Especificación de Diseño de Software (SDS)

# 1. Introducción

**Objetivo:**

El propósito de este documento es describir la arquitectura y diseño del Sistema de Encomiendas, definiendo los componentes, interacciones y responsabilidades funcionales que permitirán su implementación. El sistema busca digitalizar y optimizar los procesos de gestión de envíos, desde la recepción de solicitudes hasta la entrega final, garantizando trazabilidad, seguridad y eficiencia en la comunicación entre usuarios, conductores, gestores, funcionarios y sistemas externos.

**Alcance:**

El diseño incluye los procesos de:

* Registro de clientes y solicitudes de envío.
* Generación y envío de guías.
* Gestión del reparto de camiones y asignación de conductores.
* Actualización y consulta del estado de los envíos.
* Confirmación de entrega.
* Generación de reportes y análisis de datos.
* Integración con sistemas externos (ERP Institucional y Pasarela de Pagos).

Quedan fuera del alcance:

* La gestión física del transporte y logística en carretera.
* La infraestructura de hardware para operaciones de campo (ejemplo: dispositivos GPS, sensores de camión).
* Procesos administrativos internos del ERP que no estén relacionados con envíos.

**Definiciones y Abreviaturas:**  
Sistema de Encomiendas: Software que administra el ciclo de vida de los envíos.

* ERP: Enterprise Resource Planning, sistema empresarial utilizado para procesos administrativos.
* Pasarela de Pagos: Servicio externo encargado de procesar transacciones electrónicas.
* Guía: Documento digital que respalda y certifica el registro de un envío.
* Usuario: Cliente que solicita un servicio de encomienda.
* Destinatario: Persona que recibe un envío.
* Conductor: Responsable de transportar y actualizar el estado de los envíos.
* Gestor: Encargado de la planificación logística de camiones y rutas.
* Funcionario: Responsable de generar y validar documentos del envío.
* Analista: Usuario encargado de consultar estadísticas y reportes.

# 2. Generalidades del Proyecto

Problema a Resolver:  
 Actualmente, la gestión de encomiendas en la institución presenta dificultades debido al manejo manual de procesos como el registro de envíos, la generación de guías y la actualización del estado de los pedidos. Esto genera demoras, errores de trazabilidad, falta de comunicación en tiempo real y poca capacidad de análisis para la toma de decisiones.

El software busca resolver estos problemas mediante un sistema digital centralizado que integre clientes, conductores, gestores, funcionarios y sistemas externos.

1. Principales Retos de Diseño:  
   Diseñar una arquitectura escalable que soporte el crecimiento en la cantidad de solicitudes de envío.
2. Garantizar seguridad en las transacciones electrónicas y en la comunicación entre actores.
3. Definir un modelo de datos flexible que permita trazabilidad completa de los envíos.
4. Integrar de manera eficiente con sistemas externos (ERP y pasarela de pagos).
5. Mantener la experiencia de usuario simple y fluida, tanto en aplicaciones móviles como web.
6. Permitir operación en tiempo real, con sincronización confiable entre dispositivos y servidores.

Análisis del Diseño Conceptual Inicial (Estrategia):  
 Se plantea una **arquitectura orientada a servicios (SOA)** basada en microservicios desplegados en la nube, con un modelo Cliente-Servidor.

* Los clientes (usuarios, gestores, conductores y funcionarios) accederán a través de aplicaciones móviles y web.
* Los servicios backend estarán organizados por dominios: gestión de usuarios, gestión de envíos, logística, pagos y reportes.
* La información se almacenará en una base de datos relacional para mantener integridad, complementada con mecanismos de cache para consultas rápidas.
* La integración con sistemas externos se hará mediante **APIs RESTful seguras (OAuth 2.0, JWT)**.

Estándares de Diseño:  
**Lenguaje y framework backend:** Java con JAX-RS y/o Spring Boot.

**Frontend web:** React con Material UI.

**Móvil:** Android con Kotlin.

**Bases de datos:** PostgreSQL (principal) + Redis (cache).

**Normas de calidad:**

* Principios SOLID.
* Convenciones de nombrado UML para diagramas.
* Patrones de diseño: MVC, DAO, Observer, Singleton (donde aplique).
* Estándares de seguridad OWASP.

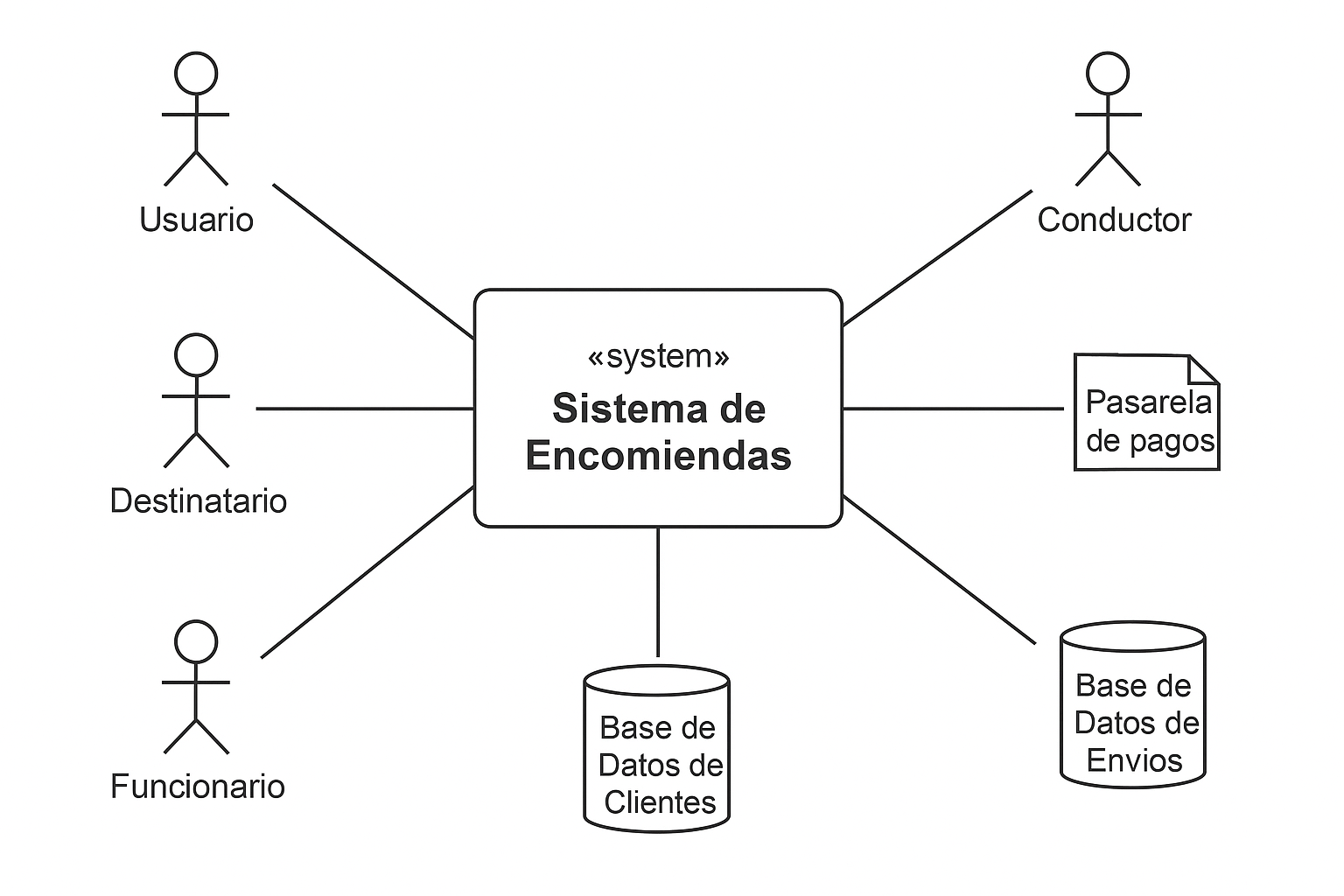
# 3. Arquitectura de Software

Estilo de Arquitectura:  
 **Cliente-Servidor con enfoque en Microservicios**, apoyado en MVC para la capa de presentación.

## Vistas de Arquitectura

Vista de Contexto:

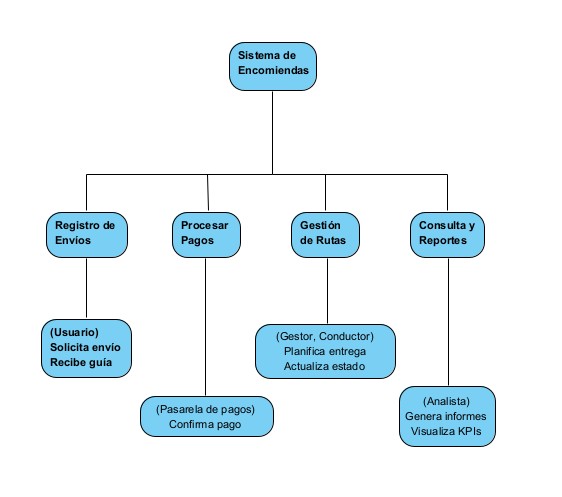
* **Entradas:** solicitudes de envío, registros de clientes, pagos, actualizaciones de estado.
* **Salidas:** guías digitales, notificaciones de estado, reportes, confirmaciones de pago.
* **Interacciones externas:** ERP institucional, pasarela de pagos.
* **Usuarios:** cliente, destinatario, conductor, gestor, funcionario, analista.



Vista Funcional:

**Módulo de Usuarios:** registro, autenticación, roles.

* **Módulo de Envíos:** creación de guías, asignación de conductores, actualización de estados.
* **Módulo de Logística:** planificación de rutas, gestión de camiones.
* **Módulo de Pagos:** integración con pasarela.
* **Módulo de Reportes:** análisis y estadísticas.



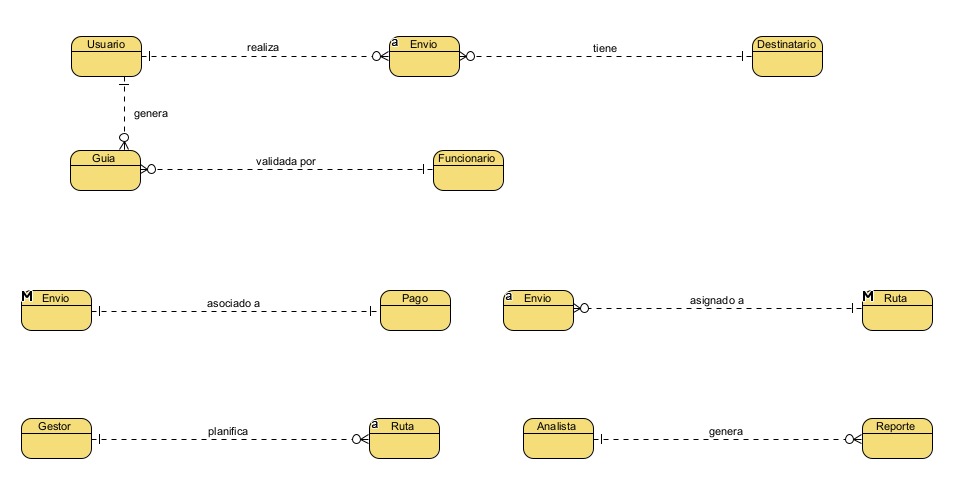
Vista de Información:

**Modelo estático:**

* Entidades principales: Usuario, Cliente, Conductor, Envío, Guía, Pago, Reporte.

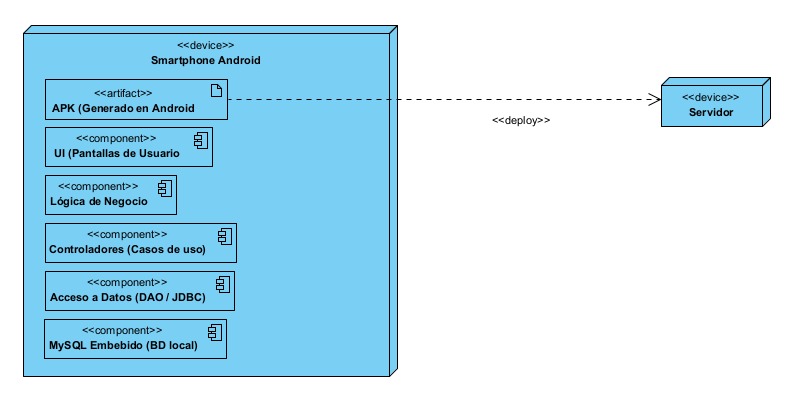
**Flujos de datos:**

* Cliente → Solicitud de envío → Generación de guía → Asignación de conductor → Actualización de estado → Entrega confirmada → Reporte.



Vista de Despliegue:

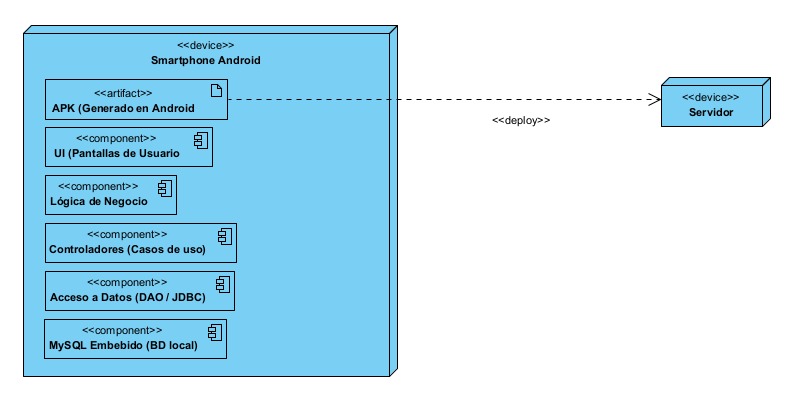
* **Servidor de Aplicaciones:** despliegue en contenedores Docker orquestados con Kubernetes.
* **Base de Datos:** PostgreSQL en servidor dedicado.
* **Clientes:** aplicaciones móviles (Android), aplicaciones web (React).
* **Integraciones externas:** ERP (SOAP/REST), Pasarela de pagos (REST).



Vista de Desarrollo:

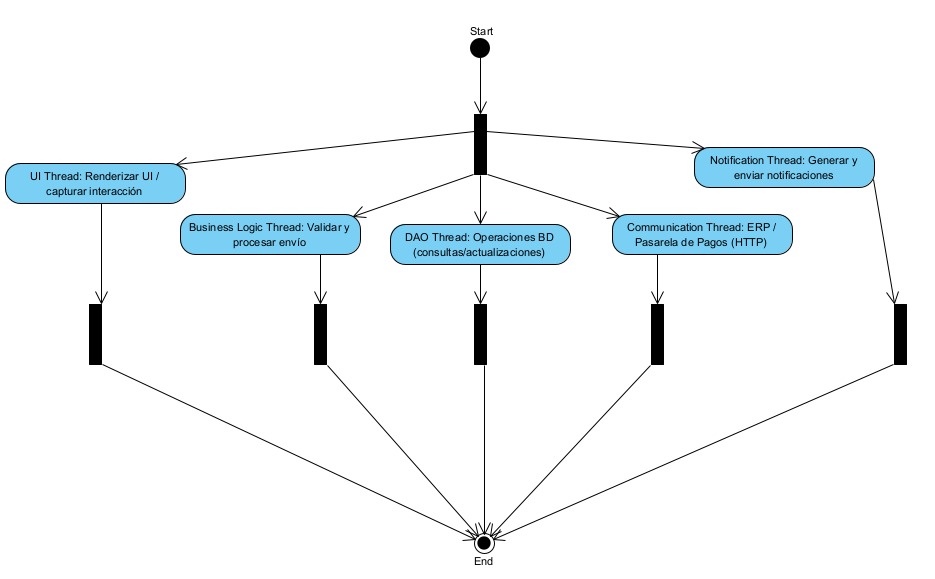
* Orden de construcción:

1. Base de datos y modelo de datos.
2. Backend (APIs REST).
3. Frontend Web y Móvil.
4. Integraciones externas.
5. Reportes y analítica.



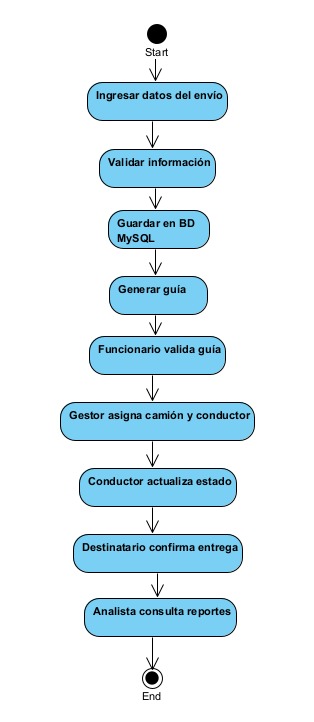
Vista de Concurrencia:

* Varias solicitudes de clientes pueden procesarse en paralelo mediante hilos y balanceadores de carga.
* Los conductores pueden actualizar estados de manera simultánea sin bloqueos, gracias a transacciones concurrentes en la base de datos.



Vista Operacional:

* Respaldos automáticos de base de datos.
* Monitoreo de logs y métricas con herramientas como Prometheus + Grafana.
* Actualizaciones continuas con CI/CD (Jenkins, GitHub Actions).



Decisiones de Arquitectura (Ciclo 29):

* Uso de **APIs REST** para comunicación entre módulos.
* Adopción de **microservicios** para escalabilidad.
* Base de datos **relacional** para consistencia, complementada con cache en memoria.
* Autenticación centralizada con **JWT/OAuth 2.0**.
* Despliegue en contenedores Docker con Kubernetes para escalabilidad y portabilidad.

# 4. Diseño Detallado de Software

Modelo Detallado de Clases:

**Clase Usuario**

* Atributos: id, nombre, correo, contraseña, rol.
* Métodos: registrar(), autenticar(), actualizarPerfil().

**Clase Envío**

* Atributos: id, origen, destino, fechaRegistro, estado, clienteId, conductorId.
* Métodos: crearGuía(), asignarConductor(), actualizarEstado(), confirmarEntrega().

**Clase Pago**

* Atributos: id, envíoId, monto, fecha, estado, pasarelaId.
* Métodos: procesarPago(), verificarEstado().

Modelo de Interfaces:

**Pantallas:**

* Registro/Login de usuario.
* Panel de cliente (crear solicitud, consultar estado).
* Panel de conductor (ver asignaciones, actualizar estado).
* Panel de funcionario (validar y generar guías).
* Panel de gestor (planificación logística).
* Panel de analista (reportes).

**APIs (ejemplos REST):**

* POST /usuarios (crear usuario).
* POST /envios (crear envío).
* PUT /envios/{id}/estado (actualizar estado).
* POST /pagos (procesar pago).
* GET /reportes (consultar estadísticas).

Modelo de Datos (EJB o equivalente):

**Entidades:**

* Cliente (idCliente, nombre, correo, teléfono).
* Conductor (idConductor, nombre, licencia, camión).
* Envío (idEnvio, origen, destino, estado, fechaRegistro, idCliente, idConductor).
* Pago (idPago, idEnvio, monto, estado, fecha).
* Guía (idGuia, idEnvio, fecha, códigoQR).

**Relaciones:**

* Cliente 1..\* → Envío.
* Envío 1..1 → Guía.
* Envío 1..1 → Pago.
* Conductor 1..\* → Envío.