**Informe del Proyecto: Aplicación de Registro de Medidores**

**Introducción**

Este informe detalla el proceso de diseño y desarrollo de una aplicación móvil para el registro de mediciones de distintos tipos de consumo (agua, luz, gas) utilizando programación orientada a objetos en Kotlin. La aplicación está destinada a registrar el tipo de gasto, el valor medido y la fecha del registro, garantizando que los datos se mantengan de manera persistente y accesible incluso después de cerrar la aplicación.

El proyecto se implementa siguiendo los requisitos solicitados, que incluyen la creación de interfaces de usuario interactivas, gestión de eventos, uso de recursos gráficos, soporte multilingüe y manejo de una base de datos SQLite utilizando la librería ROOM.

**1. Diseño y Programación de la Solución en Kotlin**

El desarrollo de la aplicación se basa en los principios de la Programación Orientada a Objetos (POO) utilizando el lenguaje Kotlin, para el desarrollo de aplicaciones Android. Se estructuró el proyecto en clases, en el proyecto llamadas fragmentos, que representan tanto las entidades como los componentes de la interfaz de usuario (UI).

**1.1. Clases Principales para la persistencia de datos**

* **Clase Medicion**: Esta clase define el modelo de datos que se utilizará para registrar las mediciones. Contiene atributos como el tipo de gasto (agua, luz, gas), el valor de la medición, y la fecha en que fue tomada.
* **Clase MedicionViewModel**: Utilizamos un ViewModel para gestionar la lógica de presentación, manteniendo los datos de las mediciones de manera independiente del ciclo de vida de las actividades o fragmentos.

**2. Diseño de la Interfaz de Usuario (UI)**

Se desarrollaron dos pantallas principales en la aplicación, cumpliendo con los requisitos de funcionalidad y diseño.

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**2.1 Pantalla de Listado de Mediciones**

En esta pantalla se muestra una lista dinámica que despliega todas las mediciones registradas por el usuario. Para gestionar las mediciones, se utiliza un RecyclerView, que permite visualizar los registros en una lista desplazable.

**Captura de pantalla:**

Texto

Descripción generada automáticamente

**2.2 Pantalla de Formulario para Registro de Mediciones**

El formulario permite al usuario ingresar una nueva medición. Se compone de varios campos, como un campo para el valor de la medición, un selector de fecha y un menú desplegable para seleccionar el tipo de medición (agua, luz, gas). También se incluye un botón para guardar la medición en la base de datos.

* **Componentes utilizados**: TextField (para el valor), DatePicker (para la fecha), DropdownMenu (para el tipo de gasto).

**Captura de pantalla:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**3. Gestión de Eventos**

**3.1. Navegación entre Pantallas**

Para la navegación entre la pantalla de listado y la pantalla del formulario, se utiliza el componente de navegación nativo de Android. Un botón en la pantalla de listado permite acceder al formulario para ingresar un nuevo registro. Este evento se maneja mediante un OnClickListener.

Texto

Descripción generada automáticamente

**3.2. Guardar una Nueva Medición**

El botón "Guardar" en el formulario desencadena un evento que valida los campos y, si es correcto, guarda el nuevo registro en la base de datos utilizando una corutina para evitar bloquear el hilo principal.

Texto

Descripción generada automáticamente

**4. Implementación del ViewModel**

Se implementa un ViewModel para gestionar la lógica de presentación de los datos. Este componente se encarga de mantener la información de las mediciones y coordina su persistencia en la base de datos. De esta forma, se garantiza que los datos persisten durante cambios de configuración como la rotación de pantalla.

**5. Uso de ROOM y Persistencia de Datos**

Para la persistencia de los registros, se utiliza la biblioteca ROOM, que facilita la interacción con una base de datos SQLite. Esto asegura que los registros se mantengan disponibles incluso después de cerrar la aplicación.

**6.1. Entidades y DAO**

* **Entidad Medicion**: Define la estructura de los registros de la base de datos.
* **DAO (Data Access Object)**: Se encarga de las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre la base de datos.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

**6.2. Persistencia Asíncrona**

Para evitar bloqueos en el hilo principal, las operaciones de escritura y lectura en la base de datos se realizan de manera asíncrona utilizando corrutinas de Kotlin, de forma que la renderización de los datos esta separada de los datos de mediciones.

Texto

Descripción generada automáticamente

**7. Implementación de Listas Dinámicas**

Para mostrar las mediciones en la pantalla de listado, se utiliza un RecyclerView (en XML), que permite cargar los datos de forma eficiente. Cada ítem de la lista se construye dinámicamente a partir de los datos guardados en la base de datos, como ya mostrado anteriormente.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

**8. Conclusión**

El desarrollo de esta aplicación ha permitido implementar una solución robusta para la gestión de mediciones de diferentes tipos de gasto, utilizando Kotlin y las mejores prácticas de desarrollo en Android. La aplicación es capaz de manejar grandes cantidades de datos de manera eficiente, y persiste los datos utilizando ROOM y corrutinas para garantizar una experiencia fluida al usuario.