

**PARQUE NACIONAL**

|  |
| --- |
| UNIVERSIDAD |

* Universidad Nacional General Sarmiento
* Materia: Programación 3
* Comisión: 01

|  |
| --- |
| ALUMNOS |

* Mayra Rossi (35.366.464) [Mayra.Rossi03@gmail.com](mailto:Mayra.Rossi03@gmail.com)
* Melina Scabini (44.756.058) [Melina.Scabini@gmail.com](mailto:Melina.Scabini@gmail.com)

|  |
| --- |
| PROFESORES |

* Patricia Bagnes
* Ignacio Sotelo

## **Introducción**

El objetivo de este trabajo práctico es implementar una aplicación visual para optimizar la red de senderos en el Parque Nacional Nahuel Huapi, minimizando el impacto ambiental de su construcción. Las estaciones (miradores, refugios, etc.) se representan como nodos de un grafo no dirigido, y los senderos como aristas con pesos (impacto ambiental, enteros en [1,10]). El objetivo es encontrar un Árbol Generador Mínimo (AGM) que conecte todas las estaciones con el menor impacto total, utilizando ambos algoritmos de Prim y Kruskal para comparar su rendimiento.

La aplicación fue diseñada en java, siguiendo un enfoque modular que separa la lógica del juego (paquete Model) de la interfaz gráfica (paquete View), con una capa de control (paquete Controller) que coordina ambas partes mediante el uso del patrón Observer para la comunicación de eventos. La integración de Observers nos permitio que las vistas reaccionen a cambios en el modelo sin acoplamiento directo, mejorando la modularidad y facilitando la extensión del sistema.

En el desarrollo, se hace la carga datos desde archivos JSON, visualiza el grafo en un mapa interactivo con JMapViewer, calcula el AGM, se muestran los senderos seleccionados como camino óptimo y se detalla el impacto total junto con su tiempo de ejecución.

Este documento describe la estructura del proyecto, los desafíos y decisiones tomadas durante el desarrollo basadas en el análisis de complejidad y buenas prácticas, con el propósito de ofrecer una visión clara del proceso llevado a cabo por el equipo.

**Buenas Prácticas**

* **Pruebas Unitarias:** Se implementaron pruebas con JUnit para las clases de negocio (NationalParkGraph, NationalParkGraphKruskal, NationalParkGraphPriJsonReader, Station, Trail) Siendo útil para detectar puntos de quiebre cubriendo casos como grafos vacíos, disconexos, archivos JSON malformados, entre otros ejemplos. Esto nos ayudó a asegurar la robustez, dando lugar a la detección y manejo temprano de errores.

### **Modularidad y Organización del Código:**

### El proyecto está estructurado en paquetes bien definidos como model, view, controller, observer, lo cual refleja un enfoque modular y respetuoso con el principio de **separación de responsabilidades**. Cada paquete tiene una tarea clara y se encarga de una parte del sistema, lo que facilita la **mantenibilidad** y la **escabilidad** del proyecto a medida que crece.

* **Nombres claros:**

Se emplean nombres descriptivos para clases y métodos, haciendo que el código sea autoexplicativo y fácil de mantener. Esta práctica, alineada con los principios de *Clean Code*, reduce la necesidad de comentarios innecesarios y mejora la legibilidad general. También se respeta la **distancia vertical**, agrupando funciones relacionadas y separando conceptos distintos con claridad, lo que facilita seguir el flujo lógico del programa sin necesidad de desplazarse innecesariamente por el archivo.

* **Encapsulado de Componentes para Reutilización:**  
  Los elementos visuales, como botones, etiquetas, paneles, se encapsularon en funciones, lo que permitió su reutilización y mantenimiento de forma independiente. Este enfoque también ayuda a mantener los archivos más ordenados, pequeños y comprensibles.
* **Desacoplamiento de Estilos con ColorPalette:**  
  El uso del componente ColorPalette permite separar los aspectos visuales del comportamiento de la aplicación. Al centralizar los colores en un único lugar, se logra una mayor coherencia, se facilita la modificación de estilos y se refuerza un diseño desacoplado y escalable.

**Decisiones de Desarrollo**

Durante el desarrollo de la aplicación, se tomaron varias decisiones clave para optimizar la implementación, garantizar la modularidad y cumplir con los requisitos del trabajo práctico. A continuación, se detallan las principales:

**Funcionalidades opcionales**

## **Implementación y Comparación de Algoritmos de Prim y Kruskal:** Se implementaron los algoritmos de Prim y Kruskal para calcular el Árbol Generador Mínimo (AGM). Se midieron sus tiempos de ejecución usando System.nanoTime() mostrando los resultados en la interfaz. Esto permite comparar su rendimiento.

## **Lectura de Datos desde Archivos:** La clase JsonReader carga estaciones y senderos desde nationalParkStations.json usando Gson.

## **Visualización de Senderos por Impacto Ambiental**: En NationalParkMap.drawTrails, los senderos se renderizan con colores según su impacto: verde ([1-3]), amarillo ([4-6]), rojo ([7-10]), definidos en ColorPalette. La actualización dinámica vía NationalParkObserver mejora la usabilidad.

## **Vistas**

A map of a country

AI-generated content may be incorrect.

## 

A map of a park

AI-generated content may be incorrect.

## **Ubicación del programa**

* <https://github.com/ScabiniMelina/NationalPark>