



INFO 1126 - Programación 3

Evaluación 3 (25%)

Proyecto: Sistema Logístico Autónomo con Drones Fase 2

Junio 2025

Problemática (Fase 2: Optimización y Visualización Geoespacial)

Correos Chile ha iniciado una segunda fase en la implementación de su red de drones autónomos. Tras superar la primera etapa de planificación y simulación básica, se requiere ahora una **visualización geoespacial realista**, una **API de integración externa** y herramientas para **toma de decisiones basadas en datos históricos**.

Esta fase busca que el sistema contemple:

- **Visualización sobre mapa real:** Visualizar todos los nodos y rutas directamente sobre una vista georreferenciada real (como el mapa de Temuco), incluyendo recorridos activos, árboles de expansión mínima y frecuencias.
- **Visualización de árboles de expansión mínima (Kruskal):** Para analizar cobertura mínima necesaria entre puntos.
- **Optimización de rutas:** Permitir calcular trayectos usando algoritmos eficientes como Dijkstra y Floyd-Warshall.
- **Estadísticas visuales y en PDF:** Los usuarios deben poder visualizar y exportar un informe en PDF con resumen de rutas, nodos, distancias y uso energético.
- **API RESTful:** Exponer la simulación para ser consultada o utilizada por servicios externos.



Objetivo

Diseñar, extender e integrar un sistema logístico de drones autónomos, que permita:




1. **Visualización de la red completa sobre un mapa interactivo real.**
2. **Selección de rutas entre nodos (almacenamiento a clientes) con control de autonomía.**
3. **Cálculo de rutas usando Dijkstra o Floyd-Warshall**, con representación gráfica y resumen energético.
4. **Visualización del Árbol de Expansión Mínima (MST)** usando Kruskal directamente en el mapa.
5. **Registro de órdenes** generadas desde rutas simuladas y visualización de su historial.
6. **Generación de informes PDF**, tanto desde la aplicación visual como desde una API externa.
7. **Consumo y prueba de una API RESTful** que sirva como backend para los datos de la simulación.

Para esto se ha dispuesto un video en youtube a modo de guía

https://www.youtube.com/watch?v=9LTdBsKqE_w



Parámetros de Simulación

- Cantidad máxima de nodos: hasta 150
- Roles de nodos:
 -  Almacenamiento: 20%
 -  Recarga: 20%
 -  Cliente: 60%
- Autonomía máxima del dron: 50 unidades de costo (suma de pesos de aristas)
- Si una ruta supera el límite de batería, se fuerza el paso por nodos de recarga.
- Todas las rutas se generan sobre grafos conectados.

Metas Funcionales Fase 2

1. Gestión avanzada de rutas

- Permitir la creación dinámica de rutas entre cualquier nodo de almacenamiento y cliente.
- Forzar la visita a estaciones de recarga cuando la ruta exceda la autonomía del dron (50 unidades de costo).
- Implementar algoritmos de búsqueda óptima:
 - **Dijkstra** para caminos más cortos con pesos reales o **Floyd-Warshall** para caminos mínimos entre todos los pares.
 - Visualización del Árbol de Expansión Mínima (MST) usando **Kruskal**.

2. Simulación interactiva

- Simulación inicial mínima: 15 nodos, 20 aristas, 10 órdenes.
- Escalabilidad soportada: hasta 150 nodos, 300 aristas, y 500 órdenes simultáneas.

Uso eficiente de estructuras de datos como:

- **AVL Tree**: para registro de rutas.
- **HashMaps**: para búsqueda rápida de clientes y pedidos.
- **Graph**: para modelar la red de transporte.



3. Análisis estadístico

- Registrar cada ruta usada en un AVL.
- Determinar las rutas más utilizadas.
- Registrar frecuencia de nodos destino y origen.

4. Garantía de conectividad

- Todos los grafos generados son conexos.
- Se evita la generación de nodos aislados.

5. Visualización geográfica

- La vista permite:
 - Calcular rutas.
 - Ver costos y distancia.
 - Completar órdenes.
 - Mostrar MST generado.
 - Visualizar resumen de vuelo.

6. Generación de informes PDF

- Desde la pestaña de “Route Analytics” y vía API, se puede generar un informe PDF descargable que contiene:
 - Tabla de pedidos.
 - Clientes con más pedidos.
 - Rutas más usadas.
 - Gráficos con distribución y uso del sistema.

7. API REST Interactiva

- Exponer la información clave del sistema a través de una API desarrollada con FastAPI, permitiendo:
 - Obtener estadísticas del sistema en formato JSON.
 - Generar y descargar informes PDF.
 - Consultar rutas, clientes, pedidos, nodos y uso de red.




Pestaña 1: Run Simulation

Propósito:

Permitir la configuración e inicio de la simulación con parámetros personalizables.




Componentes:

- **Slider**: número de nodos (**n_nodes**, entre 10 y 150).
- **Slider**: número de aristas (**m_edges**, entre 10 y 300).
- **Slider**: número de órdenes (**n_orders**, entre 10 y 300).
- **Campo informativo**: Texto informativo indicando la cantidad de Nodos cliente, almacenamiento, abastecimiento y su porcentaje.
- **Botón**:  **Start Simulation (Inicia la simulación)**

Validaciones:

- El número de aristas debe permitir un grafo conexo ($\geq n_nodes - 1$)
- El valor de nodos no debe superar 150.
- Si se hace clic sin modificar parámetros, se mantiene la configuración por defecto(ítem de componentes).

Interacciones esperadas:

- Al presionar el botón, se genera un grafo aleatorio con roles asignados proporcionalmente:
 -  20% almacenamiento
 -  20% recarga
 -  60% clientes

* Nota del docente:







Esta pestaña sirve como punto de entrada para validar la generación del entorno base y comprobar la conectividad y escalabilidad. (Manejo de parámetros)



Pestaña 2: Explore Network

Propósito:

Ofrecer una visualización geográfica real del grafo de transporte usando folium (Recomendado). Permitir el cálculo de rutas y visualización de MST en el mapa.

-  **Componentes:**
- **Mapa real (folium) con:**
 - **Nodos geolocalizados.**
 - **Aristas con pesos.**
 - **Ruta resaltada en rojo.**
- **Selectbox: Nodo origen (solo  Almacenamiento).**
- **Selectbox: Nodo destino (solo  Cliente).**
- **Radio: Algoritmo a utilizar (Dijkstra, Floyd-Warshall).**
- **Botón:  Calculate Route**
- **Botón:  Complete Delivery and Create Order**
- **Botón:  Show MST (Kruskal)**
- **Cuadro informativo: resumen de vuelo (distancia, ruta).**

Validaciones:

- La ruta debe considerar batería máxima (50) y, si es necesario, estaciones de recarga permitiendo un camino general superior a 50, pero, un uso de batería menor al utilizar estaciones de recarga.
- Solo debe permitirse crear rutas de Almacenamiento → Cliente.

Interacciones esperadas:

- Al calcular una ruta, esta se dibuja sobre el mapa con puntos intermedios y se muestra el costo.
- Al presionar "Show MST", se visualiza el Árbol de Expansión Mínima (MST) sobre el mapa en líneas discontinuas.
- El resumen de vuelo se genera dinámicamente.
- Todas las rutas y entregas quedan registradas.

* Nota docente:

Evaluar la integración con folium, el trazado correcto de rutas y MST, y la visualización geográfica precisa. Esta pestaña representa la evolución visual y algorítmica del sistema.



Pestaña 3: Clients & Orders

Propósito:

Listar los clientes activos y los pedidos generados, mostrando atributos relevantes.

Componentes:

- **Subsección:** lista de **clientes** (**st.json**)
 - Cliente: ID, nombre, tipo, total de pedidos
- **Subsección:** lista de **órdenes** (**st.json**)
 - Orden: ID, cliente asociado, cliente ID, origen, destino, status, fecha de creación, prioridad, fecha entrega, costo total.

Validaciones:

- Se muestran sólo si hay simulación activa.

Interacciones esperadas:

- Al iniciar una simulación, esta sección se autoactualiza con los datos generados.

* Nota docente:

Es útil para verificar que el sistema esté registrando correctamente los pedidos y asociándose a clientes.




Pestaña 4: Route Analytics

Propósito:

Visualizar las rutas más frecuentes utilizadas, registradas en una estructura **AVL**.

Incluir opción de exportar informe **PDF** con información de uso de rutas.

Componentes:

- **Lista:** rutas más frecuentes (clave = camino, valor = frecuencia).
- **Gráfico:** visualización del árbol AVL con etiquetas "**A → B → C\nFreq: X**" usando **networkx**.
- Botón:  Generar Informe PDF
 - Descarga un reporte con la siguiente información:
 - Rutas frecuentes.
 - Clientes más recurrentes.
 - Nodos más utilizados.
 - Gráficas de torta y barras.

Validaciones:

- Solo disponible si se han generado órdenes y rutas.
- El AVL debe reflejar el uso repetido de rutas idénticas (por ejemplo, múltiples pedidos de un mismo cliente).
- El botón genera un PDF con resumen completo del sistema.

Interacciones esperadas:

- Las rutas se ordenan por recorrido (orden lexicográfico).
- Se visualiza gráficamente la estructura del AVL para apoyar aprendizaje en estructuras balanceadas.

*Nota docente:

Parte esencial de la evaluación por uso de TDA AVL. Refuerza estructuras y visualización de árboles.



Pestaña 5: General Statistics

Propósito:

Entregar una vista global del sistema en funcionamiento, con gráficas visuales.

Componentes:

- **Gráfico de barras:** comparación entre número de nodos clientes más visitados, nodos almacenamiento más visitados y nodos de abastecimiento más visitados.
- **Gráfico de torta:** proporción entre nodos por rol.



Validaciones:

- Se requiere simulación activa para mostrar los datos.

Interacciones esperadas:

- Se actualiza automáticamente luego de iniciar una simulación.



API Endpoints

1. GET /clients/

Propósito:

Obtener la lista completa de clientes registrados en el sistema.

Respuesta esperada:

Listado JSON con todos los clientes.

2. GET /clients/{client_id}

Propósito:

Obtener la información detallada de un cliente específico por su ID.

Parámetros:

- `client_id` (path): Identificador único del cliente.

Respuesta esperada:

Datos JSON del cliente solicitado.

3. GET /orders/

Propósito:

Listar todas las órdenes registradas en el sistema.

Respuesta esperada:

Listado JSON con todas las órdenes.



4. GET /orders/orders/{order_id}

Propósito:

Obtener detalle de una orden específica por su ID.

Parámetros:

- **order_id** (path): Identificador único de la orden.

Respuesta esperada:

Datos JSON con detalle de la orden solicitada.

5. POST /orders/orders/{order_id}/cancel

Propósito:

Cancelar una orden específica.

Parámetros:

- **order_id** (path): ID de la orden a cancelar.

Validaciones:

La orden debe estar en estado cancelable.

Respuesta esperada:

Confirmación de cancelación o error si no se puede cancelar.

6. POST /orders/orders/{order_id}/complete

Propósito:

Marcar una orden específica como completada.

Parámetros:

order_id (path): ID de la orden a completar.



Validaciones:

La orden debe estar en un estado que permita completar.

Respuesta esperada:

Confirmación de estado completado o error.

7. GET /reports/reports/pdf

Propósito:

Generar y obtener el informe PDF resumen del sistema y las órdenes.

Respuesta esperada:

Archivo PDF descargable con estadísticas e información relevante.

8. GET /info/reports/visits/clients

Propósito:

Obtener el ranking de clientes más visitados en las rutas de la simulación.

Respuesta esperada:

Listado JSON ordenado de clientes según visitas.

9. GET /info/reports/visits/recharges

Propósito:

Obtener el ranking de nodos de recarga más visitados.

Respuesta esperada:

Listado JSON con nodos de recarga ordenados por visitas.



10. GET /info/reports/visits/storages

Propósito:

Obtener el ranking de nodos de almacenamiento más visitados.

Respuesta esperada:

Listado JSON con nodos de almacenamiento ordenados por visitas.

11. GET /info/reports/summary

Propósito:

Obtener un resumen general de la simulación activa.

Respuesta esperada:

JSON con estadísticas globales y datos relevantes de la simulación.



Estructura sugerida de clases y módulos

Clase / Módulo	Función	Ubicación
Graph, Vertex, Edge	Modela el grafo base (nodos, conexiones)	model/
Simulation	Controlador principal de la simulación	sim/
SimulationInitializer	Generación de grafos conectados y roles	sim/
Route, Order, Client	Representan las entidades del sistema	domain/
AVL	Almacena rutas más frecuentes	tda/
Map (hash map propio)	Acceso O(1) a clientes y órdenes	tda/
NetworkXAdapter	Adaptador visual de grafos para matplotlib	visual/
AVLVisualizer	Dibuja gráficamente el árbol AVL	visual/
dashboard.py	Interfaz principal (pestañas Streamlit)	visual/
map_builder	Construcción del mapa real (folium)	visual/map
flight_summary	Calcula distancia, batería y tiempo estimado del recorrido.	visual/map
report_generator	Genera informe PDF con rutas, estadísticas y gráficas.	visual/
api/main	Punto de entrada de la API (FastAPI).	api/
controllers/*	Rutas API RESTful: pedidos, clientes, reportes, estado del sistema.	api/controllers/*



EI

+	+	+
	Interfaz Visual	← GUI / Dashboard (Streamlit)
+	+	+
	API REST	← Exposición de datos y reportes
+	+	+
	Lógica de Aplicación	← Coordina simulaciones, rutas, análisis
+	+	+
	Dominio del Problema	← Modelos: Pedido, Cliente, Ruta
+	+	+
	Infraestructura TDA	← AVL, Diccionario (HashMap)
+	+	+
	Estructuras Base	← Graph, Vertex, Edge, Inicializador
+	+	+

proyecto debe enfocarse en una **ubicación jerárquica, dependencia lógica y modularidad**, como si construyéramos un **diagrama de arquitectura lógica por capas (layered architecture)**.

```

proyecto/
|
├── domain/          # ← Entidades del dominio (negocio)
|   ├── client.py    # ← Clase Client
|   ├── order.py     # ← Clase Order
|   └── route.py     # ← Clase Route
|
├── model/           # ← TDA del grafo base
|   ├── graph.py     # ← Clase Graph
|   ├── vertex.py    # ← Clase Vertex
|   ├── edge.py      # ← Clase Edge
|   └── __init__.py  # ← Inicializador de módulo
|
├── sim/             # ← Lógica de simulación y orquestación
|   ├── init_simulation.py # ← Generador de grafos conectados
|   └── simulation.py  # ← Clase Simulation (núcleo principal)
|
├── tda/             # ← Tipos de Datos Abstractos personalizados
|   ├── avl.py       # ← AVL funcional con key-value
|   └── hasp_map.py   # ← Hash map personalizado (tipo dict)
|
├── visual/          # ← Visualización Streamlit y adaptadores gráficos
|   ├── dashboard.py # ← Interfaz Streamlit con pestañas
|   ├── networkx_adapter.py # ← Adaptador de grafo → matplotlib
|   ├── avl_visualizer.py # ← Visualización de árbol AVL como gráfico
|   └── map/          # ← Visualización geográfica en mapa
|       ├── map_builder.py # ← Mapa base con rutas y nodos
|       └── flight_summary.py # ← Cálculo resumen de trayectos
|
└── api/             # ← API RESTful del sistema
    ├── main.py      # ← Servidor principal con FastAPI
    ├── controllers/ # ← Endpoints de acceso a datos
    │   ├── client_routes.py
    │   ├── order_routes.py
    │   ├── report_routes.py
    │   └── info_routes.py

```



Criterios de Evaluación	Descripción	Excelente (10 pts)	Bueno (8 pts)	Suficiente (4 pts)	Insuficiente (1 pt)	Peso (%)
API Restful funcional con endpoints solicitados.	API Restful completa con cada uno de los endpoints solicitados.	Funciona perfectamente sin errores	Funciona con pequeños errores	Funciona parcialmente con fallos evidentes	No funciona o tiene errores críticos	25%
Generación del informe PDF.	Generación del informe PDF con todos los puntos solicitados tanto en la API como en Dashboard.	El PDF se puede obtener desde ambas rutas con los puntos solicitados.	El PDF se puede obtener desde ambas rutas con algunos puntos solicitados.	El PDF se puede obtener sólo de una ruta.	El PDF no se puede obtener.	10%
Dashboard modular con navegación clara	Interfaz organizada en pestañas/secciones intuitivas	Navegación excelente con diseño profesional	Navegación clara con pequeños detalles	Navegación funcional pero mejorable	Interfaz confusa o difícil de usar	15%
Nuevo mapa interactivo.	Planificación considera límite de batería y puntos de recarga	Rutas dentro de autonomía	Rutas generalmente válidas (ocasionales fallos)	Rutas a veces exceden autonomía	Rutas no consideran autonomía	15%
Presentación Oral	Claridad y organización en la presentación, descripción de la problemática, diagramas, solución,	Excelente presentación oral; describe la problemática y solución de manera clara,	Buena presentación describe la problemática y solución de manera clara, aunque con detalles menores;	Presentación suficiente pero con falta de claridad y organización; la descripción de la problemática y solución es incompleta;	Presentación pobremente organizada, falta de claridad y omisión de varios elementos clave; manejo ineficiente de	25%



	demostración de código y manejo de preguntas.	responde adecuadamente las preguntas y cumple con el tiempo de 15 minutos.	responde adecuadamente las preguntas; Excede el tiempo definido.	manejo básico de preguntas ; Excede el tiempo definido.	preguntas; Excede el tiempo definido.	
Código modular y documentado	Organización del código y calidad de documentación	Código ejemplar con documentación completa	Código organizado y correctamente documentado	Código aceptable con documentación mínima	Código desorganizado sin documentar	5%
Repositorio GitHub.	Evaluación del correcto uso de GitHub como repositorio del proyecto, considerando estructura, historial y organización.	Repositorio completo, bien estructurado, con historial de commits significativo y documentación adecuada.	Repositorio funcional con estructura clara y commits que evidencian el trabajo progresivo.	Repositorio con estructura aceptable, aunque con escasa evidencia del proceso de trabajo.	Repositorio incompleto, desorganizado o inexistente.	5%



¿Qué debo entregar?

Debes preparar una presentación **PPT en grupos de 3-5** a presentar en la **semana 16 (30-06-2025 a 04-07-2025)**.

la cual debe contener:

- 1) Presentación del grupo y tema (título del problema, nombre de los integrantes, fecha y logo del departamento).
- 2) Presentación de la solución en código explicando el flujo principal de la solución.
- 3) Repositorio en Github con avance demostrable.
- 4) La presentación de la interfaz visual del problema.
- 5) La demostración de la funcionalidad total del programa.

La presentación tiene un tiempo máximo de 15 minutos y luego una tanda de preguntas de 5 minutos.

Fecha por confirmar en base al número de grupos a conformarse.