A red and black logo

AI-generated content may be incorrect.

**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**)

**Tabla de Contenido**

1. **Introducción**
2. **Objetivos**
3. **Tecnologías Utilizadas**
4. **Estructura del Proyecto**
5. **Desarrollo del Sistema**
   * Backend (Python + Flask)
   * Frontend (HTML5, CSS3, JS)
   * Comunicación entre componentes
   * Estructura de datos y algoritmos aplicados
6. **Explicación de los Algoritmos**
   * TSP (Backtracking y Greedy)
   * MST (Prim)
7. **Diseño de la Interfaz**
   * Justificación estética y funcional
   * Uso de SVG para el mapa interactivo
8. **Pruebas y Resultados**
9. **Conclusiones y Recomendaciones**
10. **Anexos (código, capturas, JSON de datos, etc.)**

**1. Introducción**

Este documento describe el desarrollo del proyecto **OptiMap**, una aplicación interactiva que permite planificar rutas óptimas entre ciudades de República Dominicana. El propósito principal es aplicar estructuras de datos avanzadas (grafos) y algoritmos clásicos (TSP y MST) dentro de una experiencia visual y didáctica.

**2. Objetivos**

* Implementar estructuras de datos tipo **grafo** y aplicar algoritmos avanzados.
* Construir una interfaz visual interactiva para facilitar la comprensión del funcionamiento de los algoritmos.
* Permitir a los usuarios explorar rutas óptimas (TSP), conexiones mínimas (MST) y rutas personalizadas.
* Integrar backend (Python) y frontend (web) en un solo producto funcional.

**3. Tecnologías Utilizadas**

* **Python 3** (algoritmos y backend)
* **Flask** (API REST)
* **HTML5, CSS3, JavaScript** (Frontend web)
* **SVG** (Gráficos vectoriales para el mapa)
* **JSON** (Almacenamiento de datos de ciudades y distancias)
* **VS Code** (Desarrollo)
* **Git y GitHub** (Alojamiento)

**4. Estructura del Proyecto**

**A screenshot of a computer

AI-generated content may be incorrect.**

**5. Desarrollo del Sistema**

**Backend (Python + Flask)**

* Se desarrolló un **API REST** con Flask.
* Las rutas /api/tsp y /api/mst permiten calcular rutas óptimas y árboles de expansión mínima a partir de los datos cargados desde mapa\_ciudades.json.
* El backend expone los resultados en formato JSON, listos para ser consumidos por el frontend.

**Frontend (HTML5 + CSS + JS)**

* Se diseñó una **SPA** (Single Page Application) usando HTML, CSS y JavaScript moderno.
* El mapa de ciudades se construyó con **SVG** para facilitar la visualización y manipulación de nodos y aristas.
* Las interacciones del usuario (selección de ciudad, cálculo de rutas, visualización de resultados) están completamente gestionadas en el frontend.

**Comunicación entre Componentes**

* La comunicación se realiza a través de llamadas fetch a la API de Flask.
* Los resultados se procesan y se reflejan inmediatamente en la interfaz gráfica.

**6. Explicación de los Algoritmos**

**TSP (Traveling Salesman Problem)**

* **Algoritmo:** Backtracking (para rutas óptimas) y Greedy (para rutas personalizadas rápidas).
* **Objetivo:** Encontrar el recorrido más corto que visite todas las ciudades una sola vez y regrese al origen.
* **Implementación:**
  + El usuario selecciona la ciudad de origen.
  + El sistema construye todas las permutaciones posibles (en el modo exhaustivo) o una solución aproximada (Greedy), calcula la distancia total y retorna la ruta óptima.

**MST (Minimum Spanning Tree, Prim)**

* **Algoritmo:** Prim (Greedy).
* **Objetivo:** Encontrar el conjunto mínimo de conexiones que unan todas las ciudades al menor costo posible (sin ciclos).
* **Implementación:**
  + El usuario puede ver cómo quedarían conectadas todas las ciudades si solo importara el costo total de las conexiones.

**7. Diseño de la Interfaz**

* **Justificación Estética:**  
  Se eligió un diseño minimalista, con colores suaves y elementos visuales claros, para resaltar los algoritmos y los resultados.
* **Mapa Interactivo (SVG):**
  + Cada ciudad es un nodo que puede ser seleccionado.
  + Las conexiones (aristas) se dibujan dinámicamente según los datos y las rutas calculadas.
  + Las rutas óptimas, MST y rutas personalizadas se resaltan con diferentes colores y estilos.
  + El usuario puede seleccionar ciudades, activar/desactivar destinos y ver visualmente cómo cambian las rutas.

**8. Pruebas y Resultados**

* Se probaron todos los algoritmos con diferentes combinaciones de ciudades.
* Los resultados fueron verificados tanto visualmente en la interfaz como mediante impresión en consola en el backend.
* Se realizaron pruebas de carga y de validación de datos para evitar errores comunes de entrada del usuario.

**9. Conclusiones y Recomendaciones**

* El proyecto cumple su objetivo didáctico y funcional.
* La integración de backend y frontend proporciona una experiencia fluida y atractiva.
* Para trabajos futuros, se podría:
  + Agregar más ciudades y rutas automáticamente desde el backend.
  + Permitir edición dinámica de datos desde la interfaz.
  + Incluir mapas geográficos reales como fondo SVG.
  + Mejorar la eficiencia del algoritmo TSP para escalabilidad.

**10. Anexos**

**A screen shot of a computer program

AI-generated content may be incorrect.**

* **Capturas de pantalla de la aplicación**

**A screenshot of a computer

AI-generated content may be incorrect.**

* **Estructura JSON utilizada**

A screenshot of a computer program

AI-generated content may be incorrect.