Plan de Implementación – Sistema de Encomiendas

# 1. Soporte de Ejecución

## 1.1 Definir criterios de completitud del diseño

• Cobertura funcional: todos los casos de uso priorizados (UC-001 a UC-006) y las historias de usuario implementadas están representados en el diseño.

• Consistencia y trazabilidad: los requisitos funcionales tienen un elemento de diseño asociado y no existen contradicciones entre diagramas.

• Validación: se realizó al menos una sesión de revisión con stakeholders y el equipo de desarrollo verificó la viabilidad técnica.

• Métricas de calidad: cobertura de diseño ≥ 80%, tiempo de respuesta ≤ 5 segundos en operaciones críticas.

## 1.2 Estándares de implementación

1.Convenciones de codificación:

Clases en PascalCase → LoginActivity, RegisterActivity, DbHelper.

Métodos y variables en camelCase → validarUsuario(), numeroGuia.

Constantes en MAYÚSCULAS\_CON\_GUIONES → MAX\_INTENTOS\_LOGIN.

Paquetes en minúsculas y organizados por funcionalidad:

db → acceso a datos (DbHelper).

ui → pantallas de usuario (LoginActivity, RegisterActivity, TrackingActivity, etc.).

util → clases auxiliares.

Uso de Javadoc para documentar clases y métodos.

**2.Uso de control de versiones (Git)**

**\*Se utilizó Git para llevar un control histórico del proyecto en Android Studio.**

**\*El repositorio se mantuvo en GitHub/GitLab (o local en el equipo de desarrollo).**

**\*Solo hubo un desarrollador principal, por lo que no se implementaron revisiones por pares ni pull requests.**

**\*Se siguió la convención de commits claros y en español, por ejemplo:**

**-feat: crear tabla shipments en DbHelper**

**-fix: corregir validación en LoginActivity**

**-style: ajustar nombres de variables en RegisterActivity**

**\*La rama principal utilizada fue main, sin necesidad de múltiples ramas de integración.**

**\*Git permitió versionar cambios importantes, hacer retrocesos cuando fue necesario y mantener un registro del avance del proyecto.**

**3.Lineamientos de seguridad y rendimiento**

**\*Validación de formularios en RegisterActivity, LoginActivity y demás pantallas críticas.**

**\*Tiempo máximo de respuesta: ≤ 5 segundos en operaciones críticas como registrar encomienda o consultar estado.**

**\*Uso de una sola base de datos SQLite (ingsoft\_logistics.db) que integra:**

**-users → autenticación y gestión de perfiles.**

**-shipments, tracking\_events → gestión y seguimiento de envíos.**

**-guides, trucks, collectors, pickups, route\_assignments → soporte logístico.**

**-Consultas optimizadas con índices en columnas críticas (email, shipment\_id, code).**

**-Separación de acceso a datos en repositorios/DAO para cada entidad.**

## 1.3 Estrategia de implementación

Fase 1 – Configuración inicial:Creación del proyecto en Android Studio, definición de paquetes (db, ui, util) y configuración del repositorio Git.

Fase 2 – Implementación de la base de datos SQLite:Definición de la clase DbHelper con la creación de las tablas (users, shipments, guides, trucks, collectors, pickups, route\_assignments, tracking\_events). Implementación de métodos para insertar, consultar, actualizar y eliminar registros.

Fase 3 – Funcionalidades de autenticación y gestión de usuarios:Desarrollo de pantallas de registro y login (RegisterActivity, LoginActivity).Validación de credenciales contra la tabla users.

Fase 4 – Funcionalidades de gestión de envíos y logística:Desarrollo de casos de uso principales (UC-001 a UC-006):

\*Registrar encomienda.

\*Consultar estado.

\*Confirmar entrega.

\*Generar guías de envío.

\*Gestionar reparto de camiones.

\*Actualizar estado de envíos.

Fase 5 – Seguridad y rendimiento:Validaciones de entrada en formularios, optimización de consultas a la BD local.

Fase 6 – Pruebas e inspecciones:Pruebas unitarias en DbHelper y lógica de negocio, pruebas instrumentadas de interfaz en Android (Espresso), y validación final del sistema.

## 1.4 Revisiones e inspecciones de código

\*Al ser un proyecto desarrollado por un solo integrante, se realizaron autorrevisiones periódicas del código en lugar de revisiones por pares.

\*Se emplearon las herramientas de Android Studio, como Lint y el Inspector de Código,también se utilizo Chatgpt para identificar problemas de estilo, posibles fugas de memoria y malas prácticas.

\*Los criterios de aceptación definidos fueron:

1.Código comentado y documentado con Javadoc.

2.Ejecución correcta de pruebas manuales sobre las funciones implementadas.

\*Cumplimiento de los estándares de codificación establecidos (nombres, estructura de paquetes, validaciones en formularios, etc.).

\*Los hallazgos detectados (advertencias del IDE o errores en pruebas) se corrigieron antes de avanzar a la siguiente fase, asegurando una mejora continua del proyecto.