# KarenFlix

**NAYA LIZCANO JAIMES**
**AURA CAMILA PICO**

**S1**

**PEDRO FELIPE GOMEZ BONILLA**

**CAMPUS LANDS**
**S1**
**RUTA NODE**
**CAJASAN**
**2025**
**TÍTULO DEL PROYECTO**

# KARENFLIX

## 1. SITUACIÓN PROBLEMA

Se necesita desarrollar una aplicación fullstack usando Node.JS con Express para el Back-end, y HTML. CSS y JavaScript para el Front-end que permita a los usuarios registrar, calificar y rankear películas, animes y series geek. Esta herramienta debe incluir funcionalidades para gestionar usuarios, reseñas, categorías y rankings, diferenciando permisos de usuario y administrador. Además, debe contar con autenticación segura, validaciones robustas y un frontend que consuma la API desarrollada.

## 2. LEVANTAMIENTO DE REQUERIMIENTOS

## 3. REQUERIMIENTOS

### 3.1. Requerimientos Funcionales

1. Gestión de usuarios
    - RF1.1: El sistema debe permitir el **registro de usuarios** con validación de datos.
    - RF1.2: El sistema debe permitir el **inicio de sesión** con autenticación mediante **JWT**.
    - RF1.3: El sistema debe soportar **roles**: usuario y administrador.
    - RF1.4: Los administradores deben poder **gestionar categorías** y **aprobar películas/series**.
    - RF1.5: El sistema debe registrar información básica de cada usuario: nombre, correo, contraseña encriptada y fecha de registro.

Aquí tienes el texto con los espacios y formato corregidos para documentos de Google:

**2. Gestión de películas y series**

- RF2.1: El sistema debe permitir el **CRUD de películas/series** (crear, leer, actualizar, eliminar).
- RF2.2: Solo los **administradores** pueden aprobar nuevas entradas.
- RF2.3: El sistema debe **validar títulos repetidos** para evitar duplicados.
- RF2.4: Cada película/serie debe incluir como mínimo:
    - Título
    - Descripción
    - Categoría
    - Año de lanzamiento
    - Imagen (opcional)

**3. Gestión de reseñas y ratings**

- RF3.1: Los usuarios deben poder **crear, editar y eliminar reseñas**.
- RF3.2: Cada reseña debe incluir:
    - Título
    - Comentario
    - Calificación (escala de 1 a 5 con decimales: 1, 1.1, 1.2… 5).
- RF3.3: Los usuarios deben poder dar **like/dislike** a reseñas de otros, pero no a las propias.
- RF3.4: El sistema debe calcular un **ranking ponderado** de películas/series considerando:
    - Calificaciones numéricas
    - Likes/dislikes
    - Fecha de la reseña

**4. Gestión de categorías**

- RF4.1: El sistema debe permitir un **CRUD de categorías**.
- RF4.2: Solo los administradores pueden crear, editar o eliminar categorías.
- RF4.3: Deben existir al menos **4 categorías iniciales** (ejemplo: Anime, Ciencia Ficción, Superhéroes, Fantasía).

**5. Ranking y listados**

- RF5.1: El sistema debe mostrar un **listado de películas/series** con ordenamiento por **popularidad y ranking**.
- RF5.2: El sistema debe permitir **filtrar películas/series por categoría**.
- RF5.3: Debe existir una **vista de detalle** para cada película/serie, incluyendo sus reseñas.

**6. Front-end (HTML + CSS + JS)**

- RF6.1: El frontend debe incluir como mínimo las siguientes pantallas:
    - Inicio
    - Registro/Login
    - Listado de películas/series
    - Detalle de película/serie
    - Panel de administración
- RF6.2: El frontend debe **consumir los endpoints del backend**.
- RF6.3: El frontend debe mostrar **mensajes de validación y error** provenientes del backend.
- RF6.4: La interfaz debe ser **amigable y responsive**.

**7. Documentación y entrega**

- RF7.1: El backend debe documentar todos los endpoints usando **Swagger**.
- RF7.2: El **README** del backend debe incluir:
    - Descripción del proyecto
    -

- Descripción del proyecto
- Tecnologías usadas
- Instrucciones de instalación y uso
- Estructura del proyecto
- Principios aplicados
- Consideraciones técnicas
- Créditos
- Link al repositorio del frontend
- RF7.3: Se debe entregar un **documento SCRUM en PDF** con:
- Roles asignados
- Definición de sprints
- Historias de usuario
- Herramienta de seguimiento
- Evidencias del proceso
- RF7.4: Se debe entregar un **video (máx. 10 minutos)** con:
- Todos los integrantes en cámara
- Breve explicación técnica del backend
- Ejemplos de código
- Demostración completa del frontend

### 3.2. Requerimientos No Funcionales

1. Arquitectura y desarrollo

- RNF1.1: El backend debe estar desarrollado en **Node.js con Express**.
- RNF1.2: La arquitectura debe ser **modular y escalable**, con carpetas organizadas en:

- /models
- /controllers
- /routes
- /middlewares
- /services
- /config
- /utils

- RNF1.3: Se debe usar **dotenv** para la configuración de variables de entorno.
- RNF1.4: Se debe configurar **CORS** para permitir la conexión desde el frontend.
- RNF1.5: El sistema debe manejar **errores centralizados** y devolver códigos HTTP correctos.

2. Seguridad
   - RNF2.1: El sistema debe implementar autenticación con **JWT (passport-jwt, jsonwebtoken)**.
   - RNF2.2: Las contraseñas deben estar encriptadas usando **bcrypt**.
   - RNF2.3: Se debe implementar **express-rate-limit** para limitar peticiones y evitar abusos.
   - RNF2.4: Validaciones de datos en endpoints usando **express-validator**.
3. Persistencia y base de datos
   - RNF3.1: El sistema debe usar **MongoDB con el driver oficial** (no mongoose).
   - RNF3.2: Se deben manejar **transacciones reales** en operaciones críticas (ej. reseñas y ratings).
   - RNF3.3: El sistema debe garantizar la **consistencia de datos** en caso de errores.

4. Versionado y documentación
    - RNF4.1: El API debe seguir **versionado semántico (semver)**.
    - RNF4.2: Todos los endpoints deben estar documentados en **Swagger**.

5. Calidad y usabilidad
    - RNF5.1: El frontend debe ser **responsive** y funcionar correctamente en dispositivos móviles y de escritorio.
    - RNF5.2: La interfaz debe ser **simple e intuitiva** para los usuarios finales.
    - RNF5.3: Los mensajes de error deben ser claros y en un lenguaje comprensible.

**4.**

| **HISTORIA DE USUARIO** | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF0.1 | **Actor** | Usuario |
| **NOMBRE DEL REQUERIMIENTO** | **Registro de usuarios** | | |
| **Descripción** | | | |
| Como usuario quiero registrarme en la aplicación para tener una cuenta personal y acceder a las funcionalidades. | | | |
| **Funcionalidad** | | | |
| El sistema debe permitir crear cuentas nuevas validando correo único, encriptando la contraseña y guardando datos en la base. | | | |
| **Criterios de aceptación** | 1. No se permite registrar un usuario repetido.<br>2. La contraseña debe encriptar con bycript antes de guardarse<br>3. Si algún campo requerido falta, el sistema devuelve error 400 con mensaje claro<br>4. El registro exitoso devuelve un JSON con datos básicos del usuario y un token JWT. | | |
| **Restricciones** | | | |
| El sistema no permitirá registros con contraseñas de más de 8 caracteres.<br>Solo se aceptan correos con formato válido. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF1.2 | **Actor** | Usuario |
| **NOMBRE DEL REQUERIMIENTO** | **Inicio de sesión** | | |
| **Descripción** | | | |
| El sistema debe permitir el inicio de sesión con autenticación mediante JWT. | | | |
| **Funcionalidad** | | | |
| Validar credenciales y generar un token de acceso. | | | |
| **Criterios de aceptación** | 1. Acceso solo con credenciales validas<br>2. El token expira despues del tiempo configurada. | | |
| **Restricciones** | | | |
| Maximo 3 intentos fallidos antes del bloqueo temporal | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF1.3 | **Actor** | Usuario/Administrador |
| **NOMBRE DEL REQUERIMIENTO** | **Inicio de sesión** | | |
| **Descripción** | | | |
| El sistema debe soportar roles: usuario y administrador. | | | |
| **Funcionalidad** | | | |
| Restringir o permitir acceso según el rol. | | | |
| **Criterios de aceptación** | 1. Los usuarios no pueden realizar acciones exclusivas de administradores.<br><br>2. Los administradores tienen acceso al panel de gestión. | | |
| **Restricciones** | | | |
| No es posible cambiar de rol desde el frontend. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF1.4 | **Actor** | Administrador |
| **NOMBRE DEL REQUERIMIENTO** | **Gestión de categorías y aprobación de películas** | | |
| **Descripción** | | | |
| Los administradores deben poder gestionar categorías y aprobar películas/series. | | | |
| **Funcionalidad** | | | |
| CRUD de categorías y validación de contenido. | | | |
| **Criterios de aceptación** | 1. Solo los administradores pueden aprobar o rechazar nuevas películas/series.<br><br>2. Cambios reflejados en la base de datos en tiempo real. | | |
| **Restricciones** | | | |
| No accesible para usuarios regulares. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF1.5 | **Actor** | Usuario |
| **NOMBRE DEL REQUERIMIENTO** | **Registro de información básica** | | |
| **Descripción** | | | |
| El sistema debe registrar nombre, correo, contraseña encriptada y fecha de registro. | | | |
| **Funcionalidad** | | | |
| Guardar información mínima de usuarios. | | | |
| **Criterios de aceptación** | 1.Todos los campos deben estar completos.<br><br>2. La fecha de registro se genera automáticamente. | | |
| **Restricciones** | | | |
| No se permite registrar usuarios sin correo. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF2.1 | **Actor** | Administrador/Usuario |
| **NOMBRE DEL REQUERIMIENTO** | **CRUD de películas y series** | | |
| **Descripción** | | | |
| El sistema debe permitir crear, leer, actualizar y eliminar películas/series. | | | |
| **Funcionalidad** | | | |
| Administrar el catálogo de contenido. | | | |
| **Criterios de aceptación** | 1.Los cambios se reflejan de inmediato en el listado. | | |
| **Restricciones** | | | |
| Solo administradores pueden modificar o eliminar. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF2.2 | **Actor** | Administrador/ |
| **NOMBRE DEL REQUERIMIENTO** | **Aprobación de entradas** | | |
| **Descripción** | | | |
| Solo los administradores pueden aprobar nuevas entradas. | | | |
| **Funcionalidad** | | | |
| Validación de contenido antes de ser público. | | | |
| **Criterios de aceptación** | 1.Una película/serie no es visible sin aprobación. | | |
| **Restricciones** | | | |
| Solo los administradores pueden modificar o eliminar. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF2.3 | **Actor** | Administrador |
| **NOMBRE DEL REQUERIMIENTO** | **Validación de títulos repetidos** | | |
| **Descripción** | | | |
| El sistema debe validar títulos repetidos para evitar duplicados. | | | |
| **Funcionalidad** | | | |
| Revisión automática al registrar. | | | |
| **Criterios de aceptación** | 1.Si ya existe un título idéntico, se rechaza el registro. | | |
| **Restricciones** | | | |
| Coincidencia exacta en mayúsculas/minúsculas. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF2.4 | **Actor** | Administrador |
| **NOMBRE DEL REQUERIMIENTO** | **Atributos minimos de peliculas/series** | | |
| **Descripción** | | | |
| Cada película/serie debe incluir título, descripción, categoría, año y una imagen opcional. | | | |
| **Funcionalidad** | | | |
| Estándar mínimo de datos. | | | |
| **Criterios de aceptación** | 1.Los campos obligatorios deben estar completos. | | |
| **Restricciones** | | | |
| El año debe ser numérico y mayor a 1900. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF3.1 | **Actor** | Usuario |
| **NOMBRE DEL REQUERIMIENTO** | **Gestión de reseñas y ratings** | | |
| **Descripción** | | | |
| Los usuarios deben poder crear, editar y eliminar reseñas. | | | |
| **Funcionalidad** | | | |
| Permitir la interacción con el catálogo. | | | |
| **Criterios de aceptación** | 1.El usuario solo puede editar/eliminar sus propias reseñas. | | |
| **Restricciones** | | | |
| No se pueden enviar reseñas vacías. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF3.2 | **Actor** | Usuario |
| **NOMBRE DEL REQUERIMIENTO** | **Contenido de reseña** | | |
| **Descripción** | | | |
| Cada reseña debe incluir título, comentario y calificación en escala 1 a 5 (decimales permitidos). | | | |
| **Funcionalidad** | | | |
| Estándar de reseñas. | | | |
| **Criterios de aceptación** | 1.Calificación debe estar entre 1 a 5 | | |
| **Restricciones** | | | |
| No se permiten comentarios con menos de 10 caracteres. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF3.3 | **Actor** | Usuario |
| **NOMBRE DEL REQUERIMIENTO** | **Likes/dislikes** | | |
| **Descripción** | | | |
| Los usuarios deben poder dar like/dislike a reseñas de otros, pero no a las propias. | | | |
| **Funcionalidad** | | | |
| Estándar de reseñas. | | | |
| **Criterios de aceptación** | 1. Un usuario solo puede votar una vez por reseña. | | |
| **Restricciones** | | | |
| No se permite auto-votarse. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF3.4 | **Actor** | Sistema |
| **NOMBRE DEL REQUERIMIENTO** | **Ranking ponderado** | | |
| **Descripción** | | | |
| El sistema debe calcular un ranking ponderado de películas/series considerando calificaciones, likes/dislikes y fecha de la reseña. | | | |
| **Funcionalidad** | | | |
| Ordenar contenido de manera justa. | | | |
| **Criterios de aceptación** | 1. Ordenar contenido de manera justa. | | |
| **Restricciones** | | | |
| El algoritmo no es modificable por usuarios. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF4.1 | **Actor** | Administrador |
| **NOMBRE DEL REQUERIMIENTO** | **CRUD de categorías** | | |
| **Descripción** | | | |
| El sistema debe permitir crear, leer, actualizar y eliminar categorías. | | | |
| **Funcionalidad** | | | |
| Administración de generos | | | |
| **Criterios de aceptación** | 1. Solo administradores pueden gestionar categorías. | | |
| **Restricciones** | | | |
| Categorías deben tener nombres únicos. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF4.2 | **Actor** | Administrador |
| **NOMBRE DEL REQUERIMIENTO** | **Categorías iniciales** | | |
| **Descripción** | | | |
| Deben existir al menos 4 categorías iniciales: Anime, Ciencia Ficción, Superhéroes, Fantasía. | | | |
| **Funcionalidad** | | | |
| Base mínima de datos. | | | |
| **Criterios de aceptación** | 1. Las categorías deben estar disponibles desde la primera ejecución. | | |
| **Restricciones** | | | |
| No pueden eliminarse todas las categorías. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF4.3 | **Actor** | Administrador |
| **NOMBRE DEL REQUERIMIENTO** | **Restricción de administración** | | |
| **Descripción** | | | |
| Solo administradores pueden crear, editar o eliminar categorías. | | | |
| **Funcionalidad** | | | |
| Control de contenido. | | | |
| **Criterios de aceptación** | 1. Usuarios solo pueden consultar categorías. | | |
| **Restricciones** | | | |
| No accesible para usuarios comunes. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF5.1 | **Actor** | Usuario |
| **NOMBRE DEL REQUERIMIENTO** | **Listado por popularidad y ranking** | | |
| **Descripción** | | | |
| El sistema debe mostrar un listado ordenado. | | | |
| **Funcionalidad** | | | |
| Ordenar catálogo. | | | |
| **Criterios de aceptación** | 1. Listado ordenado automáticamente. | | |
| **Restricciones** | | | |
| No editable por usuario. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF5.2 | **Actor** | Usuario |
| **NOMBRE DEL REQUERIMIENTO** | **Filtrado por categoria** | | |
| **Descripción** | | | |
| El sistema debe permitir filtrar por categoría. | | | |
| **Funcionalidad** | | | |
| Mejorar búsqueda. | | | |
| **Criterios de aceptación** | 1.Resultados coinciden con filtro. | | |
| **Restricciones** | | | |
| Se debe elegir categoría válida. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF5.3 | **Actor** | Usuario |
| **NOMBRE DEL REQUERIMIENTO** | **Vista de detalle** | | |
| **Descripción** | | | |
| Cada película/serie debe tener vista de detalle con reseñas. | | | |
| **Funcionalidad** | | | |
| Mostrar información completa. | | | |
| **Criterios de aceptación** | 1. Detalle muestra información y reseñas. | | |
| **Restricciones** | | | |
| Solo con peliculas aprobadas. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF6.1 | **Actor** | Usuario/Administrador |
| **NOMBRE DEL REQUERIMIENTO** | **Pantallas mínimas** | | |
| **Descripción** | | | |
| El frontend debe incluir Inicio, Registro/Login, Listado, Detalle, Panel admin. | | | |
| **Funcionalidad** | | | |
| Navegación mínima. | | | |
| **Criterios de aceptación** | 1.Todas las pantallas disponibles. | | |
| **Restricciones** | | | |
| Panel admin solo para administradores. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF6.2 | **Actor** | Sistema |
| **NOMBRE DEL REQUERIMIENTO** | **Consumo de Endpoints** | | |
| **Descripción** | | | |
| El frontend debe consumir endpoints del backend. | | | |
| **Funcionalidad** | | | |
| Comunicación API. | | | |
| **Criterios de aceptación** | 1.Datos cargan desde el backend. | | |
| **Restricciones** | | | |
| No hay datos mockeados. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF6.3 | **Actor** | Sistema |
| **NOMBRE DEL REQUERIMIENTO** | **Mensaje de validación y error** | | |
| **Descripción** | | | |
| El frontend debe consumir endpoints del backend. | | | |
| **Funcionalidad** | | | |
| Comunicación API. | | | |
| **Criterios de aceptación** | 1.Datos cargan desde el backend. | | |
| **Restricciones** | | | |
| No hay datos mockeados. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF6.4 | **Actor** | Usuario |
| **NOMBRE DEL REQUERIMIENTO** | **Interfaz responsive** | | |
| **Descripción** | | | |
| El frontend debe ser amigable y responsive. | | | |
| **Funcionalidad** | | | |
| Usabilidad en distintos dispositivos. | | | |
| **Criterios de aceptación** | 1.Funciona en móviles y escritorio. | | |
| **Restricciones** | | | |
| No compatible con navegadores obsoletos. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF7.1 | **Actor** | Desarrollador |
| **NOMBRE DEL REQUERIMIENTO** | **Documentación de endpoints** | | |
| **Descripción** | | | |
| El backend debe documentar todos los endpoints usando Swagger. | | | |
| **Funcionalidad** | | | |
| Referencia de API. | | | |
| **Criterios de aceptación** | 1.Todos los endpoints visibles en Swagger. | | |
| **Restricciones** | | | |
| Documentación sincronizada con API. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF7.2 | **Actor** | Desarrollador |
| **NOMBRE DEL REQUERIMIENTO** | **README backend** | | |
| **Descripción** | | | |
| El README debe contener detalles del proyecto. | | | |
| **Funcionalidad** | | | |
| Guía de instalación y uso. | | | |
| **Criterios de aceptación** | 1.Contiene descripción, tecnologías, instalación, estructura, créditos y link frontend. | | |
| **Restricciones** | | | |
| Archivo obligatorio en el repo | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF7.3 | **Actor** | Equipo |
| **NOMBRE DEL REQUERIMIENTO** | **Documento SCRUM** | | |
| **Descripción** | | | |
| Se debe entregar un PDF con roles, sprints, historias de usuario y evidencias. | | | |
| **Funcionalidad** | | | |
| Planeación formal | | | |
| **Criterios de aceptación** | 1.Documento completo adjunto. | | |
| **Restricciones** | | | |
| Formato PDF obligatorio. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RF7.4 | **Actor** | Equipo |
| **NOMBRE DEL REQUERIMIENTO** | **Video entrega** | | |
| **Descripción** | | | |
| Video máximo 10 min mostrando backend, frontend y equipo. | | | |
| **Funcionalidad** | | | |
| Demostración | | | |
| **Criterios de aceptación** | 1.Aparecen todos los integrantes.<br><br>2.Incluye demo completa. | | |
| **Restricciones** | | | |
| Duración max 10min | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RNF1.1 | **Actor** | Equipo de desarrollo |
| **NOMBRE DEL REQUERIMIENTO** | **Arquitectura y desarrollo** | | |
| **Descripción** | | | |
| Backend en Node.js con Express. | | | |
| **Funcionalidad** | | | |
| Base tecnológica. | | | |
| **Criterios de aceptación** | 1.Proyecto ejecuta en Node.js+Express | | |
| **Restricciones** | | | |
| No usar otro framework. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RNF1.2 | **Actor** | Equipo técnico |
| **NOMBRE DEL REQUERIMIENTO** | **Arquitectura modular** | | |
| **Descripción** | | | |
| Arquitectura organizada en carpetas (/models, /controllers, etc.). | | | |
| **Funcionalidad** | | | |
| Estructura escalable. | | | |
| **Criterios de aceptación** | 1.Código modular. | | |
| **Restricciones** | | | |
| Debe respetar la estructura definida. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RNF1.3 | **Actor** | Sistema |
| **NOMBRE DEL REQUERIMIENTO** | **Dotenv** | | |
| **Descripción** | | | |
| Uso de dotenv para variables de entorno. | | | |
| **Funcionalidad** | | | |
| Configuración segura. | | | |
| **Criterios de aceptación** | 1.Variables sensibles en `.env`. | | |
| **Restricciones** | | | |
| No subir `.env` al repo. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RNF1.4 | **Actor** | Sistema |
| **NOMBRE DEL REQUERIMIENTO** | **Cors** | | |
| **Descripción** | | | |
| Configuración de CORS para permitir frontend. | | | |
| **Funcionalidad** | | | |
| Comunicación segura. | | | |
| **Criterios de aceptación** | Frontend puede consumir API. | | |
| **Restricciones** | | | |
| Solo orígenes definidos. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RNF2.1 | **Actor** | Usuario/Sistema |
| **NOMBRE DEL REQUERIMIENTO** | **Autenticación JWT** | | |
| **Descripción** | | | |
| Errores centralizados con códigos HTTP correctos. | | | |
| **Funcionalidad** | | | |
| Acceso seguro. | | | |
| **Criterios de aceptación** | 1.Tokens válidos para acceso. | | |
| **Restricciones** | | | |
| Tokens expirados no válidos. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RNF2.2 | **Actor** | Usuario/Sistema |
| **NOMBRE DEL REQUERIMIENTO** | **Contraseñas encriptadas** | | |
| **Descripción** | | | |
| Uso de bcrypt para encriptar contraseñas. | | | |
| **Funcionalidad** | | | |
| Protección de datos. | | | |
| **Criterios de aceptación** | 1.Protección de datos. | | |
| **Restricciones** | | | |
| Algoritmo bcrypt obligatorio. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RNF2.3 | **Actor** | Sistema |
| **NOMBRE DEL REQUERIMIENTO** | **Rate limit** | | |
| **Descripción** | | | |
| Implementar express-rate-limit para limitar peticiones. | | | |
| **Funcionalidad** | | | |
| Prevenir abusos. | | | |
| **Criterios de aceptación** | 1.Usuarios no pueden hacer spam de requests. | | |
| **Restricciones** | | | |
| Máximo configurado por IP. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RNF2.4 | **Actor** | Sistema |
| **NOMBRE DEL REQUERIMIENTO** | **Validaciones** | | |
| **Descripción** | | | |
| Validar datos en endpoints con express-validator. | | | |
| **Funcionalidad** | | | |
| Integridad de datos. | | | |
| **Criterios de aceptación** | 1.Datos inválidos rechazan request. | | |
| **Restricciones** | | | |
| Uso obligatorio de express-validator | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RNF3.1 | **Actor** | Sistema |
| **NOMBRE DEL REQUERIMIENTO** | **Persistencia y base de datos** | | |
| **Descripción** | | | |
| Uso de MongoDB (sin mongoose). | | | |
| **Funcionalidad** | | | |
| Persistencia de datos. | | | |
| **Criterios de aceptación** | 1.Conexión estable y CRUD funcional. | | |
| **Restricciones** | | | |
| Prohibido mongoose. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RNF3.2 | **Actor** | Sistema |
| **NOMBRE DEL REQUERIMIENTO** | **Transacciones reales** | | |
| **Descripción** | | | |
| Manejo de transacciones en operaciones críticas. | | | |
| **Funcionalidad** | | | |
| Consistencia en reseñas/ratings. | | | |
| **Criterios de aceptación** | 1.Cambios atómicos en reseñas. | | |
| **Restricciones** | | | |
| Transacciones obligatorias. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RNF3.3 | **Actor** | Sistema |
| **NOMBRE DEL REQUERIMIENTO** | **Consistencia de datos** | | |
| **Descripción** | | | |
| Garantizar consistencia ante errores. | | | |
| **Funcionalidad** | | | |
| Robustez. | | | |
| **Criterios de aceptación** | 1.No se guardan datos corruptos. | | |
| **Restricciones** | | | |
| Rollback automático. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RNF4.1 | **Actor** | Equipo |
| **NOMBRE DEL REQUERIMIENTO** | **Versionado y documentación** | | |
| **Descripción** | | | |
| Control de versiones. | | | |
| **Funcionalidad** | | | |
| Robustez. | | | |
| **Criterios de aceptación** | 1.Versionado 1.0.0, 1.1.0, etc. | | |
| **Restricciones** | | | |
| No usar versiones arbitrarias. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RNF4.2 | **Actor** | Equipo tecnico |
| **NOMBRE DEL REQUERIMIENTO** | **Swagger** | | |
| **Descripción** | | | |
| Documentar todos los endpoints en Swagger. | | | |
| **Funcionalidad** | | | |
| Guía de API. | | | |
| **Criterios de aceptación** | 1.Swagger actualizado. | | |
| **Restricciones** | | | |
| Documentación obligatoria. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RNF5.1 | **Actor** | Usuario |
| **NOMBRE DEL REQUERIMIENTO** | **Interfaz intuitiva** | | |
| **Descripción** | | | |
| La interfaz debe ser simple y amigable. | | | |
| **Funcionalidad** | | | |
| Facilidad de uso. | | | |
| **Criterios de aceptación** | 1.Navegación en máximo 3 pasos. | | |
| **Restricciones** | | | |
| Estándares de usabilidad. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RNF5.2 | **Actor** | Usuario |
| **NOMBRE DEL REQUERIMIENTO** | **Mensajes claros** | | |
| **Descripción** | | | |
| Los mensajes de error deben ser comprensibles | | | |
| **Funcionalidad** | | | |
| Guía al usuario. | | | |
| **Criterios de aceptación** | 1.Mensajes claros y en lenguaje natural. | | |
| **Restricciones** | | | |
| Prohibido lenguaje técnico en frontend. | | | |

| HISTORIA DE USUARIO | | | |
|---|---|---|---|
| **Prioridad: Alta** | | | |
| **CÓDIGO DEL REQUERIMIENTO:** | RNF5.3 | **Actor** | Equipousuario |
| **NOMBRE DEL REQUERIMIENTO** | **Responsividad** | | |
| **Descripción** | | | |
| El frontend debe ser responsive. | | | |
| **Funcionalidad** | | | |
| Soporte móvil y escritorio. | | | |
| **Criterios de aceptación** | 1.Swagger actualizado. | | |
| **Restricciones** | | | |
| No soporta navegadores obsoletos. | | | |