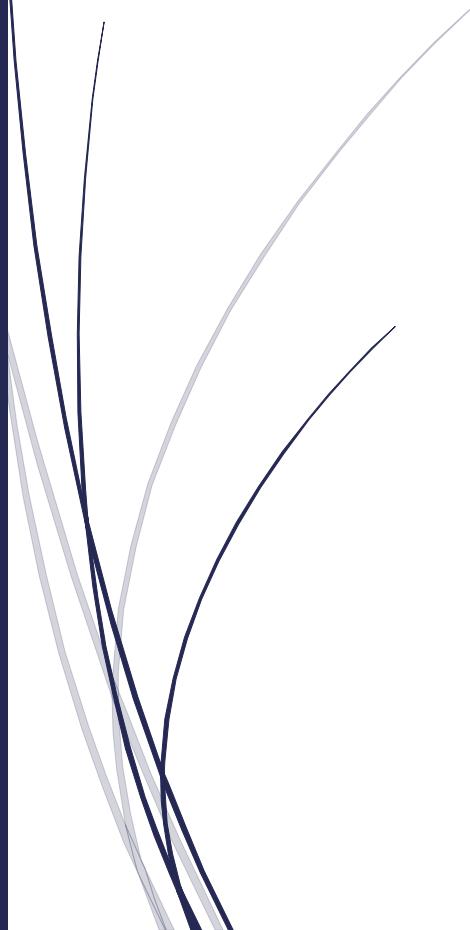




27-5-2025

# Informe: Red de Senderos

- 
- Ludmila Ruiz
  - Nahuel Godoy
  - Juan Gallardo
  - Gaston Ocampo

# Informe Técnico: Red De Senderos

## Introducción:

El presente informe describe el desarrollo de una aplicación visual para la planificación de senderos en un Parque Nacional, minimizando el impacto ambiental mediante el uso de grafos y algoritmos de árbol generador mínimo (AGM). El sistema permite cargar estaciones, definir senderos, calcular rutas óptimas y visualizar los resultados.

## Arquitectura del Sistema:

El proyecto sigue un modelo MVC (modelo-vista-controlador)

Paquetes y clases principales:

PAQUETES	CLASES	RESPONSABILIDAD
controlador	main	Gestiona la lógica de interacción entre modelo y vista.
	controlador	
model	estacion	Maneja la estructura de datos y algoritmos (grafo, AGM con Prim).
	grafo	
	sendero	
	prim	
vista	MainWindow	Interfaz gráfica para visualizar estaciones, senderos y resultados
	PanelBotones	
	PanelMapa	

## Funcionalidades Implementadas:

El proyecto realiza las siguientes funcionalidades, **carga y visualización del grafo**, ya que agrega estaciones, define senderos con su impacto ambiental ingresado por el usuario y se logra visualizar un mapa interactivo.

**Cálculo del árbol generador mínimo** , se utilizó el algoritmo de PRIM, en donde se muestra los senderos seleccionados en el AGM y su impacto.

**Persistencia de datos** ya que se puede guardar y cargar grafos desde archivos JSON (JsonManager).

Además, se pudo agregar color a los senderos según el impacto ambiental que estos tengan (ej: rojo para alto impacto, verde para bajo impacto).

## Análisis del algoritmo de Prim

```
public static List<Sendero> prim(List<Estacion> estaciones, List<Sendero> senderos, Estacion inicio) {
    Map<Estacion, List<Sendero>> grafo = construirGrafoAdyacencia(estaciones, senderos);
    Set<Estacion> visitados = new HashSet<>();
    PriorityQueue<Sendero> cola = new PriorityQueue<>();
    List<Sendero> agm = new ArrayList<>();

    visitados.add(inicio);
    cola.addAll(grafo.get(inicio));

    while (!cola.isEmpty() && visitados.size() < estaciones.size()) {
        Sendero actual = cola.poll();
        Estacion destino = actual.getFin();
        if (visitados.contains(destino)) continue;

        visitados.add(destino);
        agm.add(actual);

        for (Sendero siguiente : grafo.get(destino)) {
            if (!visitados.contains(siguiente.getFin())) {
                cola.add(siguiente);
            }
        }
    }
}
```

- **Entrada:** Lista de estaciones, senderos y nodo inicial.
- **Salida:** Lista de senderos del AGM.
- **Complejidad:**  $O(E \log V)$  (usando cola de prioridad).

## Pruebas Realizadas

Se creo un paquete de TEST para verificar que los métodos implementados en la clase model, cumplan con la lógica planteada.

Se valido que no se pueda crear un sendero con una estación inexistente, también se valido que se pueda unir con estaciones existentes.

Además, se verifica que no se permitan loops, es decir, no se puede crear senderos con una única estación.

Se valido que el AGM retorne el camino mínimo.

## Conclusiones

El proyecto desarrollado bajo la arquitectura **MVC** logró cumplir con los objetivos principales: gestionar estaciones y senderos mediante un grafo, calcular caminos mínimos con el algoritmo de Prim, y persistir los datos en formato JSON.

## URL- Branch Master

<https://github.com/LudmilaGRuiz/RedSenderos.git>