Diseño de Arquitectura - Proyecto LogiPy

Resumen: Este documento presenta el diseño de arquitectura del prototipo LogiPy — un cotizador de envíos inteligente desarrollado en Python. Incluye el *diagrama de clases* que refleja la estructura final del código entregado (clases principales, atributos, métodos y relaciones), el esquema de la base de datos y decisiones arquitectónicas relevantes.

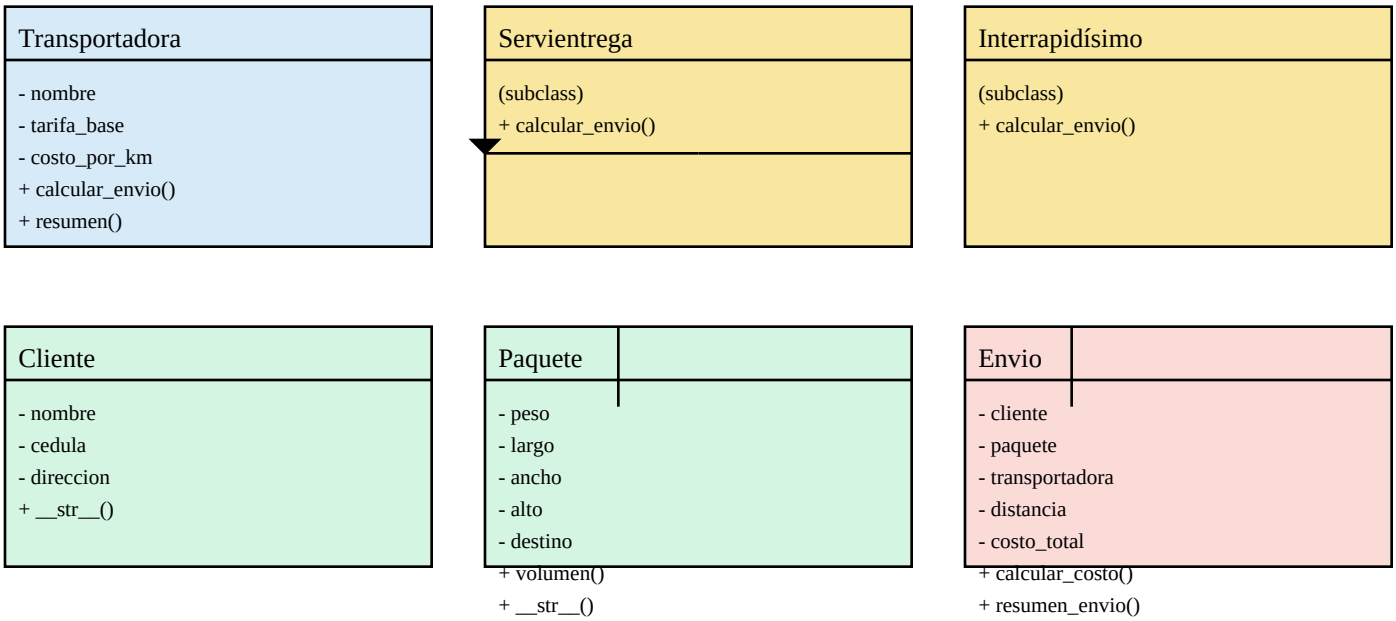
Alcance y objetivos

El propósito de este documento es: Documentar las clases principales del sistema LogiPy y sus relaciones. Proveer un diagrama que sirva como referencia para desarrollo y mantenimiento. Explicar decisiones de diseño (POO, persistencia con SQLite, extensibilidad).

Resumen de clases

Clase	Responsabilidad principal	Atributos clave	Métodos clave
Cliente	Representar usuario que realiza envíos	nombre, cedula, direccion	__str__()
Paquete	Información física del paquete	peso, largo, ancho, alto, destino	volumen(), __str__()
Transportadora	Modelo base para compañías de envío	nombre, tarifa_base, costo_por_km	calcular_envio(), resumen()
Servientrega	Regla de cálculo específica	inherit from Transportadora	calcular_envio() (descuento)
Interrapidisimo	Regla de cálculo específica	inherit from Transportadora	calcular_envio() (recargo)
Envio	Agrega cliente + paquete + transportadora para calcular costo total	cliente, paquete, transportadora, distancia, costo	calcular_costo(), resumen_envio()

Diagrama de Clases (UML - vista simplificada)



Descripción detallada de las clases

Transportadora: Clase abstracta conceptual que contiene los parámetros tarifarios (como tarifa_base y costo_por_km). Su método *calcular_envio()* define la fórmula general, que las subclases pueden extender (polimorfismo).

Servientrega y **Interrapidisimo**: Subclases que implementan reglas propias: por ejemplo, descuento por peso liviano o recargo por rapidez. Esto demuestra polimorfismo.

Cliente: Entidad ligera que representa al remitente; se usa para generar el resumen del envío.

Paquete: Contiene propiedades físicas (peso, dimensiones) y método para calcular volumen, útil para cálculos avanzados (tarifa volumétrica).

Envio: Agrega y coordina las demás clases (composición). Al invocar *calcular_costo()* delegará el cálculo a la transportadora seleccionada y almacenará el resultado.

Esquema de Base de Datos (SQLite)

Se utilizó SQLite para persistencia ligera. Tabla principal en prototipo: Transportadora(id INTEGER PRIMARY KEY AUTOINCREMENT, nombre TEXT NOT NULL, tarifa_base REAL NOT NULL, costo_por_km REAL NOT NULL, recargo_peso_por_kg REAL NULL) En versiones futuras se plantean tablas adicionales: Cliente, Paquete, Envio, TarifasHistoricas.

Mapeo código ↔ diseño

Cada caja del diagrama se corresponde directamente con una clase en el código: - La clase *Transportadora* define atributos y métodos base. - *Servientrega* y *Interrapidisimo* sobrescriben *calcular_envio()*. - *Envio* compone Cliente y

Paquete y usa la instancia de Transportadora para delegar el cálculo. El bloque de pruebas (if __name__ == '__main__') crea instancias y demuestra la funcionalidad, sirviendo como conjunto de pruebas manuales para el hito 1.

Decisiones de diseño y extensibilidad

- **POO**: Se aplica para poder ampliar fácilmente con nuevas transportadoras y reglas tarifarias. - **Persistencia con SQLite**: Sencilla, no requiere servidor y suficiente para prototipos. - **Separación de responsabilidades**: Cliente, Paquete y Envio tienen responsabilidades claras. - **Extensiones futuras**: Agregar tarifas volumétricas, seguros, tiempos de entrega, y una interfaz web.

Escenario de uso (flujo)

1. El usuario registra transportadoras en la base de datos (o se cargan por defecto). 2. Se crea un Cliente y un Paquete con sus datos. 3. Se crea un Envio vinculando Cliente, Paquete y la Transportadora elegida. 4. Se invoca *calcular_costo()*, que delega a la transportadora y devuelve el costo. 5. El sistema muestra el resumen y persiste (en versiones futuras) el registro del envío.

Documento elaborado por: Grupo LogiPy. Fecha: (inserte fecha de entrega).

Apéndice: Notación UML (texto)

```
Transportadora
- nombre: str
- tarifa_base: float
- costo_por_km: float
+ calcular_envio(distancia, peso): float
+ resumen(): str

Servientrega : Transportadora
+ calcular_envio(distancia, peso): float

Interrapidisimo : Transportadora
+ calcular_envio(distancia, peso): float

Cliente
- nombre: str
- cedula: str
- direccion: str
+ __str__(): str

Paquete
- peso: float
- largo: float
- ancho: float
- alto: float
- destino: str
+ volumen(): float

Envio
- cliente: Cliente
- paquete: Paquete
- transportadora: Transportadora
- distancia: float
- costo_total: float
+ calcular_costo(): float
+ resumen_envio(): str
```