**Tabla de Contenido**

[1](#_heading=h.hhsf9ipax8qq) Objetivo 2

[2](#_heading=h.cpoei860bk2t) Estrategia 2

[2.1](#_heading=h.r4dh854waxam) Criterios de la estrategia 2

[2.2](#_heading=h.p7uqsla6c45p) Estrategia general 2

[2.3](#_heading=h.i2ltm5f4it92) Diseño Conceptual 2

[2.3.1](#_heading=h.nubcuaspi7lz) Módulos 2

[2.3.2](#_heading=h.s0z8dxsm2i9q) Funcionalidades asociadas 3

[2.4](#_heading=h.ef7zr16z2415) Estimación preliminar del proyecto 3

# Objetivo

El objetivo de este documento es definir la estrategia general del desarrollo de la aplicación móvil creada en **Android Studio**, enfocada en la gestión de envíos y operaciones logísticas.

El proyecto busca ofrecer una solución práctica, moderna y funcional para administrar usuarios, mercancías y envíos dentro de una plataforma Android.

A través de este documento se establecen los lineamientos de diseño, organización de módulos, estimaciones y criterios técnicos que guiarán el desarrollo.

# Estrategia

La aplicación se desarrolla bajo una arquitectura **modular** que divide la lógica en capas bien definidas: **Activities** (interfaz e interacción), **Models** (estructuras de datos), **Database** (persistencia) y **Utils** (herramientas auxiliares).  
 Se emplea un enfoque **iterativo**, permitiendo construir, probar y mejorar cada componente en ciclos cortos.  
 El lenguaje principal es **Java**, con uso de **XML** para el diseño visual y **SQLite** como base de datos local.  
 El control de versiones se realiza mediante **GitHub** para garantizar trazabilidad y trabajo colaborativo.

## Criterios de la estrategia

Mantener una arquitectura clara y separada por capas (Activities, Models, Database, Utils).

Asegurar la integridad y consistencia de datos mediante SQLite y control de transacciones.

Aplicar diseño responsivo con interfaces adaptables a distintos dispositivos Android.

Reutilizar componentes mediante clases auxiliares centralizadas en el paquete **utils**.

Documentar cada módulo y mantener control de cambios versionado en Git.

Validar el correcto funcionamiento con pruebas en emuladores y dispositivos físicos.

## Estrategia general

El desarrollo se basará en el ciclo **planificación → implementación → prueba → integración**, donde cada módulo se completará por etapas funcionales.

Las **Activities** manejarán la interfaz y eventos del usuario, los **Models** representarán los objetos del sistema (usuarios, envíos, recolectores, vehículos), la **Database** contendrá las operaciones CRUD, y **Utils** centralizará métodos comunes (validaciones, formatos, manejo de Intents o Toasts).

Cada entrega incluirá código operativo, verificación funcional y actualización del control de cambios.

## Diseño Conceptual

El diseño conceptual de la aplicación se basa en una **arquitectura modular** que separa claramente la lógica de negocio, la interfaz y la gestión de datos.  
 El proyecto está organizado en cuatro componentes principales: **Activities**, **Models**, **Database** y **Utils**, los cuales interactúan de manera coordinada para mantener un código limpio, reutilizable y fácil de mantener.

Cada módulo cumple una función específica dentro del flujo general de la aplicación:

* **Activities:** manejan la interfaz gráfica del usuario (UI) y los eventos generados por sus acciones. Cada Activity representa una pantalla diferente del sistema, como inicio de sesión, registro, creación de envío o menú principal.
* **Models:** representan las entidades o clases de datos del sistema (por ejemplo: User, Shipment, Truck, Collector). Estas clases almacenan la información y la transportan entre los distintos módulos.
* **Database:** contiene las clases encargadas del acceso a datos, como la clase DbHelper, que crea y actualiza la base de datos SQLite, y los métodos CRUD (crear, leer, actualizar, eliminar).
* **Utils:** agrupa herramientas auxiliares y métodos reutilizables, como validaciones, formateo de datos, envío de mensajes con Toast, y manejo de Intent mediante putExtra y getExtra.

La comunicación entre módulos se da de la siguiente manera:

1. **El usuario** interactúa con una **Activity**.
2. La **Activity** toma los datos ingresados y los convierte en objetos del paquete **Models**.
3. Si se requiere guardar o consultar información, la **Activity** se comunica con el módulo **Database**.
4. Las **Utils** se utilizan en cualquier parte del código para simplificar tareas comunes y evitar repeticiones.

Esta estructura garantiza una alta cohesión dentro de cada módulo y bajo acoplamiento entre ellos, lo que facilita la escalabilidad y mantenimiento del sistema.

### Módulos

| **Módulo** | **Descripción** |
| --- | --- |
| **Activities** | Contiene las pantallas principales de la app (Login, Registro, Home, Envíos, Recolección). Gestiona la interacción del usuario. |
| **Models** | Define las clases de datos (User, Shipment, Truck, Collector, TrackingEvent) usadas en la aplicación. |
| **Database** | Implementa la base de datos local mediante SQLiteOpenHelper y gestiona la creación de tablas y consultas SQL. |
| **Utils** | Contiene funciones auxiliares (manejo de Toasts, validaciones, formateo de texto, paso de datos entre Activities). |

### Funcionalidades asociadas

| **Módulo** | **Funcionalidad** |
| --- | --- |
| Activities | Mostrar vistas, capturar datos, navegar entre pantallas. |
| Models | Representar entidades del sistema y transportar datos. |
| Database | Crear, leer, actualizar y eliminar información (CRUD). |
| Utils | Validar formularios, mostrar mensajes, convertir datos, manejar extras. |

## Estimación preliminar del proyecto

| **Ciclo 1** | | |
| --- | --- | --- |
| **Funcionalidad** | **Tamaño (LOC)** | **Tiempo (Horas)** |
| Login y Registro de Usuario | 420 | 10 |
| Gestión de envíos (crear, listar, eliminar) | 620 | 12 |
| Implementación de DbHelper y estructura SQLite | 480 | 10 |
| Modelado de clases (User, Shipment, Truck, Collector, TrackingEvent) | 400 | 8 |
| Funciones auxiliares (Utils: Toasts, putExtra, validaciones) | 250 | 5 |
| Interfaz principal y navegación entre pantallas (HomeActivity) | 350 | 7 |
| Pruebas, depuración y ajustes finales | 300 | 6 |
|  |  |  |
| TOTAL | 2820 | 58 Horas |

| **CONTROL DE CAMBIOS** | | |
| --- | --- | --- |
| **Fecha** | **Descripción** | **Autor(es)** |
| 25/08/2025 | Documento de Estrategia | LE |