



UNIVERSIDAD FRANCISCO DE VITORIA

ESCUELA POLITÉCNICA SUPERIOR

GRADO EN INGENIERÍA INFORMÁTICA

PROYECTO FINAL DE GRADO

MODALIDAD INGENIERÍA

GESTIÓN Y SEGUIMIENTO DE SERIES PARA GRUPOS FAMILIARES

Diego Viñals Lage

Convocatoria de junio 2024

Tutor: Manuel Raposo León

CALIFICACIÓN DEL PROYECTO FINAL DE GRADO

CUALITATIVA:	
NUMÉRICA:	

Conforme Presidente:	Conforme Secretario:
Fdo.:	Fdo.:

Conforme Vocal:	Conforme Vocal:	Conforme Vocal:
Fdo.:	Fdo.:	Fdo.:

Lugar y fecha: Pozuelo de Alarcón, a ____ de _____ de 202__

“Taking on a challenge is a lot like riding a horse. If you're comfortable while you're doing it, you're probably doing it wrong.” – Ted Lasso, Temporada 1, episodio 1, 5:57

*A mi familia, Papá, Mamá, Pablo, Javi y Almu. Por apoyarme y ayudarme en
todo momento.*

A Sofía

Agradecimientos

Aquí se incluirían los agradecimientos personales y profesionales. No olvidarse de agradecer la ayuda recibida, especialmente si se ha realizado el proyecto gracias a una beca, contrato o algún tipo de subvención o apoyo.

Este apartado es opcional. Si no hubiera agradecimientos, debe eliminarse esta sección. Sin embargo, pensad bien pues siempre hay a quien dar las gracias por nuestros logros personales.

Resumen

Este Proyecto Final de Grado aborda el desarrollo de una innovadora aplicación móvil diseñada para mejorar la experiencia de visualización de series en grupos de amigos o familiares. A lo largo del documento, se explora desde la etapa de investigación previa, identificando la necesidad de tal herramienta, hasta las fases de diseño, desarrollo, e implementación de la solución propuesta. Se detalla el plan de trabajo seguido, adoptando una metodología específica que asegura el cumplimiento de objetivos.

También discute el desarrollo técnico del proyecto, incluyendo el análisis de requisitos, el diseño de la interfaz de usuario, la configuración del entorno del servidor, y el despliegue de la aplicación. Paralelamente, el documento refleja mi recorrido académico en la Universidad Francisco de Vitoria, enfatizando cómo el proyecto culmina mi formación en Ingeniería Informática. Finalmente, se consideran las implicaciones éticas y el impacto social del proyecto, subrayando la contribución de esta aplicación a la comunidad de espectadores y su relevancia en el contexto actual de consumo de medios.

Palabras claves

FamilySeriesTrack, Series, Gestión, API, Seguimiento.

Abstract

This Final Degree Project addresses the development of an innovative mobile application designed to enhance the viewing experience of TV series in groups of friends or family. Throughout the document, it explores everything from the preliminary research stage, identifying the need for such a tool, to the design, development, and implementation phases of the proposed solution. The work plan followed is detailed, adopting a specific methodology that ensures the achievement of objectives.

The document also discusses the technical development of the project, including the requirements analysis, the design of the user interface, the setup of the server environment, and the deployment of the application. In parallel, the document reflects my academic journey at the Universidad Francisco de Vitoria, emphasizing how the project culminates my training in Computer Engineering. Finally, the ethical implications and social impact of the project are considered, highlighting the contribution of this application to the viewer community and its relevance in the current context of media consumption.

Keywords

FamilySeriesTrack, Series, Management, API, Tracking.

Índice de Contenidos

1. Introducción.....	1
2. Investigación previa	3
2.1. Filmaffinity	3
2.2. NextEpisode	4
2.3. SeriesGuide	5
2.4. TV Time	6
2.5. Funcionalidades Ausentes en Común.....	7
3. Objetivos.....	9
3.1. Objetivo general	9
3.2. Lista de objetivos específicos	10
3.3. Métodos de Validación	10
4. Plan de Desarrollo del Proyecto	11
4.1. Metodología	11
4.2. Tecnologías	13
4.3. Plan de desarrollo del proyecto.....	14
4.3.1. PT1 – Análisis de Requisitos.....	15
4.3.2. PT2 – Diseño de Interfaz de Usuario.....	16
4.3.3. PT3 – Desarrollo y configuración del Backend	16
4.3.4. PT4 – Desarrollo del Frontend	17
4.3.5. PT5 – Integración de UI con Backend	18
4.3.6. PT6 – Pruebas y Calidad	19
4.3.7. PT7 – Despliegue en Android e iOS.....	19
4.4. Plan de Trabajo	20
4.5. Recursos.....	22
4.5.1. Recursos Técnicos	22
4.5.2. Recursos Humanos	22
4.6. Costes.....	22
4.7. Condicionantes y Limitaciones	23

4.7.1.	Error con la API en iOS	23
4.7.2.	Error Despliegue en Android	24
4.7.3.	Error con tamaño de posters	25
4.7.4.	Notificaciones.....	25
4.7.5.	Autenticación en local.....	26
5.	Desarrollo de la Solución Técnica.....	27
5.1.	PT1 – Análisis de Requisitos	27
5.2.	PT2 – Diseño de Interfaz de Usuario	31
5.3.	PT3 – Desarrollo del Backend.....	32
5.3.1.	MariaDB.....	33
5.3.2.	PhpMyAdmin.....	35
5.3.3.	Tfg_Backend	35
5.3.4.	Cloudflare	37
5.3.5.	Traefik.....	37
5.3.6.	Portainer	38
5.4.	PT4 – Desarrollo Frontend	38
5.4.1.	Estructura de Directorios	38
5.4.2.	Pantallas y Navegación.....	40
5.5.	PT5 – Integración de UI con Backend	41
5.6.	PT6 – Pruebas y Calidad	42
5.7.	PT7 – Despliegue en Android e iOS.....	42
5.7.1.	Build IOs y Android.....	43
5.7.2.	Despliegue en iOS.....	44
5.7.3.	Despliegue en Android.....	45
6.	Resultados.....	47
6.1.	Resultados Obtenidos	47
6.2.	Validacion de los Resultados.....	48
7.	Implicaciones Éticas e Impacto Social	49
7.1.	Introducción	49
7.2.	Desarrollo	49
7.2.1.	Ley de Protección de Datos.....	50

8. Mi Recorrido en la UFV	51
8.1. El PFG como culminación de mi camino universitario	51
8.2. Vinculación con mi futuro profesional	52
9. Conclusiones	53
9.1. Conclusiones	53
9.2. Trabajo a futuro	54
10. Otros Méritos del Proyecto	55
11. Bibliografía	57
Anexo A: Requisitos de Usuario	61
Requisitos Funcionales	61
Requisitos No Funcionales	63
Anexo B: Diseño del Sistema	65
Diagrama	65
Detalles	66
Anexo C: Detalles de la BBDD	67
Usuarios	67
Usuario_Grupo	68
Grupos	68
Series	69
Capítulo	69
Visualizaciones	70
Comentarios Serie	70
Anexo D: Detalles de Pantallas	73
Pantalla Bienvenida	73
Pantalla Inicio Sesión	74
Pantalla Home	75
Pantalla Ajustes	77

Pantalla Crear Grupo.....	78
Pantalla Editar Grupo	79
Pantalla Detalle de Serie	80
Pantalla Comentarios	81
Pantalla Detalle de Temporada.....	83
Pantalla Crear Cuenta	84
Pantalla Calendario	85

Índice de Tablas

<i>Tabla 1: Objetivos Específicos. Elaboración Propia.....</i>	<i>10</i>
<i>Tabla 2: Análisis de Requisitos. Elaboración Propia.....</i>	<i>15</i>
<i>Tabla 3: Diseño IU. Elaboración Propia.....</i>	<i>16</i>
<i>Tabla 4: Configuración del entorno del Servidor. Elaboración Propia</i>	<i>16</i>
<i>Tabla 5: Implementación BBDD. Elaboración Propia.....</i>	<i>17</i>
<i>Tabla 6: Desarrollo de API. Elaboración Propia</i>	<i>17</i>
<i>Tabla 7: Implementación de la Estructura Base Frontend. Elaboración Propia</i>	<i>17</i>
<i>Tabla 8: Desarrollo de Pantallas y Navegación. Elaboración Propia</i>	<i>18</i>
<i>Tabla 9: Integración IU con el Backend. Elaboración Propia</i>	<i>18</i>
<i>Tabla 10: Pruebas y Calidad. Elaboración Propia</i>	<i>19</i>
<i>Tabla 11: Despliegue. Elaboración Propia</i>	<i>19</i>
<i>Tabla 12: Costes.</i>	<i>23</i>
<i>Tabla 13: Costes Recursos Humanos.....</i>	<i>23</i>
<i>Tabla 14: Resultados Obtenidos. Elaboración Propia.</i>	<i>47</i>
<i>Tabla 15: Validación Objetivos Específicos. Elaboración Propia.....</i>	<i>48</i>
<i>Tabla 16: Requisitos Funcionales. Elaboración Propia.....</i>	<i>61</i>
<i>Tabla 17: Requisitos No Funcionales.....</i>	<i>63</i>

Índice de Figuras

<i>Ilustración 1: Metodología en cascada típica.</i>	11
<i>Ilustración 2: Diagrama de Gantt. Elaboración Propia</i>	21
<i>Ilustración 3: Error Gradle. Fuente: EAS Build.</i>	24
<i>Ilustración 4: EAS Build Android correcto. Fuente: EAS</i>	25
<i>Ilustración 5: ¿Ves series regularmente? Elaboración Propia</i>	28
<i>Ilustración 6: ¿En qué plataforma sueles ver las series? Elaboración Propia</i>	28
<i>Ilustración 7: ¿Conoces alguna aplicación para hacer seguimiento de las series? Elaboración Propia</i>	29
<i>Ilustración 8: ¿Te gustaría que hubiera una manera más fácil de gestionar la visualización de series? Elaboración Propia</i>	29
<i>Ilustración 9: Características vs Recuento. Elaboración Propia</i>	30
<i>Ilustración 10: ¿Considerarías cambiar a una aplicación como FST? Elaboración Propia</i>	30
<i>Ilustración 11: Esquema BBDD. Elaboración Propia</i>	34
<i>Ilustración 12: Diagrama de Conexión API. Elaboración Propia</i>	36
<i>Ilustración 13: Estructura Directorios. Elaboración Propia</i>	38
<i>Ilustración 14: EAS Build para iOS. Fuente: EAS</i>	43
<i>Ilustración 15: EAS Build iOS finalizado. Fuente: EAS</i>	43
<i>Ilustración 16: FST en Transporter. Fuente: Transporter App</i>	44
<i>Ilustración 17: FST en Apple Store Connect. Fuente: Apple</i>	44
<i>Ilustración 18: FST en Apple Store. Fuente: Apple Store</i>	45
<i>Ilustración 20: Diseño del Sistema. Elaboración Propia</i>	65
<i>Ilustración 21: Esquema BBDD. Elaboración Propia</i>	67
<i>Ilustración 22: Tabla Usuarios. Elaboración Propia</i>	68
<i>Ilustración 23: Tabla Usuario_Grupo. Elaboración Propia</i>	68
<i>Ilustración 24: Tabla Grupos. Elaboración Propia</i>	69
<i>Ilustración 25: Tabla Series. Elaboración Propia</i>	69
<i>Ilustración 26: Tabla Capítulos. Elaboración Propia</i>	70
<i>Ilustración 27: Tabla Visualizaciones. Elaboración Propia</i>	70
<i>Ilustración 28: Tabla Comentarios Serie. Elaboración propia</i>	71
<i>Ilustración 29: Pantalla Bienvenida. Elaboración Propia</i>	73
<i>Ilustración 30: Pantalla Inicio de Sesión. Elaboración propia</i>	74
<i>Ilustración 31: Pantalla Home. Elaboración Propia</i>	75

<i>Ilustración 32: Pantalla Ajustes. Elaboración Propia.</i>	<i>77</i>
<i>Ilustración 33: Pantalla Crear Grupo. Elaboración Propia.</i>	<i>78</i>
<i>Ilustración 34: Pantalla Editar Grupo. Elaboración Propia.</i>	<i>79</i>
<i>Ilustración 35: Pantalla Detalles Serie. Elaboración Propia.</i>	<i>80</i>
<i>Ilustración 36: Pantalla Comentarios. Elaboración Propia.....</i>	<i>81</i>
<i>Ilustración 37: Pantalla Detalle de Temporada. Elaboración Propia.</i>	<i>83</i>
<i>Ilustración 38: Pantalla Crear Cuenta. Elaboración Propia.</i>	<i>84</i>
<i>Ilustración 39: Pantalla Calendario. Elaboración Propia.....</i>	<i>85</i>

Lista de Acrónimos

Acrónimo	Significado
FST	FamilySeriesTrack
TMDb	The Movie Data Base
BBDD	Base de Datos
IU	Interfaz de Usuario
UX	Experiencia de Usuario
EAS	Expo Application Services
OTA	Over The Air
iOS	iPhone Operator System
IPA	iOS App Store Package
AAB	Android App Bundle
APK	Android Package Kit

1. INTRODUCCIÓN

En la era actual, donde la tecnología y el entretenimiento digital juegan un papel central en nuestras vidas, la experiencia de ver series de televisión se ha transformado en una actividad compartida que trasciende las barreras físicas. Este proyecto de fin de carrera introduce "FamilySeriesTrack", una innovadora aplicación diseñada para dispositivos iPhone y Android que revoluciona la manera en que las familias y amigos siguen y disfrutan sus series de televisión favoritas, manteniendo a todos los usuarios sincronizados, sin importar sus compromisos individuales o ubicaciones geográficas.

La inspiración para este proyecto surge de una experiencia personal en mi hogar, donde el amor compartido por las series de televisión ha sido una constante. Sin embargo, con el crecimiento y la independencia de cada miembro de la familia, incluyendo cambios geográficos como la mudanza de mis padres a Estados Unidos y los diversos compromisos individuales, se ha vuelto un desafío mantenernos al día y sincronizados en nuestras series favoritas. "FamilySeriesTrack" nace de la necesidad de coordinar y enriquecer estas experiencias compartidas, superando los obstáculos de horarios conflictivos y distancias físicas.

La aplicación permite a los usuarios crear grupos familiares o de amigos para compartir las series que están viendo, utilizando una API que ofrece acceso a una amplia gama de información sobre las series, incluyendo títulos, nombres de episodios y descripciones. Una característica clave es la capacidad de marcar episodios como "vistos", generando notificaciones para otros miembros del grupo, facilitando así el seguimiento del progreso de visualización. Además, la aplicación integra un calendario para mostrar fechas de lanzamiento de nuevos episodios y permite la planificación de sesiones de visualización grupal.

Para fomentar la interacción y el debate, "FamilySeriesTrack" incluirá un sistema de calificación y valoración, espacios para comentarios y secciones de discusión abierta. Estas funciones permitirán a los usuarios compartir opiniones y disfrutar de una experiencia comunitaria más rica.

"FamilySeriesTrack" es una aplicación diseñada para ser intuitiva y accesible, disponible en Apple Store y Google Play. Permite a los usuarios registrarse, crear perfiles personales y formar grupos familiares usando nombres de usuario para sincronizar series y episodios vistos.

2. INVESTIGACIÓN PREVIA

En el contexto actual de la era digital, el consumo de series de televisión ha evolucionado hacia una práctica social y culturalmente enriquecedora que trasciende las barreras geográficas. Este cambio ha generado una creciente demanda de herramientas tecnológicas avanzadas que no solo faciliten el seguimiento individual de series, sino que también potencien y gestionen las experiencias compartidas de visualización en comunidades, ya sean familiares o de amigos.

Ante este escenario, se observa una diversificación en el mercado de aplicaciones móviles orientadas a satisfacer estas necesidades, proponiendo soluciones que van más allá del simple seguimiento de episodios. Estas aplicaciones buscan transformar el consumo de series en una experiencia colectiva más rica y conectada, integrando funcionalidades que permiten a los usuarios descubrir nuevos contenidos en función de sus intereses compartidos, recibir alertas sobre estrenos y actualizaciones relevantes, y acceder a plataformas de streaming de manera directa.

No obstante, el desafío radica en diseñar un sistema que consolide efectivamente estas funcionalidades en un entorno interactivo que fomente la comunicación y la interacción social entre sus usuarios. Esto implica no solo la recomendación y discusión de contenidos, sino también la posibilidad de crear espacios virtuales donde grupos de amigos o familiares puedan gestionar colectivamente sus preferencias de visualización, establecer maratones de series, y compartir sus experiencias y opiniones en tiempo real.

El problema a resolver se enfoca en la creación de una plataforma integral que responda a la necesidad de una gestión más social y colaborativa del consumo de series de televisión, proporcionando un espacio donde la tecnología sirva como puente para enriquecer las relaciones y las experiencias compartidas, independientemente de las distancias físicas. Se procederá a examinar uno a uno las posibles soluciones encontradas a este problema.

2.1. FILMAFFINITY

Filmaffinity sobresale en el panorama digital como una plataforma esencial para los amantes del cine y las series, gracias a una serie de características que la hacen particularmente atractiva. Su extenso catálogo, que abarca desde los clásicos del cine hasta las últimas novedades en series, permite a los usuarios explorar una amplia gama de géneros y épocas.

Lo que realmente distingue a Filmaffinity es su enfoque en la personalización: a través de su sistema de recomendaciones, basado en un algoritmo que analiza las calificaciones y preferencias de cada usuario, la plataforma es capaz de sugerir títulos que se ajustan a los gustos individuales, facilitando así el descubrimiento de nuevos contenidos que de otro modo podrían pasar inadvertidos.

Además de las recomendaciones personalizadas, Filmaffinity permite a los usuarios calificar películas y series, escribir reseñas detalladas y crear listas personalizadas, lo que no solo enriquece la experiencia del usuario, sino que también contribuye a una comunidad activa y participativa. Esta capacidad de interactuar con el contenido y con otros usuarios añade una capa de profundidad a la experiencia de navegación, permitiendo a los cinéfilos compartir sus opiniones y descubrir perspectivas diferentes. [1]

La interfaz de usuario de Filmaffinity es otra de sus fortalezas, ofreciendo un diseño intuitivo y fácil de navegar que hace accesible su vasta base de datos a una amplia audiencia, desde espectadores ocasionales hasta aficionados al cine más dedicados. Los usuarios pueden buscar fácilmente información detallada sobre películas y series, incluyendo reparto, director, sinopsis y opiniones de otros usuarios, lo que facilita la toma de decisiones informadas sobre qué ver.

Sin embargo, a pesar de estas características positivas, Filmaffinity presenta ciertas limitaciones, especialmente en lo que respecta a la funcionalidad de grupo. La plataforma se centra en la experiencia individual del usuario, ofreciendo pocas o ninguna opción para la gestión de contenido en un contexto grupal. Esto deja un espacio para innovaciones como FamilySeriesTrack, que busca llenar este vacío ofreciendo funcionalidades específicas diseñadas para mejorar la experiencia de visualización en grupo, permitiendo a los usuarios crear grupos con familiares y amigos, compartir recomendaciones, y organizar sesiones de visionado colectivas.

2.2. NEXTEPISODE

Al profundizar en la funcionalidad de aplicaciones como NextEpisode, que es popular en iOS por su interfaz limpia y soporte para notificaciones push, se observa una tendencia clara hacia la personalización y la facilidad de uso en el ámbito de las aplicaciones para la gestión de series de televisión. NextEpisode, al permitir a los usuarios descubrir nuevas series y recibir recomendaciones personalizadas, ejemplifica cómo la tecnología puede adaptarse a las preferencias individuales y fomentar la exploración de contenidos nuevos y relevantes. [2]

La capacidad de estas aplicaciones para notificar a los usuarios sobre episodios nuevos, cancelaciones de series, o anuncios de fechas de estrenos añade un valor significativo a la experiencia de seguimiento de series. La integración de un calendario que organiza los próximos episodios y muestra una cuenta regresiva para ellos es un ejemplo de cómo estas herramientas buscan simplificar y enriquecer la gestión del tiempo dedicado al entretenimiento.

Además, el aspecto social y comunitario de estas aplicaciones no se puede subestimar. La posibilidad de descubrir nuevos shows basados en popularidad o próximos estrenos, junto con recibir recomendaciones personalizadas basadas en los gustos del usuario, no solo mejora la experiencia individual, sino que también promueve la interacción dentro de la comunidad de usuarios. Este enfoque en la personalización y la comunidad refleja un entendimiento profundo de los hábitos de consumo de entretenimiento actuales, donde las recomendaciones y las opiniones de otros espectadores juegan un papel crucial en la decisión de qué serie ver a continuación.

Al analizar estas características, es evidente que el diseño y desarrollo de una aplicación móvil para la gestión de series de televisión en grupos familiares o de amigos debe centrarse en la facilidad de uso, la personalización y la incorporación de elementos sociales. La integración eficiente de estas funcionalidades puede crear una plataforma que no solo sirva para hacer seguimiento de las series favoritas, sino que también fomente la conexión y el intercambio entre usuarios con intereses similares, enriqueciendo así la experiencia colectiva de ver televisión.

Aunque NextEpisode se destaca por su interfaz limpia, soporte para notificaciones push y funcionalidades que facilitan el seguimiento individual de series, incluyendo un calendario de próximos episodios y recomendaciones personalizadas, su enfoque se centra primordialmente en la gestión individual de usuario. Esta orientación hacia el usuario único limita las oportunidades de interacción y gestión colectiva, un aspecto fundamental para grupos de amigos o familiares que desean compartir su pasión por las series de manera conjunta.

2.3. SERIESGUIDE

SeriesGuide se presenta como una solución destacada en el ámbito de las aplicaciones destinadas al seguimiento de series de televisión, especialmente valorada por aquellos usuarios que prefieren una experiencia libre de anuncios y sin costes. Al ser un proyecto de código abierto, SeriesGuide se beneficia de la contribución continua de una comunidad activa de desarrolladores y usuarios, lo que permite una evolución constante de la aplicación basada en feedback real y necesidades específicas del público. [3]

Esta orientación hacia la comunidad no solo asegura que la aplicación se mantenga actualizada con las últimas funcionalidades y correcciones, sino que también fomenta un entorno de colaboración en el que cualquier persona con los conocimientos técnicos adecuados puede contribuir al desarrollo y mejora de la plataforma. Esta apertura es especialmente atractiva para los entusiastas de la tecnología y el software libre, quienes valoran la transparencia y la capacidad de personalizar o ajustar la aplicación según sus preferencias personales.

Aunque el diseño de SeriesGuide puede no competir con la estética moderna de otras aplicaciones más recientes, su funcionalidad sólida y dedicación a una experiencia de usuario sin interrupciones compensan ampliamente cualquier posible deficiencia visual. La aplicación permite a los usuarios seguir sus series favoritas, marcar episodios vistos, recibir

notificaciones de nuevos lanzamientos y descubrir nuevos contenidos basados en sus gustos y hábitos de visionado.

La gratuidad de SeriesGuide, combinada con su política de no incluir anuncios, ofrece una experiencia de usuario centrada y respetuosa, distinguiéndola dentro de un mercado a menudo saturado de opciones freemium que recurren a la publicidad como modelo de ingresos. Esta característica es particularmente apreciada en un contexto donde la experiencia del usuario puede verse fácilmente comprometida por interrupciones y distracciones no deseadas.

Aunque SeriesGuide ofrece notables ventajas como su naturaleza de código abierto, ausencia de coste y de publicidad, se encuentra con una limitación importante: su exclusividad para usuarios de Android. Esta restricción deja fuera a un amplio sector de usuarios potenciales que utilizan iOS, limitando así su alcance en un mercado tecnológico que valora cada vez más la compatibilidad y accesibilidad entre diferentes plataformas. En un contexto donde la flexibilidad y la capacidad de adaptarse a diversos sistemas operativos son esenciales, la disponibilidad limitada de SeriesGuide podría considerarse una barrera significativa para su adopción más generalizada.

Por otro lado, FamilySeriesTrack se presenta como una propuesta más inclusiva, al estar desarrollada tanto para Android como para iOS. Esta estrategia no solo demuestra un compromiso con la universalidad y la accesibilidad, sino que también amplía su potencial de mercado al ser accesible para una audiencia más diversa.

FamilySeriesTrack busca ofrecer una experiencia de usuario uniforme y enriquecedora, sin importar el dispositivo móvil que se utilice, lo cual es un paso adelante hacia la integración y el disfrute compartido de contenidos televisivos entre grupos de amigos y familias, superando las barreras impuestas por las diferencias en los sistemas operativos. Este enfoque hacia la inclusión y la interoperabilidad posiciona a FamilySeriesTrack como una alternativa prometedora en el panorama de aplicaciones para el seguimiento de series, potenciando la conexión entre personas a través de sus intereses compartidos en el entretenimiento televisivo.

2.4. TV TIME

TV Time se ha establecido como una herramienta indispensable para los aficionados a las series, gracias a su amplia gama de funcionalidades diseñadas para mejorar la experiencia de seguimiento de series. Con TV Time, no solo puedes llevar un registro detallado de las series que estás viendo, sino que también ofrece la posibilidad de sumergirse en una comunidad activa de espectadores con intereses similares. Esta función de comunidad permite a los usuarios comentar episodios, compartir sus opiniones y teorías, e incluso descubrir nuevas perspectivas sobre sus series favoritas. [4]

Además, TV Time se destaca por su sistema de recomendaciones personalizadas. Basándose en tu historial de visionado y en las interacciones dentro de la aplicación, TV Time es capaz de sugerir nuevas series que se alinean con tus gustos y preferencias previas. Esta

característica es especialmente valiosa en un mercado saturado de opciones, donde descubrir contenido relevante y de calidad puede ser abrumador.

Otra de las funcionalidades apreciadas en TV Time es su capacidad para mantener a los usuarios informados sobre los lanzamientos de nuevos episodios. Al añadir series a tu lista, la aplicación automáticamente te notifica cuando un nuevo episodio está disponible, asegurando que nunca te pierdas un estreno. Esta función es particularmente útil en el contexto actual, donde el modelo de lanzamiento de episodios puede variar significativamente entre diferentes plataformas de streaming.

La interfaz de usuario de TV Time también merece reconocimiento. Intuitiva y fácil de navegar, permite a los usuarios gestionar sus series con facilidad, marcar episodios como vistos, y explorar el extenso catálogo de series disponibles. La aplicación ha logrado encontrar un equilibrio entre ofrecer una rica funcionalidad y mantener una experiencia de usuario simplificada, lo que la convierte en una opción accesible tanto para espectadores casuales como para entusiastas de las series más dedicados.

Aunque TV Time es reconocida por sus completas funcionalidades para el seguimiento de series, incluyendo la interacción con una comunidad de aficionados y la recepción de recomendaciones personalizadas, una de sus limitaciones reside en su enfoque predominantemente individual en cuanto al seguimiento y gestión de series. La plataforma, aunque rica en características sociales y de comunidad, no ofrece herramientas específicas diseñadas para el manejo de perfiles múltiples o la gestión de grupos familiares o de amigos que deseen compartir de manera conjunta sus experiencias de visionado.

En contraste, FamilySeriesTrack propone una solución directa a esta necesidad, al incorporar explícitamente la gestión de grupos en su diseño. Al permitir la creación de grupos familiares o de amigos dentro de la aplicación, FamilySeriesTrack facilita la organización de sesiones de visionado colectivas, el intercambio de recomendaciones y la discusión de episodios, todo ello en un entorno privado y exclusivo para cada grupo. Esta característica responde a la tendencia creciente de compartir experiencias de entretenimiento en entornos digitales, potenciando la conexión entre sus usuarios más allá de la simple recomendación o comentario en una comunidad abierta.

2.5. FUNCIONALIDADES AUSENTES EN COMÚN

Una característica comúnmente ausente en las aplicaciones populares de seguimiento de series es la implementación de funcionalidades orientadas a grupos. La mayoría de estas plataformas están diseñadas con un enfoque individual, permitiendo a los usuarios gestionar sus propios seguimientos de series, marcar episodios vistos, y recibir recomendaciones personalizadas. Sin embargo, esta aproximación individualista omite una dimensión crucial del consumo de medios en la actualidad: la experiencia compartida. En un mundo cada vez más conectado, donde ver series se ha convertido en una actividad social y un punto de encuentro para familias y amigos, la falta de herramientas para gestionar y compartir estas experiencias en grupo representa una notable limitación. Este vacío funcional deja de lado las oportunidades para fortalecer vínculos mediante el disfrute colectivo de contenidos,

subrayando un área de mejora y diferenciación para nuevas aplicaciones que busquen cubrir esta necesidad.

En este contexto, FamilySeriesTrack se posicionaría como una solución innovadora y destacada frente al resto de aplicaciones de seguimiento de series al implementar la funcionalidad en grupo. Esta característica única aborda directamente la necesidad de compartir y disfrutar de las series de televisión en compañía, permitiendo a los usuarios crear grupos con familiares y amigos para discutir episodios, compartir recomendaciones y planificar sesiones de visionado colectivas.

Al ofrecer un espacio dedicado para la gestión y disfrute compartido, FamilySeriesTrack no solo enriquece la experiencia de ver series, sino que también fomenta la interacción y el fortalecimiento de las relaciones personales a través del entretenimiento compartido. Esta funcionalidad distingue a FamilySeriesTrack de otras aplicaciones en el mercado, presentándola como una alternativa atractiva para aquellos que buscan una experiencia más social y conectada en el mundo del entretenimiento televisivo.

En conclusión, a pesar de las diversas funcionalidades que ofrecen las aplicaciones existentes para el seguimiento de series, ninguna logra abordar de manera efectiva el desafío principal de gestionar y sincronizar el avance de visionado dentro de un grupo familiar. Esta carencia subraya la necesidad de una solución que permita a los miembros de un grupo familiar o de amigos mantener un control colectivo sobre el progreso de cada uno en las series que ven juntos, facilitando una experiencia compartida más cohesiva y organizada. Esta solución sería FamilySeriesTrack.

3. OBJETIVOS

En cualquier proyecto, la claridad y precisión de los objetivos son cruciales para su éxito. Este apartado se dedica a establecer y detallar los objetivos que guiarán el desarrollo y la implementación del proyecto. Se divide en tres secciones esenciales, cada una abordando un aspecto diferente pero complementario de las metas.

En el objetivo general se presenta la visión amplia y el propósito fundamental del proyecto. Este objetivo encapsula la razón de ser del proyecto y establece el alcance general de lo que se busca lograr. Es una declaración que responde al 'para qué' del proyecto, ofreciendo una perspectiva global de las intenciones.

En los objetivos específicos se desglosa el objetivo general en componentes más pequeños y medibles. Cada objetivo específico es un paso concreto hacia la realización del objetivo general, proporcionando claridad y dirección en el proceso de desarrollo. Estos objetivos son peldaños esenciales que permiten medir el progreso y asegurar que cada aspecto del objetivo general se aborde eficazmente.

Finalmente, se aborda cómo se va a verificar y evaluar el éxito en el logro de los objetivos. Esta sección describe las técnicas y procedimientos que se utilizarán para asegurar que los objetivos, tanto generales como específicos, se cumplan de manera satisfactoria. Los métodos de validación son herramientas cruciales para la evaluación continua del proyecto, permitiendo ajustar y afinar las estrategias a medida que se avanza.

3.1. OBJETIVO GENERAL

El objetivo general del proyecto "FamilySeriesTrack" es facilitar el seguimiento conjunto de series de televisión favoritas para grupos de usuarios como familias o amigos.

La aplicación que se quiere desarrollar tiene como finalidad simplificar la gestión de las series que disfrutan los miembros del grupo, asegurando que todos estén sincronizados en cuanto a los episodios vistos. Esto incluye características como la creación de grupos familiares y notificaciones de episodios vistos, mejorando así la coordinación y el disfrute compartido de series de televisión.

3.2. LISTA DE OBJETIVOS ESPECÍFICOS

Tabla 1: Objetivos Específicos. Elaboración Propia.

Código	Objetivos
OB-01	Facilitar la coordinación de visualización en grupos: Mejora la experiencia individual y comunitaria de ver series de televisión, permitiendo una gestión personalizada y colectiva eficiente y enriquecedora.
OB-02	Mejorar la toma de decisiones colectivas sobre qué ver: Mejora eficientemente la coordinación entre usuarios de un mismo grupo, resolviendo el problema común de seguir el progreso de cada miembro en sus series. Esto previene confusiones y spoilers indeseados. Además, simplifica la elección de series para ver en grupo, ofreciendo opciones basadas en el progreso colectivo de visualización, lo que ahorra tiempo en debates y búsquedas.
OB-03	Enriquecer la experiencia compartida de visualización: Enriquece significativamente la experiencia de ver series de televisión. Ya no se limita a ver un episodio; ahora, se transforma en una experiencia interactiva donde se puede compartir opiniones y puntos de vista con el resto del grupo.
OB-04	Proporcionar una interfaz intuitiva y accesible: Garantiza que la aplicación sea accesible para usuarios de todas las edades y niveles de habilidad tecnológica, mejorando la experiencia de usuario general.
OB-05	Estadísticas de visualización: Añadir una pestaña de estadísticas donde los usuarios del grupo familiar puedan ver el número de capítulos o series que han visto a lo largo de los años, contribuyendo a hacer de FamilySeriesTrack una herramienta más valiosa para los amantes de las series.

3.3. MÉTODOS DE VALIDACIÓN

- ✓ Pruebas Funcionales: Verificar que cada función de la aplicación (como la creación de perfiles de usuario, sincronización de series vistas, notificaciones, etc.) funciona según lo previsto. Se valida el OB-01 y OB-02.
- ✓ Pruebas de Usabilidad: Evaluar la facilidad de uso de la aplicación con usuarios reales para asegurarse de que la interfaz es intuitiva y amigable. Se valida el OB-04.
- ✓ Feedback de los Usuarios: Obtener retroalimentación de los usuarios beta o de un grupo de prueba para ajustar y mejorar la aplicación antes de su lanzamiento final. Se valida el OB-01, OB-02, OB-03 y OB-04.
- ✓ Revisiones de Cumplimiento de Requisitos: Comparar las características y funcionalidades de la aplicación con los objetivos específicos del proyecto para asegurarse de que se han cumplido todos los requisitos. Se valida el OB-05.
- ✓ Pruebas de Integración y Continuidad: Asegurar que todos los componentes de la aplicación (como la base de datos, la API de series, el sistema de notificaciones) trabajen juntos de manera fluida y sin errores. Se valida el OB-05.

4. PLAN DE DESARROLLO DEL PROYECTO

4.1. METODOLOGÍA

La metodología en cascada, que se implementará en este proyecto, es un enfoque tradicional en la gestión de proyectos de software, conocido por su estructura lineal y rigurosa. Este método se distingue por su secuencia de fases claramente definidas, cada una de las cuales debe completarse antes de pasar a la siguiente. Es una metodología ideal para proyectos donde los requisitos son claros desde el inicio y es poco probable que cambien significativamente durante el desarrollo. [5]

La metodología en cascada típicamente sigue estas fases:

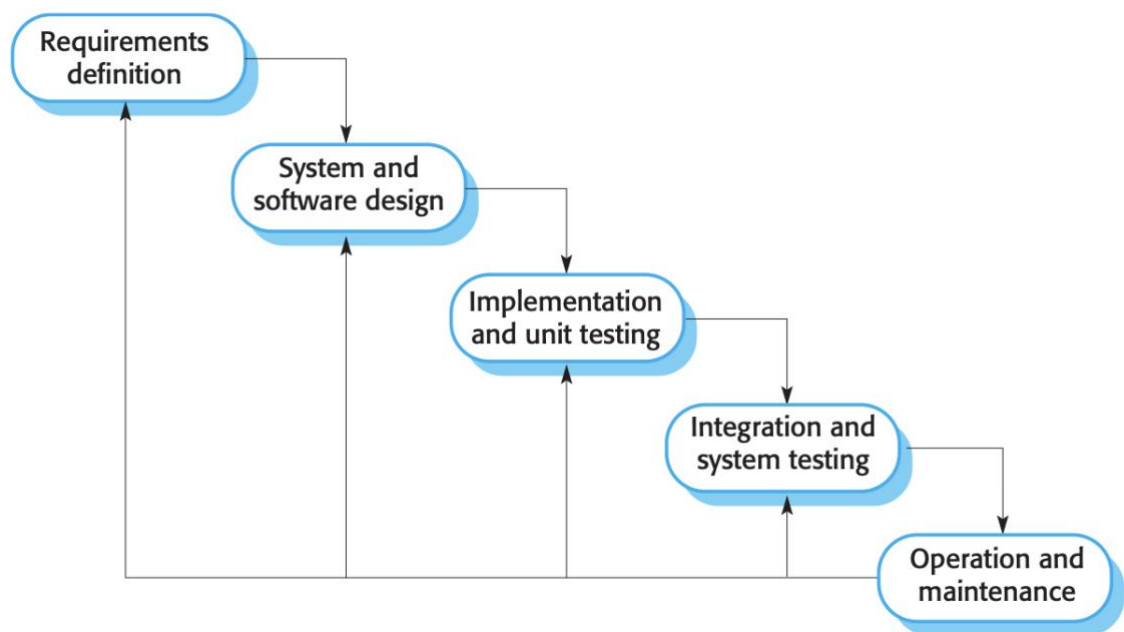


Ilustración 1: Metodología en cascada típica.

El proceso de desarrollo del proyecto comienza con el Análisis de Requisitos, una fase inicial crítica en la que se realiza una recolección y análisis detallados de los requisitos del proyecto. Esta etapa es esencial para comprender por completo lo que se necesita antes de proceder al diseño. La claridad y precisión en esta fase son fundamentales para el éxito de todo el

proyecto, ya que cualquier error o malentendido en esta etapa puede tener consecuencias significativas en las fases posteriores.

A continuación, se procede a la fase de Diseño del Sistema y Software, en la que, basándose en los requisitos recopilados, se planifica la arquitectura del sistema. Durante esta etapa, se toman decisiones críticas sobre cómo funcionará el software y cómo se organizarán sus distintos componentes. Esta fase es crucial para sentar las bases de la estructura y funcionamiento del software, estableciendo un esquema claro y coherente que guiará la implementación.

La fase de Implementación y Codificación es el momento en el que se escribe el código real del software, basándose en el diseño previamente establecido. Esta es una fase de construcción intensa, donde el producto funcional comienza a tomar forma. Durante esta etapa, se convierten los planes y diseños en un software tangible y operativo, prestando especial atención a la adherencia al diseño y a los requisitos definidos.

Una vez desarrollado el software, este entra en la fase de Pruebas del Sistema. En esta etapa, el software se somete a pruebas rigurosas para identificar y corregir cualquier error o problema. El objetivo principal es asegurar que el software funcione exactamente según los requisitos definidos, cumpliendo con todas las expectativas y necesidades del usuario final. Esta fase es crucial para garantizar la calidad y la fiabilidad del software antes de su despliegue.

Finalmente, el proyecto entra en la fase de Despliegue. En esta última etapa, el software se pone en funcionamiento y se hace accesible para los usuarios. Tras el despliegue, se lleva a cabo un mantenimiento continuo para resolver problemas operativos que puedan surgir y para realizar actualizaciones necesarias. Esta fase es vital para garantizar que el software siga siendo funcional y relevante a lo largo del tiempo, adaptándose a las necesidades cambiantes y a las nuevas condiciones del entorno.

Durante cada fase, se centrará exclusivamente en las tareas pertinentes a esa etapa, completándolas antes de pasar a la siguiente. Esta aproximación secuencial permitirá enfocarse en aspectos específicos del proyecto sin la distracción de tareas simultáneas, asegurando así una mayor atención al detalle y calidad en cada paso.

Al final de cada fase, se realizará una revisión y evaluación detallada del trabajo realizado. Esto incluirá verificar que los objetivos de esa fase se hayan cumplido satisfactoriamente antes de proceder a la siguiente. Esta evaluación será fundamental para asegurar que cada aspecto del proyecto cumpla con los estándares establecidos, permitiendo ajustes y correcciones antes de avanzar.

Adoptando la metodología en cascada en un contexto individual, se podrá beneficiar de su estructura definida y su enfoque en la finalización completa de cada fase antes de avanzar. Esto resultará en una gestión del proyecto más metódica y controlada, lo cual es especialmente valioso en un entorno unipersonal donde la claridad y la organización son claves para el éxito del proyecto.

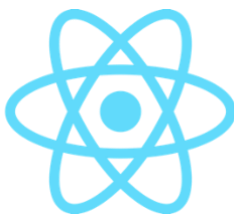
4.2. TECNOLOGÍAS

Las tecnologías utilizadas en este proyecto se pueden clasificar en dos categorías principales: frontend y backend. El frontend se refiere a la parte de la aplicación con la que interactúa el usuario, incluyendo la interfaz de usuario y la experiencia de usuario.

Por otro lado, el backend se ocupa de la lógica de la aplicación, el manejo de bases de datos, la autenticación de usuarios y la integración de APIs. El backend es crucial para el procesamiento de datos, la seguridad de la aplicación y la garantía de que las solicitudes del usuario se manejen de manera eficiente.



Figma: Figma es una herramienta avanzada de diseño de interfaz de usuario (UI) y experiencia de usuario (UX) que funciona en la nube, proporcionando una plataforma versátil para diseñadores y equipos de desarrollo. Su capacidad para facilitar la creación, el prototipado y la colaboración en proyectos de diseño gráfico y de interfaz la hace indispensable en el proceso de desarrollo de aplicaciones y sitios web modernos. [6] Su uso se ha vuelto cada vez más común en las etapas iniciales del desarrollo de software, particularmente en la conceptualización y diseño de interfaces de usuario intuitivas y atractivas.



React Native: Es un framework de código abierto para crear aplicaciones móviles nativas usando JavaScript y React. permite a los desarrolladores escribir código en JavaScript y renderizarlo con componentes nativos de iOS y Android, lo que significa que una sola base de código puede funcionar en ambas plataformas, por lo que con un solo código se puede desarrollar la misma aplicación para ambas plataformas. [7]



Expo: Es una herramienta y una plataforma para aplicaciones universales de React Native. Proporciona un conjunto de herramientas y servicios para facilitar el desarrollo y la implementación de aplicaciones React Native. Expo ofrece características como la actualización en vivo del código, lo que permite a los desarrolladores enviar actualizaciones directamente a los usuarios sin pasar por el proceso de publicación de la tienda de aplicaciones, se puede ver cómo se comporta la aplicación en un simulador en el propio ordenador. [8]

Para el desarrollo del backend se ha decidido utilizar MariaDB y phpMyAdmin en un servidor en casa con Docker. Vamos a desglosar y explicar cada una de estas herramientas.



Docker: Es una plataforma de contenedores que permite empaquetar aplicaciones y sus dependencias en un contenedor virtualizado que puede ejecutarse en cualquier máquina Linux, Windows o MacOS. Esto asegura que la aplicación se ejecute de manera idéntica en cualquier entorno. [9]



MariaDB: Es un sistema de gestión de bases de datos relacional, por lo que está diseñado para ser compatible con MySQL, lo que significa que los comandos, interfaces, bibliotecas y API que funcionan en MySQL también deberían funcionar en MariaDB. Se necesita una base de datos robusta y confiable para almacenar y gestionar datos y esta es una de las mejores opciones. [10]



PhpMyAdmin: Es una herramienta de software libre diseñada para administrar MySQL o MariaDB a través de una interfaz web. Permite gestionar fácilmente la base de datos MariaDB. La interfaz gráfica de usuario es útil para visualizar y editar datos, realizar el mantenimiento de la base de datos y desarrollar SQL sin la necesidad de utilizar la línea de comandos. [11]



The Movie Database: La API de TMDb proporciona acceso programático a su vasta colección de datos cinematográficos y televisivos. Permite a los desarrolladores buscar y recuperar información de su base de datos, incluyendo, pero no limitado a detalles de películas y programas de TV, información de elenco y equipo, imágenes, videos, clasificaciones de usuario y mucho más. [12]



Express: Se trata de un framework para Node.js diseñado para facilitar la creación de aplicaciones web y API de manera rápida y sencilla. Será de utilidad para la API ya que hará de conexión entre la base de datos y la aplicación. [13]



Transporter: Transporter es una aplicación de Apple diseñada para simplificar el proceso de subir aplicaciones al App Store Connect. Facilita a los desarrolladores y productores de contenido la entrega de sus aplicaciones, actualizaciones de aplicaciones, metadatos y capturas de pantalla a Apple. [14]

4.3. PLAN DE DESARROLLO DEL PROYECTO

El enfoque adoptado para la gestión y desarrollo de "FamilySeriesTrack" es uno que asegura meticulosidad y estructura a lo largo del ciclo de vida del proyecto. Para lograr este fin, el proceso de desarrollo se ha organizado en paquetes de trabajo claramente definidos. Cada paquete de trabajo está designado con un código único, que actúa como referencia y facilita

la trazabilidad y la gestión del proyecto. Además, se proporciona una descripción detallada para cada paquete, delineando el alcance y los objetivos específicos a alcanzar.

Para cada paquete de trabajo, se han identificado las entradas necesarias; estas son los recursos, la información y los documentos previos que deben ser procesados o utilizados en la ejecución de las tareas. Asimismo, se han definido las salidas esperadas, siendo estas los entregables, resultados o productos que se generan tras la realización de las actividades de cada paquete.

Dentro de cada paquete de trabajo, se detallan las actividades específicas a llevar a cabo, proporcionando un marco de acción claro para el equipo de desarrollo y asegurando que todos los esfuerzos están alineados con los objetivos del proyecto. Estas actividades están diseñadas para avanzar sistemáticamente hacia la conclusión de cada etapa del desarrollo, con hitos claros y verificables.

En secciones subsiguientes de este documento, se mostrará cómo se ha llevado a cabo el desarrollo de estos paquetes de trabajo.

4.3.1. PT1 – Análisis de Requisitos

El análisis de requisitos es un paso crucial al inicio de cualquier proyecto, enfocado en entender y definir claramente las necesidades y características que el producto o sistema debe cumplir. Este proceso implica recoger y documentar información sobre las expectativas y requerimientos de clientes, usuarios finales y otros stakeholders, incluyendo funciones, características, limitaciones y especificaciones necesarias para el éxito del proyecto. Su objetivo principal es obtener una comprensión detallada de los requisitos funcionales y no funcionales del sistema a desarrollar.

Para lograr este objetivo, se diseñará y distribuirá una encuesta dirigida al público general. Esta estrategia nos permitirá comprender directamente qué es lo que desean y necesitan los potenciales usuarios, facilitando la recopilación de los requisitos fundamentales para el proyecto. La encuesta será elaborada con preguntas claras y concisas, abarcando aspectos esenciales como funcionalidades deseadas, preferencias de interfaz, expectativas de rendimiento y preocupaciones de seguridad.

Tabla 2: Análisis de Requisitos. Elaboración Propia

Nombre	Recopilación y Análisis de Requisitos
Código	PT01-AR
Descripción	Realizar un análisis detallado de los requisitos necesarios para el desarrollo de la aplicación "FamilySeriesTrack", incluyendo la recopilación de necesidades de los usuarios finales y la definición de la funcionalidad esencial de la aplicación.
Entradas	Encuestas y Feedback de usuarios potenciales.
Salidas	Requisitos funcionales y no funcionales.
Actividades	<ul style="list-style-type: none">✓ Realizar encuestas a grupos focales.✓ Investigación Previa

4.3.2. PT2 – Diseño de Interfaz de Usuario

Este componente esencial del proyecto se enfoca en la tarea de diseñar una interfaz que no solo sea estéticamente agradable sino también intuitiva, facilitando así una experiencia de usuario excepcional que supere las expectativas previstas. A lo largo de este proceso, se establecerán parámetros definidos por especificaciones de requisitos detalladas y directrices de diseño de UI/UX, que servirán como punto de partida para el desarrollo de mockups y prototipos interactivos de alta fidelidad en Figma.

Tabla 3: Diseño IU. Elaboración Propia

Nombre	Diseño de UI/UX
Código	PT02-DIU
Descripción	Diseñar la interfaz de usuario y la experiencia de usuario para "FamilySeriesTrack", garantizando una navegación intuitiva y una estética atractiva que cumpla con las expectativas de los usuarios.
Entradas	<ul style="list-style-type: none">✓ Especificaciones de requisitos del paquete PT01-AR.✓ Directrices de diseño y estándares de UI/UX.
Salidas	Mockups y prototipos de alta fidelidad en Figma.
Actividades	<ul style="list-style-type: none">✓ Crear wireframes y prototipos interactivos.✓ Realizar pruebas de usabilidad y ajustar el diseño según los resultados.

4.3.3. PT3 – Desarrollo y configuración del Backend

Este componente del proyecto se enfoca en el desarrollo y configuración del backend, un pilar esencial para sustentar la operatividad y la lógica de nuestra aplicación. Durante esta fase, se procederá a configurar el servidor y a establecer una base de datos robusta y segura, creando así una infraestructura sólida que soporte eficazmente las necesidades de nuestro sistema. Adicionalmente, se desarrollará una API personalizada, diseñada para facilitar una comunicación fluida y efectiva entre el frontend y el backend.

Configuración del Servidor

Tabla 4: Configuración del entorno del Servidor. Elaboración Propia

Nombre	Configuración del Servidor
Código	PT03.1-BCK
Descripción	Preparar y configurar el entorno de servidor para "FamilySeriesTrack", garantizando que la infraestructura esté lista para el desarrollo y despliegue de la aplicación.
Entradas	<ul style="list-style-type: none">✓ Documento de arquitectura del sistema.✓ Especificaciones del servidor y requisitos de hardware.
Salidas	<ul style="list-style-type: none">✓ Servidor en la nube configurado.✓ Informe de configuración del servidor.✓ Archivo Docker-Compose
Actividades	<ul style="list-style-type: none">✓ Configurar Docker✓ Configurar conexión al servidor

Configuración de la Base de Datos

Tabla 5: Implementación BBDD. Elaboración Propia

Nombre	Configuración de la Base de Datos
Código	PT03.2-BCK
Descripción	Diseñar y configurar la base de datos para "FamilySeriesTrack", asegurando un esquema optimizado para el rendimiento y la integridad de los datos.
Entradas	Requisitos de almacenamiento de datos.
Salidas	<ul style="list-style-type: none">✓ Base de datos configurada y optimizada.✓ Documentación del esquema de la base de datos.
Actividades	<ul style="list-style-type: none">✓ Diseñar el esquema de la base de datos.✓ Configurar la base de datos en el servidor.✓ Realizar la optimización y ajustes de rendimiento.

Desarrollo de API para conexión con BBDD

Tabla 6: Desarrollo de API. Elaboración Propia

Nombre	Desarrollo de API para conexión con BBDD
Código	PT03.3-BCK
Descripción	Crear y documentar la API necesaria para la gestión de usuarios, y todos los elementos de la BBDD.
Entradas	Requisitos de la API
Salidas	<ul style="list-style-type: none">✓ API para la gestión de usuarios.✓ Documentación de la API.
Actividades	Diseñar Endpoints de la API.

4.3.4. PT4 – Desarrollo del Frontend

Esta fase del proyecto se centra en el desarrollo del frontend y la creación de las interfaces de usuario, constituyendo la cara visible de nuestra aplicación.

Estructura Base del Frontend

Tabla 7: Implementación de la Estructura Base Frontend. Elaboración Propia

Nombre	Estructura Base del Frontend
Código	PT04.1-FRNT
Descripción	Establecer la estructura base y el framework del frontend para "FamilySeriesTrack", definiendo la arquitectura de carpetas y configuraciones iniciales del proyecto.
Entradas	<ul style="list-style-type: none">✓ Guía de estilo y requisitos de UI/UX.✓ Documento de arquitectura del sistema.

Nombre	Estructura Base del Frontend
Salidas	<ul style="list-style-type: none"> ✓ Estructura base del proyecto frontend. ✓ Configuraciones iniciales implementadas.
Actividades	<ul style="list-style-type: none"> ✓ Definir la arquitectura de carpetas del proyecto. ✓ Configurar el entorno de desarrollo frontend. ✓ Instalar librerías y dependencias necesarias.

Pantallas y Flujo de Navegación

Tabla 8: Desarrollo de Pantallas y Navegación. Elaboración Propia

Nombre	Pantallas y Flujo de Navegación
Código	PT04.2-FRNT
Descripción	Implementar las pantallas detalladas en los prototipos de UI/UX y desarrollar el flujo de navegación para la aplicación "FamilySeriesTrack".
Entradas	Prototipos de UI del paquete PT02-DIU
Salidas	<ul style="list-style-type: none"> ✓ Pantallas de la aplicación desarrolladas. ✓ Navegación entre pantallas implementada.
Actividades	<ul style="list-style-type: none"> ✓ Desarrollar las pantallas de la aplicación según los prototipos. ✓ Configurar el sistema de rutas y navegación. ✓ Probar la usabilidad y coherencia de la navegación.

4.3.5. PT5 – Integración de UI con Backend

Tabla 9: Integración IU con el Backend. Elaboración Propia

Nombre	Integración de UI con Backend
Código	PT05-INT
Descripción	Conectar los componentes de la interfaz de usuario con el backend de "FamilySeriesTrack", asegurando una interacción fluida y funcional entre el frontend y el backend.
Entradas	<ul style="list-style-type: none"> ✓ API endpoints del paquete PT03.3-BCK. ✓ Código fuente de componentes de UI del paquete PT04.2-FRNT
Salidas	<ul style="list-style-type: none"> ✓ Frontend integrado con servicios backend. ✓ Aplicación Completa
Actividades	<ul style="list-style-type: none"> ✓ Codificar la lógica de conexión con el backend (peticiones HTTP, manejo de estados). ✓ Realizar pruebas de integración para asegurar la comunicación efectiva entre frontend y backend. ✓ Refinar la interacción usuario-frontend basado en Feedback de pruebas.

4.3.6. PT6 – Pruebas y Calidad

Esta etapa es fundamental para validar la funcionalidad, el rendimiento y la seguridad de la aplicación, alineándola con los más altos estándares de calidad establecidos. Utilizaremos la aplicación completa del paquete PT04-FRNT como entrada y llevaremos a cabo una serie de casos de prueba detallados y criterios de aceptación que fueron desarrollados meticulosamente durante la fase de análisis. Las salidas esperadas de este proceso son un reporte de pruebas minucioso, una lista de problemas identificados y tickets de seguimiento para cada defecto encontrado, garantizando así una mejora continua. Las actividades clave incluirán el desarrollo, la ejecución de casos de prueba y un análisis profundo de los resultados para documentar cualquier incongruencia, asegurando que el producto final sea de la más alta calidad antes de su lanzamiento.

Tabla 10: Pruebas y Calidad. Elaboración Propia

Nombre	Pruebas y Aseguramiento de la Calidad
Código	PT06-PC
Descripción	Realizar pruebas integrales para validar la funcionalidad, el rendimiento y la seguridad de la aplicación "FamilySeriesTrack", asegurando que se cumplan todos los requisitos y estándares de calidad.
Entradas	<ul style="list-style-type: none">✓ Aplicación completa del paquete PT05-INT✓ Casos de prueba y criterios de aceptación desarrollados durante la fase de análisis.
Salidas	<ul style="list-style-type: none">✓ Reporte detallado de pruebas, incluyendo resultados de pruebas de funcionalidad, rendimiento y seguridad.✓ Lista de problemas identificados y tickets de seguimiento para su corrección.
Actividades	<ul style="list-style-type: none">✓ Desarrollo y ejecución de casos de prueba.✓ Análisis de resultados y documentación de defectos.

4.3.7. PT7 – Despliegue en Android e iOS

En este punto, la aplicación completa del paquete PT04.3-FRNT está lista para ser presentada al mundo. Nuestro enfoque estará en las tiendas de aplicaciones, subiendo y asegurando la conformidad de la aplicación con los requisitos de la Apple Store y Google Play. Esto incluye la creación de descripciones detalladas del producto y la producción de capturas de pantalla atractivas, que servirán para atraer y convencer a los potenciales usuarios de la calidad y utilidad de la aplicación.

Tabla 11: Despliegue. Elaboración Propia

Nombre	Despliegue en Android e iOS
Código	PT07-DSP
Descripción	Realizar las actividades finales necesarias para lanzar la aplicación "FamilySeriesTrack" en las tiendas de aplicaciones y preparar el material de marketing correspondiente.
Entradas	Aplicación completa del paquete PT04.3-FRNT
Salidas	Aplicación subida a Apple Store y Google Play.

Nombre	Despliegue en Android e iOS
Actividades	<ul style="list-style-type: none"> ✓ Cumplimiento de los requisitos de las tiendas de aplicaciones. ✓ Creación de descripciones de productos y capturas de pantalla.

4.4. PLAN DE TRABAJO

En este apartado se detalla el Diagrama de Gantt que se ha seguido meticulosamente a lo largo del proyecto. El Diagrama de Gantt es una herramienta esencial para la gestión y planificación del proyecto, proporcionando una representación visual del cronograma y las fases de desarrollo. Este diagrama ha sido fundamental para asegurar que todas las actividades y tareas del proyecto se completen de manera organizada y eficiente.

Uno de los aspectos clave reflejados en el Diagrama de Gantt es la programación de entregas periódicas de la memoria al tutor. Estas entregas han sido cruciales para mantener una comunicación constante y efectiva, permitiendo recibir retroalimentación y orientación durante el desarrollo del proyecto. Cada entrega ha sido cuidadosamente planificada y alineada con las distintas fases del proyecto, asegurando que cada segmento de la memoria corresponda con el progreso y los hitos alcanzados.

El Diagrama de Gantt incluye no solo las fases de desarrollo técnico, como la codificación, pruebas y despliegue, sino también las etapas de investigación, diseño, y revisión. La inclusión de las entregas de la memoria en este cronograma ha sido vital para garantizar que la documentación del proyecto esté siempre actualizada y sincronizada con el avance práctico del mismo.

El Diagrama de Gantt ha sido una herramienta dinámica, actualizándose regularmente para reflejar cambios en el cronograma, ajustes en las prioridades y la incorporación de nuevas tareas o actividades según fue necesario. Esta flexibilidad ha sido clave para adaptarse a los desafíos emergentes y para aprovechar oportunidades de mejora durante el desarrollo del proyecto.

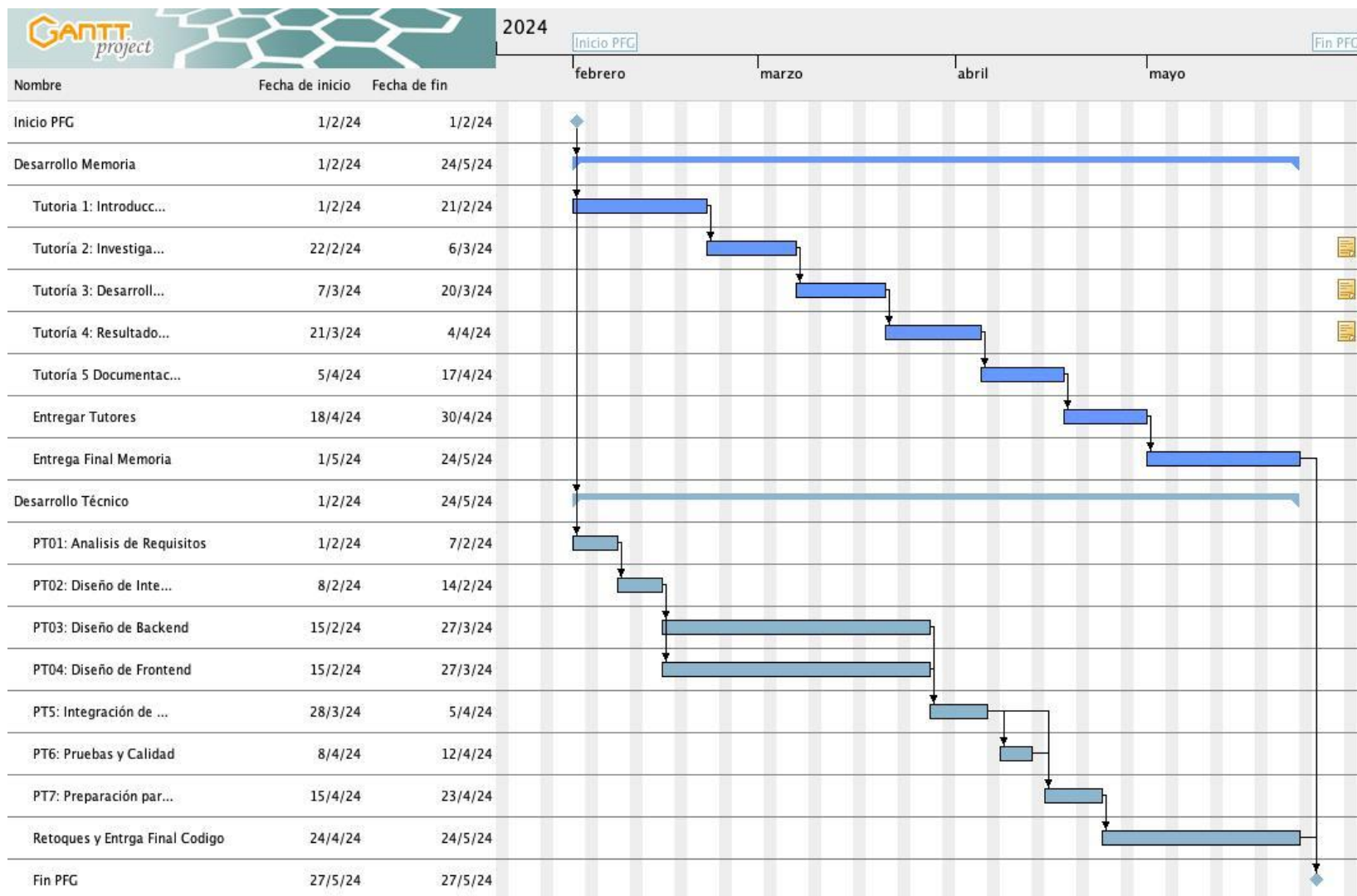


Ilustración 2: Diagrama de Gantt. Elaboración Propia

4.5. RECURSOS

4.5.1. Recursos Técnicos

Herramientas de Desarrollo: En el desarrollo del proyecto se utilizaron varias herramientas de desarrollo de software. Para el desarrollo de la aplicación móvil, se emplearon entornos integrados de desarrollo (IDE) como Visual Studio Code. Además, se hizo uso de Expo, una plataforma que simplifica el desarrollo de aplicaciones móviles multiplataforma utilizando React Native. Para el diseño de la interfaz de usuario, se utilizó Figma.

Plataformas y Tecnologías: El proyecto se desarrolló utilizando tecnologías modernas para aplicaciones móviles, específicamente React Native y Expo. Estas tecnologías permiten crear aplicaciones móviles para iOS y Android utilizando un único código base en JavaScript. Para el frontend web, se utilizó React Native Web, una extensión de React Native que permite renderizar componentes de React en navegadores web.

4.5.2. Recursos Humanos

En cuanto a los recursos humanos, el proyecto "FamilySeriesTrack" ha sido llevado a cabo por un equipo compuesto principalmente por mí, quien ha desempeñado múltiples roles, desde el desarrollo técnico hasta la gestión del proyecto en su totalidad. He asumido la responsabilidad de todas las etapas del proyecto, desde la concepción inicial hasta la implementación final, demostrando habilidades técnicas sólidas y una capacidad excepcional para la planificación y ejecución.

Además, he contado con la guía y supervisión constante de mi tutor, cuya experiencia y conocimientos han sido fundamentales para orientarme a lo largo del proceso. Mi tutor ha proporcionado asesoramiento experto, brindando dirección y apoyo en momentos clave, lo que ha contribuido significativamente al éxito del proyecto.

4.6. COSTES

Todos los recursos materiales utilizados en el proyecto "FamilySeriesTrack" han sido adquiridos de manera gratuita, lo que ha permitido minimizar los costos de desarrollo. La infraestructura de la base de datos se encuentra alojada en un servidor local existente, eliminando la necesidad de costosos servicios en la nube. Sin embargo, cabe destacar que las únicas excepciones a esta gratuidad son las licencias necesarias para publicar la aplicación en las tiendas de aplicaciones de Apple y Google. Estas licencias, requeridas para distribuir la aplicación en las respectivas plataformas iOS y Android, representan el único gasto monetario asociado con el proyecto. Aunque estas licencias son necesarias para alcanzar una audiencia más amplia y asegurar la disponibilidad de la aplicación en los principales mercados de aplicaciones móviles, se ha trabajado diligentemente para optimizar y reducir al mínimo los costos asociados con el desarrollo y despliegue de "FamilySeriesTrack".

Tabla 12: Costes.

Plataforma	Precio
Google Play	25\$ Tarifa Única. [15]
Apple Store	99\$ Tarifa Anual.
Cloudflare (Dominio)	9.77\$ Tarifa Anual.

En cuanto a los recursos humanos, el coste sería el siguiente.

Tabla 13: Costes Recursos Humanos.

Cargo	Salario	Total
Proyect Manager	30,00 \$/hora	19.680 \$
Analista de requisitos	20,00 \$/hora	800 \$
Diseñador	20,00 \$/hora	800 \$
Desarrollador Frontend	25,00 \$/hora	6.000 \$
Desarrollador Backend	25,00 \$/hora	4.800 \$
Especialista BBDD	30,00 \$/hora	3.120 \$
Testers Pruebas	15,00 \$/hora	600 \$
Encargado de Despliegue	20,00 \$/hora	1.120 \$
Total		36,920 \$

El total se calcula siguiendo las horas previstas para cada miembro del equipo con el diagrama de Gantt anterior.

4.7. CONDICIONANTES Y LIMITACIONES

4.7.1. Error con la API en iOS

Durante el proceso de despliegue de la aplicación, nos enfrentamos a un obstáculo significativo relacionado con la conectividad de la API en dispositivos iOS. Aunque la aplicación funcionaba sin problemas en un entorno local utilizando `'npx expo start'`, observamos que al desplegarla en dispositivos iOS, la conexión a la API fallaba. Después de varios días de investigación y pruebas, se descubrió que el problema radicaba en la política de seguridad de iOS, la cual rechaza y bloquea las conexiones no seguras que utilizan el protocolo `'http://'` [16]

La aplicación se estaba intentando conectar a la API utilizando dicho protocolo, lo que entraba en conflicto con estas restricciones de seguridad. La solución a este problema fue implementar una conexión segura utilizando `'https://'`, lo que no solo cumplió con los requisitos de seguridad de iOS, sino que también mejoró la seguridad general de la comunicación entre la aplicación y la API. Este cambio aseguró que la aplicación pudiera establecer una conexión segura con la API, independientemente del entorno en el que se desplegara, resolviendo así el problema de conectividad en dispositivos iOS.

4.7.2. Error Despliegue en Android

Al enfrentar un error con Gradle durante el proceso de construcción para Android, es necesario actualizar a Java 17 para solventar el inconveniente. Esto se debe a que Gradle, una herramienta automatizada de construcción ampliamente utilizada en el desarrollo de aplicaciones Android, puede requerir una versión específica de Java para funcionar correctamente, en función de las características del proyecto o las actualizaciones de la propia herramienta. Para realizar esta actualización, se debe descargar e instalar Java 17 desde el sitio web oficial de Oracle o mediante una distribución de OpenJDK.

Posteriormente, es crucial configurar la variable de entorno JAVA_HOME para apuntar al directorio de instalación de Java 17 y asegurarse de que el PATH incluya la ruta al binario de Java. Además, se debe revisar y ajustar la configuración de Gradle en el proyecto, incluyendo la actualización de la versión de Gradle si es necesario, para garantizar la compatibilidad con Java 17, lo cual puede implicar la modificación de archivos como gradle.properties y gradle/wrapper/gradle-wrapper.properties [17]. Este proceso asegura que el entorno de desarrollo esté correctamente configurado para aprovechar las funcionalidades que ofrece Java 17, evitando así errores de compilación y otros problemas relacionados con la versión de Java.

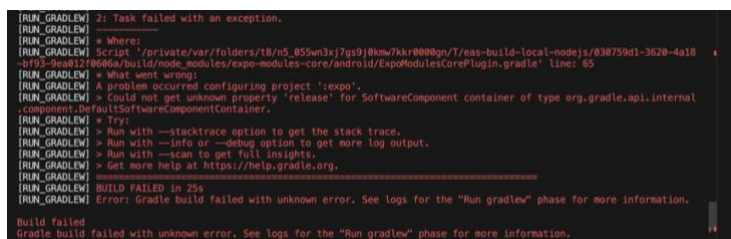


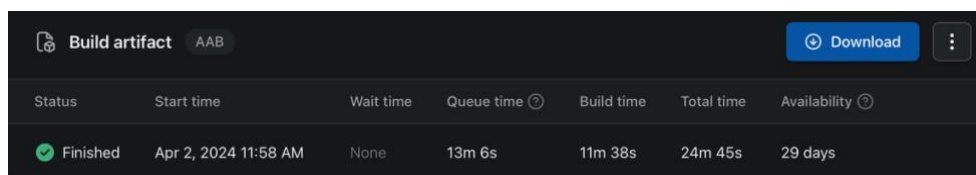
Ilustración 3: Error Gradle. Fuente: EAS Build.

Tras actualizar a Java 17 y resolver el problema inicial con Gradle, surge un nuevo error relacionado con las licencias del NDK (Native Development Kit) de Android, indicando que no están aceptadas. Para solucionar este inconveniente, es necesario ejecutar ciertos comandos que permiten aceptar explícitamente las licencias del NDK. Esta información se sacó de ChatGPT al cual se le preguntó cómo podía aceptar las licencias en Mac. [18]

Para esto se tuvo que hacer lo siguiente:

1. Abrir una Terminal
2. Navegar al Directorio del SDK de Android: Nuestra ubicación era la siguiente `'~/Android/Sdk'`
3. Ejecutar el SDK Manager para Aceptar las Licencias: Acepta las licencias que detecte que esten sin aceptar. Se utiliza el comando `'./sdkmanager --licenses'`.
4. Aceptar la licencia que nos pide.

Con estos pasos completados, se ejecutó el proceso de construcción (build) en Android y todo funciona perfectamente.



Build artifact		AAB		Download		
Status	Start time	Wait time	Queue time	Build time	Total time	Availability
Finished	Apr 2, 2024 11:58 AM	None	13m 6s	11m 38s	24m 45s	29 days

Ilustración 4: EAS Build Android correcto. Fuente: EAS

4.7.3. Error con tamaño de posters

Al ejecutar la aplicación en dispositivos Android, ya sea mediante Expo o con la aplicación ya compilada, todo funcionaba correctamente excepto en la presentación de los pósteres de series, y la aplicación se cerraba inesperadamente sin saber cuál es el error. Este problema solo ocurría en Android y no en iOS.

Tras investigar, se descubrió que el tamaño de los pósteres se había definido como '33,33%' para que tres de ellos encajaran por fila. Sin embargo, Android no interpreta correctamente los porcentajes cuando se utilizan comas, a diferencia de iOS, que sí lo hace. Al modificar esta configuración a '33%', el problema se solucionó, logrando que la visualización de los pósteres funcionara correctamente tanto en Android como en iOS.

4.7.4. Notificaciones

Se contempló la implementación de un sistema de notificaciones en tiempo real que informara a los miembros de un grupo cada vez que uno de ellos visualizara un nuevo capítulo, con el objetivo de fomentar una experiencia de usuario colaborativa y enriquecedora. Aunque inicialmente se logró desarrollar un prototipo funcional en el entorno local a través de la aplicación Expo Go, la extensión de su funcionalidad a un entorno de producción presentó retos significativos. La operatividad completa del sistema requería una integración con los servicios de notificaciones push de Apple y Google, lo que implicaba obtener y configurar perfiles y tokens específicos, una tarea que demostró ser compleja y extensa.

Tras un período prolongado de investigación y múltiples intentos, se decidió no proseguir con esta característica debido a las dificultades técnicas que impedían su implementación y el tiempo considerable que esto demandaba. Esta decisión estratégica se tomó para concentrar los esfuerzos en otras áreas cruciales del desarrollo y garantizar la entrega de una aplicación funcional dentro de los plazos establecidos.

El proceso de exploración en torno a las notificaciones push brindó un aprendizaje valioso, especialmente en el manejo de las políticas y herramientas de las plataformas móviles. Aunque la implementación no se llevó a cabo, los conocimientos adquiridos son un recurso que se puede aplicar en proyectos futuros, subrayando la importancia de la adaptabilidad y una gestión eficiente del tiempo en el desarrollo de software.

4.7.5. Autenticación en local

En la fase de desarrollo, se intentó incorporar la autenticación biométrica, incluyendo Face ID y huella dactilar, para que los usuarios no necesitaran iniciar sesión cada vez que accedieran a la aplicación. Esta funcionalidad buscaba ofrecer una mayor comodidad y seguridad en el acceso a la plataforma, aprovechando las capacidades de hardware de los dispositivos modernos.

Sin embargo, al igual que con las notificaciones, este aspecto del proyecto se enfrentó a obstáculos significativos. A pesar del tiempo dedicado a la investigación y las pruebas, y aunque se lograron avances en el entorno de desarrollo, la implementación plena de estas características resultó ser más compleja de lo anticipado. Las complicaciones surgieron tanto en la comprensión del flujo de autenticación biométrica como en su integración eficaz dentro de la aplicación.

Ante la necesidad de cumplir con los plazos establecidos y la persistencia de estos desafíos técnicos, se tomó la decisión de posponer la autenticación biométrica. Esta resolución permitió redirigir el enfoque hacia elementos esenciales del proyecto que garantizaran su funcionalidad y estabilidad para la entrega final.

La experiencia obtenida durante el proceso de exploración de la autenticación biométrica ha enriquecido el conjunto de habilidades disponibles para futuros desarrollos, proporcionando una base sobre la cual se puede construir posteriormente. Este aprendizaje refleja la capacidad para trabajar con tecnologías emergentes y subraya la necesidad de una gestión flexible del proyecto, donde se deben establecer prioridades claras para el logro de los objetivos principales.

5. DESARROLLO DE LA SOLUCIÓN TÉCNICA

La explicación detallada de cada PT permitirá una visión clara del avance del proyecto, ofreciendo una perspectiva sobre cómo cada componente contribuye al conjunto del sistema y cómo se integran para formar una solución cohesiva y funcional.

Al detallar el proceso de desarrollo de cada paquete de trabajo, se proporcionará una descripción exhaustiva de las tareas ejecutadas, los desafíos encontrados, y las soluciones implementadas para superarlos. Este enfoque nos permitirá también presentar los resultados parciales obtenidos en cada etapa, mostrando el progreso incremental hacia la realización completa del proyecto.

5.1. PT1 – ANÁLISIS DE REQUISITOS

En este paquete de trabajo, se ha completado el análisis de requisitos, cuyo resultado incluye tanto los requisitos funcionales como los no funcionales de nuestro sistema. Para lograr esto, se ha diseñado y llevado a cabo una encuesta mediante Google Forms, consistente en una serie de preguntas seleccionadas para extraer la información necesaria y cumplir con nuestro objetivo.

Esta encuesta ha sido importante para entender directamente desde los usuarios finales, cuáles son sus necesidades específicas, preferencias y expectativas respecto al sistema que estamos desarrollando. Las preguntas abarcaron temas desde funcionalidades específicas, usabilidad, accesibilidad, hasta requisitos de seguridad y rendimiento esperado, garantizando así una amplia gama de información importante.

Además, la utilización de Google Forms como herramienta para la encuesta nos ha permitido recopilar las respuestas de manera organizada y eficiente, facilitando el análisis posterior de los datos. La flexibilidad de esta herramienta también ha permitido ajustar las preguntas en función de los comentarios preliminares, mejorando la calidad de la información obtenida.

Con los datos recopilados, hemos podido identificar claramente las prioridades de los usuarios y otros criterios esenciales que guiarán el diseño y desarrollo del sistema. Este proceso ha sido fundamental para asegurar que el producto final no solo cumpla con las expectativas de los usuarios, sino que también responda de manera efectiva a sus problemas y necesidades reales.

Los datos recopilados se muestran las ilustraciones 3-8.

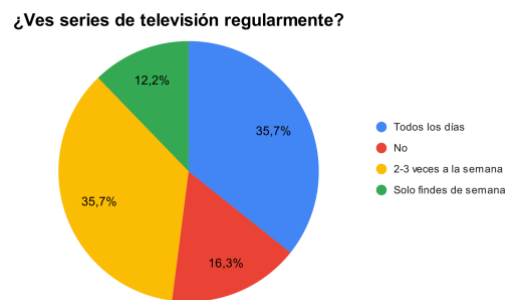


Ilustración 5: ¿Ves series regularmente? Elaboración Propia

El gráfico que se muestra en la ilustración 3 es relevante para el análisis de requisitos ya que proporciona información sobre los hábitos de consumo de series de televisión de un grupo de personas. Según los datos del gráfico, la mayoría de las personas ven series de televisión todos los días o 2-3 veces por semana, lo que sugiere una alta frecuencia de uso potencial para la aplicación.

Nos ayuda a entender características clave:

- ✓ Funcionalidad de notificaciones.
- ✓ Sincronización de estado de visualización.
- ✓ Calendario de series.

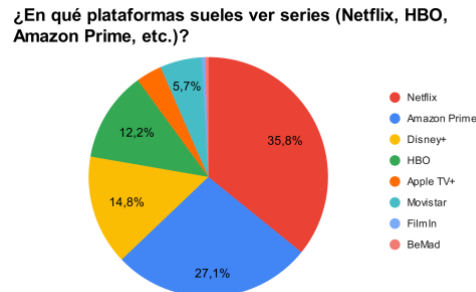


Ilustración 6: ¿En qué plataforma sueles ver las series? Elaboración Propia

Este gráfico proporciona información valiosa sobre las plataformas de streaming más populares utilizadas por el público para ver series. Conocer en qué servicios de streaming se consumen más series es útil para añadir funcionalidades a la aplicación que permitan a los usuarios seleccionar y mostrar el canal o plataforma donde están viendo cada serie. Esto podría tener varias implicaciones y beneficios para el diseño de la aplicación.

De este grafico podemos sacar estos aspectos clave:

- ✓ Integración de plataformas: Se puede considerar integrar en la aplicación la plataforma de streaming por donde se ve dicha serie.
- ✓ Organización del contenido por plataforma: Si los usuarios del grupo están viendo series en diferentes plataformas, la aplicación podría organizar la información de visualización por cada plataforma para mejorar la gestión.

¿Conoces o utilizas alguna aplicación para hacer seguimiento de las series que ves?

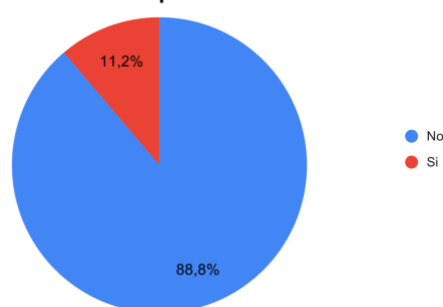


Ilustración 7: ¿Conoces alguna aplicación para hacer seguimiento de las series? Elaboración Propia

La primera imagen muestra que una gran mayoría de los encuestados, el 88,8%, no conoce o no utiliza una aplicación para hacer seguimiento de las series que ven. Esto representa una oportunidad significativa para la aplicación. El hecho de que haya tantos usuarios potenciales que no están actualmente usando una solución de seguimiento significa que hay un mercado considerable por capturar.

Este gráfico respalda la investigación previa realizada, en la cual se concluye que no existen aplicaciones que gestionen el seguimiento de series de forma colectiva para grupos.

¿Te gustaría que hubiera una manera más fácil de gestionar la visualización de series en grupo?

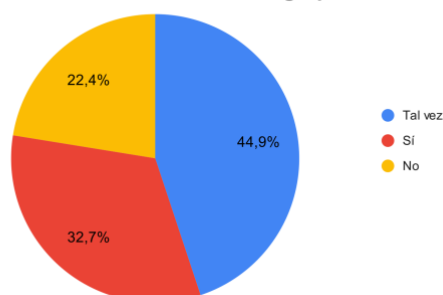


Ilustración 8: ¿Te gustaría que hubiera una manera más fácil de gestionar la visualización de series? Elaboración Propia

Esta imagen complementa la información indicando que un 44,9% de los usuarios sí estaría interesado en una manera más fácil de gestionar la visualización de series en grupo, con un adicional 22,4% que podría considerarlo ("Tal vez"). Este interés manifiesto sugiere que hay una demanda para las funcionalidades que la aplicación planea ofrecer.

Estos dos gráficos son fundamentales para el proceso de definición de requisitos de diseño para la aplicación. La falta de conocimiento general sobre aplicaciones de seguimiento colectivo de series señala la necesidad de centrarse en una UI y una UX que sean intuitivas y accesibles, incluso para aquel usuario que podría no estar familiarizado con este tipo de tecnología.



Ilustración 9: Características vs Recuento. Elaboración Propia

Este gráfico de barras ilustra las funcionalidades que los usuarios han solicitado para una aplicación de seguimiento de series en grupo. La altura de las barras indica la frecuencia con la que se ha mencionado cada característica, lo que ofrece una visión clara de las prioridades de los usuarios. Por ejemplo, la función más solicitada, como indica la barra más alta, podría ser una lista compartida que permita a los usuarios gestionar las series que están viendo como un grupo.

Las siguientes barras, de menor altura, sugieren otras funcionalidades importantes, como chat grupal, notificaciones de nuevos episodios y la capacidad de marcar episodios como vistos.

Este análisis es crucial para la fase de diseño de la aplicación, ya que proporciona una guía directa sobre qué características deben ser incorporadas para satisfacer las necesidades y deseos del usuario final. Las funcionalidades que aparecen con mayor frecuencia serán probablemente las que aporten más valor a los usuarios y, por tanto, son las que se deberían implementar en las primeras versiones de la aplicación para asegurar su adopción y satisfacción del usuario.

El gráfico también destaca las características menos solicitadas, que pueden ser consideradas para desarrollos futuros o mejoras iterativas de la aplicación. Priorizar estas características según la demanda del usuario garantiza que los recursos de desarrollo se utilicen de manera efectiva y que el producto final se alinee estrechamente con las expectativas del mercado.



Ilustración 10: ¿Considerarías cambiar a una aplicación como FST? Elaboración Propia

El gráfico muestra que un significativo 41,8% de los usuarios encuestados estarían dispuestos a cambiar a una nueva aplicación que ofrezca mejores funcionalidades para la visualización en grupo de series, mientras que un 25,5% se muestra indeciso al responder "Tal vez". Esto indica que hay un potencial del 67,3% del mercado encuestado que podría estar interesado en probar la aplicación. Es una clara señal de que, si se ofrecen las características correctas y

una experiencia de usuario superior, hay una oportunidad considerable de captar usuarios que buscan optimizar su manera de consumir series en grupo.

Además de la valiosa información recogida a través de cuestionarios, el desarrollo de los requisitos de usuario se ha enriquecido con la investigación y análisis de aplicaciones existentes en otros sectores que emplean sistemas de grupos para diferentes propósitos. Un ejemplo notable es la aplicación Life360 [19], que, si bien se centra en la localización y el seguimiento de familiares, comparte la dinámica de grupo parecida a la que se quiere implementar en FST.

La manera en que Life360 maneja la formación de grupos, las interacciones entre los miembros y las notificaciones en tiempo real son aspectos que se han considerado relevantes y que pueden trasladarse al contexto del seguimiento de series televisivas en grupos.

El análisis de estas aplicaciones ha permitido adaptar y refinar ciertas funcionalidades para satisfacer las necesidades específicas de nuestro ámbito de entretenimiento. Por ejemplo, al observar cómo Life360 facilita la conectividad y comunicación entre sus usuarios, se han obtenido ideas para implementar un sistema de notificaciones y estado de visualización que mantenga a los usuarios de FST informados sobre la actividad de visualización de su grupo familiar o de amigos.

La salida del paquete de trabajo **PT01-AR** queda reflejada en [el Anexo A](#).

5.2. PT2 – DISEÑO DE INTERFAZ DE USUARIO

Para comenzar este proceso, se llevó a cabo una fase de investigación y recopilación de ideas, analizando diversas aplicaciones existentes en el mercado para identificar características de diseño y funcionalidades que pudieran ser relevantes y aplicables al proyecto en cuestión. Este análisis preliminar permitió esbozar las primeras ideas y conceptos para la interfaz de usuario.

Posteriormente, estas ideas fueron llevadas a Figma, una herramienta especializada en diseño de interfaces y prototipado. Figma facilitó el proceso de diseño al permitir la creación de maquetas interactivas que se podían iterar y modificar de manera eficiente. Específicamente, las capacidades de Figma, como el uso de componentes reutilizables, la creación de prototipos interactivos y las herramientas avanzadas de colaboración [20], probaron ser indispensables.

Estas funcionalidades no solo optimizaron el flujo de trabajo, sino que también promovieron una mayor cohesión y consistencia a través del proyecto, lo que justifica claramente su elección sobre otras herramientas disponibles en el mercado. A medida que el proyecto avanzaba, se encontró necesario realizar ajustes en el diseño de la interfaz. Algunas decisiones iniciales, aunque parecían lógicas en las etapas tempranas, demostraron ser menos prácticas o intuitivas cuando se contextualizaron dentro del flujo completo de la aplicación.

Este proceso iterativo de diseño fue esencial para refinar la interfaz de usuario. A través de sucesivas revisiones y pruebas, se logró un diseño más coherente y funcional que mejoraba la experiencia del usuario. El uso de Figma como herramienta de diseño y prototipado jugó un papel crucial en este proceso, permitiendo al equipo visualizar cambios en tiempo real y colaborar de manera más efectiva para alcanzar un consenso sobre las mejores prácticas de diseño para la aplicación.

Aparte de lo anterior, también se implementó un enfoque metódico para integrar los requisitos del usuario, identificados en el documento PT01-AR, en el diseño de la interfaz. Este documento resultó ser un recurso invaluable, ya que proporcionó una comprensión detallada de las expectativas y necesidades del usuario final, orientando al equipo sobre cómo y dónde implementar ciertas funcionalidades dentro de la aplicación para maximizar su accesibilidad y eficacia.

Para iniciar el proceso de diseño, fue necesario establecer un nuevo proyecto en Figma. Dado que este espacio de trabajo inicial se presentaba sin elementos predefinidos, se procedió a la adición de diversas pantallas nuevas, seleccionando específicamente el tamaño correspondiente al de un iPhone 15. Esta elección se basó en la consideración de que dicho tamaño representa un equilibrio adecuado entre las distintas dimensiones de dispositivos disponibles en el mercado, facilitando así un diseño más universal y adaptable a diversas plataformas.

Tras establecer las bases fundamentales del proyecto en Figma, incluyendo la selección del tamaño de las pantallas y la definición de la paleta de colores, el siguiente paso en el proceso de diseño consiste en materializar los diseños específicos de la interfaz, comenzando con las pantallas cruciales de inicio de sesión y creación de usuario. Este enfoque inicial permite establecer las puertas de entrada a la aplicación, asegurando que los usuarios experimenten una bienvenida intuitiva y amigable desde el primer momento.

Siguiendo la lógica y el flujo de acciones que un usuario típico llevaría a cabo dentro de la aplicación, el diseño se expande secuencialmente a otras áreas críticas. Este método garantiza que el desarrollo del diseño sea centrado en el usuario, priorizando la usabilidad y la experiencia general a lo largo de todo el recorrido dentro de la aplicación.

El desarrollo del diseño de la aplicación avanza de forma estructurada, siguiendo un camino que imita la experiencia de usuario deseada.

La salida de **PT02-DIU** se utilizó posteriormente para la creación del frontend de la aplicación.

5.3. PT3 – DESARROLLO DEL BACKEND

El servidor que se ha usado es un AZW SER en el cual tiene montado un sistema operativo Ubuntu Desktop 22.04.2 LTS la cual es una distribución de Linux conocida por su facilidad de uso y popularidad, especialmente entre aquellos que se inician en Linux. Es parte de la familia de sistemas operativos Linux, basados en Unix, y opera bajo un modelo de código abierto, lo que significa que su código fuente está disponible gratuitamente para ser utilizado, modificado y distribuido. [21]

Se ha escogido Ubuntu porque proporciona un entorno robusto y versátil para desarrolladores, administradores de sistemas y empresas, asegurando un equilibrio óptimo entre facilidad de uso y potencia técnica para satisfacer las necesidades de cualquier servidor.

Además, para la implementación y operación de Docker, la utilización de Ubuntu representa una opción más conveniente en comparación con Windows [22], debido a varias razones fundamentales. Ubuntu ofrece una integración natural y eficiente con Docker, gracias a su arquitectura basada en Linux y su compatibilidad con contenedores, facilitando así el proceso de configuración y despliegue de aplicaciones en contenedores.

Por otro lado, aunque Windows ha mejorado su compatibilidad con Docker a través de Windows, este aún presenta limitaciones en comparación con un sistema nativamente Linux [23], tanto en términos de rendimiento como de compatibilidad con ciertos aspectos de Docker y aplicaciones específicas. Además, el costo de licencias de Windows representa un factor adicional a considerar para organizaciones y desarrolladores que buscan optimizar sus recursos financieros.

La selección de Ubuntu sobre Windows para la ejecución de Docker no solo se justifica por razones técnicas y de compatibilidad, sino también por consideraciones económicas, destacando la importancia de elegir un entorno que maximice la eficiencia operativa y la rentabilidad.

Para asegurar que el servidor funcione de manera eficiente y segura, la infraestructura se ha organizado en varios contenedores Docker. Esta configuración utiliza seis contenedores principales: phpMyAdmin para la gestión de la base de datos MariaDB, la base de datos MariaDB propiamente dicha, Portainer para la gestión de los contenedores, un contenedor dedicado para la API, Cloudflare para la seguridad y optimización de la red, y Traefik como proxy inverso y gestor de certificados SSL.

Todos estos contenedores están interconectados a través de una red personalizada llamada 'diegosNetwork', utilizando el controlador bridge. Esta red facilita una comunicación fluida y segura entre los contenedores, permitiéndoles operar como si estuvieran en la misma máquina física, pero manteniendo un alto grado de aislamiento y seguridad. Este enfoque mejora significativamente la seguridad y la eficiencia en la comunicación de los contenedores, minimizando la exposición al anfitrión y a redes externas.

5.3.1. MariaDB

Este contenedor se basa en la imagen "linuxserver/mariadb", que sirve como la base de datos del proyecto. Para garantizar la comunicación efectiva entre este contenedor y otros, fue necesario asignar un nombre de host específico. Además, se configuraron diversos elementos esenciales para su funcionamiento óptimo, incluyendo el volumen para persistir los datos, una red dedicada para facilitar la interconexión entre contenedores, puertos específicos para permitir el acceso externo, y variables de entorno necesarias para personalizar la configuración de MariaDB.

La configuración detallada del volumen asegura que los datos se mantengan seguros y accesibles incluso después de reiniciar o destruir el contenedor. La red dedicada simplifica la comunicación entre contenedores, permitiendo que se "escuchen" entre sí sin interferencias externas. La especificación de puertos garantiza que la base de datos sea accesible a través de la red, mientras que las variables de entorno permiten ajustes finos de la configuración interna de MariaDB, como la asignación de contraseñas, nombres de usuario, y otros parámetros cruciales para la seguridad y el rendimiento.

La implementación de la base de datos es crucial en nuestro proyecto, se ha llevado a cabo un análisis detallado de los requisitos, identificando la información específica que necesitamos almacenar en nuestra base de datos, para esto se ha empezado con lo más básico, los usuarios y después se ha ido avanzando hacia lo más complejo.

El núcleo de esta estructura es la tabla "Usuarios", que sirve como el pilar fundamental en el que se registran los detalles cruciales de cada usuario. Esta tabla incluye campos para almacenar un identificador único por usuario, el nombre y apellidos, así como el nombre de usuario y contraseña, facilitando así la autenticación y personalización de la experiencia del usuario dentro del sistema.

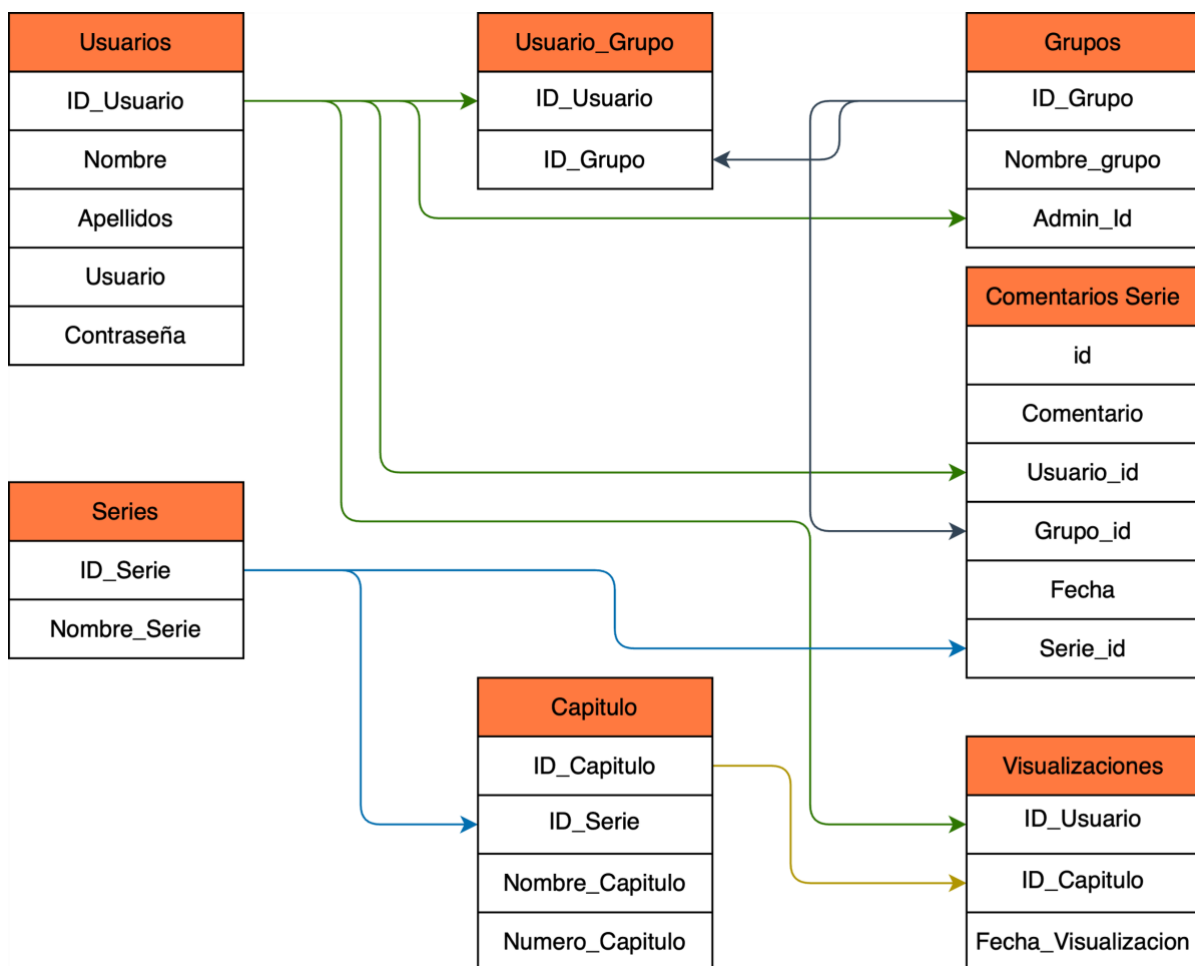


Ilustración 11: Esquema BBDD. Elaboración Propia

Para este esquema de base de datos, se han implementado disparadores que se activan cuando se elimina una serie o un grupo. Estos disparadores se encargan de eliminar de

manera automática toda la información relacionada en la base de datos, asegurando la integridad y la coherencia de los datos. Estas operaciones de eliminación en cascada son esenciales para mantener la base limpia y ordenada.

Se detalla la explicación de cada tabla en el [Anexo C](#).

5.3.2. PhpMyAdmin

El contenedor en cuestión utiliza la imagen "phpmyadmin/phpmyadmin" para proporcionar una interfaz gráfica de usuario para la administración de la base de datos MariaDB. Se ha asignado el nombre de host phpmyadmin y se ha configurado para reiniciarse automáticamente, asegurando su disponibilidad continua. Se ha establecido un enlace con el contenedor de MariaDB, permitiendo una conexión directa y fiable entre ambos servicios.

La configuración de la red 'diegosNetwork' asegura que el contenedor de phpMyAdmin esté en la misma red que el contenedor de MariaDB, lo que facilita su intercomunicación. Los puertos están mapeados de manera que el puerto 4442 del host se redirige al puerto 80 del contenedor, permitiendo el acceso a la interfaz de phpMyAdmin a través del navegador web.

Además, se han establecido variables de entorno específicas, como PMA_HOST, que apunta al contenedor de MariaDB, y MYSQL_ROOT_PASSWORD, que proporciona la contraseña necesaria para acceder a la base de datos como usuario root, contribuyendo a la seguridad y el acceso controlado a la base de datos. [24]

5.3.3. Tfg_Backend

Se ha establecido un contenedor denominado tfg_backend, diseñado específicamente para alojar y ejecutar la API del backend desarrollada en Node.js. Este proceso comenzó con la creación de una carpeta destinada al backend, dentro de la cual se colocó el archivo esencial backend.js, que contiene toda la lógica de la API. Para facilitar la ejecución de esta API en un entorno aislado y controlado, se creó un Dockerfile [25] en la raíz del proyecto.

Este archivo de configuración es crucial ya que detalla cómo debe construirse la imagen Docker del contenedor, siguiendo una serie de instrucciones precisas. Estas instrucciones abarcan desde la selección de una imagen base de Node.js (por ejemplo, la versión alpina por su ligereza), pasando por la copia del código fuente al contenedor, hasta la ejecución de comandos necesarios para instalar dependencias y exponer el puerto adecuado para las comunicaciones. Finalmente, se define un comando para iniciar la aplicación, asegurando que backend.js se ejecute correctamente dentro del contenedor.

Para garantizar una comunicación eficaz y segura entre el contenedor tfg_backend y otros servicios, especialmente con el enrutador y proxy inverso Traefik, se asignó una dirección IP fija al contenedor dentro de una red personalizada denominada diegosNetwork. Esta red utiliza el controlador bridge, favoreciendo así la interconexión entre contenedores de manera que pueden comunicarse como si residieran en la misma máquina física, manteniendo al mismo tiempo un alto grado de aislamiento.

La elección de asignar una IP fija resulta estratégica para la configuración en Traefik, permitiendo una gestión precisa del tráfico entrante. Mediante reglas definidas en Traefik, se redirige el tráfico destinado a la API hacia la IP fija del contenedor tfg_backend, optimizando la ruta de las solicitudes de los usuarios y asegurando que estas sean procesadas eficientemente por la API. [26]

La API desempeña un papel crucial al servir como intermediario entre la base de datos y la aplicación, estableciendo un canal de comunicación que permite la interacción con los datos almacenados. En este contexto, la aplicación hace llamadas a distintos endpoints de la API, cada uno diseñado con un propósito específico, como la visualización, modificación, o eliminación de datos. Estas operaciones son fundamentales para la gestión eficiente de la información y la dinámica de interacción del usuario con la aplicación.

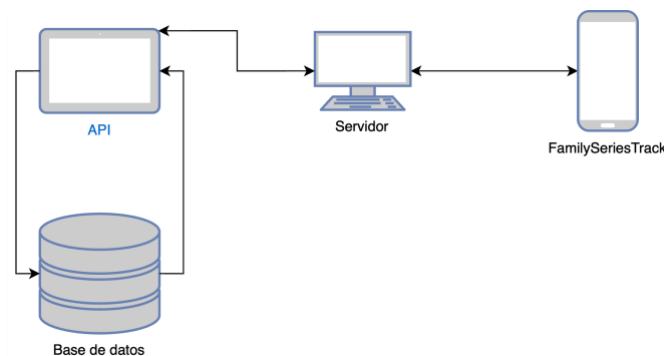


Ilustración 12: Diagrama de Conexión API. Elaboración Propia

Para el desarrollo de esta API, se ha creado un archivo denominado backend.js, el cual encapsula la lógica necesaria para establecer la conexión con la base de datos y definir los endpoints requeridos. Este archivo se basa en Express, un framework de Node.js que facilita la creación de APIs al ofrecer un conjunto robusto de características para aplicaciones web y móviles. Express simplifica el manejo de solicitudes HTTP y la configuración de respuestas a enviar al cliente, permitiendo así una estructura organizada y modular para el desarrollo del backend.

El archivo backend.js incluye la configuración inicial de Express, estableciendo parámetros esenciales para el funcionamiento del servidor, como el puerto de escucha y los middlewares necesarios para el procesamiento de datos en formato JSON y la gestión de formularios. A continuación, se detallan los endpoints, cada uno asociado a rutas específicas que definen las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) aplicables a los usuarios, grupos, series y visualizaciones dentro de la base de datos. Estos endpoints son esenciales para interactuar con la aplicación, permitiendo realizar operaciones como el registro e inicio de sesión de usuarios, la gestión de grupos y series, y el registro de visualizaciones de capítulos, entre otros.

Un endpoint se refiere a una URL específica a la que se puede hacer una solicitud HTTP para realizar una operación en un servidor web. Estas operaciones pueden incluir la obtención, creación, modificación o eliminación de datos. Los endpoints son, esencialmente, puntos de entrada a los servicios que ofrece una API. Los endpoints de nuestra API son los siguientes:

5.3.4. Cloudflare

Este contenedor Docker está específicamente configurado para ejecutar un túnel de Cloudflare, una herramienta vital para exponer de forma segura servicios internos a Internet mediante el mantenimiento de un tráfico cifrado y protegido. Al utilizar este contenedor, cualquier solicitud al DNS es interceptada por el túnel, que actúa como un conducto seguro, redirigiendo el tráfico hacia Traefik, un proxy inverso y un balanceador de carga. Esta redirección es crucial porque Traefik se encarga de gestionar y distribuir el tráfico entrante a los contenedores adecuados, según la configuración establecida.

Simplifica enormemente la configuración de red al eliminar la necesidad de configuraciones complejas de firewall y NAT en el entorno local. Esto no solo reduce la carga administrativa, sino que también mejora la escalabilidad y la fiabilidad del sistema, al permitir que los servicios se comuniquen a través de un canal estable y seguro, sin importar las fluctuaciones o restricciones en la infraestructura de red local.

5.3.5. Traefik

Se ha elegido usar Traefik por su capacidad de manejar eficientemente el tráfico entrante a tu infraestructura de contenedores, proporcionando un manejo automático del descubrimiento de servicios, balanceo de carga y encriptación SSL, todo mientras ofrece una interfaz gráfica para una gestión simplificada. Esta herramienta no solo optimiza las operaciones de red dentro de los servicios, sino que también fortalece la seguridad y la escalabilidad de tu infraestructura. [27]

Por ejemplo, cuando se llama a *<https://adminmariadb.lapspartbox.com>*, se inicia una serie de interacciones coordinadas por Traefik y protegidas por Cloudflare, asegurando que el tráfico sea seguro y eficientemente dirigido. Primero, la solicitud pasa a través de Cloudflare, que actúa como un proxy inverso y un firewall, protegiendo y cifrando tu conexión. Cloudflare también minimiza la latencia y mejora la seguridad mediante la optimización del tráfico web y la protección contra ataques DDoS y otros tipos de amenazas cibernéticas.

Una vez que Cloudflare procesa y cifra la solicitud, esta se envía a Traefik, que está configurado para actuar como el proxy inverso dentro de la infraestructura de contenedores. Traefik escucha en los puertos configurados (80 para HTTP y 443 para HTTPS), y gracias a su integración con Docker, tiene conocimiento de todos los servicios y sus respectivas configuraciones en la red de contenedores. [28]

Para la URL mencionada, Traefik consulta sus reglas de enrutamiento dinámico para determinar a cuál servicio debe redirigir la solicitud. En este caso, identifica que la solicitud corresponde al servicio de phpMyAdmin, configurado previamente en su tabla de ruteo. Traefik entonces redirige la solicitud al contenedor correspondiente que aloja el servicio de phpMyAdmin, pasando por la red interna del Docker configurada para este propósito.

El servicio de phpMyAdmin recibe la solicitud y procesa la interfaz de gestión de bases de datos MariaDB, permitiéndote administrar tu base de datos a través de una interfaz web. Este proceso se lleva a cabo de manera transparente y segura, proporcionando un acceso

eficiente y protegido a tus bases de datos, todo bajo la capa de seguridad adicional y las optimizaciones de rendimiento ofrecidas por Cloudflare y la gestión de tráfico inteligente de Traefik. Se puede ver el diagrama del sistema en el Anexo B: Diseño del Sistema.

5.3.6. Portainer

El contenedor Portainer se utiliza para simplificar la gestión y el mantenimiento de entornos de contenedores Docker. Esta herramienta ofrece una interfaz de usuario (UI) visual e intuitiva que facilita la administración de diversos aspectos de los contenedores, como imágenes, redes y volúmenes. Principalmente, se ha utilizado para permitir la visualización de los logs de la API desde cualquier lugar, proporcionando un acceso conveniente y centralizado a la información crucial para el monitoreo y la resolución de problemas en tiempo real.

5.4. PT4 – DESARROLLO FRONTEND

5.4.1. Estructura de Directorios

Para desarrollar la estructura base del frontend, se ha adoptado una metodología acorde con las mejores prácticas establecidas en proyectos desarrollados con React Native. React Native permite la construcción de aplicaciones móviles nativas usando JavaScript y React. La arquitectura de un proyecto en React Native se organiza en torno a componentes reutilizables, lo que facilita el desarrollo modular y mejora la eficiencia del proceso de desarrollo.

La estructura de directorios típica en un proyecto de React Native incluye varias carpetas y archivos clave que organizan el código fuente de la aplicación y sus recursos. [29] La estructura que se ha seguido para el proyecto es la siguiente.

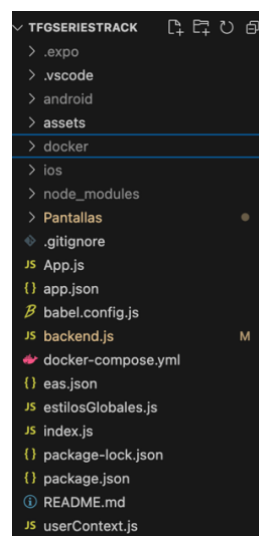


Ilustración 13: Estructura Directorios. Elaboración Propia

- ✓ .expo/: Esta carpeta contiene archivos relacionados con Expo, que es un marco y una plataforma para aplicaciones universales de React. Es utilizado para el desarrollo y construcción de aplicaciones React Native.
- ✓ .vscode/: Este directorio incluye la configuración específica de Visual Studio Code, como las preferencias de usuario o configuraciones del workspace para el proyecto.
- ✓ android/ y ios/: Contienen los proyectos y configuraciones específicos para compilar la aplicación en Android e iOS, respectivamente. Esto es lo que se subirá a Apple Store y Google Play una vez terminado.
- ✓ assets/: Almacena recursos estáticos como imágenes, fuentes que la aplicación utilice.
- ✓ node_modules/: Incluye las dependencias del proyecto instaladas a través de NPM (Node Package Manager).
- ✓ Pantallas/: Es un directorio donde se almacenan los componentes de las pantallas de la aplicación.
- ✓ .gitignore: Un archivo de configuración para Git que especifica los archivos y directorios no rastreados que deben ignorarse en las operaciones del repositorio.
- ✓ App.js: El archivo principal de entrada de una aplicación React Native, que define la composición de la UI a nivel raíz.
- ✓ app.json: Contiene la configuración de la aplicación a nivel de meta, como el nombre, versión y configuración específica para Expo.
- ✓ babel.config.js: Configuración para Babel, una herramienta que se utiliza para convertir código ECMAScript 2015+ en una versión compatible con versiones anteriores de JavaScript.
- ✓ docker-compose.yml: Un archivo de Docker Compose para la creación de los contenedores necesarios de la aplicación.
- ✓ eas.json: Específico de Expo Application Services (EAS), que gestiona la construcción y despliegue de aplicaciones creadas con Expo.
- ✓ estilosGlobales.js: Contiene estilos CSS en JavaScript que se aplican de manera global en toda la aplicación.
- ✓ index.js: A menudo es el punto de entrada a la aplicación React Native para el registro del componente raíz con AppRegistry.
- ✓ package-lock.json y package.json: Estos archivos gestionan las dependencias del proyecto, con package.json declarando las dependencias y package-lock.json asegurando una instalación consistente de estas.
- ✓ README.md: Un archivo Markdown con información acerca del proyecto, incluye instrucciones de instalación, uso y contribución al proyecto.
- ✓ useContext.js: Un archivo que está utilizando el Context API de React para manejar el estado global de la autenticación del usuario y su información.

Este proceso de creación de la estructura te le da hecho React Native, solo había que crear el proyecto en el terminal. Para crear este proyecto se ha abierto un terminal, se ha navegado hasta donde se quiere guardar el proyecto, y se ha ejecutado las siguientes líneas de código: `'npx create-expo-app TFGFamilySeriesTrack'`, después, `'cd TFGFamilySeriesTrack'` y `'npx expo start'` para ejecutarlo.

Para ejecutar el proyecto y revisar su funcionamiento o apariencia, es necesario ejecutar `'npx expo start'`. [30]

Lo importante es la carpeta "Pantallas", donde se han creado los archivos necesarios para desarrollar la aplicación.

5.4.2. Pantallas y Navegación

El desarrollo de este paquete de trabajo ha sido guiado por los resultados del paquete **PT2-DIU**, que establece el diseño a seguir para la creación de la aplicación. Este diseño ha servido como hoja de ruta para el desarrollo y la implementación de las funcionalidades de la aplicación.

Primero, se han creado las pantallas dentro de la carpeta 'Pantallas', utilizando la nomenclatura CamelCase, por ejemplo, un archivo se llamaría 'HomeScreen.js'. Este paso es crucial para organizar el proyecto y facilitar la identificación de cada pantalla. Posteriormente, estas pantallas han sido incorporadas al archivo 'App.js', lo cual es esencial para integrarlas en el flujo principal de la aplicación.

Al añadir las pantallas a 'App.js', se establece la estructura de navegación y se hace posible que el usuario interactúe con diferentes secciones de la aplicación. Esta integración garantiza una experiencia de usuario cohesiva y funcional, por lo que el usuario podría navegar por todas las pantallas.

Todas las pantallas están divididas en unos bloques, los cuales cada uno tiene una función.

- ✓ Importaciones y Estado: Se importan bibliotecas de React, React Native, y otras utilidades. Se establece el estado inicial para gestionar datos de usuario, series, y otros elementos visuales.
- ✓ Navegación y Contexto: Utiliza `useNavigation` y `useRoute` para manejar la navegación entre pantallas, y `useUser` para acceder al contexto del usuario y sus datos.
- ✓ UI Componentes: Define componentes visuales como `View`, `Text`, `Button`, y `ScrollView`, que estructuran el layout de la pantalla y permiten interacciones del usuario.
- ✓ Funcionalidades: Implementa funciones para interactuar con APIs externas (e.g., obtener series), gestionar estados (e.g., refrescar datos), y navegar entre pantallas.
- ✓ Estilos: Aplica estilos para dar formato a los componentes UI, utilizando tanto estilos locales definidos en el mismo archivo como estilos globales importados.

Queda detallado cada pantalla de navegación en el [Anexo D](#).

5.5. PT5 – INTEGRACIÓN DE UI CON BACKEND

Para llevar a cabo la integración efectiva de la interfaz de usuario con el backend, es imprescindible contar con los recursos y desarrollos completados en etapas previas del proyecto. Específicamente, es necesario tener acceso a la salida del paquete de trabajo **PT04.2-FRNT**, que contiene el código fuente del frontend de las pantallas. Este código es crucial, ya que define la estructura visual y la interactividad de la aplicación, asegurando que los usuarios finales puedan navegar e interactuar de manera intuitiva con la plataforma.

Además, resulta igualmente fundamental disponer de la API completamente desarrollada y operativa, tal como se especifica en el paquete de trabajo **PT03.3-BCK**. Esta API debe incluir todos los endpoints necesarios para permitir una comunicación fluida entre el frontend y el backend [31]. Estos endpoints son los puntos de acceso a través de los cuales el frontend puede solicitar datos a la base de datos, enviar nueva información, actualizar datos existentes o realizar cualquier otra operación.

Para asegurar el correcto funcionamiento de la aplicación, se ha adoptado un enfoque metódico y detallado en el desarrollo y la integración de sus componentes. Este proceso se ha centrado principalmente en la interacción entre la interfaz de usuario, la API y la base de datos, así como en la gestión eficiente del tráfico web y la seguridad. [32]

Inicialmente, se llevó a cabo una revisión exhaustiva de cada pantalla de la aplicación con el objetivo de determinar los datos y funcionalidades específicas requeridos por los usuarios en diferentes puntos de la interfaz. Este análisis permitió identificar claramente las necesidades de información y operación en cada segmento de la aplicación. A partir de esta identificación, se seleccionaron los endpoints apropiados de la API, los cuales proveen los datos necesarios o realizan las operaciones requeridas para satisfacer las demandas de cada pantalla.

En cada segmento de la interfaz, se desarrollaron funciones específicas que ejecutan ciertos endpoints, adaptándose a los requisitos únicos de cada pantalla. Este enfoque asegura que la información se muestre o modifique adecuadamente según lo dictado por las funciones diseñadas para cada contexto. Los detalles sobre cada endpoint se encuentran detallados en la siguiente web: <https://apitfg.lapspartbox.com/api-docs>

Una pieza clave en la infraestructura de la aplicación es la gestión del tráfico y la seguridad, facilitada por la integración de Cloudflare y Traefik. Cloudflare actúa como el primer punto de contacto cuando se realiza una solicitud a la web de la API, optimizando la entrega de contenido a través de su red CDN y proporcionando una capa adicional de seguridad al filtrar amenazas antes de que las solicitudes lleguen a la infraestructura principal.

Una vez que la solicitud ha sido procesada por Cloudflare, es redirigida al contenedor Traefik, que desempeña un papel crucial como proxy inverso. Traefik está configurado para gestionar el tráfico interno, dirigiendo las solicitudes al contenedor tfg-backend de acuerdo con reglas específicas. Dentro de este contenedor, la API procesa la solicitud, interactúa con la base de datos según sea necesario, y prepara la información para ser enviada de vuelta al usuario. La respuesta sigue el camino inverso a través de Traefik y Cloudflare, asegurando que los datos lleguen de forma segura y eficiente al usuario final.

Con la finalización de este paquete de desarrollo, la aplicación se considera completa. Este hito incluye no solo la interfaz de usuario y las funcionalidades asociadas, sino también la implementación efectiva de la base de datos y la operatividad continua de la API. Esta integración garantiza el funcionamiento fluido y coordinado de todos los componentes de la aplicación.

El siguiente paso implica proporcionar la aplicación a un grupo seleccionado de usuarios para su uso y evaluación. La retroalimentación obtenida de estos usuarios será invaluable para identificar posibles errores o áreas de mejora. Tras incorporar las sugerencias y realizar las mejoras necesarias, la aplicación estará lista para su despliegue en las plataformas correspondientes.

5.6. PT6 – PRUEBAS Y CALIDAD

La aplicación ha sido distribuida a un grupo selecto de usuarios, quienes han sido cuidadosamente elegidos para participar en esta fase inicial. Para facilitar la recopilación de opiniones y valoraciones, se ha proporcionado a estos usuarios un formulario de Google. Este formulario les permite compartir su retroalimentación sobre la aplicación, evaluar distintos aspectos de su funcionamiento y experiencia de usuario, y sugerir mejoras o reportar problemas específicos.

La selección manual de estos usuarios permite asegurar que el feedback recibido sea relevante y provenga de una muestra representativa del público objetivo de la aplicación. Este proceso es crucial para identificar áreas de mejora y garantizar que la aplicación cumpla con las expectativas y necesidades de sus usuarios finales antes de su lanzamiento oficial en las plataformas correspondientes.

Este enfoque iterativo hacia el desarrollo y mejora de la aplicación es fundamental para afinar la experiencia del usuario, corregir errores técnicos y enriquecer el contenido y funcionalidades según las preferencias del usuario.

Es importante destacar que los usuarios seleccionados incluyen tanto a usuarios de Apple como de Android, asegurando así la capacidad de probar la aplicación en ambos sistemas operativos.

5.7. PT7 – DESPLIEGUE EN ANDROID E IOS

Para el lanzamiento, se utilizaron las herramientas de Expo y EAS para compilar nuestra aplicación tanto para iOS como para Android. EAS es un conjunto de servicios de construcción y distribución que facilitan el proceso de desarrollo y lanzamiento de aplicaciones creadas con Expo. Ofrece una solución de construcción en la nube, permitiendo compilar aplicaciones para iOS y Android sin necesitar configurar entornos de desarrollo locales complejos.

EAS proporciona características avanzadas, como compilaciones personalizables, actualizaciones OTA, y la capacidad de manejar activos y secretos de manera más segura y

eficiente. Esto simplifica significativamente el proceso de preparación y publicación de aplicaciones en las tiendas de aplicaciones.

5.7.1. Build IOs y Android

En el caso de aplicaciones desarrolladas con herramientas como Expo y gestionadas a través de servicios como EAS (Expo Application Services), el build se realiza generalmente en la nube.

Para iOS, este proceso implica la creación de un archivo IPA (iOS App Store Package), que es el formato de paquete de aplicaciones utilizado por iOS para distribuir e instalar aplicaciones. Para Android, el proceso genera un archivo APK (Android Package Kit) o un archivo AAB (Android App Bundle), que son los formatos utilizados por Android para el mismo propósito.

Cada vez que se realice un nuevo build, es necesario actualizar el número de versión en el archivo `'app.json'`. Esto se hace para reflejar cambios o actualizaciones en la aplicación.

Una vez actualizada la versión en `'app.json'`, se ejecutará el comando `'npx expo prebuild'` para crear los archivos ejecutables de ios y Android, después ejecutaremos `'eas build --profile ios'` para iOS, y `'eas build --profile android'` para Android, iniciando así el proceso de compilación para cada plataforma respectivamente. [33]

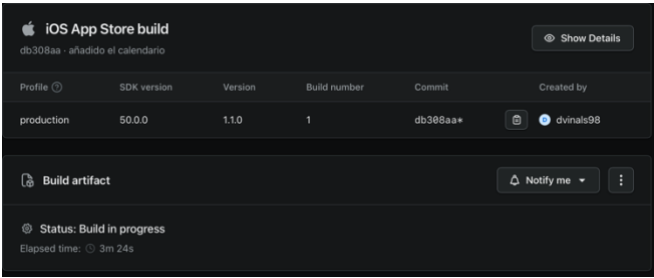


Ilustración 14: EAS Build para iOS. Fuente: EAS

Al ejecutar esta sentencia, los archivos `'ipa'` y `'apk'` estarán disponibles en la nube para ser descargados y posteriormente subidos a sus respectivas tiendas.

En la siguiente imagen se puede ver como el proceso de compilación ha terminado y se puede descargar el archivo `'ipa'`

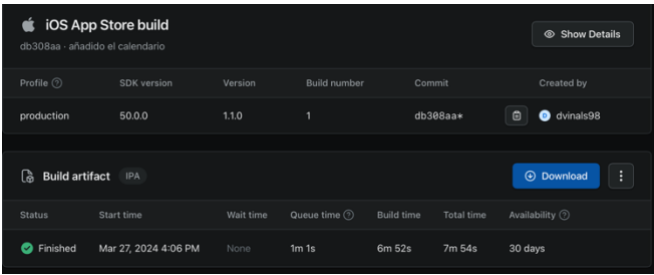


Ilustración 15: EAS Build iOS finalizado. Fuente: EAS.

5.7.2. Despliegue en iOS

Una vez que el archivo ‘.ipa’ ha sido descargado, el siguiente paso es cargarlo en la aplicación Transporter, herramienta oficial de Apple diseñada específicamente para este propósito.

Este proceso no solo incluye la carga de la aplicación en sí, sino también la entrega de metadatos esenciales, capturas de pantalla y otro contenido relacionado, lo cual es crucial para una presentación exitosa en la tienda. Transporter simplifica y agiliza el acceso de los desarrolladores al ecosistema de Apple, convirtiéndose en una herramienta indispensable para la distribución de aplicaciones en iOS. [34]

En la siguiente imagen se puede ver la aplicación cargada en Transporter.

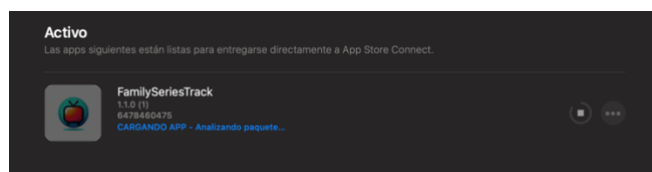


Ilustración 16: FST en Transporter. Fuente: Transporter App.

Una vez que la aplicación ha sido subida utilizando la aplicación Transporter, es necesario dirigirse al sitio web de App Store Connect. En esta plataforma, los desarrolladores deben complementar la carga con información adicional esencial para la publicación de la aplicación. Esto incluye la introducción de metadatos detallados, como la descripción de la aplicación, palabras clave, categorías pertinentes, información de contacto para soporte, y cualquier detalle relevante que facilite a los usuarios entender la funcionalidad y los beneficios de la aplicación.

La imagen muestra la aplicación con la versión 1.1.0 en Apple Store Connect, preparada para iniciar la fase de pruebas. Una vez completado este proceso, la aplicación estará lista para ser publicada en la App Store.

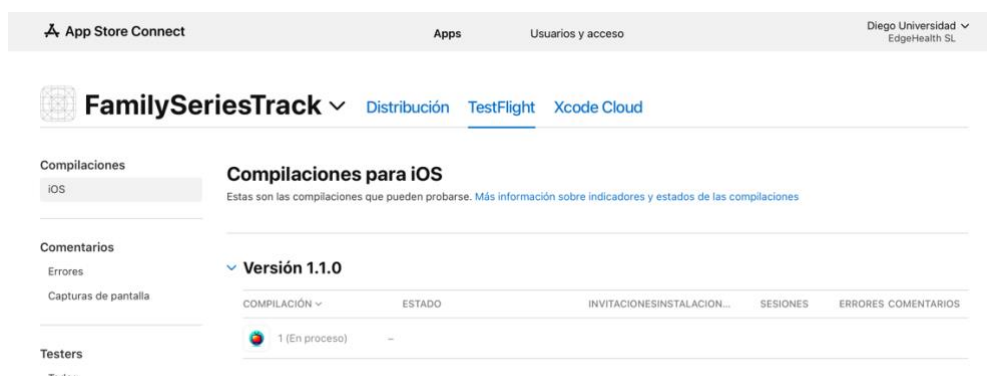


Ilustración 17: FST en Apple Store Connect. Fuente: Apple.

Además, es importante subir capturas de pantalla y videos de vista previa que muestren la interfaz de usuario y las características principales de la aplicación. Estos elementos visuales son fundamentales para captar la atención de los usuarios potenciales en el App Store. La versión de la aplicación también debe ser especificada, junto con notas de lanzamiento que describan las nuevas funciones o mejoras incluidas en la actualización actual.

Este paso en App Store Connect es importante para asegurar que la aplicación cumpla con todas las directrices de revisión de Apple y proporcione una experiencia clara y transparente para los usuarios finales. Solo después de completar adecuadamente todos estos requerimientos y enviar la aplicación para su revisión, estará en camino de ser aprobada y finalmente disponible en el App Store para su descarga por parte de los usuarios.

Se pudo completar todo este proceso y subir la aplicación al Apple Store de forma gratuita, como se puede ver en la imagen siguiente.

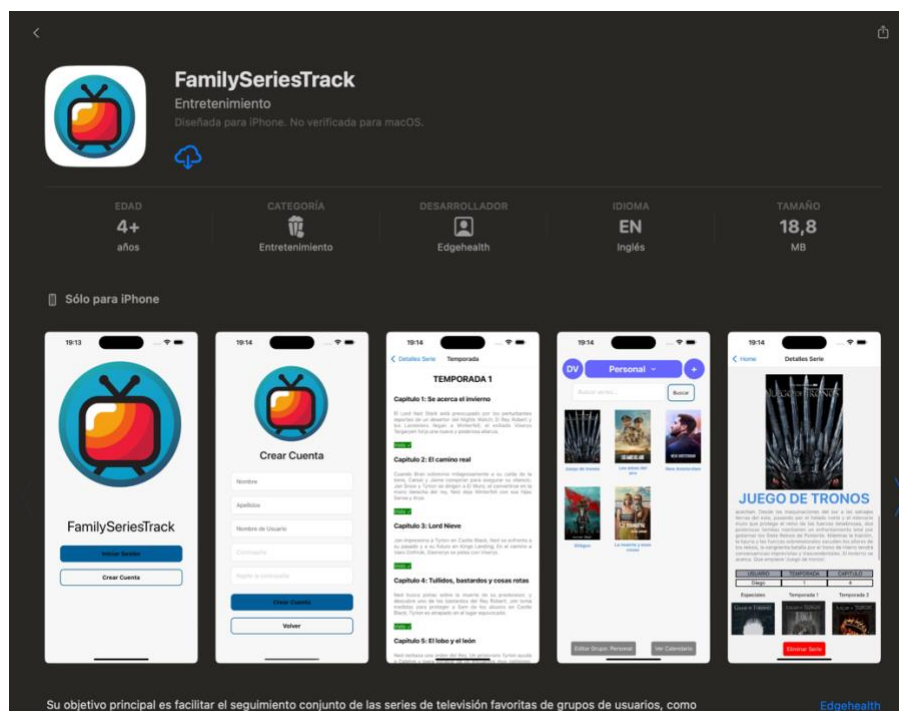


Ilustración 18: FST en Apple Store. Fuente: Apple Store.

5.7.3. Despliegue en Android

Para publicar una aplicación en Android, el primer paso esencial es crear una cuenta en Google Developer, requisito indispensable para poder subir aplicaciones a la plataforma.

Este proceso se caracteriza por su relativa simplicidad, especialmente en comparación con el proceso de publicación en iOS, ya que en Android se necesita principalmente una cuenta de Google Developer y el archivo `‘.apk’` o `‘.aab’` generado por `‘eas build’`.

Con este archivo descargado, abrimos Google Play Console, creamos una nueva versión y subimos el archivo `‘.aab’`. Una vez subido, tenemos un enlace disponible para poder testearla con testers internos, aunque necesitarán proporcionarnos su email. Luego se puede promocionar la aplicación a una prueba abierta, la cual tendrá que validar Google para verificar si es viable. Una vez pasado este periodo, Google nos proporciona un enlace privado con el que podemos descargar la aplicación. [35]

Es crucial completar con cuidado todos los detalles que Google Play solicita para tu aplicación. Esto incluye una descripción detallada, capturas de pantalla que muestren la funcionalidad y el diseño de la aplicación, y una clasificación de contenido adecuada. Estos elementos son

fundamentales para comunicar a los potenciales usuarios qué hace tu aplicación, cómo luce y para quién es apropiada.

Con todos los detalles completados a conciencia y la aplicación lista para el público, el siguiente paso es la publicación. Este proceso implica someter tu aplicación a una revisión por parte de Google Play, una etapa que varía en duración, pudiendo extenderse desde unas pocas horas hasta varios días.

Una vez la aplicación ha sido aprobada, estará disponible para su descarga e instalación por parte de los usuarios en Google Play Store.

6. RESULTADOS

El resultado del proyecto ha sido la creación de una aplicación móvil diseñada tanto para Android como para iOS. Esta aplicación ofrece una plataforma robusta y accesible, diseñada para satisfacer las necesidades específicas de los usuarios en ambos sistemas operativos, garantizando así una amplia accesibilidad y una experiencia de usuario coherente y eficiente en diferentes dispositivos.

6.1. RESULTADOS OBTENIDOS

Al enfocarse en facilitar un seguimiento conjunto, la aplicación ha introducido funciones innovadoras que no solo mejoran la experiencia de visualización, sino que también fomentan una interacción significativa entre los usuarios. Los resultados obtenidos son los siguientes.

Tabla 14: Resultados Obtenidos. Elaboración Propia.

Objetivo General	Facilitar el seguimiento conjunto de series de televisión favoritas para grupos de usuarios, como familias o amigos. Este objetivo se logra permitiendo la creación de grupos donde los usuarios pueden añadir las series que están viendo, marcar los capítulos como vistos, y visualizar el progreso de los demás miembros del grupo.
Mejorar la Experiencia	El sistema de seguimiento grupal mejora la experiencia de ver series ya que permite a los usuarios compartir sus progresos y descubrimientos.
Fomento de la Interacción	Mayor interacción y discusión entre los miembros sobre los contenidos visualizados, para ello se ha añadido un espacio de comentarios, esencial para la interacción en grupo.
Despliegue	La aplicación ha sido desplegada tanto en iOS como en Android, esto amplía la accesibilidad y utilidad para una mayor base de usuarios.

6.2. VALIDACION DE LOS RESULTADOS

A continuación, se detallará con precisión hasta qué punto se han alcanzado los objetivos específicos del proyecto.

Tabla 15: Validación Objetivos Específicos. Elaboración Propia.

Objetivo Especifico	Validación
OB-01	Se ha alcanzado con éxito a través de la aplicación desarrollada. Al permitir que cada integrante del grupo actualice su progreso individual en las series que están siguiendo, la aplicación ha mejorado significativamente la manera en que los grupos coordinan sus sesiones de visualización. Esta funcionalidad es especialmente útil en grupos donde los miembros tienen horarios variados o viven en diferentes zonas horarias, ya que cada persona puede ver a su propio ritmo sin perder la sincronía del grupo.
OB-02	<p>Este objetivo específico se ha cumplido efectivamente mediante la implementación de características en la aplicación que permiten una toma de decisiones colectiva más informada y democrática sobre qué series ver.</p> <p>Al proporcionar una plataforma donde todos los miembros del grupo pueden ver las series disponibles, junto con detalles como descripciones y episodios vistos por otros, se facilita la discusión y la selección consensuada de contenidos.</p> <p>La función de comentarios también juega un papel crucial, ya que permite a los usuarios expresar sus opiniones y preferencias sobre las series, fomentando un diálogo abierto que ayuda al grupo a tomar decisiones que satisfacen los intereses de todos. Así, la aplicación no solo mejora la organización de las sesiones de visualización, sino que también asegura que las elecciones hechas reflejen verdaderamente las preferencias del grupo, aumentando la satisfacción y la cohesión entre sus miembros.</p>
OB-03	<p>Este objetivo se ha logrado plenamente a través de la implementación de una pestaña de comentarios específica para cada serie dentro de la aplicación. Esta funcionalidad enriquece significativamente la experiencia de visualización compartida, permitiendo a los usuarios no solo seguir juntos la trama de las series, sino también compartir y discutir sus puntos de vista, teorías y emociones en tiempo real o después de ver un episodio.</p> <p>Al fomentar un espacio interactivo de discusión, la aplicación transforma la actividad de ver series de un acto pasivo a una experiencia interactiva y comunitaria.</p>
OB-04	Este objetivo se ha cumplido exitosamente, ya que la aplicación ha sido diseñada con una interfaz intuitiva y accesible que se adapta a diferentes dispositivos y tamaños de pantalla. Esta flexibilidad asegura que los usuarios puedan disfrutar de una experiencia de usuario coherente y eficiente, ya sea en smartphones, tabletas o incluso en pantallas más grandes.
OB-05	Este objetivo no se ha alcanzado completamente, ya que la aplicación no ofrece una pestaña de estadísticas detalladas de visualización. Aunque puedes saber cuántos capítulos has visto por temporada, no hay estadísticas completas disponibles.

7. IMPLICACIONES ÉTICAS E IMPACTO SOCIAL

7.1. INTRODUCCIÓN

En este apartado se reflexionará sobre las responsabilidades éticas que emergen en la creación de una plataforma tecnológica, las cuales no se limitan solo a los usuarios directos, sino que alcanzan un espectro más amplio en la sociedad.

Del mismo modo, examinaremos cómo nuestro proyecto puede influir en la comunidad, afectando comportamientos, normas sociales y la dinámica de grupo. Finalmente, discutiremos las leyes y regulaciones que enmarcan el desarrollo y lanzamiento de aplicaciones móviles, y cómo el cumplimiento de estas no solo es una obligación legal, sino también una declaración de nuestro compromiso con la ética profesional y el respeto por los derechos de nuestros usuarios.

7.2. DESARROLLO

En la actualidad, con el acceso inmediato a una amplia gama de series y películas gracias a las plataformas de streaming, es común que las personas opten por disfrutar de sus programas favoritos en solitario. Este fenómeno puede convertir la experiencia de ver series en algo bastante aislado. Sin embargo, la aplicación FST emerge como una solución con un significativo impacto social.

FST trasciende el concepto tradicional de las aplicaciones para ver series. Proporciona una forma novedosa y estimulante de hacer que la visualización de televisión sea una experiencia compartida. La aplicación permite a los usuarios comentar los episodios que ven y discutirlos con amigos o familiares, incluso si no se encuentran físicamente juntos. . Esta característica cobra especial relevancia para aquellos que, debido a circunstancias como el trabajo, estudios o incluso situaciones inesperadas como una pandemia, se encuentran lejos de sus seres queridos.

Discutir un episodio con otros puede enriquecer considerablemente la experiencia de visualización. Permite a los usuarios considerar diferentes perspectivas que quizás no habían evaluado anteriormente. Esta interacción no solo hace que el contenido sea más cautivador, sino que también puede fortalecer los lazos emocionales y cognitivos entre los usuarios que comparten sus opiniones y reacciones.

Al facilitar que los usuarios compartan sus experiencias al ver series, FST contribuye a que esta actividad sea menos individualista. En un contexto en el que parecemos estar constantemente absortos en nuestras pantallas, compartir lo que vemos puede mejorar nuestra conexión con los demás. Esto es crucial, ya que promueve una mejor comprensión y empatía entre las personas.

A largo plazo, una aplicación como FST tiene el potencial de transformar nuestras interacciones sociales. Al convertir la visualización de televisión en una actividad más comunitaria y menos aislada, FST podría fomentar una mayor unidad social, incluso en un mundo dominado por la tecnología.

7.2.1. Ley de Protección de Datos

En el ámbito del desarrollo de software, la protección de datos personales no solo constituye una obligación legal, sino que es también un compromiso ético fundamental. Las aplicaciones que manejan información sensible, tales como nombres, apellidos y contraseñas, deben cumplir rigurosamente con las leyes y regulaciones de protección de datos como la Ley General de Protección de Datos en Europa y leyes similares en otras regiones. Este cumplimiento garantiza la privacidad y la seguridad de los usuarios, reflejando un respeto profundo por su autonomía y dignidad.

El marco legal establecido por estas regulaciones obliga a las aplicaciones a implementar medidas técnicas y organizativas que prevengan el acceso no autorizado [36], la pérdida accidental o la destrucción de datos personales. En el desarrollo de este proyecto, se ha priorizado la integridad moral al implementar un sistema de cifrado de contraseñas. Este sistema no solo protege la información personal de los usuarios, sino que también refleja un compromiso ético con la protección de su privacidad.

El cifrado de contraseñas antes de su transmisión y almacenamiento en la base de datos es una medida de seguridad crítica. Este proceso garantiza que las contraseñas no sean accesibles en su forma original, protegiendo así la información contra potenciales interceptaciones o accesos no autorizados. Esta práctica no solo cumple con las exigencias regulatorias, sino que también manifiesta un respeto profundo por la confianza que los usuarios depositan en la tecnología.

Adoptar estas medidas de seguridad demuestra una responsabilidad ética hacia la protección de datos personales, subrayando la importancia de procesar esta información de manera segura. Al asegurar que las contraseñas y otros datos personales están cifrados, la aplicación no solo protege a los usuarios, sino que también eleva los estándares éticos de la industria, promoviendo un ambiente digital más seguro y confiable.

8. MI RECORRIDO EN LA UFV

8.1. EL PFG COMO CULMINACIÓN DE MI CAMINO UNIVERSITARIO

Durante mi trayectoria universitaria, he experimentado un notable crecimiento personal y en cuanto a mis objetivos de vida. Al ingresar a la universidad a los 21 años, mi enfoque principal era simplemente asistir a clases y aprobar asignaturas. A menudo, no dedicaba el tiempo necesario para el estudio, lo que me impedía aprobar todas las materias. En aquel entonces, mi futuro laboral me parecía algo lejano y difuso, sin gran interés de mi parte.

A lo largo de los años, he forjado valiosas amistades que han influido significativamente en mi desarrollo personal. Estos amigos no solo me apoyaron en los estudios y en la finalización de trabajos, sino que también me motivaron a comprometerme más con mis responsabilidades académicas, que era mi principal desafío.

Inicialmente, durante mis primeros años de carrera, no asistí a ninguna de las charlas organizadas por la AFC. Sin embargo, gracias al aliento de mis amigos, comencé a participar en ellas, lo que enriqueció enormemente mi formación. Un ejemplo memorable fue una charla sobre ética en la inteligencia artificial, que me ayudó a comprender los dilemas éticos en la IA y a distinguir entre lo que es ético y lo que no.

Hoy en día, me percibo como una persona trabajadora cuyo principal objetivo es terminar la carrera, ingresar al mundo laboral y formar una familia. Mis expectativas de vida han cambiado radicalmente: ahora me considero un adulto con claras aspiraciones a futuro, muy diferente de aquel joven cuyas únicas preocupaciones eran aprobar exámenes y planificar el fin de semana con amigos. Este cambio no solo refleja una maduración en mi actitud hacia los estudios y la vida profesional, sino también un reconocimiento de la importancia de la ética y la responsabilidad personal en mi futura carrera.

He elegido desarrollar este proyecto de fin de grado porque soy un apasionado de las series de televisión. En mi hogar, es tradición ver un capítulo con mi familia todos los días. Sin embargo, coordinar qué capítulo ver y mantener un registro de los episodios ya vistos puede ser complicado, especialmente con horarios tan variados. Una aplicación como la que propongo en este proyecto nos ayudaría enormemente a organizar y seguir nuestras series favoritas de manera más eficiente.

Además de mi amor por las series, otro factor que me motivó a embarcarme en este proyecto fue mi pasión por la programación. Siempre he disfrutado de los desafíos que ofrece el desarrollo de software, y este proyecto me brindó la perfecta oportunidad de aprender y aplicar técnicas nuevas de programación, como React Native. Este framework no solo es nuevo para mí, sino que también es altamente relevante en el desarrollo de aplicaciones móviles modernas. Aprender React Native no solo ha enriquecido mi conjunto de habilidades técnicas, sino que también ha aumentado mi entusiasmo por transformar una idea en una solución tecnológica funcional que podría beneficiar a muchas personas. Esta combinación de pasiones personales y profesionales hace de este proyecto de fin de grado una aventura especialmente significativa y motivadora para mí.

8.2. VINCULACIÓN CON MI FUTURO PROFESIONAL

La realización de este Proyecto Final de Grado ha sido una exploración reveladora de mis capacidades y pasiones, reafirmando mi deseo de trabajar en el desarrollo de software y soluciones tecnológicas que mejoren la vida cotidiana de las personas. A través de este proyecto, he descubierto cuán gratificante puede ser transformar una idea personal en una aplicación funcional, algo que ha intensificado mi interés en seguir una carrera en programación y desarrollo de aplicaciones móviles.

Al trabajar en este proyecto, he enfrentado desafíos que me han hecho cuestionar y a la vez fortalecer mi comprensión técnica y mi resiliencia, descubriendo que estoy más que listo y deseoso de llevar mis habilidades al ámbito profesional. Ahora, mirando hacia el futuro, estoy motivado para continuar aprendiendo y perfeccionando mis habilidades en tecnologías emergentes.

Gracias a este proyecto, me he dado cuenta de que lo que realmente me apasiona es programar y dar vida a ideas. En el futuro, me gustaría trabajar como programador o como coordinador de un equipo de programadores. Más adelante, aspiraría a ocupar roles de mayor responsabilidad, como el de mánager o un puesto ejecutivo, trabajando en proyectos de gran envergadura.

Este proyecto no solo ha servido como un puente hacia mi futuro profesional, sino que también ha solidificado mi convicción de que estoy en el camino correcto, buscando impactar positivamente en el mundo a través de la tecnología.

9. CONCLUSIONES

Se resumen a continuación las principales conclusiones derivadas del desarrollo de este proyecto. Adicionalmente, se presentan varias recomendaciones para futuras mejoras que podrían implementarse en el sistema para optimizar su funcionamiento y eficacia.

9.1. CONCLUSIONES

Las conclusiones que se han sacado de todo este proyecto son las siguientes:

- ✓ **Cumplimiento de Objetivos:** Los objetivos propuestos al inicio del proyecto se han alcanzado dentro de lo razonable, aunque no al 100%, la realización ha sido satisfactoria siguiendo fielmente la planificación establecida.
- ✓ **Desarrollo de la Aplicación:** Se ha desarrollado una aplicación funcional tanto para Android como para iOS, diseñada para facilitar la organización y seguimiento de series, tanto individualmente como en grupo.
- ✓ **Funcionalidades de la Aplicación:** Proporciona información detallada de cada serie, incluyendo descripciones y el número de capítulos. También incluye un espacio para comentarios, enriqueciendo la experiencia de visualización y permitiendo interacciones ricas, incluso con seres queridos a distancia.
- ✓ **Desarrollo Profesional y Técnico:** Me ha dado la oportunidad para profundizar en el uso de tecnologías como Docker, comprendiendo completamente su funcionamiento. También se ha aprendido a publicar la aplicación en Apple Store y Google Play, lo cual añade valor al perfil profesional. Logro de un hito significativo al subir la aplicación al Apple Store, lo que permite que cualquier persona pueda encontrar y utilizar la aplicación.
- ✓ **Desafíos y Soluciones:** Se ha aprendido un nuevo lenguaje de programación y adopción de una metodología diferente. Se ha resuelto de manera efectiva numerosos problemas a lo largo del proyecto, preparación adecuada para el ámbito profesional enseñando a ser resolutivo y autónomo en la búsqueda de soluciones.

9.2. TRABAJO A FUTURO

De cara al futuro, el potencial de expansión y mejora de la aplicación es prácticamente infinito. A medida que la tecnología evoluciona y las expectativas de los usuarios crecen, emergen nuevas oportunidades para enriquecer aún más la experiencia de visualización y gestión de series.

A continuación, se detalla una serie de mejoras y adiciones propuestas que se consideran podrían transformar significativamente cómo los usuarios interactúan con la plataforma y entre sí.

- ✓ **Notificaciones en Tiempo Real:** Implementar un sistema que alerte a los usuarios cuando se vea un capítulo nuevo o se haga un comentario en una serie, fomentando la interacción y el compromiso continuo. Esta funcionalidad se quiso implementar, pero no fue posible por la complejidad y el poco tiempo disponible.
- ✓ **Recomendaciones Basadas en Inteligencia Artificial:** Integrar IA para recomendar series basadas en las preferencias y el historial de visualización de los usuarios, lo que podría enriquecer significativamente la experiencia del usuario.
- ✓ **Detector de Spoilers mediante IA:** Desarrollar una funcionalidad que use inteligencia artificial para detectar y evitar spoilers, protegiendo a los usuarios de revelaciones no deseadas sobre tramas o desenlaces.
- ✓ **Visualización de Estadísticas Detalladas:** Incorporar herramientas que permitan a los usuarios visualizar estadísticas de su progreso de visualización tanto a nivel individual como grupal, facilitando la comparación y el compartir experiencias dentro de grupos de visualización.
- ✓ **Autenticación Biométrica:** Planea retomar e implementar la autenticación local utilizando métodos biométricos como FaceID y reconocimiento de huellas dactilares para facilitar un acceso seguro y cómodo a la aplicación.
- ✓ **Decisión de Serie para Ver en Grupo:** Añadir una funcionalidad que analice las preferencias de todos los miembros del grupo y, mediante algoritmos de votación o consenso, decida cuál serie del catálogo conjunto sería ideal para ver juntos. Esta herramienta puede considerar factores como géneros preferidos, series no vistas por algún miembro y la popularidad de las series entre los miembros del grupo.

10. OTROS MÉRITOS DEL PROYECTO

Como méritos adicionales de este proyecto, es relevante mencionar la implementación de páginas web de soporte y política de privacidad, las cuales no estaban originalmente previstas en los resultados esperados del desarrollo de la aplicación. Estas páginas se volvieron indispensables para cumplir con los requisitos de publicación de las tiendas de aplicaciones de Apple y Android.

La creación de estas páginas web fue posible gracias a la infraestructura ya establecida utilizando Docker, junto con los servicios de Cloudflare y Traefik. La configuración existente proporcionó una base sólida que permitió un despliegue eficiente y relativamente sencillo de estas soluciones web adicionales, pese a no estar contempladas en los requerimientos iniciales del proyecto.

Estas páginas no solo cumplen con los criterios necesarios para la publicación de la aplicación en los respectivos mercados digitales, sino que también ofrecen un valor añadido en términos de servicio al cliente y transparencia. La página de soporte ofrece una vía directa para que los usuarios obtengan ayuda y resuelvan sus dudas o problemas con la aplicación, mejorando así la experiencia del usuario y la satisfacción general. Por otro lado, la página de política de privacidad detalla cómo se manejan los datos personales de los usuarios, reforzando la confianza en la aplicación al asegurar el cumplimiento de las normativas vigentes sobre privacidad y protección de datos.

Las páginas se pueden encontrar en '<https://soportefst.lapspartbox.com>' y en '<https://privacidadfst.lapspartbox.com>'

El proceso de compilación y despliegue de la aplicación merece una mención especial por la complejidad y el aprendizaje autodidacta que conllevó. Utilizando Expo Application Services (EAS) para la compilación, el proyecto se enfrentó a numerosos desafíos técnicos que exigieron una capacidad de resolución de problemas de manera independiente y efectiva. A lo largo de este proceso, se identificaron y corrigieron varios errores, lo cual es un testimonio de la destreza técnica y la perseverancia del equipo de desarrollo.

Además, el despliegue de la aplicación en las tiendas de aplicaciones no solo implicó un proceso técnico de compilación, sino también una extensa investigación y aprendizaje en áreas previamente desconocidas. Esto incluyó familiarizarse con los protocolos y políticas específicas de la App Store de Apple y Google Play Store, cada una con su conjunto de requerimientos y lineamientos para aceptar y publicar nuevas aplicaciones.

La tarea de desplegar en ambas tiendas involucró una comprensión detallada de los pasos necesarios para el envío de aplicaciones, la gestión de certificados y perfiles de aprovisionamiento, y el cumplimiento de las políticas de cada plataforma en términos de privacidad, funcionalidad y diseño. Además, se aseguró de que todos los criterios, que eran extensos y complejos, se cumplieran adecuadamente.

11. BIBLIOGRAFÍA

- [1] «Globamatic,» 15 Mayo 2023. [En línea]. Available: <https://www.globamaticmedia.com/que-es-filmaffinity-y-para-que-se-usa/>. [Último acceso: 29 Febrero 2024].
- [2] M. D. Hernández, «Hipertextual,» 18 Enero 2015. [En línea]. Available: <https://hipertextual.com/2015/01/aplicaciones-para-seguir-series>. [Último acceso: 28 Febrero 2024].
- [3] Y. Fernández, «Xataka,» 15 Octubre 2022. [En línea]. Available: <https://www.xataka.com/basics/14-mejores-servicios-apps-para-seguir-controlar-series-peliculas-que-ves-tener-toda-su-informacion>. [Último acceso: 28 Febrero 2024].
- [4] S. Arteaga, «ComputerHoy,» 3 Noviembre 2018. [En línea]. Available: <https://computerhoy.com/reportajes/entretenimiento/como-llevar-seguimiento-series-que-estas-viendo-320649>. [Último acceso: 28 Febrero 2024].
- [5] i. Sommerville, Software Engineering, Pearson, 2016.
- [6] G. Garcés, 7 Enero 2022. [En línea]. Available: <https://www.hiberus.com/crecemos-contigo/ventajas-de-usar-figma-como-herramienta-de-diseno-ui/>. [Último acceso: 3 Abril 2024].
- [7] «Deloitte,» [En línea]. Available: ¿Qué es React Native?. [Último acceso: 24 Marzo 2024].
- [8] D. Borovskoy, «LinkedIn,» 14 Septiembre 2023. [En línea]. Available: <https://es.linkedin.com/pulse/qué-es-expo-10-características-que-tenes-saber-2023-denis-borovskoy>. [Último acceso: 1 Abril 2024].

- [9] AWS, «AWS,» [En línea]. Available: <https://aws.amazon.com/es/docker/>. [Último acceso: 26 Febrero 2024].
- [10] PureStorage, «PureStorage,» [En línea]. Available: <https://www.purestorage.com/es/knowledge/what-is-mariadb.html>. [Último acceso: 27 Febrero 2024].
- [11] F. G. d. Zúñiga, «Arsys,» 15 Noviembre 2021. [En línea]. Available: <https://www.arsys.es/blog/phpmyadmin>. [Último acceso: 17 Abril 2024].
- [12] L. Codina, «Lluís Codina,» 31 Enero 2017. [En línea]. Available: <https://www.lluiscodina.com/bases-de-datos-de-cine-y-television/>. [Último acceso: 5 Abril 2024].
- [13] «Kinsta,» 13 Septiembre 2023. [En línea]. Available: <https://kinsta.com/knowledgebase/what-is-express-js/>. [Último acceso: 29 Febrero 2024].
- [14] Aplpe, «Apple,» [En línea]. Available: https://itunespartner.apple.com/movies/articles/transporter_getting-set-up. [Último acceso: 2 Marzo 2024].
- [15] GooApps, «GooApps,» 21 Febrero 2022. [En línea]. Available: <https://gooapps.es/2022/02/21/cuanto-cuesta-subir-una-aplicacion-a-una-app-store/>. [Último acceso: 25 Abril 2025].
- [16] Apple, «Preventing Insecure Network Connections,» [En línea]. Available: https://developer.apple.com/documentation/security/preventing_insecure_network_connections#. [Último acceso: 31 Marzo 2024].
- [17] Sanjeev, «StackOverflow,» 18 Mayo 2022. [En línea]. Available: <https://stackoverflow.com/questions/24184579/how-to-properly-write-a-gradle-wrapper-properties-file>. [Último acceso: 2024 Marzo 5].
- [18] ChatGPT, 25 Abril 2024. [En línea]. Available: <https://chat.openai.com/share/10e3510c-3127-4073-8256-016bdc987b43>. [Último acceso: 24 Abril 2024].

- [19] Life360. [En línea]. Available: <https://apps.apple.com/us/app/life360-buscar-familia-amigos/id384830320?l=es-MX>. [Último acceso: 3 Abril 2024].
- [20] E. Fedorenko, Designing in Figma: The complete guide to designing with reusable components and styles in Figma, Independently published, 2020.
- [21] T. Kurek, «Canonical Ubuntu,» 24 Abril 2020. [En línea]. Available: <https://ubuntu.com/blog/ubuntu-server-20-04>. [Último acceso: 14 Marzo 2024].
- [22] «tecnofaq,» [En línea]. Available: <https://tecnofaq.com/es-docker-mejor-windows-o-linux/>. [Último acceso: 14 Marzo 2024].
- [23] Soloelectronicos, «Soloelectronicos,» 4 Diciembre 2021. [En línea]. Available: https://soloelectronicos.com/2021/12/04/configuracion-de-docker-para-windows-y-wsl-para-funcionar-sin-problemas/?utm_content=cmp-true. [Último acceso: 14 Marzo 2024].
- [24] A. Chacón, 7 Julio 2023. [En línea]. Available: <https://a-chacon.com/docker/mariadb/phpmyadmin/2023/07/07/docker-compose-mariadb-phpmyadmin.html>. [Último acceso: 26 Febrero 2024].
- [25] B. Öggl y M. Kofler, Docker: Practical Guide for Developers and DevOps Teams, Rheinwerk, 2023.
- [26] J. Turnbull, The Docker Book: Containerization is the new virtualization, 2015.
- [27] M. Morejón, «Manuel Morejón,» 23 Febrero 2021. [En línea]. Available: <https://mmorejon.io/blog/traefik-2-configuracion-avanzada-docker-compose/>. [Último acceso: 4 Febrero 2024].
- [28] J. M. Ramírez, «MasQteclas,» 29 Marzo 2020. [En línea]. Available: <https://www.masquetecclas.com/curso/docker-qnap/4-proxy-inverso-traefik-2/>.
- [29] C. R. Serrano, APRENDE REACT NATIVE DESDE CERO EN ESPAÑOL: DESARROLLA APLICACIONES HIBRIDAS Y HERMOSAS CON JAVASCRIPT, 2021.

- [30 Expo, «Expo,» [En línea]. Available: <https://docs.expo.dev/tutorial/create-your-first-app/>.
] [Último acceso: 27 Marzo 2024].
- [31 S. Patni, Pro RESTful APIs: Design, Build and Integrate with REST, JSON, XML and JAX-RS,
] Apress, 2017.
- [32 B. Jin, S. Sahni y A. Shevat , Designing Web APIs: Building APIs That Developers Love,
] O'Reilly Media, 2018.
- [33 Expo, «Expo Docs,» [En línea]. Available: <https://docs.expo.dev/build/setup/>. [Último acceso:
] 23 Marzo 2024].
- [34 o. ak, iOS app Development for Novices - iOS App Creation with Swift, Xcode - Paving Your
] Way to App Store Success, Independently published , 2024.
- [35 Android App Development: From Concept to Code, Independently published, 2023.
]
- [36 BOE, «Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía
] de los derechos digitales.,» Madrid, 2018.

ANEXO A: REQUISITOS DE USUARIO

En este anexo se detallan los requisitos de usuario, estos requisitos son el resultado del paquete de trabajo **PT01-AR**.

REQUISITOS FUNCIONALES

Los requisitos funcionales definen las acciones, procesos y capacidades que un sistema o aplicación debe ofrecer para cumplir con los requisitos y demandas de los usuarios. Constituyen la base sobre la cual se construye y desarrolla el software o sistema, detallando las funcionalidades y servicios necesarios que debe proporcionar. Estos criterios son fundamentales para el proceso de desarrollo, ya que especifican de manera precisa las tareas, funciones y operaciones que el sistema debe ser capaz de realizar, asegurando que se cumplan las metas y objetivos de los usuarios de forma efectiva.

Se clasificarán los defectos en tres niveles de severidad: muy críticos (3), críticos (2) y no críticos (1). Esta categorización nos permitirá priorizar eficientemente las correcciones, asegurándonos de que los errores que más impactan la experiencia del usuario y la funcionalidad de la aplicación sean atendidos con la máxima urgencia.

Tabla 16: Requisitos Funcionales. Elaboración Propia.

Código	Requisitos funcionales	Criticidad
RF-01	Los usuarios deben poder registrarse en la aplicación usando un nombre de usuario.	3
RF-02	Los usuarios deben poder iniciar sesión usando su nombre de usuario y contraseña.	3
RF-03	La aplicación permitirá editar nombre, apellidos, nombre de usuario y contraseña.	3
RF-04	Los usuarios deben poder crear grupos familiares o de amigos.	3
RF-05	Los usuarios deben poder unirse a grupos existentes cuando el usuario principal crea el grupo.	3
RF-06	Los usuarios deben poder ver una lista de todos los miembros en sus grupos.	3
RF-07	Los usuarios deben poder agregar series a su lista personal y grupal.	3
RF-08	Los usuarios deben poder marcar episodios como vistos.	3
RF-09	Los usuarios deben poder marcar episodios como no vistos.	3

Código	Requisitos funcionales	Criticidad
RF-10	La aplicación debe mostrar el progreso de visualización de las series para cada miembro del grupo.	3
RF-11	Los usuarios deben recibir notificaciones cuando un miembro del grupo marca un episodio como visto.	2
RF-12	Los usuarios deben ser notificados sobre los próximos episodios de sus series agregadas.	2
RF-12	La aplicación debe ofrecer un calendario con las fechas de estreno de nuevos episodios.	2
RF-13	Los usuarios deben poder programar sesiones de visualización grupales en el calendario.	1
RF-14	Los usuarios deben poder calificar y comentar sobre episodios y series.	3
RF-15	La aplicación debe permitir discusiones grupales dentro de cada grupo por serie.	3
RF-16	La aplicación no permitirá entrar en el dialogo de los episodios no vistos para evitar <i>spoilers</i> .	2
RF-17	La aplicación debe ofrecer una opción para que los usuarios puedan dejar de seguir o eliminar series de su lista.	3
RF-18	La aplicación debe permitir a los usuarios ver estadísticas de su actividad de visualización, como el total de episodios vistos o el tiempo total invertido en ver series.	1
RF-19	Los usuarios deben poder filtrar las series y episodios en su lista según diferentes criterios, como género, año de lanzamiento, o si están vistos/no vistos.	1
RF-20	La aplicación debe permitir a los usuarios configurar recordatorios para las fechas de estreno de nuevos episodios.	2
RF-21	Los usuarios deben poder buscar series por nombre.	3
RF-22	Los usuarios deben poder compartir su actividad o recomendaciones de series en redes sociales directamente desde la aplicación.	1
RF-23	La aplicación debe soportar la creación de listas personalizadas de series o episodios, como "Favoritos", "Para ver", o "Vistos recientemente".	1
RF-24	La aplicación debe permitir a los usuarios personalizar las notificaciones que desean recibir, como notificaciones de nuevos episodios, cambios en el grupo, o recomendaciones.	2
RF-25	Los usuarios deben poder invitar a nuevos miembros a sus grupos a través de correo electrónico o un enlace de invitación.	2
RF-26	La aplicación debe permitir a los usuarios establecer metas de visualización, como completar una serie antes de una fecha específica.	2
RF-27	La aplicación debe proporcionar resúmenes de episodios para ayudar a los usuarios a ponerse al día rápidamente sin necesidad de ver episodios anteriores.	3
RF-28	Los usuarios deben poder sortear episodios o series de forma aleatoria para decidir qué ver a continuación.	2
RF-29	Debe existir una funcionalidad para exportar e importar la lista de series y el progreso de visualización, facilitando la transición entre dispositivos o la compartición de listas con otros usuarios.	1

<i>Código</i>	<i>Requisitos funcionales</i>	<i>Criticidad</i>
<i>RF-30</i>	Los usuarios deben poder configurar alertas para recordatorios de sesiones de visualización grupales planificadas.	2
<i>RF-31</i>	Los usuarios deben poder acceder a un historial de cambios para cada serie, incluyendo actualizaciones de episodios y modificaciones en la información de la serie.	1
<i>RF-32</i>	La aplicación debe permitir a los usuarios cambiar el orden de visualización de las series en su lista, ya sea de forma manual o según criterios predefinidos (por ejemplo, alfabéticamente, por fecha de añadido, etc.).	2
<i>RF-33</i>	La aplicación debe ofrecer la posibilidad de visualizar trailers o teasers de series directamente dentro de la plataforma.	1
<i>RF-34</i>	Los usuarios deben poder ver una clasificación de las series más populares dentro de la aplicación, basada en las interacciones y preferencias de todos los usuarios.	1

REQUISITOS NO FUNCIONALES

Los requisitos no funcionales se refieren a las cualidades, estándares y atributos que un sistema o aplicación debe poseer, pero que no están directamente vinculados con las funciones específicas que realiza. Estos requisitos abarcan aspectos como el rendimiento, seguridad, usabilidad, compatibilidad, y escalabilidad, estableciendo cómo debe comportarse el sistema en términos de eficiencia, fiabilidad y otras características clave. Mientras que los requisitos funcionales se centran en "qué" debe hacer el sistema, los no funcionales se enfocan en el "cómo" debe operar y ser experimentado por los usuarios, garantizando que la solución final no solo sea funcional sino también robusta, segura y agradable de usar.

Tabla 17: Requisitos No Funcionales

<i>Código</i>	<i>Requisitos no funcionales</i>
<i>RNF-01</i>	La interfaz de usuario debe ser intuitiva y fácil de navegar para usuarios de todas las edades.
<i>RNF-02</i>	La aplicación debe permitir discusiones grupales dentro de cada grupo por serie.
<i>RNF-03</i>	La aplicación debe cargar las páginas y los datos de las series en menos de 2 segundos.
<i>RNF-04</i>	La aplicación debe ser capaz de manejar el aumento de usuarios y datos sin degradar el rendimiento.
<i>RNF-05</i>	La aplicación debe implementar controles de acceso para asegurar que solo los miembros autorizados puedan ver o editar información sensible del grupo.
<i>RNF-06</i>	Todos los datos personales y de actividad deben ser encriptados tanto en tránsito como en reposo para proteger contra accesos no autorizados.
<i>RNF-07</i>	La aplicación debe ser compatible con las versiones más recientes de los sistemas operativos iOS (iOS14) y Android (Android13).

<i>Código</i>	<i>Requisitos no funcionales</i>
<i>RNF-08</i>	La aplicación debe estar optimizada para funcionar en una amplia gama de dispositivos móviles, incluyendo tabletas y smartphones, con diferentes tamaños de pantalla y resoluciones.
<i>RNF-09</i>	Debe soportar la integración con otras aplicaciones y servicios de terceros, como sistemas de calendario y redes sociales, mediante APIs.
<i>RNF-10</i>	La aplicación debe incluir un sistema de registro de errores (logging) detallado para facilitar el diagnóstico y la solución de problemas.
<i>RNF-11</i>	Debe haber un proceso establecido para la actualización y el despliegue de nuevas versiones de la aplicación, minimizando el tiempo de inactividad y garantizando la compatibilidad hacia atrás.

ANEXO B: DISEÑO DEL SISTEMA

DIAGRAMA

Para ilustrar más claramente cómo está montado el sistema y las conexiones entre los contenedores Docker y el servidor, sería útil crear un diagrama detallado que muestre cada componente y su interacción.

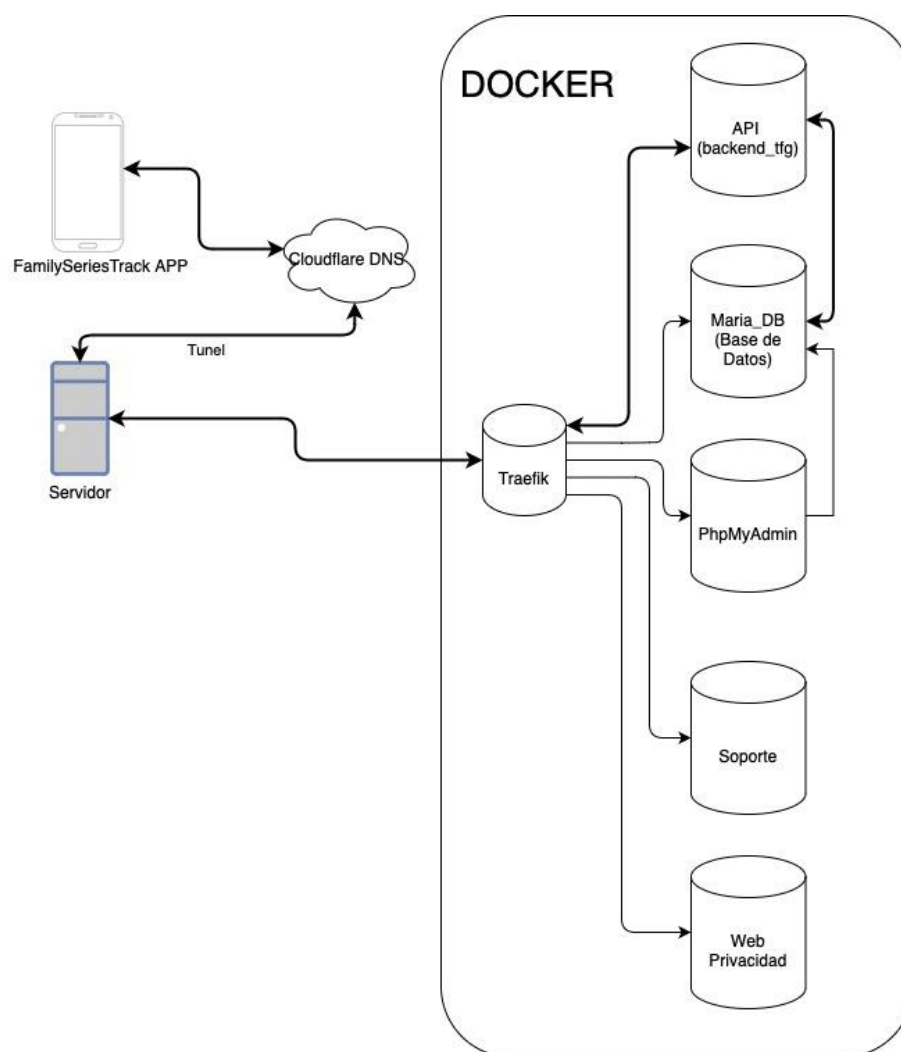


Ilustración 19: Diseño del Sistema. Elaboración Propia

DETALLES

Los componentes del sistema son los siguientes:

- ✓ FamilySeriesTrack APP: Es la aplicación móvil que los usuarios finales utilizan para interactuar con el servicio.
- ✓ Cloudflare DNS: Un servicio de gestión de DNS que también proporciona seguridad y rendimiento. Al hacer una solicitud, la app primero contacta con Cloudflare.
- ✓ Túnel: Este representa una conexión segura entre Cloudflare y el servidor, esta usando Cloudflare Tunnel, lo que garantiza que el tráfico esté protegido de amenazas externas y no esté expuesto directamente en internet.
- ✓ Servidor: La máquina donde se ejecutan los contenedores Docker.
- ✓ Traefik: Es un proxy inverso y balanceador de carga que maneja las conexiones entrantes y las dirige hacia el contenedor apropiado.
- ✓ API (backend_tfg): La API del backend, que gestiona las lógicas de llamada de la aplicación para la base de datos.
- ✓ MariaDB (Base de Datos): El sistema de gestión de bases de datos donde se almacenan los datos necesarios para la aplicación.
- ✓ PhpMyAdmin: Una interfaz web para administrar la base de datos MariaDB.
- ✓ Soporte y Web Privacidad: Contenedores adicionales que son webs en el que se da soporte e información sobre la privacidad.

A continuación, vamos a examinar el flujo del sistema cuando un usuario de la aplicación desea crear un perfil (se puede ver en el diagrama con la línea con más grosor):

1. Inicio de la Solicitud: El usuario utiliza la aplicación para crear un nuevo usuario, llenando los datos necesarios y enviando el formulario.
2. DNS y Túnel: La APP envía la solicitud al servidor a través de la resolución de DNS de Cloudflare y a través del túnel seguro establecido entre Cloudflare y el servidor.
3. Enrutamiento con Traefik: Traefik recibe la solicitud y la dirige al contenedor de la API.
4. Procesamiento en la API: La API (backend_tfg) procesa la solicitud. Valida los datos proporcionados y ejecuta la lógica de creación de usuario.
5. Interacción con la Base de Datos: Si la validación es exitosa, la API envía una solicitud a MariaDB para crear un nuevo registro en la tabla de usuarios.
6. Confirmación y Respuesta: Una vez que el usuario se crea en la base de datos, la API devuelve una respuesta a Traefik, que a su vez la envía de vuelta a través de Cloudflare y finalmente al dispositivo del usuario.
7. Finalización en la APP: La aplicación móvil recibe la respuesta y, dependiendo del resultado (éxito o error), muestra un mensaje correspondiente al usuario.

ANEXO C: DETALLES DE LA BBDD

Para proporcionar una explicación más profunda y precisa sobre el esquema de la base de datos, primero desglosaremos sus componentes principales y luego detallaremos cada uno de ellos.

A continuación, cada aspecto será explicado con un nivel de detalle que facilitará la comprensión de cómo interactúan las diferentes partes del esquema para cumplir con los requerimientos específicos del sistema de base de datos.

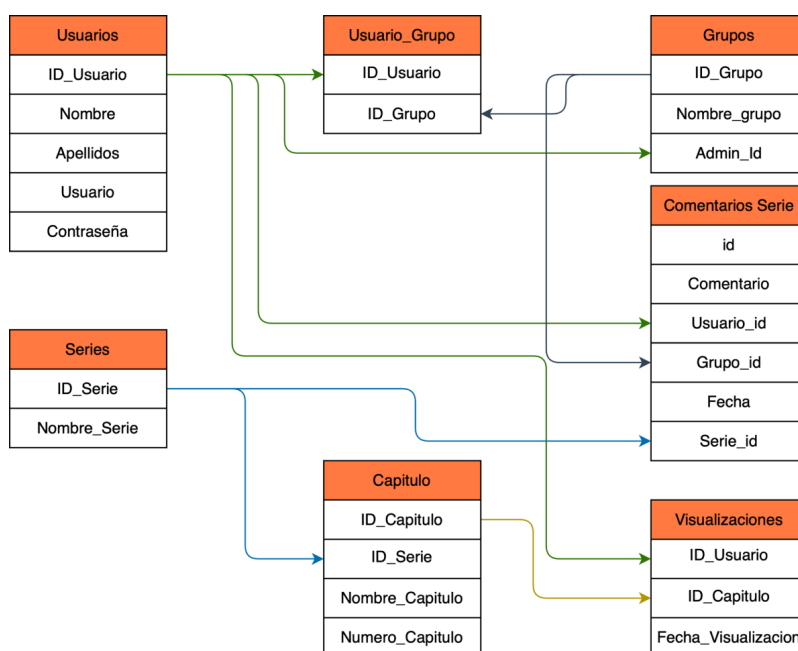


Ilustración 20: Esquema BBDD. Elaboración Propia.

USUARIOS

Esta tabla contiene toda la información con respecto al usuario de la aplicación, necesario para el inicio de sesión. En la pestaña crear usuario se rellena la información de esta tabla.

- ✓ ID_Usuario: Este sería el identificador único para cada usuario en la base de datos.
- ✓ Nombre: Este campo se utiliza para almacenar el nombre propio del usuario.

- ✓ Apellidos: Similar al campo de Nombre, pero utilizado para almacenar el apellido o los apellidos del usuario.
- ✓ Usuario: Este es el nombre de usuario que la persona utilizará para iniciar sesión en la aplicación o sistema. Es una representación alfanumérica que el usuario elige al crear una cuenta y suele ser única en el sistema para evitar confusiones.
- ✓ Contraseña: Aquí se almacenaría la contraseña del usuario. Cabe destacar que esta contraseña se cifra para no saber cuál es la original.

Usuarios
ID_Usuario
Nombre
Apellidos
Usuario
Contraseña

Ilustración 21: Tabla Usuarios. Elaboración Propia

USUARIO_GRUPO

Para gestionar la complejidad de las relaciones entre usuarios y grupos, se emplea la tabla "Usuario_Grupo". Esta tabla actúa como un puente en la relación muchos a muchos entre usuarios y grupos, permitiendo una asociación flexible en la que un usuario puede pertenecer a múltiples grupos y, a su vez, un grupo puede comprender varios usuarios. Esta relación es esencial para organizar los usuarios en categorías o equipos.

- ✓ ID_Usuario: Es el ID del usuario que está en el grupo.
- ✓ UD_Grupo: ID del grupo del usuario.

Usuario_Grupo
ID_Usuario
ID_Grupo

Ilustración 22: Tabla Usuario_Grupo. Elaboración Propia

GRUPOS

Por otro lado, la tabla "Grupos" se dedica exclusivamente a almacenar la información de los grupos formados por los usuarios. Cada entrada en esta tabla contiene el identificador único

del grupo y su nombre, lo que permite una identificación clara y directa de cada colectivo dentro del sistema.

- ✓ ID_Grupo: Id único para identificar el grupo,
- ✓ Nombre_Grupo: Nombre del grupo que se ha creado
- ✓ Admin_id: El id del creador del grupo. Solo este podrá eliminar el grupo por completo.

Grupos
ID_Grupo
Nombre_grupo
Admin_Id

Ilustración 23: Tabla Grupos. Elaboración Propia

SERIES

Adicionalmente, la interacción de los usuarios con el contenido multimedia se refleja en la tabla "Series", que está directamente vinculada a la tabla "Usuarios" mediante el identificador del usuario. Esta tabla alberga la información de las series asociadas a cada usuario, permitiendo así un seguimiento personalizado de los contenidos de interés para cada individuo.

- ✓ ID_Serie: Es el identificador único de la serie, este nos viene dado desde la API The Movie Database.
- ✓ ID_Usuario: Es el id del usuario que está viendo esa serie, nos viene dado de la tabla Usuarios.

Series
ID_Serie
ID_Usuario

Ilustración 24: Tabla Series. Elaboración Propia

CAPÍTULO

La estructura se expande con la tabla "Capítulo", que detalla cada episodio perteneciente a las series. Aquí se almacena el identificador del capítulo, a qué serie pertenece, el nombre del capítulo, su número y la temporada correspondiente.

- ✓ ID_Capitulo: ID único del capítulo, dado por la API The Movie Database
- ✓ ID_Serie: ID de la serie de ese capítulo.
- ✓ Nombre_Capitulo: Nombre del capítulo.

- ✓ Numero_Capitulo: Numero del capítulo.
- ✓ Numero_Temporada: Numero de la temporada a la que pertenece ese capítulo.

Capitulo
ID_Capitulo
ID_Serie
Nombre_Capitulo
Numero_Capitulo
Número Temporada

Ilustración 25: Tabla Capítulos. Elaboración Propia

VISUALIZACIONES

Finalmente, la tabla "Visualizaciones" juega un papel crucial al registrar las interacciones específicas de los usuarios con los capítulos. Mediante la conexión entre las tablas "Usuarios" y "Capitulo" a través de sus identificadores y el registro de la fecha de visualización, se logra un seguimiento detallado del consumo de contenidos. Esta información es invaluable para comprender las preferencias de los usuarios y mejorar las recomendaciones de contenidos, así como para análisis estadísticos y de comportamiento dentro del sistema.

- ✓ ID_Usuario: Id del usuario que ha visto ese capítulo.
- ✓ Id_Capitulo: Id del capitulo que se ha visto.
- ✓ Fecha_Visualizacion: Fecha de cuando se vio el capitulo.

Visualizaciones
ID_Usuario
ID_Capitulo
Fecha_Visualizacion

Ilustración 26: Tabla Visualizaciones. Elaboración Propia

En conjunto, esta estructura de base de datos no solo garantiza una organización eficiente y sistemática de la información relacionada con los usuarios y los contenidos multimedia, sino que también facilita la implementación de funcionalidades avanzadas y personalizadas, mejorando significativamente la experiencia del usuario en el sistema.

COMENTARIOS SERIE

Esta tabla es una tabla donde se guardan los comentarios que se han hecho sobre una serie de televisión.

- ✓ Id: Este es el identificador único de cada comentario.
- ✓ Comentario: Este campo almacena el texto del comentario que el usuario ha hecho sobre una serie.
- ✓ Usuario_id: Este campo relaciona el comentario con el usuario que lo ha hecho. Sería una clave foránea que apunta al identificador único de un usuario en la tabla de usuarios. Esto permite rastrear quién ha realizado cada comentario.
- ✓ Grupo_id: Este campo indicaría si el comentario está asociado a un grupo específico, lo cual es útil si la aplicación permite a los usuarios formar grupos para discutir series. Sería otra clave foránea que apunta al identificador único de un grupo en una tabla de grupos.
- ✓ Fecha: Este campo registra la fecha y hora en que se hizo el comentario. Es importante para mostrar cuándo se publicó cada comentario y para mantener un registro cronológico.
- ✓ Serie_id: Finalmente, este campo es una clave foránea que vincula el comentario con una serie de televisión específica en la base de datos. Esto permite saber a qué serie pertenece cada comentario.

Comentarios Serie
id
Comentario
Usuario_id
Grupo_id
Fecha
Serie_id

Ilustración 27: Tabla Comentarios Serie. Elaboración propia.

ANEXO D: DETALLES DE PANTALLAS

PANTALLA BIENVENIDA

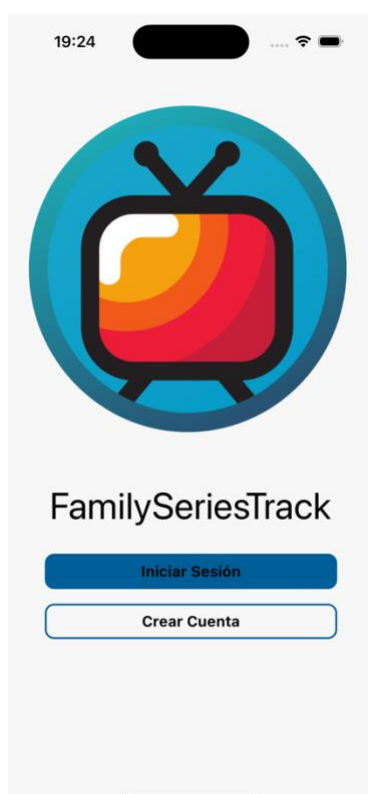


Ilustración 28: Pantalla Bienvenida. Elaboración Propia.

El estado local `todosUsuarios` se utiliza para almacenar información de usuarios obtenida a través de una solicitud HTTP. Hay funciones como `imprimirUsuarios`, que imprime detalles de los usuarios en la consola, y dos manejadores de eventos, `handleLoginPress` y `handleCreateAccountPress`, para los botones de iniciar sesión y crear cuenta, respectivamente.

El primero realiza una solicitud a un endpoint de una API para obtener los usuarios, los almacena en el estado `todosUsuarios`, y luego navega a la pantalla de inicio de sesión (`LogInScreen`), mientras que el segundo simplemente navega a la pantalla de registro de nuevos usuarios (`SignUp`).

El componente también incluye un logo y botones para iniciar sesión o crear una nueva cuenta, aplicando estilos globales y específicos para organizar visualmente la interfaz. Estos estilos se definen tanto en el archivo de estilos globales (`globalStyles`) como en una constante `styles` dentro del componente, asegurando que la aplicación mantenga una apariencia coherente y adaptativa a diferentes tamaños de pantalla.

En esta pantalla, los usuarios se encuentran con dos opciones: iniciar sesión y crear una cuenta. Al seleccionar la opción de iniciar sesión, serán dirigidos a una pantalla dedicada a este proceso. Por otro lado, si eligen crear cuenta, serán llevados a otra pantalla donde podrán registrar su nombre, apellidos, nombre de usuario y contraseña.

PANTALLA INICIO SESIÓN



Ilustración 29: Pantalla Inicio de Sesión. Elaboración propia

En esta pantalla usuarios introducen su nombre de usuario y contraseña. Si las credenciales son correctas, se les permite acceder a la aplicación; de lo contrario, se muestra una alerta indicando que deben ingresar correctamente sus datos.

La pantalla utiliza varios hooks y componentes de React Native, como `useState` para gestionar el estado de los inputs de usuario y contraseña. También se integra un contexto personalizado, `useUser`, para establecer los datos del usuario en el contexto global de la aplicación tras un inicio de sesión exitoso.

La función `handleLogin` maneja el proceso de inicio de sesión. Realiza una petición POST a una API, enviando el nombre de usuario y la contraseña. Si la respuesta indica un éxito, la función actualiza el contexto del usuario con los datos recibidos y navega a la pantalla principal (Home). En caso de error o si las credenciales son incorrectas, se muestra una alerta al usuario.

La interfaz de usuario incluye un campo de texto para el nombre de usuario, otro para la contraseña (con texto oculto por seguridad), y botones para iniciar sesión o volver a la pantalla anterior. Los estilos aplicados aseguran una presentación clara y accesible, con un esquema de colores coherente y elementos visuales como el logo de la aplicación.

El componente también emplea `TouchableWithoutFeedback` y `Keyboard` de React Native para mejorar la usabilidad, permitiendo a los usuarios ocultar el teclado al tocar cualquier área fuera de los campos de texto.

El usuario deberá tocar el cuadro de texto e ingresar sus credenciales. Si estas resultan ser incorrectas, se activará una alerta, y será necesario que las reingrese. En caso de que las credenciales sean correctas, el usuario será dirigido a la página de inicio.

PANTALLA HOME

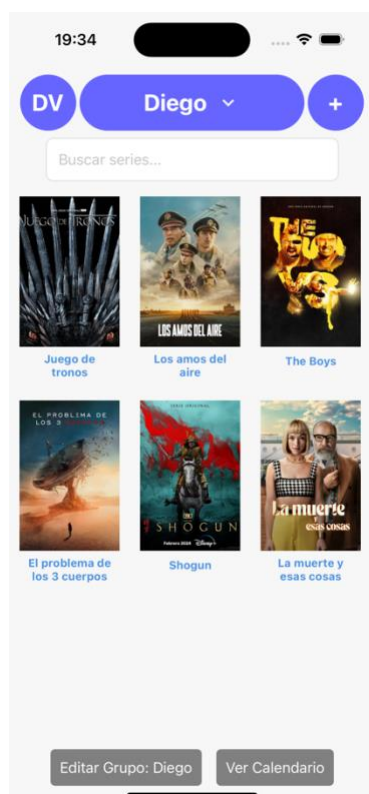


Ilustración 30: Pantalla Home. Elaboración Propia.

El código de la pantalla de inicio Incluye módulos de React (React, useEffect, useState), componentes de React Native para la interfaz de usuario (como View, Text, TouchableOpacity, etc.), utilidades de navegación de @react-navigation/native, un contexto personalizado useUser para acceder a los datos del usuario, estilos globales, y otros componentes para la selección de elementos e iconos.

En cuanto a los estados, el componente aprovecha el hook useState para manejar varios estados locales, incluyendo datos del usuario, de series, detalles específicos de series y grupos, además de controlar aspectos de la interfaz de usuario como la visibilidad y selección de elementos en la interfaz, tales como el menú desplegable y la barra de búsqueda. También se encarga de la gestión del refresco de datos mediante gestos de actualización.

El código de la pantalla de inicio Incluye módulos de React (React, useEffect, useState), componentes de React Native para la interfaz de usuario (como View, Text, TouchableOpacity, etc.), utilidades de navegación de @react-navigation/native, un contexto personalizado useUser para acceder a los datos del usuario, estilos globales, y otros componentes para la selección de elementos e iconos.

En cuanto a los estados, el componente aprovecha el hook useState para manejar varios estados locales, incluyendo datos del usuario, de series, detalles específicos de series y grupos, además de controlar aspectos de la interfaz de usuario como la visibilidad y selección de elementos en la interfaz, tales como el menú desplegable y la barra de búsqueda. También se encarga de la gestión del refresco de datos mediante gestos de actualización.

La carga de datos se facilita a través del uso de useEffect y useFocusEffect, que sirven para inicializar el componente o reaccionar al reenfoque de la pantalla, respectivamente. Estos efectos invocan funciones como llamarAGrupos y obtenerSeries, las cuales realizan solicitudes a una API para actualizar el estado local con datos relevantes a los grupos y series asociadas al usuario.

La navegación y gestión de la interfaz de usuario forman una parte crítica de la funcionalidad del componente. Se utiliza useNavigation para manejar la navegación entre pantallas y useRoute para acceder a parámetros de la ruta actual. La lógica incluida permite al usuario interactuar con diversos elementos de la interfaz, como botones e inputs de texto, facilitando

acciones como la búsqueda de series, la selección de grupos, el refresco de datos, y la navegación hacia los detalles de las series o la edición de grupos.

Dentro de las funcionalidades específicas del componente, se destacan la búsqueda de series por nombre a través de una API, la gestión de grupos que permite a los usuarios ver los grupos a los que pertenecen, añadir nuevos grupos, y editarlos, así como la visualización de series asociadas a un grupo seleccionado, permitiendo ver detalles o acceder al calendario de series.

En esta pantalla, los usuarios tienen la capacidad de buscar series de televisión. Esta búsqueda se realiza mediante una llamada a la API de TMDb (The Movie Database), que devuelve una información detallada sobre la serie de televisión buscada. Una vez que esta información es recibida, se almacena en la base de datos de la aplicación para su administración y utilización futura. Posteriormente, los detalles de la serie se muestran en la pantalla de manera atractiva y fácil de acceder, facilitando a los usuarios la interacción intuitiva con la información relevante de la serie.

Cabe destacar que la API de TMDb sirve únicamente como fuente de información para obtener los detalles de las series. El seguimiento de los capítulos vistos por cada usuario y la gestión de esta información se realizan a través de la base de datos interna de la aplicación y su propia API. Este enfoque permite personalizar la experiencia de usuario y gestionar eficientemente los datos relacionados con las preferencias y el progreso de visualización de cada usuario dentro de la aplicación.

En la pantalla referenciada como ilustración 24, los usuarios tienen la opción de cambiar de grupo a través de una barra de navegación ubicada en la parte superior. Este componente facilita la selección del grupo cuyos detalles se quieren explorar. Solo las series que están siendo vistas en común por todos los miembros del grupo seleccionado serán mostradas.

Además, se ofrece la posibilidad de añadir un nuevo grupo mediante el icono de suma (+). Al seleccionarlo, los usuarios serán redirigidos a una pantalla donde podrán crear un nuevo grupo. La edición del grupo actual es posible mediante un botón situado en la esquina inferior izquierda, junto con la opción de visualizar el calendario. Este último mostrará un calendario con las fechas de estreno de futuros episodios de series.

Al pulsar el botón situado en la esquina superior izquierda, identificado con las iniciales del usuario, se accede a una pantalla donde es posible editar el perfil del usuario y cerrar sesión.

Si se selecciona alguna de las series mostradas, se abrirá una nueva pantalla que presenta las temporadas de la serie junto con información detallada de cada una, mostrando la imagen, el título y las temporadas disponibles de la serie.

PANTALLA AJUSTES

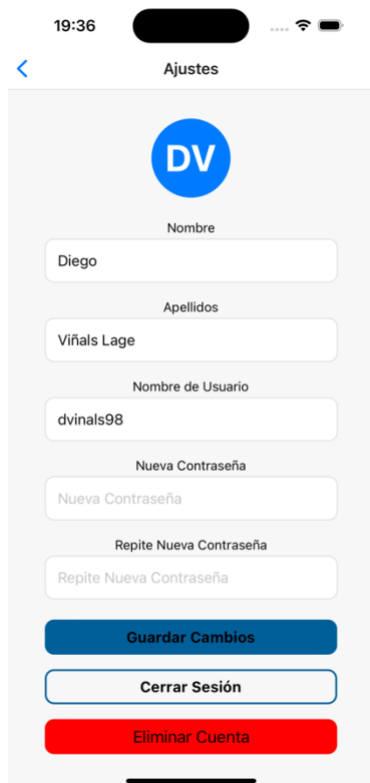


Ilustración 31: Pantalla Ajustes. Elaboración Propia.

A esta pantalla se accede pulsando el círculo de arriba a la izquierda con las iniciales en la pantalla de inicio, aquí, los usuarios pueden actualizar su información personal, como nombre, apellidos, nombre de usuario y contraseña. También incluye una opción para cerrar sesión y volver a la pantalla de bienvenida.

El componente Settings utiliza useState para manejar estados locales como el mensaje de error y los datos del usuario actual (nombre, apellidos, usuario, y contraseñas). Se emplea el hook personalizado useUser para acceder y modificar el contexto del usuario, permitiendo actualizar estos datos globalmente en la aplicación.

La interfaz de usuario consta de campos de texto para ingresar los datos actualizados y botones para guardar los cambios o cerrar sesión.

Al presionar el botón de guardar, se verifica que todos los campos estén llenos y que las contraseñas coincidan y cumplan con los criterios de validación antes de proceder con la actualización.

Los estilos utilizados aseguran que la pantalla sea visualmente atractiva y coherente con el resto de la aplicación, con elementos como un círculo que muestra las iniciales del usuario y campos de texto estilizados para la entrada de datos.

En la pantalla mostrada, se permite a los usuarios actualizar su información personal, incluyendo nombre, apellidos, nombre de usuario y contraseña. Para editar estos datos, simplemente deben tocar en el respectivo cuadro de texto y escribir la nueva información. Para la contraseña, es necesario introducirla dos veces para confirmación; si las entradas no coinciden, el sistema bloqueará la actualización del perfil. Además, se presenta un botón etiquetado "Eliminar cuenta", el cual, al ser presionado, desencadenará una alerta emergente pidiendo confirmación para proceder. Si el usuario confirma la acción, la cuenta y todos los datos asociados serán eliminados permanentemente. Optar por cerrar sesión finalizará la sesión activa del usuario, redirigiéndolo automáticamente a la página de inicio.

PANTALLA CREAR GRUPO



Ilustración 32: Pantalla Crear Grupo. Elaboración Propia.

Este componente aprovecha varios hooks de React, como `useState` y `useEffect`, para manejar el estado local, como el nombre del grupo, los miembros del grupo, y si actualmente se está editando el nombre del grupo.

La funcionalidad principal del componente incluye la edición del nombre de un grupo y la visualización de sus miembros. Al iniciar, realiza una petición al servidor para obtener los datos actuales del grupo, que se actualizan automáticamente gracias al uso de `useEffect`. Los usuarios pueden editar el nombre del grupo, lo cual activa un campo de texto. Una vez que el usuario desenfoca el campo o guarda los cambios, se envía una solicitud de actualización al servidor.

Además, ofrece la opción de eliminar un grupo a través de un botón, que muestra una alerta de confirmación para evitar eliminaciones accidentales. La interacción con el servidor para actualizar y eliminar grupos es manejada mediante la API.

La interfaz de usuario se compone de textos, botones y listas, organizados de manera que facilita la interacción. La estética se mantiene coherente y accesible, utilizando estilos definidos tanto localmente en `StyleSheet` como estilos globales importados, asegurando que la experiencia del usuario sea agradable y uniforme en toda la aplicación..

En esta interfaz, el usuario tendrá la capacidad de ingresar el nombre del grupo que desea conformar mediante el campo de texto provisto. Posteriormente, se visualizará el nombre del usuario fundador del grupo en la sección correspondiente a 'Usuario 1'. Al seleccionar el botón señalado con el símbolo de adición (+), se procederá a incluir un 'Usuario 2', continuando de esta manera hasta alcanzar un límite de seis miembros dentro del grupo. E En cada usuario, se debe escribir el nombre de la persona que pertenecerá al grupo.

Finalmente, al activar el botón para añadir el grupo, este se establecerá y se redirigirá al usuario a la pantalla principal.

PANTALLA EDITAR GRUPO

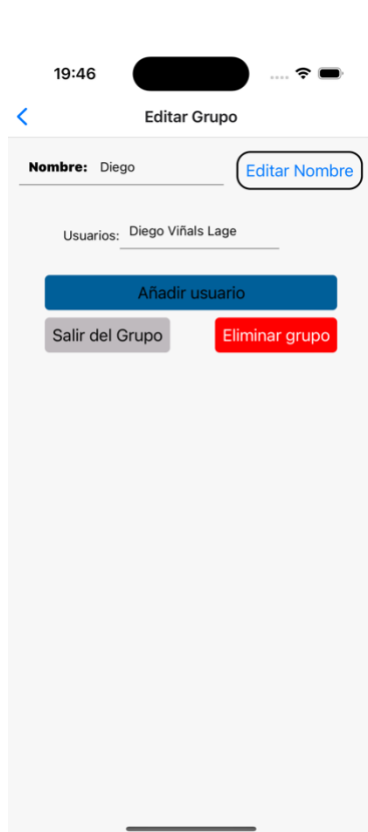


Ilustración 33: Pantalla Editar Grupo. Elaboración Propia.

Este componente permite la edición del nombre de un grupo y la visualización de sus miembros. Al cargarse, solicita los datos actuales del grupo al servidor, refrescando automáticamente estos datos mediante `useEffect`. Los usuarios pueden editar el nombre del grupo activando un campo de texto editable, cuyos cambios se guardan al desenfocar el campo o guardar los cambios, enviando una actualización al servidor.

Además, se presenta una funcionalidad para eliminar grupos, mediante un botón que desencadena una alerta de confirmación, previniendo así eliminaciones accidentales. La comunicación con el servidor para las acciones de actualizar y eliminar grupos se realiza a través de una API.

La interfaz del componente se compone de elementos como textos, botones y listas, organizados de manera intuitiva para facilitar la interacción del usuario.

Se emplea una estética coherente y accesible, con estilos definidos tanto localmente en `StyleSheet` como estilos globales importados, garantizando una experiencia de usuario agradable y uniforme a lo largo de la aplicación.

Esta interfaz se encuentra accesible desde la pantalla principal de la aplicación, mediante la selección del botón ubicado en la esquina inferior izquierda. En esta pantalla, se exhibe de manera detallada toda la información pertinente a un grupo específico, incluyendo su nombre y los miembros que lo conforman. Al interactuar con el botón destinado para la edición del nombre del grupo, los usuarios tienen la posibilidad de modificar este dato para posteriormente guardar los cambios efectuados. Además, la interfaz ofrece la opción de eliminar el grupo en cuestión; sin embargo, el botón para eliminar el grupo solo estará activado si el usuario es el administrador, es decir, el que creó el grupo.

Se ha añadido un botón para salir del grupo, permitiendo que el usuario se retire sin necesidad de eliminar el grupo completo. También se incluye un botón para añadir usuarios; al pulsar este botón, aparecerá un campo de texto donde se podrá escribir el nombre del usuario a añadir al grupo.

PANTALLA DETALLE DE SERIE

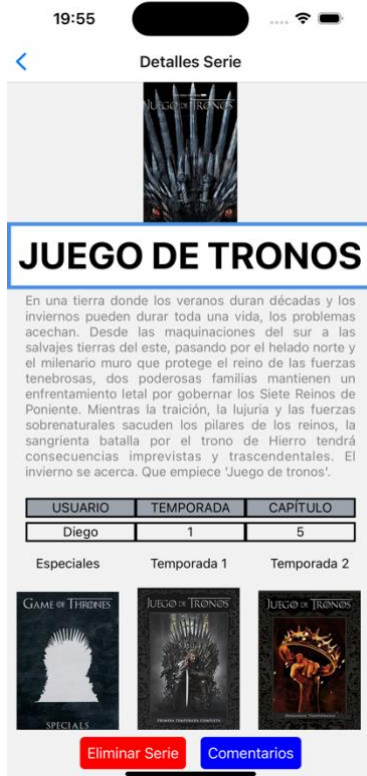


Ilustración 34: Pantalla Detalles Serie. Elaboración Propia.

El componente realiza peticiones a una API para obtener los detalles de la serie específica, incluyendo su poster, nombre, descripción, y temporadas disponibles, utilizando el hook `useEffect` y el `useFocusEffect` para refrescar estos datos automáticamente cuando la pantalla gana foco. Los usuarios tienen la capacidad de ver qué miembros del grupo están siguiendo la serie, incluyendo la temporada y el episodio más reciente que han visto.

Para eliminar una serie de la lista de un usuario, se incluye un botón que desencadena una alerta de confirmación. Esta precaución ayuda a prevenir eliminaciones accidentales y mejora la experiencia del usuario al asegurar una interacción intencionada. La eliminación efectiva se realiza a través de una petición `DELETE` a una API, manejando la respuesta y navegando de vuelta a la pantalla principal como indicación de que la acción se completó con éxito.

El componente se beneficia de un diseño de interfaz de usuario intuitivo y accesible, con uso de `ScrollView` para permitir la visualización de contenido extendido, `TouchableOpacity` para interacciones táctiles en elementos como posters de temporadas, y un esquema de estilos definido a través de `StyleSheet` para mantener una apariencia coherente y atractiva. Los estilos se centran en la claridad y accesibilidad, usando colores contrastantes, espaciado adecuado, y elementos visuales como imágenes de series y temporadas para enriquecer la experiencia del usuario.

En esta pantalla se muestran los detalles de la serie seleccionada desde la pantalla de inicio, incluyendo la fotografía, el título de la serie y una breve descripción. Se presenta una tabla con los nombres de los usuarios del grupo, indicando la temporada y el capítulo en el que se encuentra cada usuario.

Además, se exhiben las temporadas disponibles de la serie, cada una con su respectiva imagen, las cuales pueden ser seleccionadas para acceder a todos los capítulos de dicha temporada. También se incluye un botón que permite eliminar la serie y, consecuentemente, los capítulos que se han marcado como vistos.

Se ha añadido un botón adicional que llevará al usuario al espacio de comentarios de esa serie, donde podrá leer y participar en discusiones relacionadas con la serie, compartir opiniones e interactuar con otros espectadores.

PANTALLA COMENTARIOS

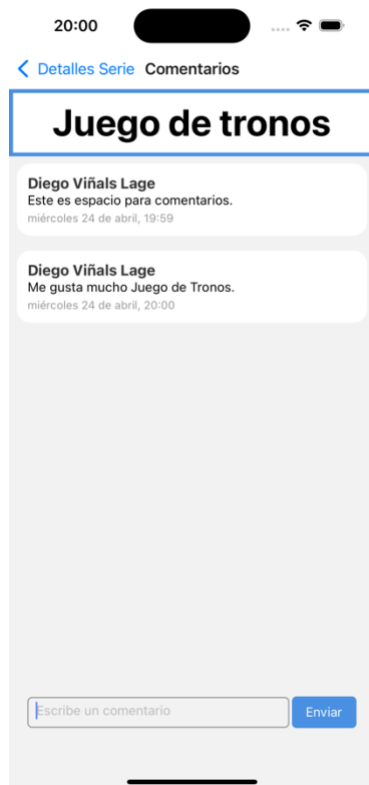


Ilustración 35: Pantalla Comentarios. Elaboración Propia.

El componente `ComentariosSerie` se encarga de gestionar la interacción de los usuarios con los comentarios relacionados con una serie específica dentro de un grupo, empleando una combinación de técnicas modernas de desarrollo en React Native. Este componente es fundamental para fomentar la participación y el intercambio de opiniones entre los miembros del grupo que siguen la serie.

Utiliza el hook `useEffect` para realizar peticiones a una API y obtener los comentarios de la serie por grupo, actualizando automáticamente estos datos cuando se modifican las dependencias especificadas. Esta automatización es crucial para mantener a los usuarios informados sobre las discusiones más recientes sin necesidad de recargar manualmente la página.

A través de otro `useEffect`, se maneja la visualización automática del final del área de comentarios cada vez que el teclado aparece, lo que mejora la experiencia de usuario al facilitar la escritura de nuevos comentarios. Además, un temporizador ajusta el desplazamiento al final de la lista de comentarios cada vez que estos cambian, asegurando que los usuarios siempre vean los comentarios más recientes sin esfuerzo.

El componente proporciona una interfaz para escribir y enviar comentarios mediante un formulario integrado. Al enviar un comentario, se realiza una petición POST a la API para añadir el nuevo comentario a la base de datos. La respuesta de la API es manejada adecuadamente para actualizar la interfaz y reflejar el comentario recién añadido, potenciando la interactividad en tiempo real.

El componente ajusta su comportamiento y estilo basado en la plataforma, usando `KeyboardAvoidingView` para manejar las diferencias en la visualización del teclado entre iOS y Android. Esto asegura que la experiencia del usuario sea uniforme, independientemente del dispositivo utilizado.

El uso de ScrollView para los comentarios permite a los usuarios explorar fácilmente largas listas de entradas. La interacción táctil es manejada por TouchableOpacity, haciendo que los elementos interactivos no solo sean visibles sino también gratificantes al tacto.

Al escribir un comentario en el espacio dedicado y pulsar el botón de enviar, el comentario es enviado a la base de datos mediante una petición POST a la API correspondiente. Este proceso no solo incluye el envío del comentario sino también la inclusión del identificador del usuario, el grupo y la serie específica, asegurando que el comentario se asocie correctamente en la base de datos.

Los usuarios pueden escribir y enviar comentarios a través de una interfaz dedicada. Al pulsar enviar, se realiza una petición POST a la API, que incluye el comentario junto con los identificadores del usuario, grupo y serie, asegurando que se asocie correctamente en la base de datos.

Tras enviar un comentario, el componente actualiza automáticamente el estado local, añadiendo el comentario a la lista visible en la interfaz del usuario sin necesidad de recargas, permitiendo una interacción fluida y continua.

El componente realiza peticiones a la API cada 10 segundos para verificar si hay nuevos comentarios de otros usuarios. Utiliza comparaciones de cadenas JSON para detectar y mostrar cambios, manteniendo así a los usuarios al tanto de las últimas conversaciones sin esfuerzo manual.

PANTALLA DETALLE DE TEMPORADA



Ilustración 36: Pantalla Detalle de Temporada.
Elaboración Propia.

Al cargarse, el componente efectúa llamadas a una API externa para recuperar los detalles de la temporada seleccionada, incluyendo una lista de episodios. También hace peticiones a una API interna para obtener y marcar los episodios que el usuario ya ha visto, lo cual se visualiza mediante un cambio en los botones asociados a cada episodio, permitiendo marcar o desmarcar episodios como vistos. Esto se logra a través de la función `obtenerCapitulosVistos` que se invoca cada vez que el componente gana foco o cuando ciertas dependencias cambian, asegurando que la información se mantenga actualizada.

Para interactuar con los episodios, se proporcionan dos acciones principales: `marcarVisto` y `eliminarVisto`. Estas acciones permiten al usuario gestionar su progreso en la temporada, registrando o eliminando episodios de su lista de visualizados.

Estas interacciones son manejadas mediante peticiones POST a un servidor, con un manejo de estado local que refresca la lista de episodios vistos para reflejar los cambios inmediatamente.

El diseño de la interfaz utiliza `ScrollView` para permitir la navegación vertical a través de los episodios y utiliza `TouchableOpacity` para los botones, proporcionando una experiencia de usuario táctil e interactiva. Los estilos se definen mediante `StyleSheet`, asegurando una presentación coherente y atractiva que mejora la legibilidad y la usabilidad del componente.

Esta pantalla es una de las más sencillas; muestra todos los capítulos de la temporada seleccionada, incluyendo su título y una breve descripción de cada capítulo. Para cada capítulo hay un botón: si el capítulo no ha sido visto, se muestra un botón azul con la leyenda 'Marcar como visto'. Al pulsarlo, el capítulo se marcará como visto y el botón cambiará a uno que dice 'Visto'. Si se pulsa este último, el capítulo se eliminará de la lista de vistos.

Al regresar a la pantalla de detalles de la serie, se actualizarán los episodios que el usuario ha visto.

PANTALLA CREAR CUENTA



Ilustración 37: Pantalla Crear Cuenta. Elaboración Propia.

El componente `SignUp` permite a los usuarios crear una cuenta proporcionando su nombre, apellidos, nombre de usuario y una contraseña que debe cumplir con ciertos criterios de seguridad, como tener al menos 8 caracteres, incluir una letra mayúscula y un número.

Al iniciar, el componente establece estados iniciales para almacenar la información del usuario y un mensaje de error en caso de que ocurra algún problema durante el proceso de validación o registro. Utiliza `TextInput` para recoger los datos del usuario y `TouchableOpacity` para los botones que manejan la acción de registro y la opción de volver a la pantalla anterior.

Para el registro, se implementa una función `handleSignUp` que primero verifica si las contraseñas ingresadas coinciden y si cumplen con los requisitos mínimos de seguridad.

Si las validaciones son exitosas, se procede a crear un objeto de usuario con un ID único generado por la función `generarIdUnico` y se intenta registrar al usuario mediante una petición `POST` a un servidor. El resultado de esta operación se comunica al usuario a través de alertas.

El diseño utiliza `StyleSheet` para definir el estilo de los componentes UI, como los campos de texto y botones, además de un logo personalizado que se muestra en la parte superior. También se emplea `TouchableWithoutFeedback` para ocultar el teclado cuando se toca fuera de los campos de entrada, mejorando la experiencia de usuario.

En la parte superior, se encuentra el logo, seguido por el título “Crear Cuenta” que indica claramente el propósito de la pantalla. Debajo del título, se disponen varios campos de texto donde el usuario debe introducir su nombre, apellidos, nombre de usuario y contraseña. Cada campo está claramente etiquetado para su fácil identificación.

Justo debajo de los campos para la contraseña y su confirmación, hay dos botones rectangulares; el primero, destacado en un color más intenso, es para “Crear Cuenta”, lo que sugiere que es el siguiente paso lógico tras rellenar la información solicitada. El segundo botón, para cancelar o volver atrás en el proceso, está etiquetado como “Volver”.

PANTALLA CALENDARIO

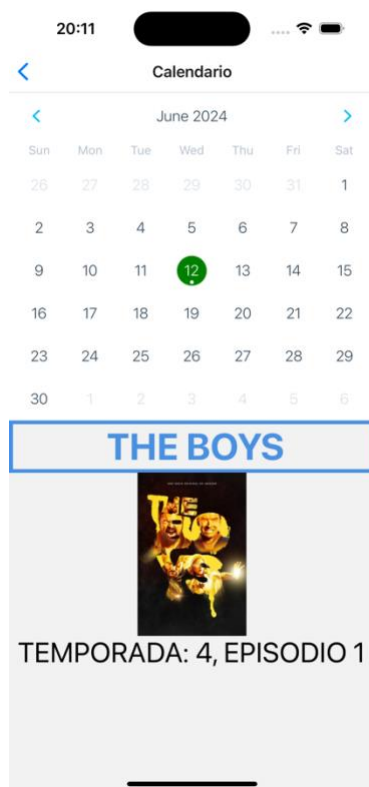


Ilustración 38: Pantalla Calendario. Elaboración Propia.

Utiliza varias librerías y componentes de React Native, así como la API de The Movie Database (TMDb) para obtener los detalles de las series y los próximos episodios.

Al iniciar, el componente carga los detalles de las series asociadas a un usuario y a un grupo específico, si se ha seleccionado uno. Las fechas de los próximos episodios se marcan en el calendario, y al seleccionar una fecha, se muestran los detalles del episodio programado para esa fecha, incluyendo un póster de la serie, el número de temporada y episodio, y una descripción.

El estado `seriesDetalles` contiene la información de las series obtenida de la API de TMDb, mientras que `markedDates` mantiene un registro de las fechas marcadas en el calendario.

La lógica para marcar las fechas de los próximos episodios se realiza en un efecto secundario que se dispara cuando el estado `seriesDetalles` cambia.

El componente `Calendar` de `react-native-calendars` se utiliza para renderizar el calendario, y se configura para marcar las fechas de los próximos episodios y resaltar la fecha seleccionada por el usuario. La función `obtenerSeriesDelUsuario` realiza una petición a un servidor backend (no especificado) para obtener los identificadores de las series que sigue el usuario dentro de un grupo específico, y luego se hace otra petición a la API de TMDb para obtener los detalles de estas series.

La función `formatDate` se utiliza para formatear las fechas de los episodios de manera legible, y la función `poster` para renderizar el póster de la serie con el estilo definido.

Este componente también muestra cómo manejar el estado y efectos secundarios en React Native, el uso de promesas y `async/await` para realizar peticiones HTTP, y la integración con APIs externas para enriquecer la funcionalidad de las aplicaciones móviles.

En esta pantalla, se aprecia un calendario que automáticamente se ubica en el día actual. Los días que cuentan con el estreno de un nuevo episodio están señalados con puntos debajo de la fecha correspondiente.

Al seleccionar uno de estos días, en la parte inferior se despliegan los detalles del episodio que se estrena. En caso de no haber estrenos en la fecha seleccionada, la parte inferior no mostrará información alguna.