalidad	
Protección de datos mediante backups regulares.	
Criterios de aceptación	1. Se debe realizar backup diario automático. 2. Se debe probar procedimientos de recuperación mensualmente. 3. Los backups deben almacenarse en ubicación segura.
Restricciones	
R	

5. METODOLOGÍA

Para el presente proyecto se va a utilizar el marco de trabajo SCRUM que implica el esfuerzo de colaboración para crear un producto nuevo, este se va a utilizar en conjunto con la metodología ágil Kanban ya que es la que se ajusta de una mejor manera a los requerimientos del proyecto permitiendo una mayor efectividad al realizar el software solicitado por el cliente.

El equipo de trabajo está seccionado de la siguiente manera.
Scrum Master: Juan David Saavedra Jaimez
Product Owner: Simon Dante Salamanca
Equipo Scrum: Simon Dante Salamanca, Juan David Saavedra Jaimez

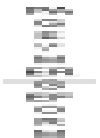
6. EVIDENCIA DE PLANTEAMIENTO DE PLATAFORMA DE TRABAJO

Con el fin de garantizar la trazabilidad y el cumplimiento del trabajo colaborativo en el desarrollo de la plataforma, se presenta la evidencia de la metodología aplicada y los entregables generados por el equipo:

Repositorio del proyecto

1. Se creó y gestionó un repositorio en GitHub donde se evidencia:

- El correcto uso de ramas para organizar el flujo de trabajo.



- La implementación de roles de trabajo colaborativo asignados a cada integrante.

- El uso de conventional commits para mantener un historial claro y estandarizado.

2. Link del repositorio:

https://github.com/Dante-Sal/Proyecto_Express_S1_SalamancaDante-SaavedraJuan-frontend

https://github.com/Dante-Sal/Proyecto_Express_S1_SalamancaDante-SaavedraJuan-backend

- Tablero Scrum

1. Se implementó un tablero Scrum en Jira que documenta el ciclo de vida del proyecto en todas sus etapas:

- Historias de usuario con sus respectivos criterios de aceptación.

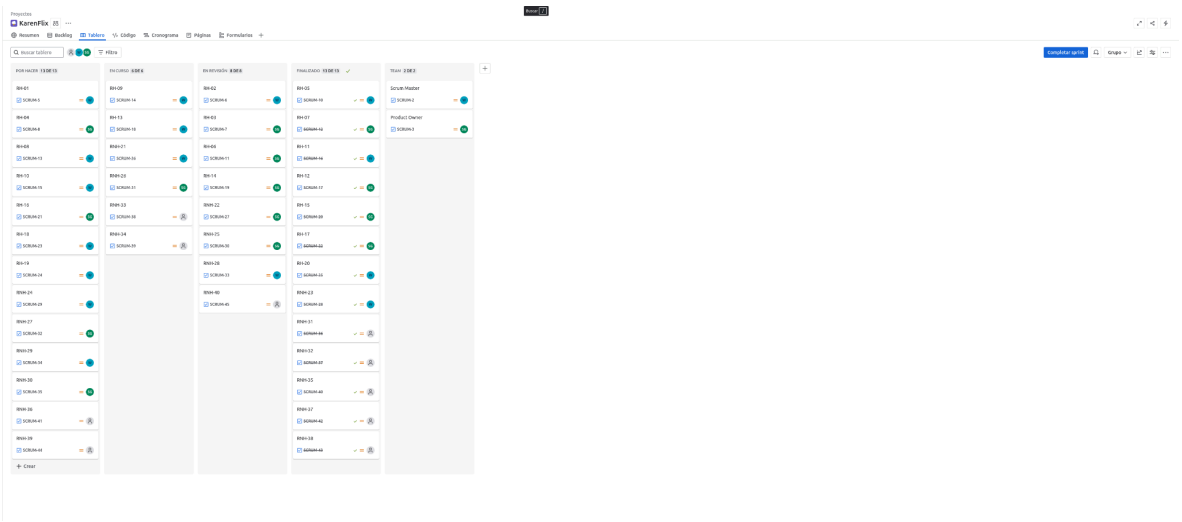
- Tareas y requerimientos detallados, asignados a responsables.

- Priorización y estimación de tiempos de desarrollo.

- Estados de avance: pendiente, en progreso, en revisión y finalizado.

2. Evidencia:

https://karenflix.atlassian.net/jira/software/projects/SCRUM/boards/1/backlog?atlOrigin=eyJpIjoiMmVjNTdhYmVlYzU0NDI2ZjlhNjA2MGViYWJjMzcxNGUiLCJwIjoiaiJ9



Además del código y del tablero, se ha incluido evidencia de calidad mediante pruebas automatizadas y scripts de inicialización. Las pruebas unitarias y de integración cubren la lógica de las validaciones de modelo y los casos de uso que requieren transacciones en MongoDB; en el repositorio se documenta el comando para ejecutar la suite de pruebas y las condiciones necesarias para ejecutarlas en un entorno de CI o local de pruebas. Se han incorporado scripts que permiten inicializar la base de datos, poblarla con datos de ejemplo y ejecutar flujos de demostración que generan las salidas esperadas para los evaluadores. La política de commits, las pull requests con revisiones, las etiquetas de versiones y los scripts de despliegue figuran en la documentación de entrega como evidencia del cumplimiento de buenas prácticas de ingeniería de software y de la correcta coordinación del equipo.

7. CONCLUSIONES

En términos generales, el proyecto del Gestor de Portafolio de Proyectos Freelance alcanzó los objetivos planteados en el levantamiento de requerimientos y demostró la viabilidad técnica y organizativa de la solución propuesta. La elección de Node.js como plataforma de desarrollo y del driver oficial de MongoDB para la persistencia permitió implementar operaciones transaccionales y garantizar la integridad de los datos en procesos críticos como la conversión de propuestas en proyectos y el registro de movimientos financieros. La adopción de principios de diseño orientado a objetos y SOLID, junto con la aplicación de patrones de diseño documentados en el código, contribuyó a una arquitectura clara y extensible que facilita el mantenimiento y la incorporación de mejoras futuras. Las pruebas automatizadas y los scripts de inicialización incluidos en el repositorio aportaron evidencia reproducible del correcto funcionamiento de los flujos críticos, y la documentación entregada permitió que evaluadores y miembros externos del equipo pudieran verificar los criterios de aceptación definidos.

La retrospección sobre el uso del tablero ágil como plataforma de coordinación evidencia aportes significativos al proceso productivo. El tablero, alojado en la plataforma indicada, facilitó la visualización continua del estado de las historias de usuario y de las tareas técnicas, permitiendo al equipo identificar cuellos de botella y priorizar ítems que afectaban la integridad del sistema, tales como validaciones de modelo y operaciones transaccionales. El esquema pull-based aplicado en el tablero y los límites de trabajo en curso favorecieron la finalización de tareas antes de iniciar nuevas actividades, lo que redujo la fragmentación del trabajo y mejoró la calidad de los entregables. Las pull requests asociadas a las tarjetas del tablero proporcionaron un flujo ordenado de revisiones y retroalimentación, integrando control de calidad y trazabilidad en cada incremento del producto.

No obstante, la retrospección también revela áreas de mejora importantes para proyectos futuros. La estimación de tiempos y la granularidad de las tareas podrían haberse refinado para facilitar una asignación de carga más equilibrada entre los miembros y reducir repeticiones por ajuste de requerimientos. La instrumentación de métricas cuantitativas (por ejemplo lead time, cycle time, tasa de rework) quedó a nivel cualitativo en esta etapa; incorporar el registro sistemático de estas métricas permitiría tomar decisiones mejor fundadas sobre priorización y capacidad del equipo. Asimismo, la automatización más amplia de pipelines de integración continua que ejecuten pruebas de integración contra un entorno controlado de MongoDB aceleraría las validaciones y disminuiría la dependencia de ejecuciones manuales para evidencias de comportamiento transaccional.

Como lecciones aprendidas, resultó clave mantener una comunicación estrecha entre definición de requerimientos, modelado de datos y diseño de transacciones, dado que decisiones tempranas en la estructura de datos repercutieron directamente en la complejidad de las operaciones atómicas y en la necesidad de mecanismos de compensación. La práctica de documentar patrones y criterios de aceptación en las tarjetas del tablero facilitó la revisión por pares y la generación de evidencias reproducibles. Finalmente, la experiencia confirmó que una aproximación incremental, sustentada en un tablero ágil y en revisiones frecuentes, es adecuada para proyectos académicos y de alcance realista, pues permite adaptar la solución ante hallazgos técnicos sin comprometer la entrega.

En cuanto a recomendaciones para la continuidad del proyecto, se sugiere priorizar la implantación de métricas operativas en el tablero, consolidar un pipeline de CI/CD que incluya pruebas de integración automatizadas y mejorar la formación en estimación de tareas para equilibrar la carga por sprint o ciclo. Igualmente, conviene documentar y versionar procedimientos operativos críticos (respaldo y restauración, scripts de migración, políticas de roles) en un manual operativo dentro del repositorio para facilitar la transferencia de conocimiento. En conjunto, las conclusiones y la retrospectiva muestran que el proyecto alcanzó un nivel sólido de cumplimiento funcional y técnico, a la vez que proporciona una hoja de ruta clara para profesionalizar y escalar la solución en iteraciones posteriores.