Samir Sayah Moammer Rodriguez 1106232
Estructura de Datos y Algoritmos II
Proyecto Final

Curso: IDS344 - Estructura de Datos y Algoritmos II
Nombre del Estudiante: Samir Sayah Moammer Rodriguez
Profesor: Jose Ramon Romero
Proyecto Final: Ruta Óptima del Viajero en Mapa Interactivo (**Optimap**)

Informe de Avance – Parte 1

El presente informe corresponde a la primera fase del desarrollo del proyecto final titulado OptiMap, una aplicación interactiva para calcular rutas óptimas entre ciudades utilizando estructuras de datos y algoritmos de optimización. En esta etapa se establecieron las bases del proyecto, incluyendo la planificación, preparación del entorno de trabajo, carga de datos iniciales y construcción de la estructura de grafo que será utilizada por los algoritmos en las siguientes fases.

Objetivo de esta Fase

Establecer la base técnica del proyecto mediante:

- Configuración completa del entorno de desarrollo.
- Creación de la estructura inicial de carpetas y archivos.
- Implementación de la clase **GradoCiudades** para cargar y construir el grafo a partir de un archivo JSON.
- Verificación del funcionamiento mediante ejecución y salida en consola.

Preparación del Entorno de Desarrollo

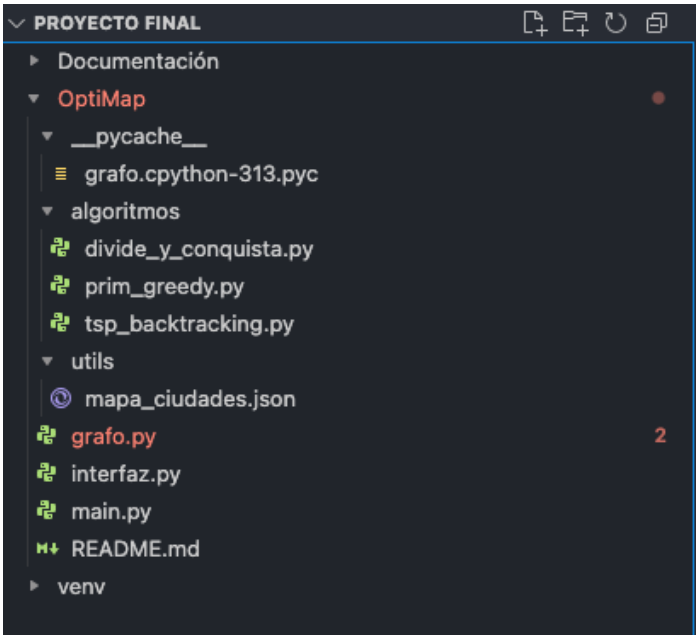
Se realizó la instalación y configuración de los siguientes elementos:

Componente	Acción Realizada
Python 3	Instalado por medio de la terminal.
Homebrew	Instalado como gestor de paquetes por la terminal.
Visual Studio Code	Instalado y configurado con extensión de Python desde el IDE.
Entorno Virtual	Creado con python3 -m venv venv
Librerías	Networkx, matplotlib, PyQt5

Samir Sayah Moammer Rodriguez 1106232
Estructura de Datos y Algoritmos II
Proyecto Final

Estructura del Proyecto

La estructura de carpetas y archivos creada fue la siguiente:



Datos de Entrada: Ciudades de República Dominicana

En vez de usar etiquetas genéricas (A, B, C...), usé ciudades reales de República Dominicana para dar un contexto más significativo al proyecto:

Ciudades usadas:

- Santo Domingo
- Santiago
- La Romana
- Puerto Plata
- San Cristóbal
- San Pedro de Macorís

Las distancias entre ellas fueron aproximadas basándose en rutas reales por carretera.

El archivo **mapa_ciudades.json** contiene las ciudades y sus conexiones con distancias como valores, estructurado para ser leído fácilmente por el programa.

Desarrollo Algorítmico

Samir Sayah Moammer Rodriguez 1106232
Estructura de Datos y Algoritmos II
Proyecto Final

Implementación del Grafo

Se implementó la clase GrafoCiudades en el archivo grafo.py la cual:

- Lee el archivo JSON de ciudades y distancias.
- Construye el grafo con networkx.Graph().
- Crea nodos y aristas ponderadas.

Se validó correctamente el funcionamiento al imprimir los nodos y aristas en consola desde main.py

Ejemplo de salida:

```
venvsamir@Samirs-MacBook-Pro optimap % python3 main.py
Ciudades cargadas:
['Santo Domingo', 'Santiago', 'La Romana', 'Puerto Plata', 'San Cristóbal', 'San Pedro de Macorís']
Aristas del grafo:
Santo Domingo <-> Santiago con distancia 155
Santo Domingo <-> La Romana con distancia 125
Santo Domingo <-> Puerto Plata con distancia 210
Santo Domingo <-> San Cristóbal con distancia 28
Santo Domingo <-> San Pedro de Macorís con distancia 71
Santiago <-> Puerto Plata con distancia 45
Santiago <-> La Romana con distancia 225
Santiago <-> San Pedro de Macorís con distancia 210
La Romana <-> San Pedro de Macorís con distancia 45
La Romana <-> San Cristóbal con distancia 145
Puerto Plata <-> San Cristóbal con distancia 185
San Cristóbal <-> San Pedro de Macorís con distancia 105
```

TSP Backtracking

Se implementó una solución exhaustiva para el problema del viajante, utilizando Backtracking para evaluar todas las permutaciones posibles desde una ciudad de origen. El sistema retorna la ruta más corta entre todas las posibles.

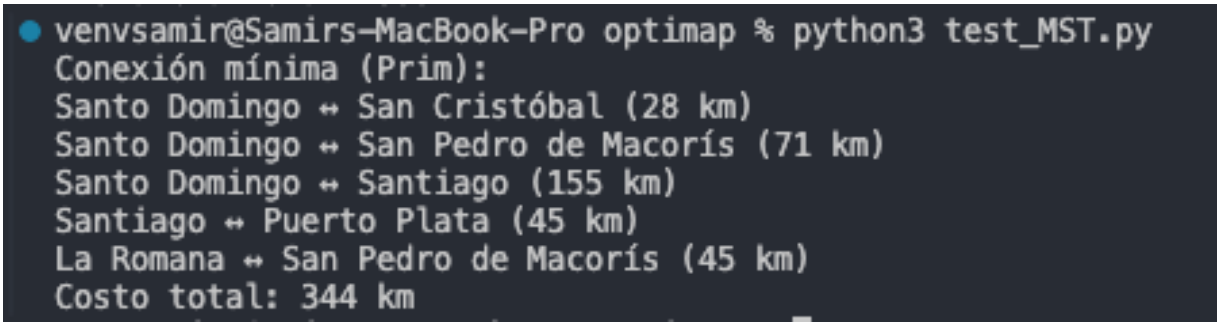
Ejemplo de Salida:

```
venvsamir@Samirs-MacBook-Pro optimap % python3 test.py
Ruta óptima (TSP): Santo Domingo → San Cristóbal → Puerto Plata → Santiago → La Romana → San Pedro de
 Macorís → Santo Domingo
Distancia total: 599 km
```

Árbol de Expansión Mínima (Prim – Greedy)

Se implementó un segundo algoritmo basado en la técnica Greedy para encontrar el conjunto mínimo de conexiones entre todas las ciudades, evitando ciclos.

Ejemplo de Salida:



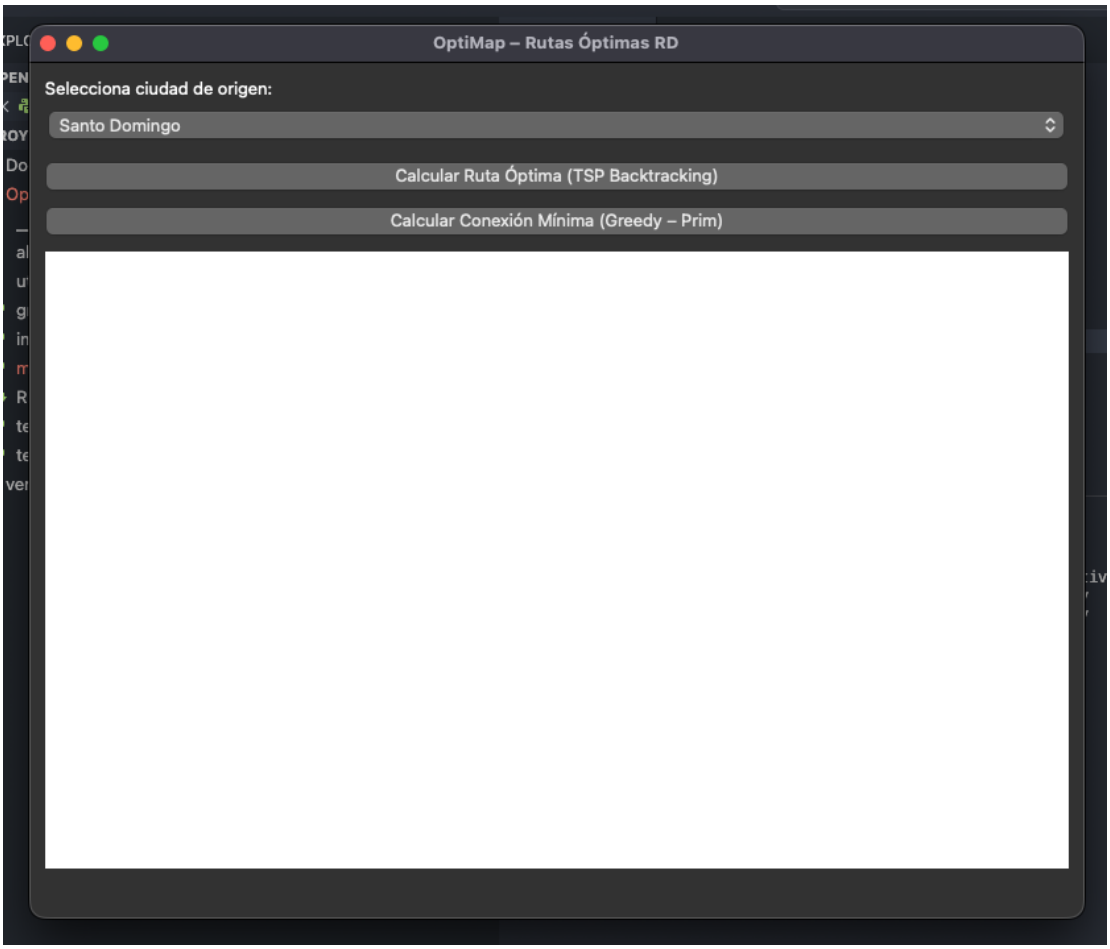
Interfaz Gráfica:

La aplicación incluye:

- Selector de ciudad de origen.
- Dos botones principales:
  - Calcular ruta óptima (TSP)
  - Calcular conexión mínima (Prim)
- Visualización del grafo completo.
- Rutas resaltadas según el algoritmo:
  - Rojo (TSP)
  - Verde (MST)
- Resultados mostrados como texto permanente al pie de la interfaz.

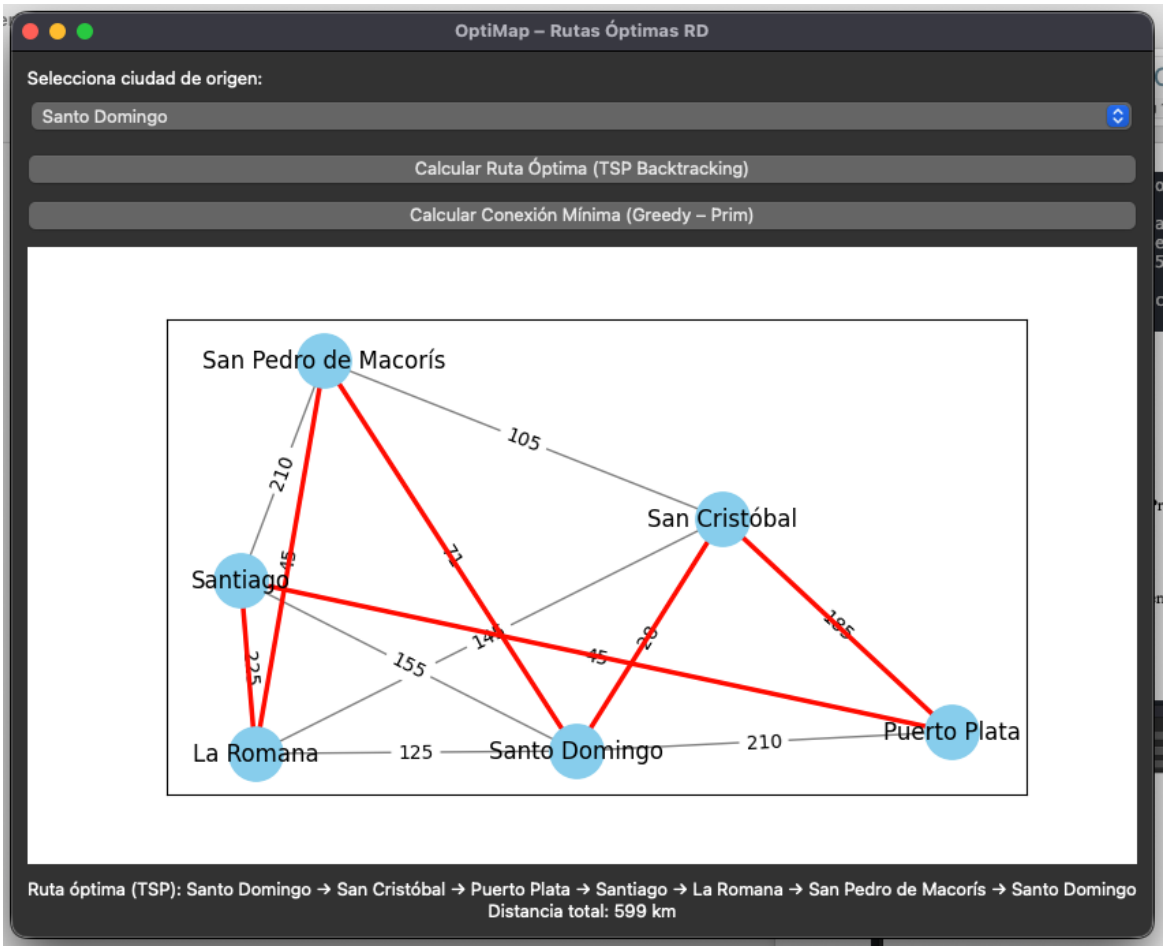
Ejemplo visual:

Captura al iniciar el programa:



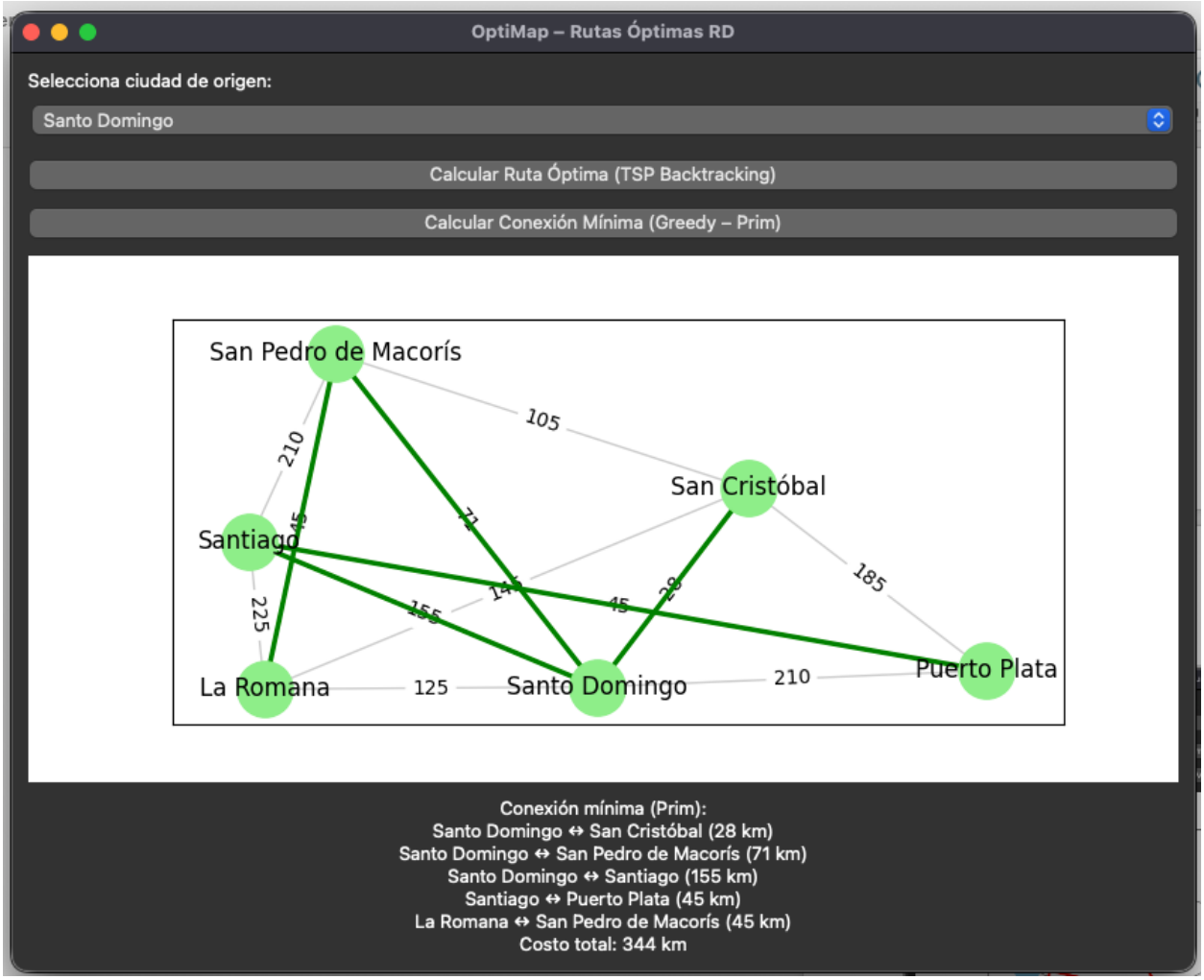
Samir Sayah Moammer Rodriguez 1106232
Estructura de Datos y Algoritmos II
Proyecto Final

Calcular la Ruta Optima:



Samir Sayah Moammer Rodriguez 1106232
Estructura de Datos y Algoritmos II
Proyecto Final

Calcular Conexión Minima:



Próximas mejoras:

- Permitir selección múltiple de destinos personalizados.
- Mejorar la Interfaz grafica, haciendola mas facil de identificar las rutas.
- Hacerlo mas interactivo.
- Expandir los grafos.