Manual Técnico

1. Información General del Proyecto

1.1. Resumen

Spacius es una aplicación móvil nativa para la plataforma Android, desarrollada bajo el lenguaje **Kotlin**. Su objetivo es la gestión y reserva de espacios públicos mediante una interfaz intuitiva y geolocalización.

1.2. Stack Tecnológico

El sistema se construye sobre las siguientes tecnologías clave:

- **Lenguaje:** Kotlin.
- **Entorno de Desarrollo (IDE):** Android Studio (narwhal).
- **Arquitectura:** MVVM (Model - View - View Model).
- **Backend & Base de Datos:** Google Firebase (Serverless).
- **Servicios de Mapas:** Google Maps SDK for Android.

2. Arquitectura del Software

El proyecto sigue el patrón de diseño **MVVM (Model-View-ViewModel)**, lo que permite separar la lógica de negocio de la interfaz de usuario, facilitando el mantenimiento y las pruebas.

2.1. Componentes del Patrón

1. **View (Vista):**
    - Corresponde a las Activities y Fragments (archivos .kt y layouts .xml).
    - Su única función es mostrar datos al usuario y capturar eventos (clicks).
    - *En Spacius:* LoginActivity, HomeActivity, ReservaFragment.
2. **ViewModel:**
    - Actúa como intermediario. Recibe las acciones de la Vista (ej. "Usuario hizo click en reservar") y se comunica con el Modelo.
    - Utiliza LiveData o StateFlow para actualizar la interfaz automáticamente cuando los datos cambian.
3. **Model (Modelo):**

- Maneja la lógica de datos y la comunicación con Firebase.
- Aquí residen los Repositories (Repositorios) que envían y traen información de la nube.

3. Entorno de Desarrollo y Configuración

3.1. Requisitos del Sistema (SDKs)

El proyecto está configurado para abarcar una amplia gama de dispositivos modernos, manteniendo compatibilidad con versiones anteriores estables.

- **Min SDK:** Versión **24** (Android 7.0 Nougat). Esto garantiza que la aplicación funcione en aproximadamente el 94% de los dispositivos activos a nivel mundial.
- **Target SDK:** Versión **36** (Android 15). La aplicación está optimizada para las últimas características de seguridad y rendimiento de Android.
- **Compilación:** Se utiliza **Java 17** (JavaVersion.VERSION_17) para la compatibilidad de código fuente.

3.2. Tecnologías de Construcción (Build Tools)

- **Sistema de Construcción:** Gradle con **Kotlin DSL** (build.gradle.kts), ofreciendo mayor seguridad de tipos y autocompletado en la configuración.
- **Inyección de Vistas:** Se utiliza **ViewBinding** (viewFeatures { viewBinding = true }).
  - *Justificación Técnica:* Reemplaza al tradicional findViewById, proporcionando seguridad contra nulos (Null Safety) y seguridad de tipos al interactuar con los elementos de la interfaz XML.
- **Procesamiento de Anotaciones:** Se utiliza **KSP** (Kotlin Symbol Processing) (alias(libs.plugins.ksp)), que es más rápido que el antiguo KAPT, optimizando los tiempos de compilación.

3.3. Seguridad y Gestión de Claves

El proyecto implementa una gestión segura de credenciales para servicios externos (como Google Maps):

- **Ocultamiento de API Keys:** Las claves sensibles no están hardcodeadas en el código fuente ni en el repositorio.

- **Implementación:** Se leen dinámicamente desde el archivo local.properties (no versionado) o variables de entorno del sistema.
- **Inyección en Manifiesto:** Gradle inyecta la clave MAPS_API_KEY directamente en los manifestPlaceholders durante el tiempo de compilación.

4. Estructura del Proyecto (Android)

El código fuente de Spacius sigue una organización modular basada en paquetes semánticos, lo que facilita la escalabilidad y la localización de archivos. A continuación se describe la distribución de los directorios principales dentro de app/src/main/java/com/example/spacius:

4.1. Paquetes Principales

- **/view (o /ui):** Contiene todos los componentes visuales.
  - **/activities:** Las pantallas contenedor (MainActivity, LoginActivity).
  - **/fragments:** Las pantallas de navegación interna (HomeFragment, MapFragment, ProfileFragment).
  - **/adapters:** Clases intermediarias para llenar listas (RecyclerViews) como el listado de parques o el historial de reservas.
- **/viewmodel:** Contiene las clases ViewModel (ej. ReservaViewModel).
  - Estas clases mantienen el estado de la UI y sobreviven a los cambios de configuración (como rotar la pantalla).
  - Gestionan la lógica de presentación y comunican la Vista con el Modelo.
- **/model (o /data):** La capa de datos.
  - **/models (Data Classes):** Clases simples de Kotlin que representan los objetos (Usuario, Reserva, Lugar).
  - **/repository:** Clases encargadas de decidir de dónde obtener los datos (Firebase) y enviarlos al ViewModel.
- **/utils:** Clases de utilidad general (ej. convertidores de fecha, constantes, validadores de formularios).
- **SpaciusApp.kt:** Clase de aplicación que inicializa contextos globales o librerías al arrancar la app.

Manual Técnico

5. Diseño de Base de Datos (Firebase)

Spacius utiliza una arquitectura **Serverless** apoyada en **Google Firebase**. Los datos se almacenan en **Cloud Firestore**, una base de datos NoSQL flexible, escalable y en tiempo real.

A continuación, se detalla el **Diccionario de Datos** con las colecciones principales que sustentan la lógica de negocio:

5.1. Colección: users (Usuarios)

Almacena la información de perfil de cada ciudadano registrado. El Document ID coincide con el UID generado por Firebase Authentication.

Campo	Tipo	Descripción
uid	String	Identificador único del usuario.
name	String	Nombre completo (ej. "Danny Freire").
email	String	Correo electrónico de contacto.
phone	String	Número celular para validación.
photoUrl	String	URL de la imagen de perfil (alojada en Storage).
createdAt	Timestamp	Fecha de creación de la cuenta.

5.2. Colección: places (Espacios Públicos)

Catálogo de los lugares disponibles para reservar.

Campo	Tipo	Descripción
id	String	Identificador único del lugar.
name	String	Nombre (ej. "Canchas de Vóley").
description	String	Detalles y normas del lugar.
location	GeoPoint	Coordenadas (Latitud, Longitud) para el mapa.
imageUrl	String	URL de la foto principal del lugar.
amenities	Array	Lista de servicios (ej. ["Wifi", "Baños"]).

5.3. Colección: bookings (Reservas)

Es la colección transaccional más importante. Relaciona un usuario con un lugar en un tiempo determinado.

Campo	Tipo	Descripción
id	String	ID único de la reserva.
userId	Reference	Referencia al usuario que reservó.
placeId	Reference	Referencia al lugar reservado.
date	String	Fecha de la reserva (Formato: YYYY-MM-DD).
startTime	String	Hora de inicio (ej. "10:00").
status	String	Estado actual: CONFIRMED o CANCELLED.
timestamp	Timestamp	Fecha/hora en que se creó el registro.

5.4. Colección: notifications (Historial)

Almacena el registro de alertas que el usuario ve en la pantalla de la "campanita".

Campo	Tipo	Descripción
userId	String	Usuario dueño de la notificación.
title	String	Título (ej. "Reserva Cancelada").
message	String	Cuerpo del mensaje.
type	String	Tipo visual (ej. "success", "warning").
isRead	Boolean	true si ya fue vista, false si es nueva.

6. Instalación y Despliegue

Para finalizar el manual técnico, necesitamos explicar cómo otro programador puede descargar y correr tu código.

6.1. Requisitos Previos

- Tener instalado **Android Studio** (versión recomendada: Ladybug o superior).
- JDK 17 configurado.
- Dispositivo Android físico o Emulador con Google Play Services.

6.2. Pasos de Instalación

1. Clonar el Repositorio:

Descargue el código fuente desde el repositorio de control de versiones.

Bash

git clone <https://github.com/davmatute-lang/spacius>

2. Configuración de Claves:

Cree un archivo local.properties en la raíz del proyecto si no existe y agregue su API Key de Google Maps:

Properties

MAPS_API_KEY=AIzaSyD_KeyFalsaParaEjemploDelManual_XyZ123

3. Sincronización:

Abra el proyecto en Android Studio y presione "Sync Project with Gradle Files". Espere a que se descarguen las dependencias.

4. Configuración de Firebase:

Asegúrese de que el archivo google-services.json (proporcionado por la consola de Firebase) esté ubicado dentro de la carpeta /app.

5. Ejecución:

Seleccione el dispositivo de destino y presione el botón Run (Play).