

# Manual Técnico

## Cobros de Mercado Móvil

Desarrollo de Aplicaciones Móviles Básicas

Wilber Salvador Sorto Sánchez & Abigail Espinal Mejía



## Tabla de contenido

|  |          |
|--|----------|
| <b>Introducción .....</b>                                | <b>1</b> |
| <i>Descripción .....</i>                                 | <i>1</i> |
| <b>Contenido .....</b>                                   | <b>1</b> |
| <b>Introducción al Sistema (Stack Tecnológico) .....</b> | <b>1</b> |
| <i>Instalación y Configuración .....</i>                 | <i>2</i> |
| <i>Arquitectura del Proyecto .....</i>                   | <i>2</i> |
| <i>Diseño de Base de Datos .....</i>                     | <i>2</i> |
| <i>Implementación de Funciones Clave .....</i>           | <i>3</i> |

# Manual Técnico

## Introducción

El presente Manual Técnico tiene como finalidad documentar la arquitectura, desarrollo e implementación de la aplicación "Cobros de Mercado Móvil". Este documento está dirigido al equipo de TI y desarrolladores encargados del mantenimiento, escalabilidad y soporte del sistema. Se detallan las tecnologías utilizadas (Kotlin, Room), la estructura de la base de datos local y los procedimientos necesarios para la compilación y despliegue del código fuente en entornos Android.

## Descripción

*La documentación técnica es vital para garantizar la continuidad del proyecto. Dado que el sistema opera con lógica de negocio crítica (cálculos financieros offline y sincronización de datos locales), es necesario comprender la estructura interna de clases, la gestión de la base de datos SQLite y los permisos de hardware (GPS) para realizar futuras actualizaciones o correcciones de errores sin afectar la integridad de los datos históricos.*

## Contenido

*El documento abarca los requisitos previos de software (Android Studio, JDK), la arquitectura del proyecto basada en paquetes, el diccionario de datos de las entidades Room (Usuario, Comerciante, Cobro), fragmentos clave de código para funciones avanzadas como geolocalización, y una guía de solución de problemas comunes para desarrolladores.*

## Introducción al Sistema (Stack Tecnológico)

| <i>Tecnología</i>        | <i>Versión</i> | <i>Propósito</i>                          |
|--------------------------|----------------|---|
| <i>Kotlin</i>            | <i>1.9.20</i>  | <i>Lenguaje de programación principal</i> |
| <i>Android SDK</i>       | <i>API 29+</i> | <i>Plataforma de desarrollo</i>           |
| <i>Room Database</i>     | <i>2.6.1</i>   | <i>Persistencia de datos local</i>        |
| <i>Material Design</i>   | <i>1.11.0</i>  | <i>Componentes de UI</i>                  |
| <i>Location Services</i> | <i>21.1.0</i>  | <i>Captura de ubicación GPS</i>           |
| <i>Coroutines</i>        | <i>1.7.0</i>   | <i>Programación asíncrona</i>             |

## Instalación y Configuración

Para configurar el entorno de desarrollo:

1. **Requisitos:** Android Studio Hedgehog (2023.1.1)+ y JDK 17.
2. **Clonado:** Ejecutar git clone <https://github.com/Wilber01/CobrosMarketMovil.git>
3. **Sincronización:** Abrir el proyecto en Android Studio y permitir la sincronización de Gradle.
4. **Permisos:** Verificar en AndroidManifest.xml los permisos ACCESS\_FINE\_LOCATION.
5. **Ejecución:** Compilar apuntando a un emulador o dispositivo físico (API 29+).

## Arquitectura del Proyecto

El proyecto sigue una estructura modular clara:

- **data/:** Contiene la capa de datos.
  - **entities/:** Modelos de Room (Usuario, Comerciante, Cobro).
  - **dao/:** Interfaces de acceso a datos (Data Access Objects).
  - **database/:** Configuración de AppDatabase.
- **/:** Contiene la capa de presentación (Activities y Adapters).
  - Incluye MainActivity, RegistroCobroActivity, etc.
- **res/:** Recursos estáticos (Layouts XML, Values, Drawables).

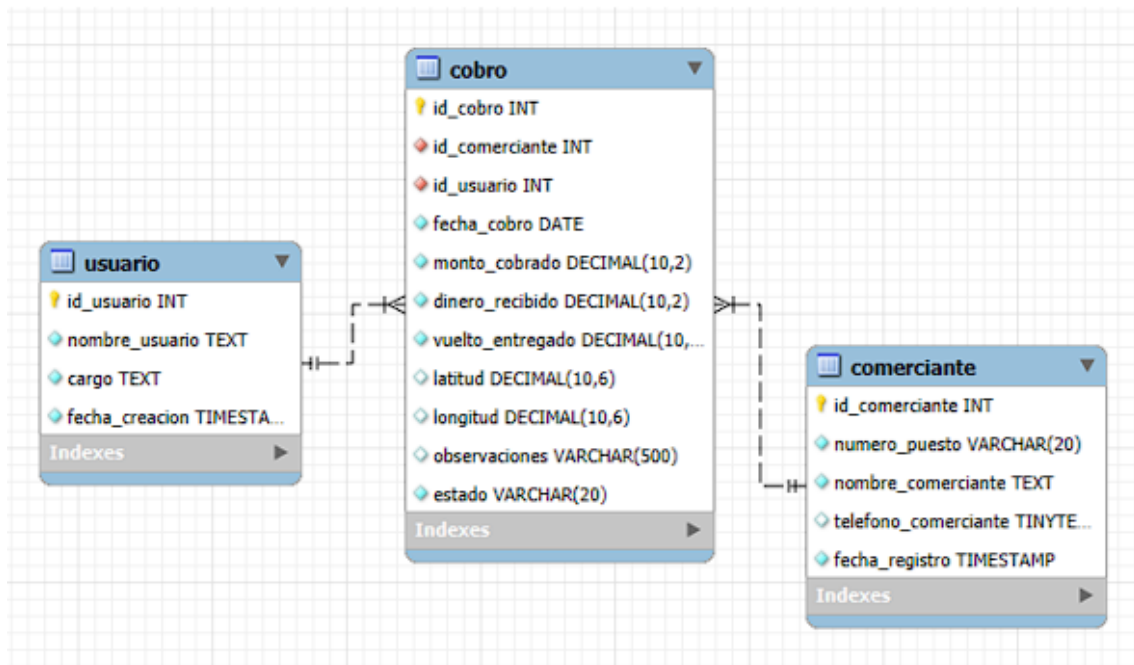
## Diseño de Base de Datos

El sistema utiliza SQLite gestionado por Room con tres tablas principales relacionadas:

1. **Usuarios (usuarios):** Almacena credenciales y roles.
  - PK: id\_usuario (INT). Campos: nombre\_usuario, cargo, fecha\_creacion.
2. **Comerciantes (comerciantes):** Catálogo de contribuyentes.
  - PK: id\_comerciante (INT). Campos: numero\_puesto (Unique), nombre, telefono.
3. **Cobros (cobros):** Registro transaccional.
  - PK: id\_cobro (INT). FKs: id\_comerciante, id\_usuario.

- *Datos: monto, dinero\_recibido, vuelto (Calc), latitud/longitud, estado.*

## Diagrama de Entidad de Relación



## Implementación de Funciones Clave

### Lógica de Negocio Relevante:

- **Vuelto Automático:** Calculado en tiempo real en la UI ( $\text{vuelto} = \text{dinero\_recibido} - \text{monto\_cobrado}$ ).
- **GPS:** Implementado con `FusedLocationProviderClient`. Se captura `lastLocation` al guardar el cobro.
- **Reportes:** Consultas SQL nativas en DAOs usando agregaciones: `SELECT SUM(monto_cobrado) FROM cobros WHERE fecha....`

### Solución de Problemas Comunes (Troubleshooting):

- **GPS no captura:** Verificar permisos en tiempo de ejecución y asegurar que el dispositivo esté al aire libre (señal GPS clara).
- **Error Comerciante Duplicado:** La restricción `UNIQUE` en `numero_puesto` impide duplicados; usar la búsqueda antes de crear.
- **Cobros no visibles:** Revisar formato de fecha `yyyy-MM-dd` en las consultas SQL, ya que SQLite es sensible al formato de texto en fechas.