PRÁCTICA 1

ARQUITECTURA SOFTWARE:

VISTA DE MÓDULOS, COMPONENTES & CONECTORES Y DISTRIBUCIÓN SOBRE EL TRABAJO DE INGENIERÍA DEL SOFTWARE

Jorge Hernández Aznar 872838
Laura Hernández Sierra 872203
Arquitectura Software 2024/2025
28/02/2025

ÍNDICE

1. Documentación de la Vista de Módulos del Sistema

1.1 Descripción General

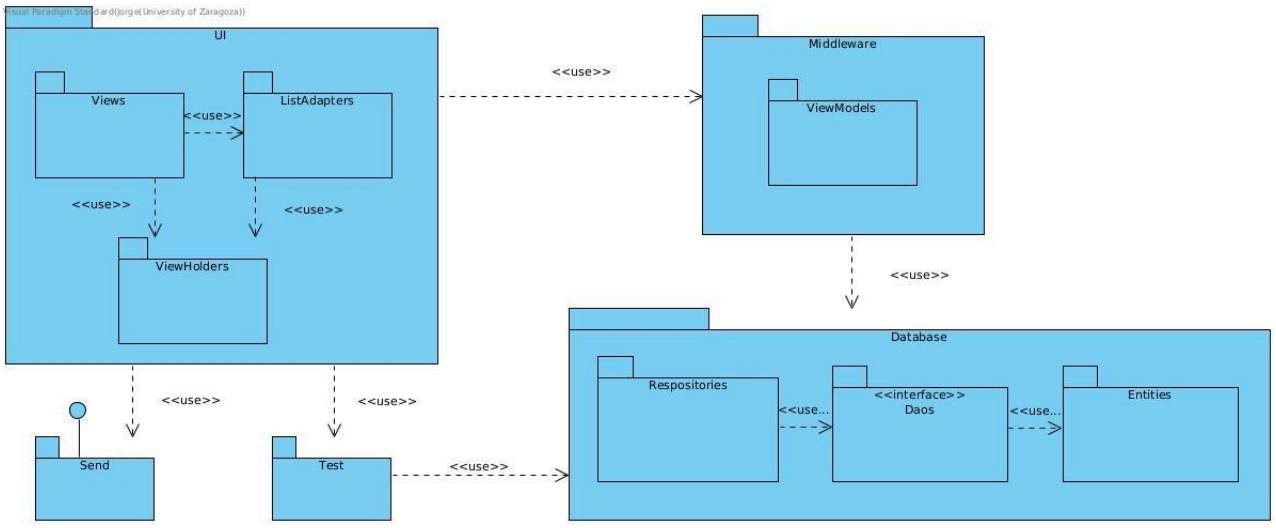
Este documento describe la vista de módulos del sistema, mostrando cómo los diferentes componentes interactúan entre sí. La arquitectura se divide en múltiples módulos bien organizados que separan responsabilidades y facilitan el mantenimiento y la escalabilidad de la aplicación. Una de las funcionalidades principales de esta vista es mostrar cómo se organiza el código fuente.

Los módulos principales incluyen:

- UI (Interfaz de Usuario): Maneja la presentación y la interacción con el usuario.
- Middleware: Actúa como una capa intermedia entre la UI y la base de datos.
- Database (Base de Datos): Gestiona la persistencia y recuperación de datos.
- Módulos adicionales (Send y Test): Funcionalidades específicas para el envío de datos y pruebas.

A continuación, se muestra un diagrama que representa los módulos anteriores y sus interrelaciones.

1.2 Vista de Módulos



1.3 Descripción de Módulos

1.3.1 Módulo UI (Interfaz de Usuario)

Este módulo se encarga de manejar la representación visual de la aplicación y la interacción con el usuario. Contiene los siguientes submódulos:

Views

- Responsable de la representación visual de la aplicación.
- Interacciones:
    - Usa ListAdapters para manejar y mostrar listas de datos dinámicamente.
    - Usa ViewHolders para optimizar la representación de vistas en listas.

ListAdapters

- Se encarga de gestionar la adaptación de datos para ser mostrados en listas dentro de la UI.
- Interacciones:
    - Usa ViewHolders para la reutilización eficiente de vistas.

ViewHolders

- Implementa un patrón de diseño para almacenar referencias a vistas y mejorar el rendimiento de las listas.
- Interacciones:
    - Es utilizado por ListAdapters para optimizar la representación de elementos en listas.

1.3.2 Módulo Middleware

El módulo Middleware actúa como intermediario entre la UI y la base de datos, gestionando la lógica de presentación y manipulación de datos.

- Responsabilidad:
    - Procesa peticiones recibidas desde la UI antes de enviarlas a la base de datos.
    - Utiliza los ViewModels para estructurar los datos antes de enviarlos a la UI.
- Interacciones:
    - Recibe solicitudes de la UI y las reenvía a la base de datos.
    - Devuelve los datos formateados a la UI.
- Dependencias:
    - UI
    - Database

1.3.3 Módulo Database

El módulo Database maneja la persistencia de datos de la aplicación y contiene los siguientes submódulos:

Repositories

- Proporciona una capa de abstracción para acceder a los datos y proveerlos al Middleware.
- Interacciones:
    - Usa DAOs para interactuar con la base de datos.

DAOs (Data Access Objects)

- Interfaz encargada de la comunicación con la base de datos.
- Interacciones:
    - Usa Entities para mapear los datos almacenados.

Entities

- Define los modelos de datos utilizados en la base de datos.
- Interacciones:
    - Es utilizada por DAOs para almacenar y recuperar información estructurada.

1.3.4 Módulo Send (Envío de Datos)

Es el módulo que gestiona el envío de información dentro de la aplicación.

- Interacciones:
    - Interactúa con la UI para recibir los datos a enviar.
- Dependencias:
    - UI

1.3.5 Módulo Test (Pruebas y Validación)

Es el módulo encargado de facilitar pruebas y validaciones dentro del sistema.

- Interacciones:
    - Se conecta con la UI y la Database para ejecutar las inserciones en la base de datos y verificar su funcionamiento.
- Dependencias:
    - UI
    - Database

1.4 Relaciones entre Módulos

- La UI usa Middleware para acceder a los datos.
- Middleware interactúa con Database a través de Repositories.
- Repositories usan DAOs para acceder a las Entities.
- Send interactúa con la UI para enviar datos
- Test depende de los DAOs y es usado por la UI para realizar pruebas del sistema.

1.5 Interfaces

1.5.1 Interfaces del Módulo DAOs (Data Access Objects)

El módulo DAOs proporciona una interfaz para acceder a los datos almacenados en la base de datos. Sirve como intermediario entre el módulo Database y los demás módulos del sistema, asegurando una comunicación estructurada y eficiente.

Funciones Principales:

- CRUD (Create, Read, Update, Delete): Proporciona métodos para la gestión de datos en la base de datos.
- Consultas Específicas: Ofrece métodos optimizados para recuperar datos según distintos criterios.
- Manejo de Conexiones: Administra la comunicación con la base de datos para optimizar el rendimiento.

Interacciones:

- Es utilizado por los Repositories dentro del módulo Database.
- Interactúa con Entities para mapear datos almacenados en la base de datos.
- Permite al módulo Test realizar validaciones mediante inserciones y consultas de prueba.

1.5.2 Interfaces del Módulo Send (Envío de Datos)

El módulo Send gestiona el envío de datos a través de distintos canales de comunicación, incluyendo WhatsApp y SMS. Este módulo abstrae la lógica de comunicación y ofrece una interfaz unificada para realizar envíos eficientes.

Funciones Principales:

- Envío de Mensajes: Permite enviar mensajes de texto y multimedia.
- Soporte para WhatsApp y SMS: Implementa métodos específicos para cada canal.

Interacciones:

- UI: Recibe los datos desde la interfaz de usuario.

2.Documentación de la Vista de Componentes y Conectores

2.1 Descripción General

Esta documentación describe la Vista de Componentes y Conectores del sistema, representando la forma en que los diferentes componentes interactúan entre sí y la naturaleza de las conexiones entre ellos. Esta vista muestra cómo se comporta el sistema en ejecución.

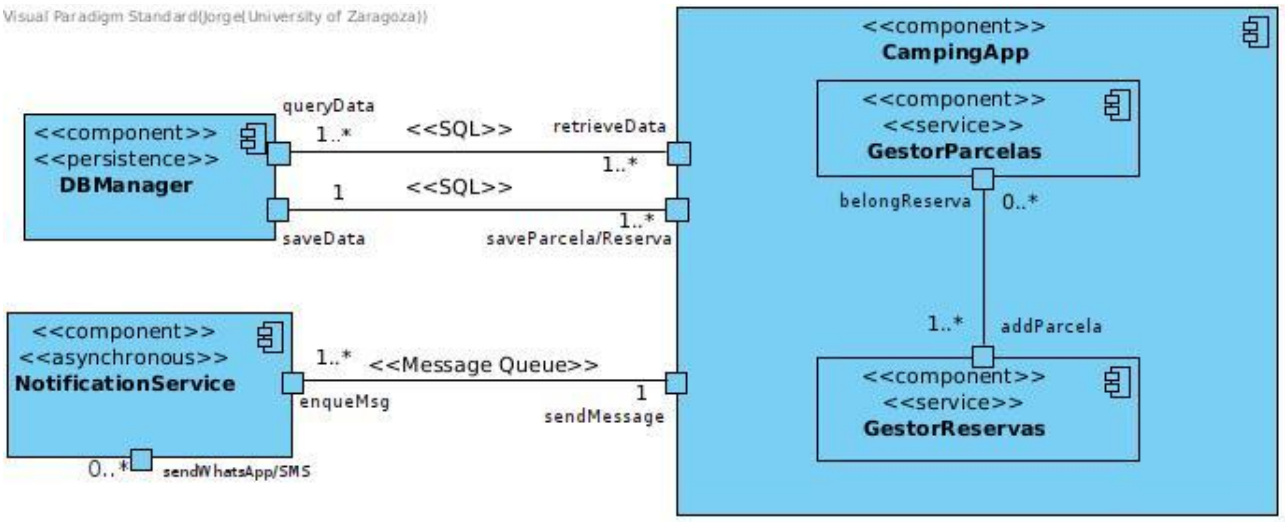
La arquitectura se basa en componentes bien definidos, organizados según su rol en el sistema y conectados mediante diferentes tipos de conectores que facilitan la comunicación y el procesamiento de datos.

Los componentes principales incluyen:

- CampingApp: la aplicación central que gestiona las reservas y parcelas.
- GestorParcelas: servicio encargado de manejar la información sobre las parcelas.
- GestorReservas: servicio encargado de administrar las reservas.
- DBManager: componente de persistencia que gestiona la base de datos.
- NotificationService: servicio de notificación asíncrono.

Los conectores definen las interacciones entre estos componentes y pueden ser de diferentes tipos. A continuación, se presenta el diagrama de esta vista, con las interacciones especificadas.

2.2 Vista de Componentes y Conectores



2.3 Descripción de componentes

2.3.1. CampingApp

Componente principal de la aplicación, que agrupa la lógica de gestión de reservas y parcelas.

- Subcomponentes:
    - GestorParcelas: Maneja la información de las parcelas.
    - GestorReservas: Gestiona la creación, modificación y cancelación de reservas. Así como la adición de parcelas a la reserva.
- Interacciones:
    - Se comunica con DBManager para la persistencia de datos, usa SQL para realizar consultas y almacenar datos.
    - Usa NotificationService para enviar notificaciones de confirmación de reservas.

2.3.2 DBManager

Componente de persistencia responsable de gestionar la base de datos.

- Responsabilidades:
    - Ejecutar consultas SQL para recuperar y almacenar información.
    - Mantener la consistencia y disponibilidad de los datos.
- Interacciones**:**
    - Responde a consultas de CampingApp.
    - Recibe datos de GestorReservas y GestorParcelas.

2.3.3. NotificationService

Componente encargado de gestionar las notificaciones de forma asincrónica.

- Responsabilidades**:**
    - Procesar mensajes de notificación en una cola de mensajes.
    - Enviar SMS o mensajes de WhatsApp según corresponda.
- Interacciones**:**
    - Recibe mensajes encolados de CampingApp.
    - Envía confirmaciones de reserva a los usuarios.

2.4 Descripción de conectores

2.4.1 SQL

Es un conector del tipo “acceso a datos”, que proporciona servicios de comunicación, ya que permite el acceso del componente CampingApp a la fuente de datos persistente mantenida por el componente DBManager.

- Utilizado por DBManager para recibir y procesar consultas.
- Relaciona CampingApp con DBManager.
- Roles:
    - queryData para consultas de información. Tiene una cardinalidad de 1..* ya que puede tener varias consultas(lecturas) de forma simultánea y concurrente.
    - saveData para almacenar nuevas reservas y parcelas. A diferencia de queryData esta tiene cardinalidad 1 ya que las escrituras tienen que ser en exclusión mutua y no puede haber escrituras concurrentes.

2.4.2 Cola de Mensajes

Es un conector del tipo “eventos”, que provee servicios de coordinación y comunicación, permitiendo la transferencia de información de manera asíncrona.

- Utilizada por NotificationService para procesar mensajes de manera asíncrona.
- Permite la comunicación diferida y desacoplada entre CampingApp y NotificationService.
- Roles:
    - enqueMsg para encolar mensajes de notificación.Puede recibir mensajes de forma concurrente, por ende la cardinalidad es 1..*.
    - sendMessage para mandar los mensajes por parte de la aplicación central. En una aplicación los mensajes solo se pueden mandar de 1 en 1.

2.4.3 Referencias Directas

Se tratan de “llamadas a procedimientos” que permiten la comunicación entre el gestor de parcelas y el gestor de reservas mediante diferentes técnicas de invocación.

- Conectan GestorParcelas y GestorReservas dentro de CampingApp.
- Roles:
    - belongReserva: Indica la asociación entre reservas y parcelas. Una parcela puede pertenecer a varias reservas distintas.
    - addParcela: Permite a GestorParcelas asignar nuevas parcelas a reservas. Una reserva consta de 1 o más parcelas.

3.Documentación de la Vista de Distribución

3.1 Vista de Despliegue

3.1.1 Descripción General

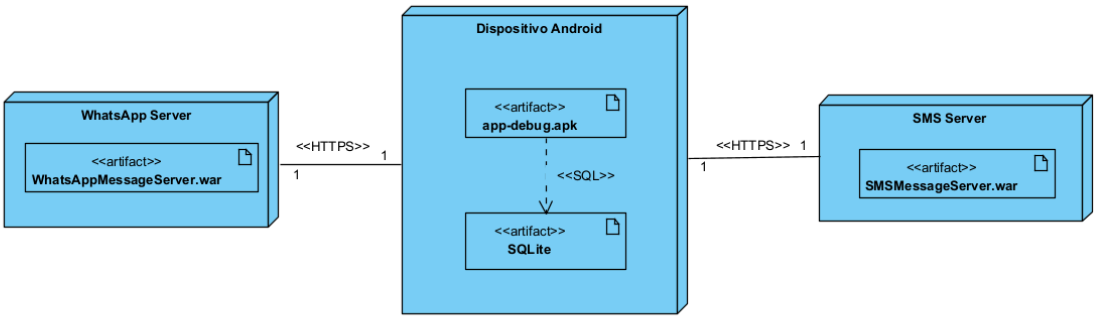
Este documento describe la vista de distribución del sistema, destacando cómo los diferentes artefactos y componentes están desplegados en los distintos nodos y la forma en que se comunican entre sí.

El sistema está compuesto por tres principales nodos de despliegue:

- Dispositivo Android: Contiene la aplicación móvil y su base de datos local.
- WhatsApp Server: Gestiona la comunicación con WhatsApp.
- SMS Server: Maneja el envío y recepción de mensajes SMS.

Todos los nodos se comunican a través del protocolo HTTPS, asegurando la transmisión segura de datos.

3.1.2 Vista de Despliegue



3.1.3 Descripción de Nodos y Artefactos

3.1.3.1 Dispositivo Android

Este nodo representa el dispositivo en el cual se ejecuta la aplicación móvil. Contiene los siguientes artefactos:

- app-debug.apk (Aplicación Android)
    - Es el archivo ejecutable de la aplicación móvil instalada en el dispositivo Android.
    - Se encarga de gestionar la interfaz de usuario y las interacciones del usuario con el sistema.
    - Se comunica con los servidores a través de HTTPS para enviar y recibir información.
    - Interactúa con la base de datos local mediante consultas SQL.

- SQLite (Base de Datos Local)
    - Es el sistema de almacenamiento local utilizado por la aplicación Android.
    - Permite almacenar mensajes, configuraciones y datos temporales sin necesidad de conexión a Internet.
    - Es accedido por app-debug.apk a través de consultas SQL.

3.1.3.2 WhatsApp Server

Este nodo representa el servidor encargado de gestionar la mensajería a través de WhatsApp. Contiene el siguiente artefacto:

- WhatsAppMessageServer.war (Aplicación Web del Servidor WhatsApp)
    - Se encarga de procesar y enviar mensajes a través de la API de WhatsApp.
    - Recibe solicitudes desde la aplicación Android a través de HTTPS.
    - Devuelve respuestas con información sobre el estado de los mensajes enviados.

3.1.3.3 SMS Server

Este nodo representa el servidor que maneja la comunicación vía SMS. Contiene el siguiente artefacto:

- SMSMessageServer.war (Aplicación Web del Servidor de Mensajería SMS)
    - Gestiona el envío y recepción de mensajes SMS.
    - Recibe peticiones desde la aplicación Android a través de HTTPS.
    - Proporciona confirmaciones sobre los mensajes enviados y recibidos.

3.1.4 Relaciones y Comunicación entre Nodos

El sistema utiliza los siguientes protocolos de comunicación:

- app-debug.apk y SQLite
    - La aplicación Android interactúa con la base de datos local SQLite utilizando consultas SQL.
- app-debug.apk y WhatsApp Server
    - La aplicación móvil envía y recibe mensajes a través del servidor de WhatsApp mediante el protocolo HTTPS.
    - WhatsApp Server procesa la solicitud y devuelve la confirmación del mensaje enviado.
- app-debug.apk y SMS Server
    - La aplicación móvil se comunica con el servidor de SMS mediante HTTPS.
    - SMS Server gestiona la transmisión del mensaje y envía una respuesta con el estado del envío.

3.1.5 Seguridad y Consideraciones de Despliegue

Para garantizar la seguridad y el correcto funcionamiento del sistema, se han implementado las siguientes medidas:

- Uso de HTTPS: Todas las comunicaciones entre el dispositivo Android y los servidores están cifradas para proteger la integridad y privacidad de los datos.
- Persistencia de Datos Locales: SQLite permite almacenar mensajes localmente, asegurando el acceso a datos cuando no hay conexión a Internet.
- Modularidad: Cada servidor se encarga de una tarea específica (WhatsApp o SMS), lo que mejora la escalabilidad del sistema.

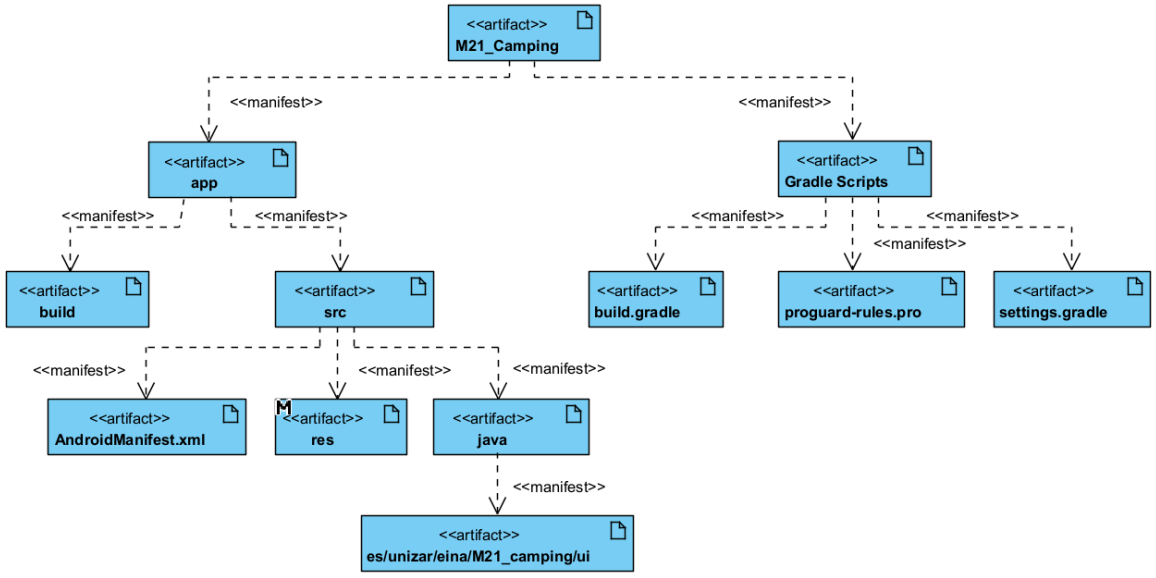
3.2. Vista de Instalación

3.2.1. Descripción General

La Vista de Instalación muestra cómo se organizan los archivos del proyecto antes y después de la instalación en un dispositivo Android. Antes de la instalación, la estructura del proyecto contiene el código fuente, recursos y configuraciones dentro del entorno de desarrollo. Una vez instalada, la aplicación se distribuye en el sistema de archivos de Android en directorios específicos, asignando componentes como el código compilado, recursos visuales, configuraciones de usuario, bases de datos y archivos generados por la aplicación. Esta organización permite el correcto funcionamiento y almacenamiento de la app en el entorno de producción.

3.2.2. Estructura del Proyecto antes de la Instalación

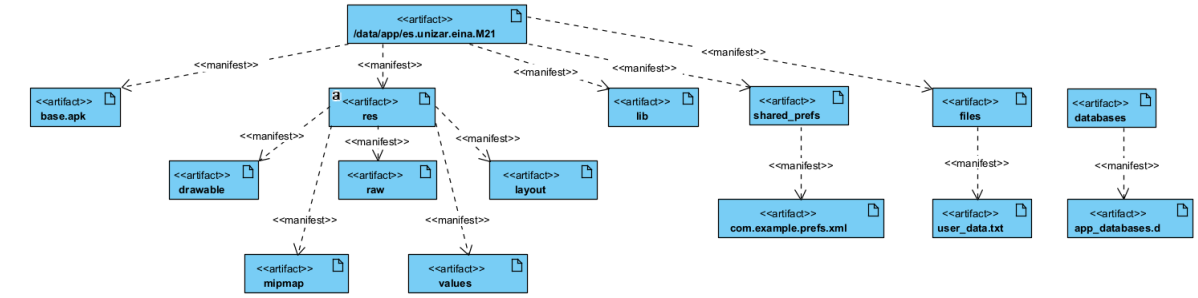
Para entender cómo se organiza todo, primero vamos a repasar la estructura de archivos del proyecto antes de la instalación. Esto incluye:



En el entorno de producción, una vez que el archivo APK se instala en el dispositivo, se distribuye en una estructura de directorios específica, que se va a mostrar a continuación.

3.2.3. Distribución de Archivos en el Sistema de Archivos del Dispositivo

Cuando la aplicación se instala, el sistema de archivos de Android organiza los archivos de la APK en una estructura como la siguiente, dentro de la carpeta de aplicaciones de Android:



Asimismo, ciertos componentes de la aplicación se asignan al sistema de archivos una vez que la app esté instalada:

Código Fuente y Archivos Compilados (base.apk)

- El archivo APK (por ejemplo, base.apk) es el archivo principal de la aplicación que se encuentra en el directorio /data/app/<es.unizar.eina.M21>/. Este archivo contiene todo el código compilado y empaquetado de la aplicación.

Archivos de Recursos (Imágenes, Layouts, etc.)

- Los archivos de recursos como imágenes (drawable/), layouts (layout/), y valores (values/ como strings.xml) se extraen del archivo APK y se colocan en los directorios adecuados dentro de la estructura de la app instalada.

Preferencias Compartidas

- Las preferencias compartidas (shared_prefs/) almacenan configuraciones específicas de la aplicación, como las configuraciones del usuario, preferencias y otros parámetros. En este directorio se guardan los archivos XML de configuración como com.example.prefs.xml.

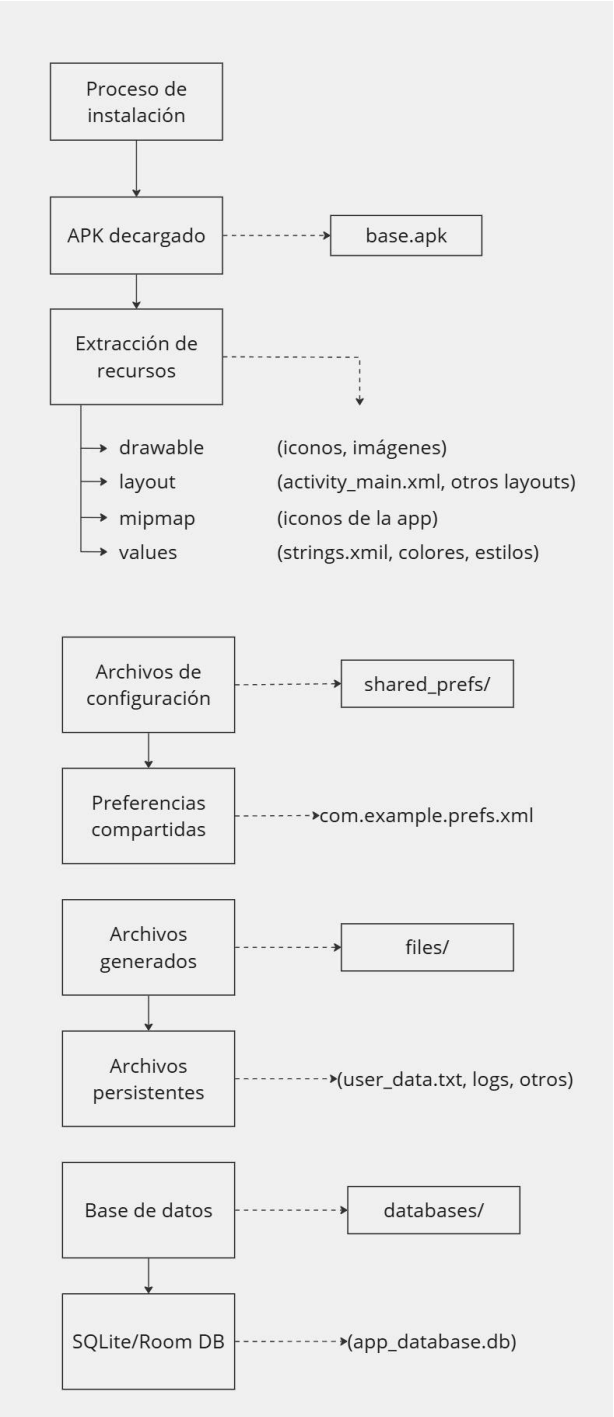
Bases de Datos Locales

- Si la app utiliza bases de datos locales (por ejemplo, con Room o SQLite), estas bases de datos se almacenan en el directorio databases/, donde se guardan los archivos de bases de datos como app_database.db.

Archivos de la Aplicación

- En el directorio files/ se almacenan los archivos creados o utilizados por la app, como archivos de texto, imágenes generadas por el usuario u otros archivos persistentes.

A continuación, se muestra esta organización de manera visual en un esquema que asocia los componentes de la aplicación con los directorios del sistema de archivos de Android.



3.3 Vista de Asignación de Trabajo

El proyecto contaba con dos desarrolladores Jorge Hernández y Laura Hernández por lo que se ha distribuido el diseño e implementación de los distintos módulos y submódulos ya comentados en la vista de módulos. Ha sido un trabajo conjunto en el que la comunicación para vincular y conectar los distintos módulos ha sido clave.

MÓDULO	SUBSISTEMA	RESPONSABLE
UI	VIEWS	Jorge Hernández
	LISTADAPTERS	Jorge Hernández
	VIEWHOLDERS	Laura Hernández
DATABASE	RESPOSITORIES	Jorge Hernández
	DAOS	Laura Hernández
	ENTITIES	Laura Hernández
MIDDLEWARE	VIEWMODELS	Jorge Hernández
TEST	PRUEBAS UNITARIAS	Laura Hernández
	PRUEBAS DE SOBRECARGA	Jorge Hernández
	PRUEBAS DE VOLUMEN	Jorge Hernández
SEND	SMS	Laura Hernández
	WHATSAPP	Laura Hernández