Proyecto TFG: App de Recomendación de Películas y Series

Resumen Ejecutivo

Una aplicación móvil multiplataforma (Android e iOS) que revoluciona cómo los usuarios descubren contenido en streaming. La solución integra filtros inteligentes por plataforma, análisis en tiempo real, comunidad social y un panel administrativo robusto para garantizar moderación efectiva y experiencia de usuario excepcional.

Objetivo principal: Facilitar el descubrimiento de películas y series adaptado a las plataformas de streaming que cada usuario contrata, eliminando la confusión y proporcionando recomendaciones personalizadas respaldadas por la comunidad.

---

1. Descripción del Problema

Contexto Actual

Los usuarios contemporáneos se enfrentan a un desafío significativo: poseen suscripciones a múltiples servicios de streaming (Netflix, HBO Max, Disney+, Prime Video, etc.) pero no cuentan con herramientas efectivas para:

- **Identificar dónde ver contenido específico** entre sus plataformas contratadas
- **Descubrir tendencias actuales** sin depender de publicidad invasiva
- **Confiar en opiniones verificadas** de una comunidad con intereses similares
- **Seguir recomendaciones personalizadas** que respeten su catálogo disponible

Esta fragmentación del mercado streaming ha generado una experiencia fragmentada para el usuario final, que debe consultar múltiples aplicaciones para tomar decisiones informadas sobre qué contenido ver.

Oportunidad de Mercado

Existe una brecha clara entre lo que ofrecen plataformas individuales (catálogos restringidos) y lo que necesitan los usuarios (visión unificada). Los principales competidores actuales carecen de:

- Integración comunitaria genuina y feed social
- Panel de administración transparente y auditable
- Filtros avanzados con análisis en tiempo real
- Experiencia móvil optimizada para descubrimiento rápido

---

2. Solución Propuesta

Pilares Estratégicos

La aplicación se construye sobre cuatro pilares fundamentales:

2.1 Filtros Multi-Plataforma Inteligentes

Los usuarios configuran sus plataformas al registrarse. El sistema filtra dinámicamente el contenido, mostrando **solo lo que pueden ver realmente**. Integración con JustWatch API garantiza información actualizada sobre disponibilidad por región.

Características:

- Selección múltiple de servicios (Netflix, HBO Max, Disney+, Prime Video, etc.)
- Filtros avanzados secundarios (género, año, país, idioma, clasificación)
- Búsqueda fuzzy por título, actor, director
- Actualización automática de disponibilidad

2.2 Análisis en Tiempo Real y Rankings

Panel de tendencias dinámico que muestra:

- **Trending Now:** Contenido más buscado y recomendado en las últimas 24 horas
- **Más Vistas:** Ranking de películas/series más vistas por usuarios activos
- **Estrenos Cine:** Estrenos de cine próximos con fechas precisas
- **Novedades por Plataforma:** Últimos agregados a cada servicio
- **Top por Género:** Ranking dentro de cada categoría temática

Cada item en ranking incluye contexto visual: etiquetas de color (Trending, Popular, Novedad), medallas (oro, plata, bronce para top 3) y explicación: "¿Por qué te recomendamos esto?"

2.3 Comunidad Social y Recomendaciones

Un feed social genuino donde la comunidad comparte:

- **Reseñas completas:** Valoración (1-10), texto extenso, etiquetas
- **Listas personalizadas:** "Mejores películas para llorar", "Series para maratón", etc.
- **Debates temáticos:** Conversaciones sobre géneros, actores, tendencias
- **Recomendaciones directas:** Posts sugiriendo contenido con justificación

Funcionalidades sociales:

- Likes y comentarios en posts/reseñas
- Sistema de reportes para contenido inapropiado
- Perfiles públicos con historiales de contribuciones
- Badges de contribuyente activo (Crítico, Influencer, Moderador)

2.4 Panel Administrativo Robusto

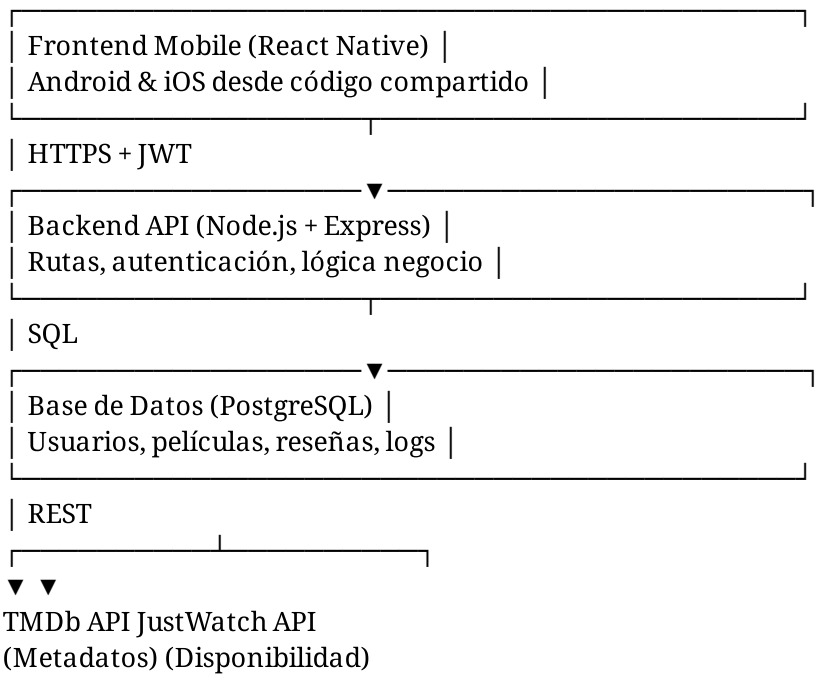
Control total y auditable de la plataforma:

- **Gestión de usuarios:** Vista/búsqueda completa, filtros por rol/actividad
- **Sistema de baneos:** Banear/desbanear con motivo, duración temporal o permanente
- **Moderación de contenido:** Revisión y eliminación de posts/reseñas inapropiadas
- **Historial completo:** Log auditable de cada acción administrativa
- **Estadísticas:** Usuarios activos, contenido más votado, reportes por tipo
- **Control de roles:** Asignación de moderadores y administradores

---

3. Stack Tecnológico

Arquitectura General



Tecnologías Seleccionadas

Componente	Tecnología	Justificación
Frontend	React Native (TypeScript)	Multiplataforma (iOS/Android), comunidad grande, rendimiento nativo, hot reload
Backend	Node.js + Express	JavaScript full-stack, escalabilidad, amplios ecosistema de librerías
Autenticación	JWT (JSON Web Tokens)	Stateless, seguro, ideal para APIs REST
Base de Datos	PostgreSQL	Relacional robusta, ACID, manejo complejo de usuarios y transacciones
APIs Externas	TMDb API, JustWatch API	Acceso a metadatos de películas y disponibilidad por plataforma
Diseño	Figma	Prototipado colaborativo, componentes reutilizables
Control de Versiones	Git + GitHub	Estándar industrial, colaboración

Por qué Node.js sobre Java/Python

Node.js seleccionado por:

- **Velocidad de desarrollo:** JavaScript compartido frontend/backend
- **Escalabilidad horizontal:** Manejo eficiente de múltiples conexiones concurrentes
- **Ecosistema npm:** Librerías especializadas (bcryptjs, jsonwebtoken, sequelize)
- **Performance:** Event-driven, no-blocking I/O
- **Tiempo de entrega:** Fase (febrero) crítica; Node reduce ciclos de iteración

---

4. Funcionalidades Clave

4.1 Autenticación y Configuración Inicial

Flujo de usuario nuevo:

1. **Registro:** Email + contraseña segura (validación de requisitos)
2. **Selección de plataformas:** Checkboxes multi-selección (Netflix, HBO, Disney+, Prime, etc.)
3. **Perfil básico:** Avatar, nombre público, biografía (opcional)
4. **Login:** Email + contraseña en accesos posteriores
5. **Recuperación:** Reset de contraseña por email con token temporal

Seguridad:

- Contraseñas hasheadas con Bcrypt (sal de 10 rondas)
- JWT con expiración (24 horas, refresh token válido 7 días)
- Validación de inputs en frontend y backend
- Middleware de autenticación en rutas protegidas
- Protección CORS y rate limiting

4.2 Pantalla Principal (Home)

Estructura visual:

```
┌─────────────────────────────────────────────────┐
│ Header: Logo + Avatar + Notificaciones
├─────────────────────────────────────────────────┤
│  RESUMEN RÁPIDO
│ "Hoy recomendamos: Oppenheimer"
│ [Ir a detalles]
├─────────────────────────────────────────────────┤
│  TRENDING NOW (Carrusel)
│ [Card1: Badge "Trending"] [Card2] ...
├─────────────────────────────────────────────────┤
│  TOP MOMENTO (Ranking Podio)
│  Barbie (8.9★)
│  Oppenheimer (8.5★)
│  Poor Things (7.8★)
├─────────────────────────────────────────────────┤
│  MÁS VISTAS
│ [Card1] [Card2] [Card3] ...
├─────────────────────────────────────────────────┤
│  POR GÉNEROS (Tabs)
│ [Acción] [Drama] [Comedia] [Sci-Fi]
│ Cards filtrables por género
├─────────────────────────────────────────────────┤
│  NOVEDADES EN TUS PLATAFORMAS
│ [Netflix] [HBO] [Disney+] (Tabs)
│ Últimas agregadas (últimos 7 días)
└─────────────────────────────────────────────────┘
```

Componentes clave:

- **Carruseles horizontales:** Scroll suave, detección de borde
- **Badges dinámicos:** "Trending" brilla con animación pulsante, colores contextuales
- **Ranking visual:** Medallas (oro/plata/bronce) + número grande de rating
- **"¿Por qué te recomendamos esto?":** Tooltip explicando criterio
- **Bottom navigation:** Home, Búsqueda, Social, Favoritos, Perfil

4.3 Detalle de Película/Serie

Información mostrada:

- Póster, título, año, duración, clasificación
- Sinopsis completa
- Actores principales + director
- Géneros, país de origen, idiomas disponibles
- **Dónde verlo:** Badges de plataformas en las que está disponible
- **Valoraciones:** Promedio TMDb, rating comunitario, número de valoraciones
- **Reseñas comunitarias:** Top 3 ordenadas por utilidad
- **Botones de acción:** Favoritos (corazón animado), Compartir, Reportar
- **Botón social:** "Ver comentarios/debate"

Microinteracciones:

- Guardar en favoritos: Animación de corazón relleno
- Swipe para pasar a siguiente movie en carrusel
- Scroll smooth entre secciones
- Feedback háptico en botones (vibración suave)

4.4 Búsqueda Avanzada

Filtros:

- Título (búsqueda fuzzy)
- Actor/Actriz
- Director
- Género (multi-selección)
- Año (rango)
- Plataforma (multi-selección, filtra según suscripciones)
- Clasificación (G, PG, PG-13, R, etc.)
- Ordenar por: Relevancia, Rating, Año, Más visto

Interfaz:

- Input de búsqueda con sugerencias en tiempo real
- Expansor de "Filtros avanzados"
- Resultados con cards consistentes
- Número total de resultados

4.5 Reseñas y Valoraciones

Flujo de reseña:

1. Usuario ve película/serie
2. Botón "Escribe una reseña" en detalle
3. Modal con campos:
    - Clasificación: 1-10 (slider visual)
    - Título de reseña
    - Texto (mín. 50 caracteres)
    - Etiquetas (Spoilers, Recomendado, No recomendado, etc.)
4. Publicar → Se muestra en comunidad

Visualización:

- Reseña por usuario (nombre, avatar, rating, fecha)
- Marcar como útil/inútil
- Responder/comentar
- Reportar si inapropiada

4.6 Módulo Social

Feed de posts:

- Posts de reseñas
- Listas personalizadas ("Top 10 películas para llorar")
- Debates temáticos
- Recomendaciones directas

Interacciones:

- Like/Unlike
- Comentarios (threaded)
- Compartir fuera de app
- Reportar contenido

Perfil público:

- Avatar, nombre, biografía
- Estadísticas (reseñas totales, listas creadas, followers)
- Historial de contribuciones
- Badges (Crítico activo, Moderador, etc.)

4.7 Favoritos

- Sistema de bookmarks por usuario
- Vista de favoritos en lista o grid
- Organización por carpetas (Viendo pronto, Top películas, etc.)
- Sincronización automática entre dispositivos (vía cuenta)

4.8 Panel Administrativo

Acceso: Solo usuarios con rol `admin`

Vistas:

1. **Dashboard:** KPIs (usuarios activos, posts hoy, reportes pendientes, usuarios baneados)
2. **Gestión de usuarios:**
    - Tabla con buscador y filtros (rol, estado, activo hoy/semana)
    - Acción rápida: Ver perfil, Ver posts, Acciones
3. **Baneos:**
    - Crear baneo: Usuario, motivo, duración (temporal/permanente)
    - Listar activos: Usuario, motivo, fecha inicio, fecha fin
    - Desbanear: Con confirmación
    - Baneo temporal: Recuento regresivo
4. **Moderación:**

- Reportes pendientes: Post reportado, razón, número de reportes
- Acción: Aprobar, Rechazar, Eliminar

5. **Logs de auditoría:**
   - Tabla: Fecha, admin, acción, usuario afectado, detalles
   - Filtrable por tipo de acción
6. **Estadísticas:**
   - Usuarios (totales, activos, nuevos esta semana)
   - Posts (totales, nuevos hoy, reportados)
   - Contenido (películas indexadas, reviews, favoritos)

---

5. Diseño y Experiencia de Usuario

5.1 Paleta de Colores

Color	Código Hex	Uso
Fondo Principal	`#141414`	Background general (tema oscuro)
Accent Primario	#E50914	Botones principales, highlights, badges "Trending"
Verde Éxito	`#10B981`	Badges "Disponible", confirmaciones
Azul Info	#3B82F6	Badges "Popular", información
Naranja Aviso	`#F97316`	Badges "Novedad", alertas
Gris Secundario	#6B7280	Textos secundarios, separadores
Blanco	`#FFFFFF`	Textos principales, contraste

5.2 Tipografía

Sistema de tipografía:

- **Headings (H1-H3):** Titillium Web Bold, tamaño 24-32px, espaciado amplio
- **Body text:** Inter Regular, tamaño 14-16px, line-height 1.6
- **Secundario:** Inter Light, tamaño 12px, para metadatos
- **Etiquetas:** Roboto Mono, tamaño 10-12px, uppercase, tracking amplio

5.3 Componentes Base

Cards de película/serie:

```
┌──────────────────────────┐
│ [Poster] │ ← Imagen 3:4 ratio
├──────────────────────────┤
│ Título (2 líneas)
│ ★ 8.5 (4.2K)
│ [Netflix] [HBO]
└──────────────────────────┘
```

Badge de etiqueta:

```
┌──────────────────┐
│  Trending │ ← Color dinámico, ícono sugestivo
└──────────────────┘
```

Botón principal:

```
┌──────────────────────────────┐
│ AGREGAR FAVORITOS │ ← Uppercase, bold, rojo #E50914
└──────────────────────────────┘
```

Ranking podio (Top 3):

```

```

Barbie
8.9 ★

Oppenheimer Poor Things
8.5 ★ 7.8 ★

5.4 Microinteracciones

Animaciones clave:

- **Badges Trending:** Pulsación sutil (opacity 0.8 → 1.0) cada 2 segundos
- **Guardar favorito:** Corazón se rellena con animación de escala (0.8 → 1.2 → 1.0)
- **Swipe de carrusel:** Transición suave con deceleration
- **Botones:** Efecto ripple al presionar (Material Design)
- **Load de reseñas:** Skeleton screens en lugar de spinners

5.5 Accesibilidad

- Contraste mínimo 4.5:1 (WCAG AA)
- Tamaños de tap target: 48x48dp mínimo
- Soporte para lectores de pantalla
- Navegación con teclado completa
- Pruebas de colorblindness (protanopía, deuteranopía)

6. Diseño de Base de Datos

Diagrama Entidad-Relación

Tabla	Descripción Rápida
users	Usuarios, roles, plataformas suscritas
platforms	Listado de plataformas (Netflix, HBO, etc.)
user_platforms	Relación usuarios-plataformas (M2M)
movies	Películas/series indexadas
reviews	Reseñas comunitarias
posts	Publicaciones sociales
favorites	Películas favoritas por usuario
movies_cache	Cache de metadatos TMDb
comments	Comentarios en posts
likes	Likes en posts/reseñas
reports	Reportes de contenido inapropiado
bans	Registro de baneos
admin_actions	Log auditable de acciones admin

Figure 1: Tablas principales del proyecto

Esquema Detallado

Tabla users

```
CREATE TABLE users (
id SERIAL PRIMARY KEY,
email VARCHAR(255) UNIQUE NOT NULL,
password_hash VARCHAR(255) NOT NULL,
username VARCHAR(50) UNIQUE NOT NULL,
avatar_url VARCHAR(255),
bio TEXT,
role ENUM('user', 'moderator', 'admin') DEFAULT 'user',
is_banned BOOLEAN DEFAULT FALSE,
ban_reason VARCHAR(255),
ban_until TIMESTAMP,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Tabla platforms

```
CREATE TABLE platforms (
id SERIAL PRIMARY KEY,
name VARCHAR(50) NOT NULL, -- 'Netflix', 'HBO Max', etc.
icon_url VARCHAR(255),
color_hex VARCHAR(7),
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Tabla user_platforms

```
CREATE TABLE user_platforms (
user_id INTEGER NOT NULL,
platform_id INTEGER NOT NULL,
subscribed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
PRIMARY KEY (user_id, platform_id),
FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
FOREIGN KEY (platform_id) REFERENCES platforms(id)
);
```

Tabla movies

```
CREATE TABLE movies (
id SERIAL PRIMARY KEY,
tmdb_id INTEGER UNIQUE NOT NULL,
title VARCHAR(255) NOT NULL,
description TEXT,
poster_url VARCHAR(255),
backdrop_url VARCHAR(255),
release_date DATE,
genre VARCHAR(100), -- CSV: 'Drama,Thriller'
director VARCHAR(255),
cast VARCHAR(500), -- CSV de actores principales
runtime INTEGER, -- en minutos
rating DECIMAL(3, 1), -- TMDb rating 0-10
country VARCHAR(100),
language VARCHAR(50),
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Tabla reviews

```
CREATE TABLE reviews (
id SERIAL PRIMARY KEY,
user_id INTEGER NOT NULL,
movie_id INTEGER NOT NULL,
rating INTEGER CHECK (rating >= 1 AND rating <= 10),
title VARCHAR(255),
text TEXT NOT NULL,
helpful_count INTEGER DEFAULT 0,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
FOREIGN KEY (movie_id) REFERENCES movies(id) ON DELETE CASCADE
);
```

Tabla posts

```
CREATE TABLE posts (
id SERIAL PRIMARY KEY,
user_id INTEGER NOT NULL,
content TEXT NOT NULL,
type ENUM('review', 'list', 'debate', 'recommendation'),
post_image_url VARCHAR(255),
likes_count INTEGER DEFAULT 0,
comments_count INTEGER DEFAULT 0,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
```

Tabla comments

```
CREATE TABLE comments (
id SERIAL PRIMARY KEY,
post_id INTEGER NOT NULL,
user_id INTEGER NOT NULL,
text TEXT NOT NULL,
likes_count INTEGER DEFAULT 0,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
FOREIGN KEY (post_id) REFERENCES posts(id) ON DELETE CASCADE,
FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
```

Tabla likes

```
CREATE TABLE likes (
id SERIAL PRIMARY KEY,
user_id INTEGER NOT NULL,
post_id INTEGER,
review_id INTEGER,
comment_id INTEGER,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
FOREIGN KEY (post_id) REFERENCES posts(id) ON DELETE CASCADE,
FOREIGN KEY (review_id) REFERENCES reviews(id) ON DELETE CASCADE,
FOREIGN KEY (comment_id) REFERENCES comments(id) ON DELETE CASCADE,
CHECK (post_id IS NOT NULL OR review_id IS NOT NULL OR comment_id IS NOT NULL)
);
```

Tabla favorites

```
CREATE TABLE favorites (
id SERIAL PRIMARY KEY,
user_id INTEGER NOT NULL,
movie_id INTEGER NOT NULL,
added_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
UNIQUE(user_id, movie_id),
FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
FOREIGN KEY (movie_id) REFERENCES movies(id) ON DELETE CASCADE
);
```

Tabla reports

```
CREATE TABLE reports (
id SERIAL PRIMARY KEY,
reported_by INTEGER NOT NULL,
post_id INTEGER,
review_id INTEGER,
comment_id INTEGER,
reason VARCHAR(100), -- 'spam', 'inappropriate', 'harassment', etc.
description TEXT,
status ENUM('pending', 'resolved', 'dismissed') DEFAULT 'pending',
resolved_by INTEGER,
resolution_note TEXT,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
resolved_at TIMESTAMP,
FOREIGN KEY (reported_by) REFERENCES users(id),
FOREIGN KEY (resolved_by) REFERENCES users(id),
FOREIGN KEY (post_id) REFERENCES posts(id) ON DELETE SET NULL,
FOREIGN KEY (review_id) REFERENCES reviews(id) ON DELETE SET NULL,
FOREIGN KEY (comment_id) REFERENCES comments(id) ON DELETE SET NULL
);
```

Tabla bans

```
CREATE TABLE bans (
id SERIAL PRIMARY KEY,
user_id INTEGER NOT NULL,
banned_by INTEGER NOT NULL,
reason VARCHAR(255),
is_temporary BOOLEAN DEFAULT TRUE,
banned_until TIMESTAMP,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
FOREIGN KEY (banned_by) REFERENCES users(id)
);
```

Tabla admin_actions

```
CREATE TABLE admin_actions (
id SERIAL PRIMARY KEY,
admin_id INTEGER NOT NULL,
action_type VARCHAR(50), -- 'ban_user', 'unban_user', 'delete_post', etc.
target_user_id INTEGER,
target_post_id INTEGER,
reason VARCHAR(255),
details JSON, -- Detalles adicionales en formato JSON
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
FOREIGN KEY (admin_id) REFERENCES users(id),
FOREIGN KEY (target_user_id) REFERENCES users(id) ON DELETE SET NULL,
FOREIGN KEY (target_post_id) REFERENCES posts(id) ON DELETE SET NULL
);
```

---

7. Planificación y Timeline

Fase 1: MVP (Diciembre - Enero)

Objetivo: Funcionalidad core sin social ni admin avanzado

Hitos:

- ✓ Setup inicial (proyecto React Native, backend Express, DB PostgreSQL)
- ✓ Autenticación (registro, login, JWT)
- ✓ Selección de plataformas
- ✓ Integración TMDb API (búsqueda, metadatos)
- ✓ Home con 3 carruseles básicos
- ✓ Detalle de película/serie
- ✓ Búsqueda simple (título, género)
- ✓ Reseñas básicas (crear, ver)
- ✓ Deploy inicial (backend en Heroku/Railway, DB en cloud)

Entregables: APK funcional, backend en producción, documentación de API

Fase 2: Core Features (Febrero)

Objetivo: Completar funcionalidades de usuario final

Hitos:

- ✓ Sistema de favoritos con sincronización
- ✓ Filtros avanzados completos
- ✓ Integración JustWatch API (disponibilidad por plataforma)
- ✓ Búsqueda por actor, director, año
- ✓ Rankings en tiempo real (Trending, Más vistas)
- ✓ Etiquetas dinámicas (Trending, Popular, Novedad, por plataforma)
- ✓ Badges de medallas (Top 3 podio)
- ✓ Perfil de usuario editable
- ✓ Histórico de actividad del usuario
- ✓ Notificaciones básicas

- ✓ Testing y optimización (performance)

Entregables: Aplicación funcional completa (iOS/Android), documentación de usuario

Deadline crítico: Final de febrero para TFG

Fase 3: Social + Moderación (Marzo)

Objetivo: Comunidad y panel admin

Hitos:

- ✓ Feed social (posts, listas)
- ✓ Sistema de comentarios y likes
- ✓ Reportes de contenido
- ✓ Perfiles públicos
- ✓ Badges de usuario (Crítico activo, etc.)
- ✓ Panel admin: gestión usuarios
- ✓ Baneos y desbaneos (temporal/permanente)
- ✓ Moderación de posts
- ✓ Log auditable de acciones
- ✓ Estadísticas del admin

Entregables: Panel admin funcional, documentación de moderación

---

Fase 4: Pulido Final (Abril)

Objetivo: Producción y entrega

Hitos:

- ✓ Optimización de performance
- ✓ Testing exhaustivo (unitario, integración, E2E)
- ✓ Mejoras UI/UX (feedback usuarios beta)
- ✓ Documentación técnica (API, DB, deployment)
- ✓ Documentación de usuario (FAQs, guías)
- ✓ Security audit
- ✓ Preparación para App Store/Play Store (si aplica)
- ✓ Presentación final TFG

Entregables: Aplicación producción-ready, documentación completa, presentación TFG

---

8. Seguridad y Privacidad

8.1 Autenticación y Autorización

Contraseñas:

- Hash con Bcrypt (salt rounds: 10)
- Validación de requisitos: mín. 8 caracteres, mayúscula, número
- Never stored plain-text

JWT (JSON Web Tokens):

- Access token: 24 horas de validez
- Refresh token: 7 días de validez
- Payload: {userId, role, email, iat, exp}
- Secret key: Variable de entorno

Middleware de autenticación:

```
const verifyToken = (req, res, next) => {
const token = req.headers['authorization']?.split(' ')[1];
if (!token) return res.status(401).json({error: 'No token'});

jwt.verify(token, process.env.JWT_SECRET, (err, user) => {
if (err) return res.status(403).json({error: 'Token inválido'});
req.user = user;
next();
});
};
```

8.2 Control de Acceso Basado en Roles (RBAC)

Permiso	user	moderator	admin
Ver películas	✓	✓	✓
Crear reseñas	✓	✓	✓
Editar perfil	✓	✓	✓
Moderar posts	✗	✓	✓
Ver reportes	✗	✓	✓
Banear usuarios	✗	✗	✓
Ver logs admin	✗	✗	✓

Table 1: Matrix de permisos por rol

8.3 Validación de Inputs

Frontend:

- Validación de tipos de dato
- Sanitización de HTML en formularios
- Máximas longitudes de campo

Backend:

- Validación con librerías como Joi o yup
- Escape de caracteres especiales
- SQL injection prevention (prepared statements)

Ejemplo:

```
const reviewSchema = yup.object({
rating: yup.number().min(1).max(10).required(),
title: yup.string().max(255).required(),
```

```
text: yup.string().min(50).max(5000).required(),
});
```

8.4 Protección de APIs

CORS (Cross-Origin Resource Sharing):

```
app.use(cors({
origin: process.env.FRONTEND_URL,
credentials: true
}));
```

Rate Limiting:

```
const rateLimit = require('express-rate-limit');
const limiter = rateLimit({
windowMs: 15 * 60 * 1000, // 15 minutos
max: 100 // máximo 100 requests por IP
});
app.use(limiter);
```

HTTPS obligatorio en producción

8.5 Privacidad de Datos

- Cumplimiento RGPD (derecho al olvido, exportación de datos)
- Encriptación de datos sensibles en tránsito (TLS 1.3)
- Política de privacidad clara y accesible
- Consentimiento explícito para recopilar datos

8.6 Auditoría y Logs

Eventos registrados:

- Login exitoso/fallido
- Cambios de rol
- Baneos/desbaneos
- Eliminación de posts
- Acciones administrativas

Retención: Mínimo 6 meses de histórico

8.7 Moderación de Contenido

Reportes:

- Usuario reporta post/reseña/comentario
- Moderador revisa (contexto, razón)
- Acción: Aprobar, Rechazar, Eliminar
- Historial auditable

---

9. Métricas de Éxito y KPIs

Para el Proyecto TFG

KPI	Meta	Medición
Usuarios registrados	50+	Al final Fase 1
Películas indexadas	5,000+	TMDb API
Reseñas comunitarias	100+	Fase 2
Funcionalidad core	100%	Checklist de hitos
Performance (Lighthouse)	>80	Auditoría
Cobertura de tests	>70%	Jest/Detox
Documentación	Completa	Readme + Wiki

Para Producción (Futuro)

- **Retención de usuarios:** 30% DAU a 7 días
- **Engagement:** Promedio 3+ reseñas por usuario activo
- **Satisfacción:** NPS >50
- **Uptime:** 99.9%

---

10. Tecnologías de Referencia

Documentación Oficial

1. TMDb API Documentation. https://www.themoviedb.org/documentation/api
2. JustWatch API Documentation. https://apis.justwatch.com/docs/api/
3. React Native Official Guide. https://reactnative.dev/docs/getting-started
4. Node.js Best Practices. https://nodejs.org/en/docs/
5. PostgreSQL Documentation. https://www.postgresql.org/docs/

Tendencias 2025

1. Node.js vs Java vs Python Backend Comparison. https://enstacked.com/node-js-vs-java-vs-python/
2. 5 Stacks Tecnológicos Dominantes 2025. https://w3pds.com/blog/5-stacks-tecnologicos-deminantes-2025/
3. Diseño de Apps Móviles 2025: Tendencias y Mejores Prácticas. https://somospixart.com/tendencias-en-diseno-de-apps-moviles-para-2025-lo-que-tu-negocio-debe-saber/
4. UX/UI Móvil 2025: Nuevas Tendencias. https://www.kando.es/blog/tendencias-en-ux-ui-movil-para-2025
5. JWT en Node.js: Seguridad en APIs. https://www.antino.com/blog/node-js-vs-java

---

Conclusión

Este proyecto TFG representa una solución completa, viable y profesional para un problema real del mercado de streaming. Combina:

- **Utilidad inmediata:** Facilita descubrimiento de contenido personalizado
- **Arquitectura escalable:** Stack moderno (React Native + Node.js + PostgreSQL)
- **Experiencia móvil excepcional:** Diseño atractivo, microinteracciones pulidas, accesibilidad
- **Comunidad y moderación:** Feed social genuino con control administrativo robusto
- **Timeline realista:** Fases claras con **deadline final de febrero para versión core**

El proyecto demuestra competencia completa en full-stack development, UX/UI design, database architecture, y project management—esencial para la evaluación de un TFG de calidad profesional.

¡Éxito en el desarrollo!