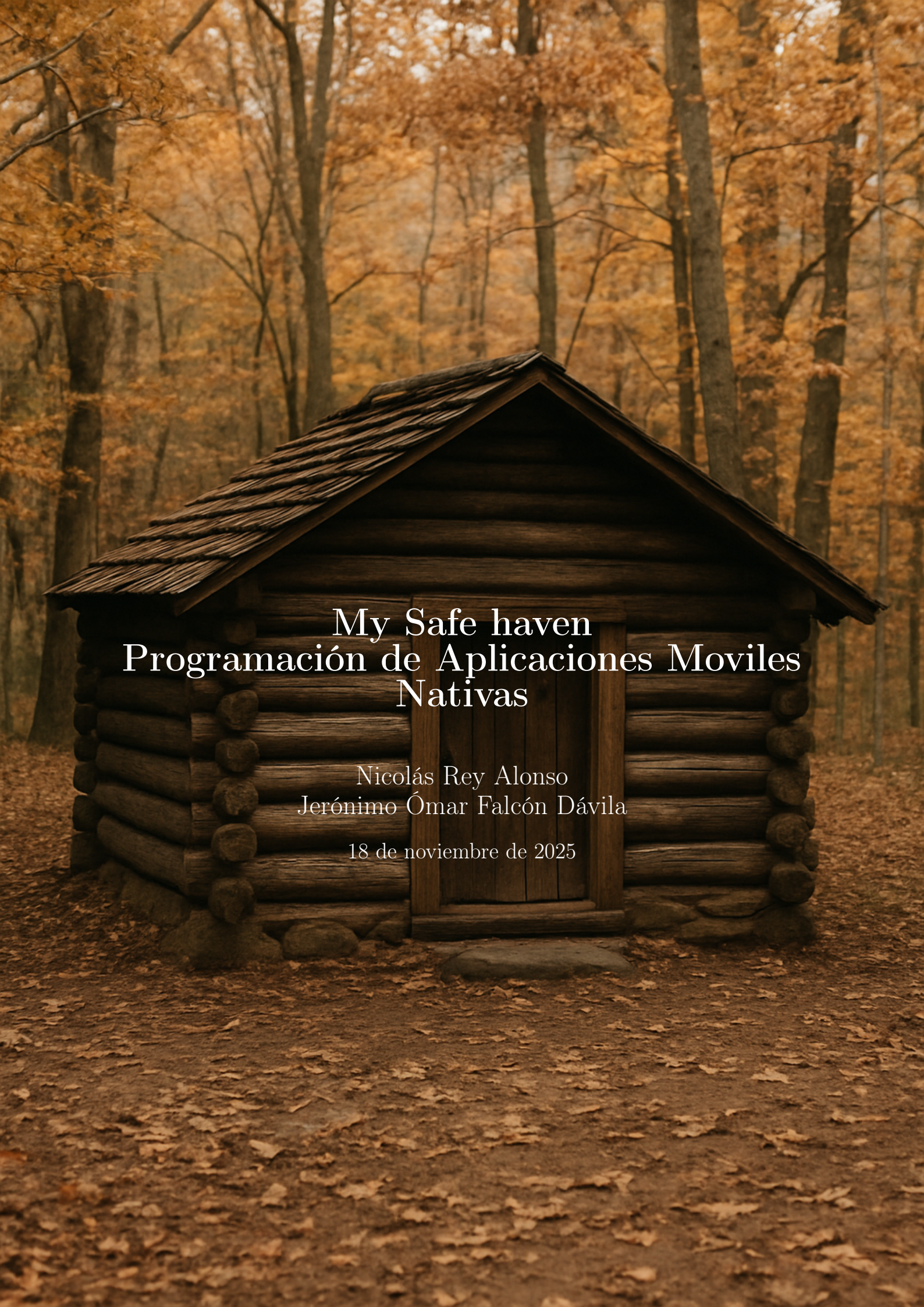

My Safe haven
Programación de Aplicaciones Moviles Nativas

Nicolás Rey Alonso
Jerónimo Ómar Falcón Dávila

18 de noviembre de 2025

Índice

1. Descripción del proyecto

El presente proyecto consiste en el desarrollo de una red social integrada en una aplicación móvil nativa denominada *My Safe Haven*. La propuesta toma como referencia diversas plataformas ampliamente reconocidas, como Instagram, Twitter y Tumblr, adoptando algunos de sus conceptos fundamentales, pero introduciendo un enfoque innovador que diferencia a nuestra aplicación del resto.

La característica central de *My Safe Haven* radica en que el acceso a las publicaciones no es completamente abierto: para visualizar los distintos posts, el usuario debe encontrarse físicamente dentro del "haven" creado por el autor del contenido. Definimos un *haven* como un tipo particular de publicación asociada a un espacio delimitado en el mundo real y compuesta por un nombre identificativo, una descripción, un conjunto de elementos que conforman su propio *feed*, así como coordenadas geográficas y un radio de alcance que determina su área de influencia.

La creación de un haven también requiere presencia física: el usuario debe encontrarse en la ubicación exacta donde desea establecerlo para poder registrarlo en la aplicación. Esta decisión de diseño se inspira en fenómenos virales que combinan elementos digitales con interacción física, como el caso de *Pokémon Go*. Nuestro objetivo es recuperar esa dimensión híbrida entre lo virtual y lo real, incorporándola en un contexto de red social que fomente la exploración, la conexión local y la interacción significativa con el entorno.

De esta manera, *My Safe Haven* busca ofrecer una experiencia social alternativa, basada en la proximidad y en la creación de espacios digitales anclados al mundo físico, promoviendo dinámicas más auténticas y situadas entre usuarios.

2. Plan de Sprint

Para el desarrollo de esta aplicación hemos decidido utilizar la metodología Scrum, ya que nos permite organizar el trabajo de forma iterativa y mantener un seguimiento constante del progreso. Gracias a sus ciclos de trabajo cortos (sprints), podemos adaptarnos con mayor facilidad a los cambios, cumplir con los objetivos definidos para cada etapa y asegurar entregas frecuentes que aporten valor al proyecto.

2.1. Plan de trabajo

Fase	Duración Estimada	Tareas
Plan Sprint Zero	16 Horas	Preparación de documentación, redacción del Plan de Sprint, realización de Mockups y definición de objetivos
Sprint 0	2 Semanas	Desarrollo del mínimo viable de la aplicación

Cuadro 1: Tabla de Plan de Trabajo

2.2. Sprint 0

Las **historias** que se implementan en este sprint son las siguientes:

Loggear en la APP

ID: 0
Prioridad: 0
Estimación: 4h

Historia:
Como: Usuario
Quiero/necesito: Loggearme en la aplicacion
Para: Poder usar la app con mis credenciales

Criterios de validación:

- Cuando no ponga bien la contraseña me debe salir el error.
- Cuando me logge, debe haber alguna indicación de que lo he hecho correctamente.

Base de datos postgres

ID: 1
Prioridad: 0
Estimación: 4h

Historia:
Como: Desarrollador
Quiero/necesito: Una base de datos postgres lanzable en mi pc
Para: Poder desarrollar la aplicación

Criterios de validación:

- Tiene que accederse por el puerto 5243.

Mockups figma

ID: 2
Prioridad: 0
Estimación: 8h

Historia:
Como: Desarrollador
Quiero/necesito: Unos mockups en figma
Para: Poder desarrollar la aplicación

Criterios de validación:

- tienen que representar el diseño de la aplicación.
- tienen que poseer las vista de movil y tablet.
- tiene que contener el tema de colores de la aplicacion.

Obtener mis havens

ID: 3
Prioridad: 0
Estimación: 8h

Historia:
Como: Usuario
Quiero/necesito: obtener mis havens
Para: poder gestionarlos y visualizarlos

Criterios de validación:

- Se deven ver los havens propios y en los que participo.
- Mis havens deven tener un botón que me permitan gestionarlos.
- Los havens deven poderse filtrarion.

Tablero trello

ID: 4
Prioridad: 0
Estimación: 8h

Historia:
Como: Desarrollador
Quiero/necesito: Un tablero trello
Para: realizar el desarrollo de los sprints

Criterios de validación:

- Debe tener todas las historias del sprint 0 a desarrollar.

Plan de sprint

ID: 5
Prioridad: 0
Estimación: 8h

Historia:
Como: Desarrollador
Quiero/necesito: Un plan de sprint
Para: realizar el desarrollo de los sprints

Criterios de validación:

- Debe representar las tareas a realizar.

Backend desplegable en docker

ID: 6
Prioridad: 0
Estimación: 2h

Historia:
Como: Desarrollador
Quiero/necesito: Un backend desplegable en docker
Para: aumentar la portabilidad y la facilidad de despliegue del backend

---

Criterios de validación:

- Debe representar las tareas a realizar.

Crear Haven

ID: 7
Prioridad: 0
Estimación: 8h

Historia:
Como: Usuario
Quiero/necesito: Crear un haven
Para: Tener mi feed personal

Criterios de validación:

- Si ya hay un haven en el mismo sitio aproximado no debe de poderse crear.
- Al crearse debe mostrarse un mensaje de creado y navegar al feed del haven.
- No debe poderme permitir poner el mismo nombre a dos havens.
- Debo poder poner información de nombre y descripción al haven.
- En la versión gratuita no puedo crear más de dos havens.
- Se me debe crear una feed vacía automáticamente al crear el haven.
- Debe poder ser publico o privado

Eliminar un haven

ID: 8
Prioridad: 0
Estimación: 4h

Historia:
Como: Usuario
Quiero/necesito: Eliminar un haven
Para: Poder gestionar mis havens

Criterios de validación:

- Solo debo poderlo eliminar si es mío.
- Al eliminarlo debe salirme un mensaje de confirmación.
- Se me deven dejar una segundos en los que pueda deshacer la acción.

Editar un haven

ID: 9
Prioridad: 0
Estimación: 4h

Historia:
Como: Usuario
Quiero/necesito: Editar un haven
Para: Cambiar su nombre o descripción

Criterios de validación:

- Solo debo poderlo editarlo si es mío.
- No debo de poder cambiarle el nombre a uno que ya exista.
- Se me deven dejar una segundos en los que pueda deshacer la acción.

Añadir posts a mi Haven Feed

ID: 10
Prioridad: 0
Estimación: 4h

Historia:
Como: Usuario
Quiero/necesito: Añadir posts en el feed de mi haven
Para: Que mis amigos lo vean cuando estén en mi haven

Criterios de validación:

- Deben poder ser de tipo texto.
- Deben de poder ser de tipo media (Música, Imagen o Vídeo).
- El texto debe ser enriquecido (Negrita etc).

Visualizar Haven Feeds

ID: 11
Prioridad: 0
Estimación: 8h

Historia:
Como: Usuario
Quiero/necesito: acceder a una haven feed
Para: ver sus publicaciones

Criterios de validación:

- Solo debo de poder verla si estoy físicamente en el haven.
- Si no estoy añadido al haven, no devo de poder verlo.
- Los posts se mostraran de más nuevos a más viejos.

Añadir usuarios a mi haven feed

ID: 12
Prioridad: 0
Estimación: 8h

Historia:
Como: Usuario
Quiero/necesito: poder añadir a usuarios a mi haven feed
Para: que vean mis publicaciones

Criterios de validación:

- Devo de poder enviar invitaciones a amigos.
- Si pasa alguien por mi haven, me debe salir una notificación que me permita invitarle.

Eliminar posts de mi Haven Feed

ID: 13
Prioridad: 0
Estimación: 2h

Historia:
Como: Usuario
Quiero/necesito: poder eliminar posts de mi haven feed
Para: desechar posts que ya no me representan

Criterios de validación:

- Debo de mostrarme un mensaje de confirmación antes de eliminar.
- Debo de tener un tiempo en el que pueda deshacerlo.

Registrarme en My Safe Haven

ID: 14
Prioridad: 0
Estimación: 4h

Historia:
Como: Usuario
Quiero/necesito: registrarme en my safe haven
Para: poder disfrutar de la red social completa

Criterios de validación:

- Debe llegarme un correo de confirmación al mail.
- La contraseña debe de ser segura, y si no lo es mostrar el error.
- El correo electrónico debe de ser válido.

3. Arquitectura

Las arquitecturas que hemos decidido utilizar para el desarrollo de esta aplicación son las siguientes:

Clean Architecture + MVVM + Jetpack (Compose/Views)

3.1. MVVM

El patrón *Model–View–ViewModel* (MVVM) nos permite separar de forma efectiva la lógica de presentación de la lógica de la interfaz. Esto facilita el mantenimiento del código y reduce el acoplamiento entre componentes. Las principales ventajas por las que se eligió este patrón son:

- Permite que las vistas contengan únicamente lógica de interfaz, manteniendo un código más limpio y fácil de extender.
- Los *ViewModels* soportan la persistencia del estado incluso frente a cambios de configuración, como la rotación de pantalla.
- Facilita el uso de herramientas modernas como *LiveData*, *StateFlow* y *Coroutines*.
- Incrementa la capacidad de realizar pruebas unitarias al separar las responsabilidades dentro del flujo de datos.

3.2. Clean Architecture

La *Clean Architecture* garantiza una separación clara entre las reglas de negocio, los datos y la interfaz de usuario. Esta división en capas mejora considerablemente la escalabilidad y mantenibilidad del sistema. Se seleccionó debido a sus siguientes beneficios:

- Define límites bien estructurados mediante capas independientes: dominio, datos y presentación.
- Permite encapsular la lógica central de negocio en casos de uso (*use cases*), facilitando la evolución del proyecto sin afectar otras capas.
- Facilita la integración de servicios externos, como APIs, sensores de ubicación o bases de datos locales.
- Reduce el acoplamiento, lo que permite sustituir componentes (por ejemplo, cambiar la base de datos local) sin modificar la lógica interna.
- Mejora considerablemente la capacidad de realizar pruebas, especialmente sobre reglas de negocio.

3.3. Jetpack (Compose/Views)

El ecosistema *Jetpack* nos proporciona herramientas modernas y altamente optimizadas para el desarrollo de interfaces y funcionalidades móviles. Dentro del proyecto empleamos tanto *Compose* como vistas tradicionales (*Views*), según las necesidades de cada pantalla. Las razones principales para utilizar estas tecnologías son:

- *Jetpack Compose* permite crear interfaces declarativas, más intuitivas y fáciles de mantener.
- Ofrece integración natural con *ViewModels*, *StateFlow* y otros componentes de arquitectura.
- Simplifica el manejo del estado y el renderizado dinámico de la UI.
- Framework optimizado por Google para el desarrollo moderno de Android, asegurando compatibilidad y evolución a largo plazo.
- La coexistencia con *Views* permite integrar partes previas del sistema o componentes aún no migrados a Compose.