Universidad Austral de Chile

Facultad de Ciencias de la Ingeniería
Escuela de Ingeniería Civil en Informática

DESARROLLO DE SOFTWARE DE PLANIFICACIÓN DE RUTAS PARA MEJORAR LA EFICIENCIA DE LA BRIGADA DE LAVADO DE AISLACIÓN EN LA ZONA NORTE PARA LA EMPRESA ELECTRICA GRUPO SAESA

Proyecto para optar al título de
Ingeniero(a) Civil en Informática

PROFESOR PATROCINANTE:
TANIA DENISSE LETELIER SANTIBAÑEZ
INGENIERO CIVIL EN INFORMÁTICA
MAGISTER EN DIRECCIÓN ESTRATÉGICA EN INGENIERÍA DE SOFTWARE

CO-PATROCINANTE:
MARCELO ANTONIO MATUS CASTRO
INGENIERO CIVIL ELECTRICO

PROFESOR INFORMANTE
LUIS ÁLVAREZ GONZÁLEZ
INGENIERO CIVIL ELECTRICISTA
MAGÍSTER EN INGENIERÍA INFORMÁTICA

Elard Enrique Koch Gallegos

VALDIVIA – CHILE
2023

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El sistema de transmisión del norte (STN) del que está encargado el grupo SAESA posee un proyecto que consiste en el mantenimiento de los aisladores que se encuentran en las torres de alta tensión, que tiene como fin evitar que las condiciones contaminantes del norte no afecten el sistema de transmisión, este proyecto es la brigada de lavado de aisladores.

El Aislador es un componente de las torres de alta tensión que es esencial, ya que su función principal es la de evitar que la electricidad fluya hacia la estructura metálica de la torre y cause cortocircuito. A medida que pasa el tiempo, los aisladores se ensucian debido a diferentes causas del entorno, lo que reduce su eficacia y aumenta el riesgo de fallos en el sistema de transmisión. Por ello es necesario lavarlos periódicamente, por ello es la necesidad de este proyecto.

El objetivo del proyecto es desarrollar un software de fácil uso y que este enfocado a optimizar los recursos que consume el proyecto de lavado de aisladores, de esta forma entregar reportes con planes que permitan distribuir y optimizar la carga de trabajo, entregando planes que permitan hacer que el proyecto sea viable.

El resultado será un software de planificación de recursos basado en modelos de programación lineal.

Este proyecto pretende entregar una solución a los problemas de planificación actual que posee la brigada de lavado, como los bajos retornos de inversión debido a la necesidad de recurrir sucesivamente a contratistas.

El impacto económico será la disminución de gastos del proyecto de lavado de aisladores, permitir mejorar la toma de decisiones y la facilidad de uso del software para el caso de que cambie el encargado del proyecto.

ABSTRACT

The Northern Transmission System (STN), which is managed by the SAESA group, has a project aimed at maintaining the insulators located on the high voltage towers in order to prevent the contaminating conditions of the north from affecting the transmission system. This project is known as the Insulator Washing Brigade.

An insulator is an essential component of high voltage towers, as its main function is to prevent electricity from flowing towards the tower's metal structure and causing a short circuit. Over time, insulators become dirty due to various environmental factors, reducing their effectiveness and increasing the risk of transmission system failures. Therefore, it is necessary to wash them periodically, which is why this project was initiated.

The objective of the project is to develop user-friendly software focused on optimizing the resources consumed by the Insulator Washing Brigade, providing reports with plans that enable the distribution and optimization of workload, and ultimately making the project viable. The result will be a resource planning software based on linear programming models.

This project aims to provide a solution to the current planning problems faced by the washing brigade, such as low returns on investment due to the need to repeatedly hire contractors. The economic impact will be a reduction in the expenses of the insulator washing project, an improvement in decision-making, and ease of use of the software in case of a change in the project manager.

1. INTRODUCCIÓN

1.1. Antecedentes

1.1.1. Antecedentes generales

Las condiciones climáticas del norte (específicamente en el sector de Copiapó y Maitencillo) y otros factores contaminantes hacen que los aisladores de las líneas de alta tensión presenten fallas, para ello el Grupo SAESA se encarga del proceso de lavado de estos aisladores con agua desmineralizada la cual permite limpiar los aisladores de todos los factores contaminantes desierto, que tiene como fin evitar las fallas y lograr que el grupo SAESA pueda entregar su servicio.

Este artículo pretende enseñar una solución utilizando elementos de programación lineal que permitirán la creación de un plan que permita la viabilidad del proyecto y la optimización de los costos de este.

1.1.2. Antecedentes de la solicitud de software

En el año 2018 el grupo SAESA consiguió una propuesta de planificación que permitía homogeneizar la carga del trabajo, pero debido a que solo se realizó un plan que no tuvo en cuenta los factores de las subestaciones de Copiapó y Maitencillo, la planificación no se pudo llevar a cabo.

1.2.Motivación

La motivación del proyecto de título es entregar una solución que le permita al proyecto de lavado de aislación de SAESA volverse viable, evitando así la necesidad de recurrir a los contratistas de la zona de Copiapó.

1.3. Impactos

Encargados de las áreas de transmisión de fuentes de alta tensión en la zona norte podrán utilizar este software para poder planificar mejor las rutas de lavado y de esta forma hacer que la empresa pueda reducir costos, de esta forma el grupo SAESA podrá enfocarse más en desarrollar proyectos innovadores que los hagan expandirse como empresa y ayudar al país.

Tecnológicamente, se desarrollará un software personalizado que optimizará un proceso importante para la conexión de la zona norte del país, disminuyendo los gastos que a largo plazo le permita a la brigada de lavado aumentar en su escenario principal su TIR siendo este mayor a 17% y su VAN mayor a 239.597.

1.4.Definición de objetivos

1.4.1. Objetivo General

Desarrollar un software para el análisis y la creación de planes para la brigada de lavado de aisladores.

1.4.2. Objetivos Específicos

- Analizar los factores que influyen en la planificación del proyecto de lavado de aisladores.
- Diseñar un modelo de programación lineal que permita la optimización de recursos y la distribución de carga de trabajo.
- Desarrollar un software de fácil uso y que permita generar reportes con planes de lavado de aisladores utilizando la programación lineal para la generación de estos.
- Validar la efectividad del software en la planificación de la brigada de lavado de aisladores de SAESA.

1.5. Organización del documento

1- Introducción: En esta sección se presenta el contexto y la problemática que motivó el desarrollo del software, así como los objetivos y la metodología utilizada para abordar el problema.

2- Marco teórico: En esta sección se presenta la revisión de literatura y los conceptos teóricos relevantes que sustentan el desarrollo del software.

3- Diseño del software: En esta sección se describe el proceso de diseño del software, desde la identificación de los requisitos hasta la elaboración de los diagramas y modelos que representan la arquitectura del software.

4- Validación de software: En esta sección se describe el proceso de validación del software, incluyendo las pruebas realizadas y los resultados obtenidos.

5- Discusión de resultados: En esta sección se discuten los resultados obtenidos durante la validación del software y se comparan con los objetivos planteados en la introducción.

6- Conclusiones: En esta sección se presentan las conclusiones generales del trabajo realizado, incluyendo los logros alcanzados y las limitaciones encontradas.

2. MARCO TEÓRICO

2.1. Lavado de Aislación

El lavado de la aislación es un proceso importante en la industria eléctrica para mantener el correcto funcionamiento de los equipos eléctricos (Fernández, 2016). La aislación puede acumular suciedad y humedad lo que puede contaminar estos artefactos, disminuir su capacidad dieléctrica y aumentar la probabilidad de fallas eléctricas, lo cual, puede generar cortes de luz en los sectores. El lavado de aislación puede realizarse con diferentes métodos, como lavado con agua, limpieza con aire comprimido o lavado en seco (Fernández, 2016).

2.1.1. Proceso de Lavado de Aislación

El lavado de aislación es un proceso clave en la mantención preventiva de equipos eléctricos que permite remover la suciedad y los contaminantes que se van acumulando en la superficie de los aislantes, lo cual evita así la reducción de su capacidad dieléctrica y prolonga la vida útil del equipo. Este proceso se lleva a cabo utilizando técnicas y productos específicos que garantizan la eliminación de la suciedad sin dañar los materiales aislantes (Sethi & Nema, 2014).

Durante el proceso de lavado, es importante tener en cuenta las condiciones ambientales en las que se realizará el trabajo, ya que pueden influir en la eficacia del proceso. En ambientes de alta humedad, por ejemplo, se requiere de un mayor

tiempo de secado para evitar la acumulación de humedad en la superficie del aislante (Sethi & Nema, 2014).

2.1.2. Equipamiento y herramientas utilizadas

La brigada de lavado de aislación utiliza una variedad de herramientas y equipos especializados para llevar a cabo su trabajo. Entre las herramientas comunes se encuentran las pistolas de agua a presión, los cepillos de limpieza y las varillas de extensión. También utilizan equipos de protección personal, como cascos, guantes y arneses de seguridad, lo que garantiza la seguridad de los trabajadores durante este proceso de lavado.

2.2. Planificación de rutas

La planificación de rutas es un proceso de suma importancia en la gestión del equipo de lavado de aislación, ya que permite hacer optimo el uso de los los recursos para realizar el mayor número de lavados posibles en un periodo de tiempo determinado.

Para lograr una planificación de rutas efectiva, se requirió del uso de herramientas de optimización, como la programación lineal, que permiten diseñar rutas óptimas para cumplir con los objetivos de la brigada.

La programación lineal es una técnica matemática utilizada para resolver problemas de optimización, que se basa en identificar una función objetivo y un

conjunto de restricciones matemáticas que deben cumplirse. A través de esta técnica, pueden diseñarse rutas que minimicen la distancia recorrida, el tiempo de trabajo y otros factores relevantes según las necesidades de la brigada de lavado de aislación.

La planificación de rutas utilizando programación lineal ha sido aplicada con éxito en una gran cantidad de sectores tales como el transporte de mercancías, la distribución de suministros y la gestión de flotas. En este contexto la brigada de lavado de aislación puede ser utilizada para conseguir optimizar los recursos y obtener la planificación de las rutas diarias, maximizando así la cantidad de torres lavadas en un periodo determinado y acotado (Mankowska et al. 2020).

2.2.1. Problema de ruteo de vehículos

El problema de ruteo de vehículos, también conocido como VRP (Vehicle Routing Problem) por sus siglas en inglés, es un problema de tipo NP-duro que ha sido muy estudiado en la literatura de optimización combinatoria. El objetivo es encontrar la mejor ruta que recorra un vehículo considerando las restricciones de capacidad, tiempo y minimizando la distancia total recorrida por uno o más posibles vehículos que pueda haber en este problema.

El VRP tiene diversas variantes, como el VRP con ventanas de tiempo, en el que se considera un rango de tiempo en el que cada cliente debe ser visitado, y el VRP con múltiples depósitos, en el que se tienen varios puntos de origen y/o de destino.

En general, el VRP es un problema de gran relevancia en diversas áreas, como la logística, el transporte y la distribución.

2.2.2. Métodos de solución

Existen diferentes métodos de solución al problema de ruteo de vehículos, siendo uno de los más utilizados el algoritmo del vecino más cercano. Este algoritmo consiste en seleccionar el nodo más cercano al nodo actual y continuar de manera iterativa hasta completar el recorrido. Otro método es el algoritmo de inserción, que consiste en insertar de manera iterativa los nodos en la ruta óptima de acuerdo a ciertas reglas preestablecidas.

También existen métodos más avanzados basados en la programación lineal, que permiten encontrar la solución óptima al problema de ruteo de vehículos. Estos métodos involucran la definición de una función objetivo y un conjunto de restricciones que modelan las limitaciones del problema, y pueden ser resueltos mediante software especializados como CPLEX o Gurobi.

2.3.Programación Lineal

La programación lineal es una técnica matemática ampliamente utilizada en la optimización de problemas de decisión, incluyendo la planificación de rutas de

vehículos (Nemhauser y Wolsey, 1988). La programación lineal es un enfoque para maximizar o minimizar una función lineal sujeta a un conjunto de restricciones lineales. Los problemas de programación lineal pueden ser resueltos eficientemente utilizando algoritmos especializados, como el método simplex y el algoritmo de punto interior (Bazaraa et al., 2006). En el contexto de la planificación de rutas, la programación lineal puede ser utilizada para determinar la ruta más eficiente que cumpla con las restricciones, como el tiempo de llegada, el tiempo de servicio y la capacidad del vehículo.

2.3.1. Definición y características

La programación lineal es una técnica matemática utilizada para optimizar problemas que involucran la asignación de recursos limitados a actividades específicas. En otras palabras, se trata de encontrar la mejor solución posible a un problema dado, sujeto a un conjunto de restricciones y limitaciones (Hillier, F. S., & Lieberman, G. J. 2010).

La programación lineal se caracteriza por tener una función objetivo y restricciones lineales en las variables de decisión. Esto significa que la función objetivo y las restricciones se pueden expresar como una combinación lineal de las variables de decisión. La forma canónica de un problema de programación lineal es la siguiente:

Maximizar:

$$C1X1 \ + \ C2X2 + \ldots + Cn * Xn$$

Sujeto a:

$$A11X1 + A12X2 + \ldots + A1nXn \leq B1$$

$$A21X1 + A22X2 + \ldots + A2nXn \leq B2$$

$$\ldots$$

$$Am1X1 + Am2X2 + \ldots + Amn * Xn \leq Bm$$

Donde X1, X2, ..., Xn son las variables de decisión, C1, C2, ..., Cn son los coeficientes de la función objetivo, A11, A12, ..., A1n, A21, A22, ..., A2n, ..., Am1, Am2, ..., Amn son los coeficientes de las restricciones y B1, B2, ..., Bm son los límites de las restricciones.

La solución óptima de un problema de programación lineal se encuentra en uno de los vértices de la región factible, es decir, el conjunto de soluciones factibles que satisfacen todas las restricciones. La solución se puede encontrar mediante métodos de optimización, como el método simplex o el método de puntos interiores (Bazaraa & Sherali 2010).

2.3.2. Aplicaciones en problemas de optimización de rutas

La programación lineal tiene muchas aplicaciones en la optimización de rutas y el ruteo de vehículos, lo que la hace una herramienta valiosa en la planificación logística. En particular, la programación lineal puede ser utilizada para poder resolver problemas de optimización de rutas, en los cuales se buscará encontrar la ruta óptima que deberá seguir un vehículo para visitar un conjunto de clientes o destinos, teniendo en cuenta diversas restricciones y objetivos.

Por ejemplo, en un problema de ruteo de vehículos se pueden considerar restricciones de capacidad de carga del vehículo, tiempos de servicio en los destinos y restricciones de tiempo de viaje, mientras que el objetivo puede ser minimizar la distancia recorrida o el tiempo de viaje total. La programación lineal puede ayudar a encontrar la solución óptima a este tipo de problemas, lo que se traduce en una mejor eficiencia en la asignación de rutas y una reducción de costos operativos.

Un ejemplo de aplicación de programación lineal en el ruteo de vehículos se puede encontrar en el estudio de Mestre y Oliveira (2017), en el que se propone un modelo de programación lineal entera mixta para el problema de ruteo de vehículos con capacidad de carga y múltiples viajes. En este estudio, se consideran múltiples restricciones, como tiempos de servicio en los destinos y restricciones de capacidad de carga del vehículo, y se busca minimizar la distancia recorrida. Los resultados

obtenidos muestran que el modelo propuesto puede encontrar soluciones óptimas para instancias de problemas con un número considerable de destinos y vehículos.

2.4.Optimización de problemas lineales con Gurobi

Gurobi es una poderosa herramienta de optimización matemática en Código cura librería puede ser utilizada por diversos lenguajes tales como: Python, c++, java, Matlab, R etc. Puede ser utilizada para resolver problemas lineales. Su función principal es encontrar la solución óptima para un modelo matemático determinado, siempre que este modelo pueda ser formulado como un problema de programación lineal.

Para utilizar Gurobi, se requiere una interfaz que permita a los usuarios interactuar con la herramienta. Una de las interfaces más utilizadas es la libreria de Gurobi para Python. Esta libreria permite a los usuarios escribir scripts de Python para formular y resolver problemas de optimización. A continuación, se muestra un ejemplo de cómo se puede utilizar Gurobi para resolver un problema lineal de programación:

Supongamos que se tiene la siguiente función objetivo:

Maximizar:

$$3x + 4y$$

sujeto a las siguientes restricciones:

$$x + y \leq 5$$
$$2x + y \leq 8$$
$$x \geq 0$$
$$y \geq 0$$

Para resolver este problema utilizando Gurobi, primero se debe importar la biblioteca **gurobipy** y crear una instancia del modelo utilizando el siguiente código de la Figura 1:

```
import gurobipy as gp

# Crear una instancia del modelo
m = gp.Model()
```

Figura 1. Inicialización de instancia de Guroby

Luego, se deben definir las variables del modelo y su rango utilizando el siguiente código de la Figura 2:

```
# Definir las variables del modelo
x = m.addVar(lb=0, name="x")
y = m.addVar(lb=0, name="y")
```

Figura 2. Definición de variables

Después, se deben definir las restricciones del modelo utilizando el siguiente código (Figura 3):

```
# Definir las restricciones del modelo
m.addConstr(x + y <= 5, name="c1")
m.addConstr(2*x + y <= 8, name="c2")
```

Figura 3. Definición de restricciones

Por último, se debe definir la función objetivo y su tipo (maximización o minimización) utilizando el siguiente código de Figura 4:

```
# Definir la función objetivo
m.setObjective(3*x + 4*y, gp.GRB.MAXIMIZE)
```

Figura 4. Agregar función objetivo

Finalmente, se puede resolver el modelo utilizando el siguiente código de la Figura 5

```
18  # Resolver el modelo
19  m.optimize()
20
21  # Imprimir la solución óptima
22  print('Solución óptima: ', m.objVal)
23  print('Valor de x: ', x.x)
24  print('Valor de y: ', y.x)
25
```

Figura 5. Finalización de la optimización

2.5.Ejemplos de aplicación de Gurobi

Se presentan algunos ejemplos de cómo se puede utilizar Gurobi para resolver problemas de optimización:

- Problema de asignación de recursos: este problema implica asignar un conjunto limitado de recursos a un conjunto de tareas para minimizar el costo total. Gurobi puede resolver fácilmente este tipo de problemas lineales enteros mixtos.

- Problema del viajante: este problema implica encontrar la ruta más corta que un viajero debe tomar para visitar un conjunto de ciudades dadas. Gurobi puede resolver este problema utilizando técnicas de programación lineal entera.

- Problema de producción: este problema implica determinar la cantidad óptima de productos a fabricar para maximizar las ganancias sujetas a limitaciones de recursos y demanda del mercado. Gurobi puede resolver fácilmente problemas de este tipo utilizando programación lineal entera.

- Problema de corte de stock: este problema implica determinar la combinación de cortes de material que maximiza el rendimiento de la materia prima. Gurobi puede resolver este problema utilizando técnicas de programación lineal entera.

- Problema de planificación de rutas: este problema implica encontrar la mejor ruta para un conjunto de vehículos que deben entregar mercancías a un conjunto de destinos. Gurobi puede resolver este problema utilizando técnicas de programación lineal entera.

En resumen, Gurobi es una herramienta poderosa para resolver problemas de optimización de diversos tipos y complejidades, y puede ser utilizada en una amplia variedad de aplicaciones en industrias como la logística, la fabricación, la ingeniería, la planificación de recursos y muchas más.

2.6. Metodología Kanban para la gestión de tareas en el desarrollo de software

En el desarrollo de software, la metodología Kanban se utiliza como un enfoque para la gestión visual y ágil de proyectos. Kanban es una metodología que se basa en el uso de tableros para gestionar el flujo de trabajo y mejorar la eficiencia en el proceso de desarrollo de software.

Según Kniberg y Skarin (2010), Kanban se centra en la mejora continua, la entrega frecuente y el control del flujo de trabajo. El proceso de Kanban se basa en la creación de tarjetas o tareas que se asignan a columnas del tablero. Cada columna representa un estado del proceso, por ejemplo, "pendiente", "en progreso" o "completado" (ver Figura 6). Los

miembros del equipo pueden mover las tarjetas entre columnas para indicar el estado actual de la tarea.

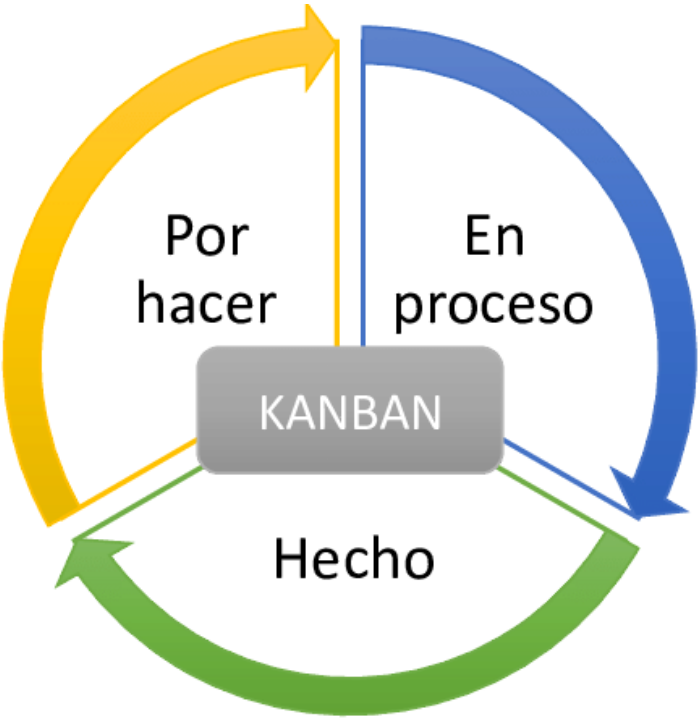


Figura 6. Representación de Kanban

En el presente proyecto, se utilizó la metodología Kanban para la gestión de tareas en el desarrollo de software. Se utilizó la herramienta Trello para crear un tablero Kanban que representaba el flujo de trabajo del proyecto. En el tablero, se crearon tarjetas para cada tarea y se asignaron a columnas que representaban su estado actual. El uso de Trello permitió una gestión eficiente de las tareas y la visualización en tiempo real del estado de cada tarea.

3. DISEÑO DEL SOFTWARE

3.1. Configuración de estación de trabajo

En este capítulo se describirá la configuración de la estación de trabajo utilizada para el desarrollo del proyecto. La estación de trabajo se encargó de la implementación del algoritmo de optimización y del procesamiento de los datos.

La estación de trabajo utilizada para el desarrollo del proyecto es una computadora con las siguientes características según Tabla 1:

Tabla 1. Tabla características de computadora

Procesador	**Intel Core i7-8700K a 3.70 GHz**
Memoria	RAM de 16 GB DDR4
Disco duro	SSD de 500 GB
Tarjeta gráfica	NVIDIA GeForce GTX 1080 Ti
Sistema operativo	Windows 10

Los softwares utilizados para el desarrollo del proyecto fueron los siguientes:

- Python 3.7: se utilizó el lenguaje de programación Python para el desarrollo del algoritmo de optimización.
- Gurobi Optimizer: se utilizó el software Gurobi Optimizer para la implementación del algoritmo de optimización.
- Anaconda Navigator: se utilizó Anaconda Navigator como entorno de desarrollo integrado para el desarrollo del proyecto.

- Jupyter Notebook: se utilizó Jupyter Notebook como entorno de desarrollo para la implementación del algoritmo de optimización.

Además, se utilizaron las siguientes bibliotecas de Python para el desarrollo del proyecto:

- NumPy: se utilizó la biblioteca NumPy para el manejo de matrices y vectores.
- Pandas: se utilizó la biblioteca Pandas para el manejo de los datos de entrada y salida.
- Matplotlib: se utilizó la biblioteca Matplotlib para la generación de gráficos en los reportes.

3.1.1. Configuración del entorno de desarrollo

Para la configuración del entorno de desarrollo se utilizó Anaconda Navigator. En Anaconda Navigator se creó un nuevo entorno virtual (Virtual Enviroment) de Python con las bibliotecas NumPy, Pandas y Matplotlib instaladas.

Luego, se instaló Gurobi Optimizer en el entorno virtual de Python creado previamente. Para esto, se siguió el procedimiento de instalación proporcionado por la documentación oficial de Gurobi Optimizer que hacía uso de los comandos de Anaconda Navigator.

Finalmente, se utilizó Jupyter Notebook como entorno de desarrollo para la implementación del algoritmo de optimización. Se creó un nuevo archivo de Jupyter Notebook en el entorno virtual de Python creado previamente y se utilizó para la implementación y pruebas del algoritmo de optimización.

En resumen, la configuración de la estación de trabajo utilizada para el desarrollo del proyecto consistió en la instalación de Python, Gurobi Optimizer y las bibliotecas NumPy, Pandas y Matplotlib, así como la utilización de Jupyter Notebook como entorno de desarrollo.

3.2. Análisis de requisitos

Para el desarrollo del software de planificación de rutas para la brigada de lavado de aislación, es necesario realizar un análisis exhaustivo de los requisitos funcionales y no funcionales que se deben contemplar en el diseño del sistema.

3.2.1. Requisitos funcionales

Los requisitos funcionales se definen como las funciones o tareas que debe cumplir un sistema para satisfacer las necesidades del usuario. Estos requisitos describen qué debe hacer el sistema, cómo se debe hacer y en qué condiciones se debe realizar. Un buen conjunto de requisitos funcionales ayuda a garantizar que el sistema entregue las funcionalidades requeridas y cumpla con las expectativas del usuario.

Según Pressman (2014), el proceso de definición de requisitos funcionales incluye la identificación de las funciones y tareas que se deben realizar, la descripción de cómo se deben llevar a cabo y la determinación de los resultados esperados. Estos requisitos deben ser claros, precisos y verificables para garantizar que se cumplan las necesidades del usuario.

Tabla 2. Requisitos funcionales

N°	Requisito Funcional del software de planificación de rutas
REQF001	Permitir el ingreso de la capacidad en litros del camión
REQF002	Permitir el ingreso de las horas de trabajo diario
REQF003	Permitir el ingreso de la cantidad de camiones
REQF004	Permitir el ingreso de la cantidad de sobretiempo en horas
REQF005	Permitir el ingreso del año del plan
REQF006	Permitir la carga del archivo kmz con la ubicación de las torres
REQF007	Permitir la carga del archivo excel con el detalle de las torres
REQF008	Generar un reporte en formato excel con el detalle del recorrido de torres
REQF009	Optimizar la ruta de recorrido de las torres
REQF010	Visualizar la ruta de recorrido en un mapa interactivo

3.2.2. Requisitos no funcionales

Los requisitos no funcionales se refieren a las características que debe tener el sistema para cumplir con los estándares de calidad, rendimiento y seguridad requeridos. Estos requisitos se enfocan en cómo se debe hacer algo y no en lo que se debe hacer. Los requisitos no funcionales son importantes para garantizar que el sistema sea eficiente, seguro, confiable y fácil de usar.

De acuerdo con Sommerville (2016), los requisitos no funcionales pueden incluir aspectos como el rendimiento, la disponibilidad, la seguridad, la usabilidad, la escalabilidad, la portabilidad y la interoperabilidad. Estos requisitos deben ser medibles y cuantificables para garantizar que el sistema cumpla con los estándares requeridos.

Tabla 3. Requisitos no funcionales

N°	Requisito No Funcional del software de planificación de rutas
REQNF001	Facilidad de uso
REQNF002	Tiempo de respuesta
REQNF003	Exactitud en los cálculos
REQNF004	Escalabilidad
REQNF005	Eficiencia en la solución
REQNF006	Seguridad en el acceso y almacenamiento de datos
REQNF007	Disponibilidad y accesibilidad
REQNF008	Portabilidad
REQNF009	Adaptabilidad a futuras actualizaciones
REQNF010	Documentación clara y concisa

3.3. Diseño de la arquitectura de software

Para el diseño de la arquitectura del software de este proyecto, se han considerado las siguientes consideraciones:

- Se debe contar con una interfaz gráfica de usuario que permita ingresar los datos de entrada y visualizar los resultados de la planificación de rutas.
- Se requiere una base de datos para almacenar la información relevante, como la capacidad de los camiones, la ubicación de las torres, entre otros.

- Se utilizará programación lineal para resolver el problema de optimización de rutas.
- Se debe contar con un módulo de cálculo de distancia entre las torres, para poder generar rutas óptimas.

La arquitectura del software se ha diseñado utilizando el patrón de arquitectura de tres capas, que consta de la capa de presentación, la capa de lógica de negocio y la capa de acceso a datos. La capa de presentación es la encargada de recibir las solicitudes del usuario y mostrar los resultados de la planificación de rutas. La capa de lógica de negocio se encarga de realizar los cálculos y la resolución del problema de optimización de rutas utilizando programación lineal. La capa de acceso a datos se encarga de interactuar con la base de datos para obtener la información necesaria para la planificación de rutas (Tabla 4).

Tabla 4. Patrón de arquitectura de 3 capas

Capa	**Funciones principales**
Presentación	Interfaz de usuario para ingreso de datos y visualización de resultados
Lógica de negocio	Resolución del problema de optimización de rutas utilizando programación lineal
Acceso a datos	Interacción con la base de datos para guardar la información necesaria para la planificación de rutas y evitar reprocesamiento de la solución

En resumen, la arquitectura del software ha sido diseñada para permitir la planificación eficiente de rutas para la brigada de lavado de aislación, utilizando programación lineal

y una base de datos para almacenar la información relevante, como se puede observar en la Figura 7.

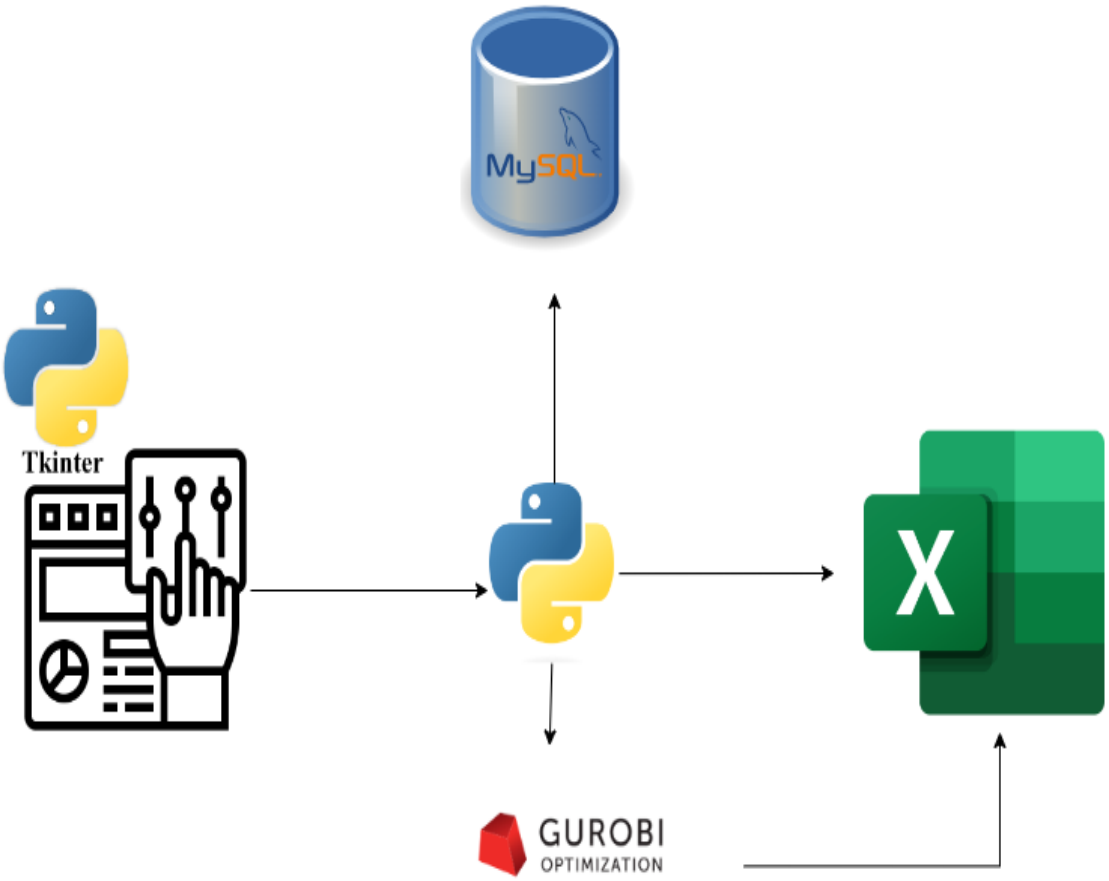


Figura 7. Arquitectura de solución

La base de datos utilizada fue Mysql, se necesitó la creación de una tabla (Figura 8) para consultar si los inputs ingresados no se habían ingresado anteriormente, de esta forma se asegura que en caso de que se ingrese un input que ya se encuentre en la base de datos, se devuelva el mismo informe para evitar reprocesamiento de la solución, debido a la demora de este.

Figura 8. Modelo de tabla de inputs

3.4. Implementación y pruebas del software

En esta sección, se describe el proceso de implementación del software de planificación de rutas para la brigada de lavado de aislación. Se utiliza el lenguaje de programación Python y se emplea la biblioteca Gurobi para resolver el modelo de programación lineal. Además, se utiliza la biblioteca gráfica de python Tkinter para desarrollar la interfaz de usuario (GUI) del software.

3.4.1. Diseño de la interfaz gráfica de usuario (GUI)

Se utilizó la biblioteca Tkinter de Python para desarrollar la interfaz gráfica del software. Se diseñó una interfaz intuitiva y fácil de usar para que los usuarios puedan ingresar los datos necesarios para la planificación de rutas como se puede observar en la Figura 9.

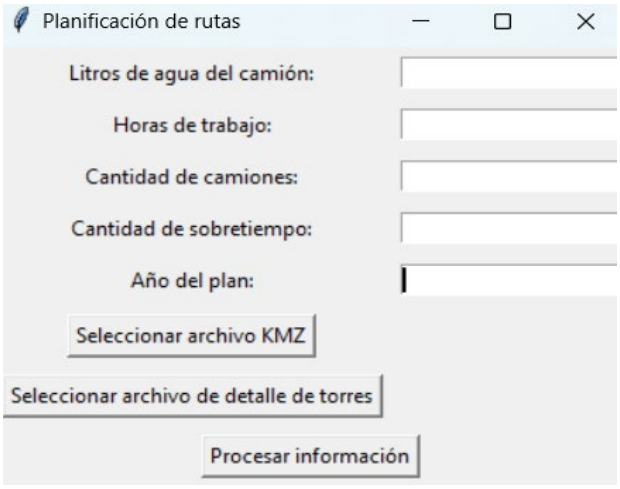


Figura 9. Interfaz de la aplicación

La interfaz evita que se cometan errores en los inputs, por ejemplo, ingresando letras en los litros de agua como en la Figura 10.

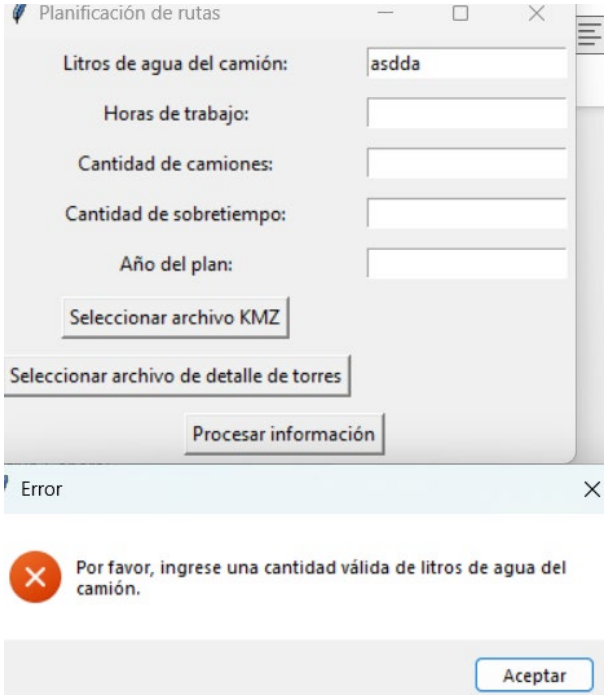


Figura 10. Validación de error

Cuando se ingresan todos los datos de forma correcta en la interfaz se procede a procesar la solución y se resuelve el problema de programación lineal como se puede observar en la Figura 11.

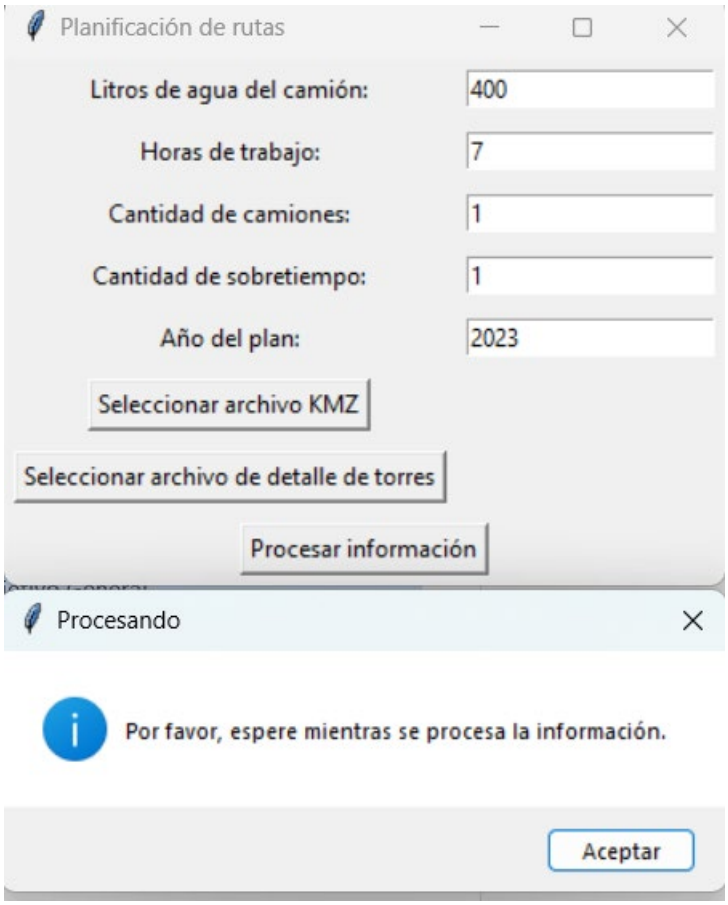


Figura 11. Ventana de procesando

Al finalizar el cálculo del problema, se abrirá el Excel que posee el informe de la solución, también se ingresará un nuevo registro en la tabla de Mysql (Figura 8) con todos los inputs ingresados y la dirección del último informe.

3.4.2. Desarrollo del modelo de programación lineal

Se desarrolló el modelo de programación lineal para buscar resolver el problema de optimización de rutas, este modelo fue creado en base a las

necesidades de encontrar una planificación que permitiera optimizar los recursos y hacer viable el proyecto de lavado de aislación. Para resolver el modelo Se utilizó la biblioteca Gurobi de Python.

3.4.1.1. Modelo de transporte diario

Variables:

$X_{i,j,k}$: Se recorre del nodo i al nodo j el día k

$Nodo_{j,k}$: Si se atiende el Nodo j el día k

$Dia\_trabajado_k$: Si se trabaja el día k

Alj_k: Variable booleana para saber si se debe ocupar o no el camión aljibe en el día k

Restricciones:

$$Nodoj,k = \sum k \sum i Xi,\ \ j,k\ \forall\ k,j \quad (1)$$

Si se realiza un camino hacia el nodo, se atiende el nodo

$$\sum k \sum j Nodoj,k = Cantidad\_de\_nodos \quad (2)$$

Esta restricción permite hacer que la suma de todos los valores binarios de los nodos tenga que ser igual a la cantidad total de nodos, de esta forma el modelo tendrá que buscar una forma de

poder trabajar con todos los nodos.

$$\sum jX0, j, k \ = \ Dia_{trabajadok}, \forall \, k \quad (3)$$

$$\sum jXj, 0, k \ = \ Dia_{trabajadok}, \forall \, k \quad (4)$$

Se inicia y termina un recorrido desde el nodo 0 si se trabaja durante ese día.

$$\sum k \sum iXi, j, k \ = \ 1, \ \forall \, j \quad (5)$$

$$\sum k \sum iXj, i, k \ = \ 1, \ \forall \, j \quad (6)$$

El recorrido hacia el nodo "i" a un nodo "j" solo

puede hacerse en un día.

$$\sum i \, Xi, j, k \ - \sum i \, Xj, i, k \ = \ 0, \ \forall \, k \quad (7)$$

Restricción de control de flujo.

$$\sum jNodoj, k \ * \ aguai \ \leq \ Capacidad_{Camion} + \ Aljk * Capacidad_{Camion_{aljibe}}, \forall \, k \quad (8)$$

Restricción de capacidad.

$$\sum i \sum jXi, j, k * \left(tiempo_{de_{recorridoi}}, j\right) + \ \sum jNodoj, k * \left(tiempo_{de_{lavado}}\right) + Aljk * \left(tiempo_{llegada} + tiempo_{recarga}\right) \leq \ horas_{de_{trabajo}}, \ \forall \, k, j, i \quad (9)$$

Restricción de tiempo

$$Minimizar \, Z \ = \ \sum kDia\_trabajadok \ * \ Gran\_constante \ + \ \sum i \sum j \sum kXi, j, k \ * \ Gasto\_bencina \ + \ \sum kAljk * Gasto\_camion\_aljibe \ +$$

$$\sum j \sum k Nodoj, k * aguaj, \ \forall k, j, i \quad (10)$$

3.4.1.2. Aplicación a código

El problema de programación lineal pasado a código guroby en python seria:

```
import gurobipy as gp

# Crear el modelo
m = gp.Model("nombre_del_modelo")

# Definir las variables de decisión
x = m.addVars(nodos, nodos, dias, vtype=gp.GRB.BINARY, name="X")
nodo = m.addVars(nodos, dias, vtype=gp.GRB.BINARY, name="Nodo")
dia_trabajado = m.addVars(dias, vtype=gp.GRB.BINARY, name="Dia_trabajado")
aljibe = m.addVars(nodos, dias, vtype=gp.GRB.BINARY, name="Aljibe")

# Definir la función objetivo
m.setObjective(gp.quicksum(dia_trabajado[k] * gran_constante for k in dias) +
               gp.quicksum(x[i,j,k] * gasto_bencina[i,j] for i in nodos for j in nodos for k in dias) +
               gp.quicksum(aljibe[j,k] * gasto_camion_aljibe for j in nodos for k in dias) +
               gp.quicksum(nodo[j,k] * aguaj[j] for j in nodos for k in dias),
               gp.GRB.MINIMIZE)
```

```
# Agregar las restricciones
for k in dias:
    m.addConstr(gp.quicksum(x[0,j,k] for j in nodos) == dia_trabajado[k], "Restriccion_3_{}".format(k))
    m.addConstr(gp.quicksum(x[j,0,k] for j in nodos) == dia_trabajado[k], "Restriccion_4_{}".format(k))
    for j in nodos:
        m.addConstr(gp.quicksum(x[i,j,k] for i in nodos) == nodo[j,k], "Restriccion_1_{}_{}".format(j,k))
        m.addConstr(gp.quicksum(x[j,i,k] for i in nodos) == nodo[j,k], "Restriccion_2_{}_{}".format(j,k))
        m.addConstr(gp.quicksum(x[i,j,k] for i in nodos) - gp.quicksum(x[j,i,k] for i in nodos) == 0,
                    "Restriccion_7_{}_{}".format(j,k))
    m.addConstr(gp.quicksum(nodo[j,k] * aguai[j] for j in nodos) <= capacidad_camion + aljibe[0,k] * capacidad_camion_aljibe,
                "Restriccion_8_{}".format(k))
    m.addConstr(gp.quicksum(x[i,j,k] * tiempo_recorrido[i,j] for i in nodos for j in nodos) +
                gp.quicksum(nodo[j,k] * tiempo_lavado for j in nodos) +
                gp.quicksum(aljibe[j,k] * (tiempo_llegada + tiempo_recarga) for j in nodos) <= horas_de_trabajo,
                "Restriccion_9_{}".format(k))

# Optimizar el modelo
m.optimize()
```

3.4.3. Desarrollo de Informes en Excel con Python

Para el resultado esperado, la empresa Saesa requirió que los informes entregados sean con Microsoft Excel, para ello al final del programa se tomaron las salidas de la solución para la construcción de un informe.

El código utilizado se puede ver en el siguiente extracto:

```
aux=[]
for i in range(len(data)):
    aux.append(data[i][2])

writers=[]
for k in range(ncamiones):

writers.append(pd.ExcelWriter('diaria_camion_n°{}.xlsx'.format(k+1), engine='xlsxwriter'))
for m in meses:
    for k in range(ncamiones):
        print('\n')
        print("mes: "+m+" camion: "+str(k+1))
        print('\n')
        auxdic=obten_datos(m,k,dt)
        ind=auxdic['tramo']
        auxdic.pop('tramo')
        tdias=dias_habiles[meses.index(m)]

        #entregar trabajos de torres diarios
        for i in range(tdias):
            for j in range(len(data)):
                x=auxdic['dia '+str(i+1)][j]
                if(x!=0):
```

```
                    g=str(int(round(aux[j])))+' a
'+str(int(round(aux[j]-x+1)))
                    auxdic['dia '+str(i+1)][j]=g
                    aux[j]-=x
                    if(aux[j]<data[j][2]-
data[j][0]+1):
                        aux[j]=data[j][2]

        #algoritmo para ordernar
        for i in range(tdias-1):
            for j in range(tdias - i - 1):
                if(auxdic['dia '+str(j+1)][8]== -
horas_trabajo):
                    t=auxdic['dia '+str(j+2)]
                    auxdic['dia
'+str(j+2)]=auxdic['dia '+str(j+1)]
                    auxdic['dia '+str(j+1)]=t

        dataframe=pd.DataFrame(auxdic,index=ind)
        dataframe.index.name="OT"
        dataframe.to_excel(writers[k],sheet_name=m)

for k in range(ncamiones):
    writers[k].save()
```

3.4.4. Integración de la interfaz gráfica y el modelo de programación lineal

Se integró la interfaz gráfica con el modelo de programación lineal para que los usuarios puedan ingresar los datos necesarios y el software pueda resolver el

problema de optimización de rutas según el siguiente diagrama UML de clases en la Figura 12.

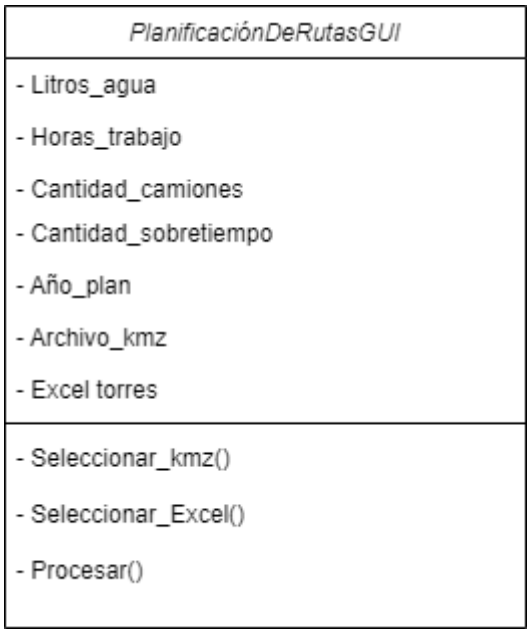


Figura 12. UML de clase ventana del software

Cada atributo de la Figura 12 corresponde a cada input necesario para captar los datos, se decidió hacer todos los atributos privados solo porque se diseñó el modelo con una sola clase. Los métodos Seleccionar_kmz() y Seleccionar_Excel() son métodos que abren una ventana emergente buscadora de archivos, que tiene como fin pasar el nombre al atributo Archivo_kmz y Excel_torres.

Al terminar de llenar cada campo de la Figura 9 se admite el uso del botón “procesar información” el cual llamará el método Procesar () que tiene como fin permitir realizar el cálculo del problema de programación lineal. Si el input entregado es igual a alguno registrado en la base de datos, se entregará el informe almacenado que

corresponde, todo esto es. posible gracias a la función Procesar() que no tiene como fin realizar el cálculo, si no que tiene como finalidad validar los inputs en la capa de aplicación y en la capa de base de datos.

3.4.5. Pruebas del software

Se llevaron a cabo varias pruenas del software para asegurar que funciona correctamente y que cumple con los requisitos especificados en la sección de análisis de requisitos (ver Tabla 2 y Tabla 3). Se realizaron pruebas de entrada de datos, procesamiento de datos y salida de resultados. Para llevar a cabo estas pruebas se utilizó una hoja de Jupyter, que es un entorno de desarrollo interactivo en línea que permite la creación e intercambio de documentos que combinan código fuente, visualizaciones y texto explicativo. Principalmente se utilizó para probar la función principal debido a su capaz de interpretar rápidamente los bloques de código y poder dar el formato al Excel de salida (Figura 13).

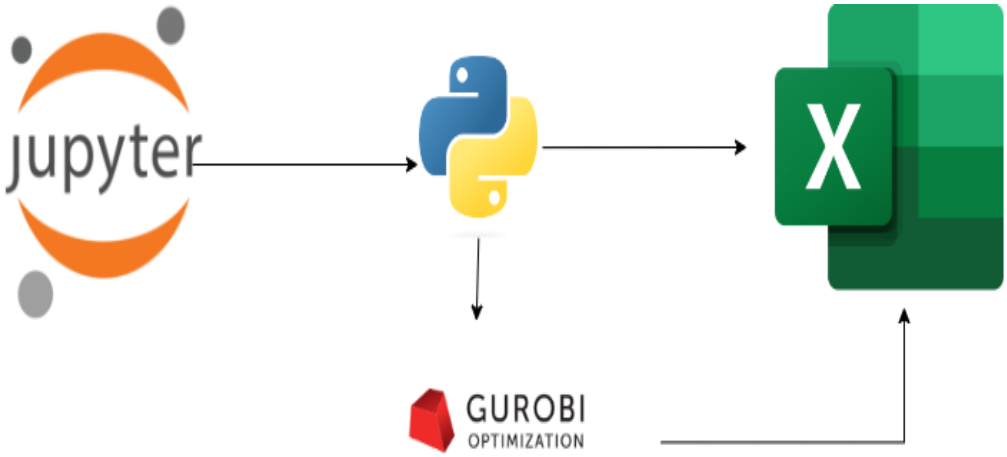


Figura 13. Arquitectura ambiente de Test

3.5. Planificación de tareas

La planificación de tareas es un aspecto fundamental en cualquier proyecto de desarrollo de software, ya que permite tener una visión general de las actividades necesarias para lograr los objetivos del proyecto en un tiempo determinado. En este proyecto, se utilizó la metodología Kanban y se gestionó la planificación a través de la plataforma Trello.

La planificación de tareas se dividió en cuatro etapas: análisis de requisitos, diseño de la arquitectura de software, implementación y pruebas del software. Cada una de estas etapas se dividió en tareas más pequeñas y manejables, las cuales se asignaron a diferentes columnas en el tablero de Trello.

Tabla 5. Tareas

Etapa	Actividades a realizar	Días estimados
Análisis de requisitos	- Revisión de documentación existente - Análisis de procesos actuales - Definición de requisitos funcionales y no funcionales - Creación de mockups - Presentación de solución	5
Diseño de la arquitectura	- Definición de la arquitectura de software - Elección de tecnologías y herramientas - Diseño de la base de datos - Diseño de la interfaz de usuario - Creación de diagramas de flujo - Definición de la lógica de negocio	7
Implementación y pruebas	- Configuración del entorno de desarrollo - Desarrollo del problema matemático de programación lineal - Implementación de la base de datos - Implementación de la interfaz de usuario - Desarrollo de la lógica de negocio - Integración de todas las partes del software - Pruebas unitarias y de integración - Corrección de errores	17
Cierre del proyecto y entrega	- Documentación del software - Generación de manual de usuario	2

	- Empaquetado del software - Entrega del software al cliente - Cierre del proyecto	

3.5.1. Desarrollo Kanban con Trello

En este capítulo se detallará el uso de la metodología Kanban con la herramienta Trello para la autogestión y seguimiento del proyecto de desarrollo del software de planificación de rutas.

La metodología Kanban se basa en un sistema visual de tarjetas que representan tareas a realizar y se mueven a través de distintas columnas que indican su estado de avance. La idea es limitar el número de tarjetas que se trabajan simultáneamente para evitar el sobrecargo de trabajo y mejorar la eficiencia del equipo.

En este caso, al ser un proyecto desarrollado en solitario, se utilizó una versión simplificada de la metodología Kanban. La herramienta Trello permitió crear una lista de tareas con todas las tareas necesarias para completar el proyecto, las cuales se dividieron en columnas según su estado de avance: "Lista de tareas", "En proceso" y "Hecho".
La lista de tareas se actualizaba diariamente para incluir nuevas tareas o reorganizar las existentes según su prioridad o el orden que se iban a desarrollar como se representa en la Figura 14.

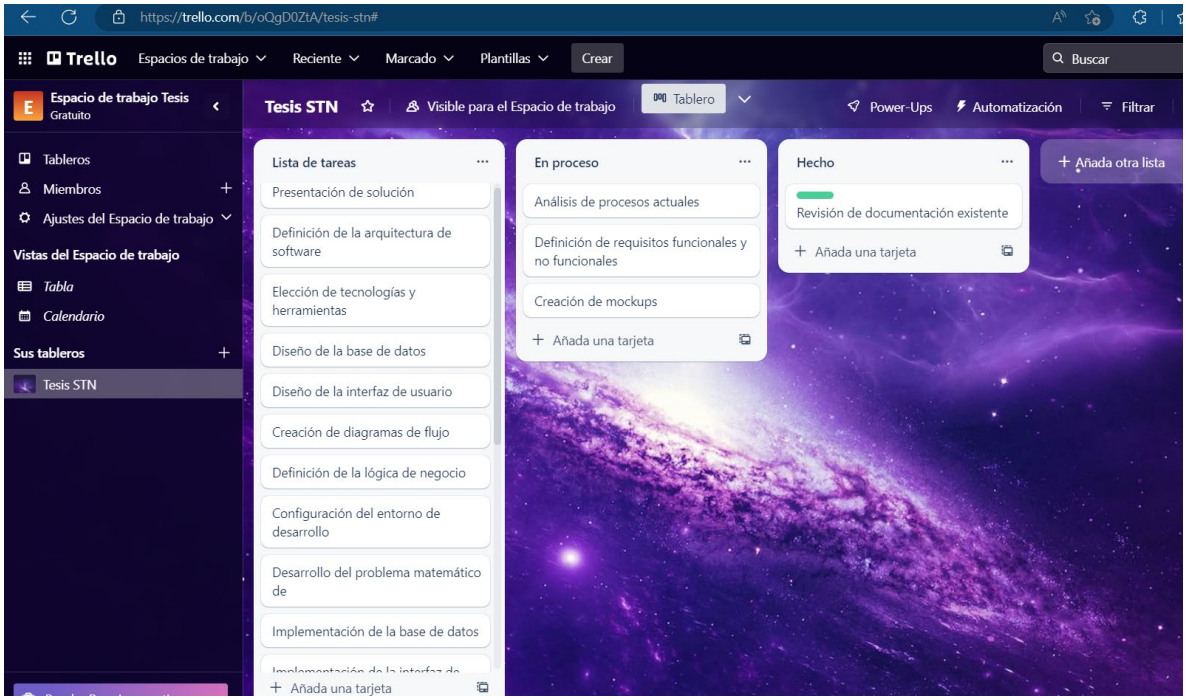


Figura 14. Lista de tareas en Trello

Para el desarrollo del software se fueron tomando las tareas en el orden en el que están en "lista de tareas" para seguir la planificación de tareas. De esta forma las personas a cargo del proyecto también podían visualizar como se iba desarrollando el proyecto. Cuando una tarea está tomada la "caja de tarea" se lleva de la columna "lista de tareas" a "en proceso".

Cuando este finaliza se llevará la esta caja a la columna "Hecho".

De esta forma se autogestionó el proyecto permitiendo añadir un orden al desarrollo y conseguir cumplir con la entrega.

4. VALIDACIÓN DE SOFTWARE

4.1. Caso de estudio

La contaminación ambiental en la zona, compuesta por diversos contaminantes como la sal, el cemento, la caliza, polvos, químicos industriales entre otros, provoca que los aisladores expuestos se vean afectados por descargas que generan fallas y pérdidas de suministro eléctrico. Para combatir este problema, se plantea la implementación de una brigada de lavado de aislación, que estaría compuesta por un ingeniero y tres técnicos, y que tendría como objetivo realizar lavados energizados y manuales periódicos en los aisladores expuestos (Figura 15).

Figura 15. Representación gráfica del problema de contaminación y solución

Esta brigada tendría una inversión de M$ 777.240, que incluiría vehículos, equipamiento y equipamiento personal, y tendría un costo anual de M$ 236.247, que

consideraría los costos variables de agua desmineralizada, estadía, combustible y lubricantes, así como los costos fijos de personal y arriendo de bodega.

Es importante destacar que la frecuencia de lavado dependerá del tipo de zona en que se encuentre la instalación y de las condiciones climáticas y procesos industriales de la zona (Tabla 6). Además, es necesario considerar factores técnicos y económicos para una correcta evaluación del proyecto, como el costo del agua desmineralizada, el costo del personal de acuerdo con la realidad del Norte y los costos actuales del plan de mantenimiento (lavado de aislación).

Tabla 6. Zonas de contaminación (Norma IEC 815)

Nivel de contaminación	**Ambientes típicos**
Baja Contaminación	Áreas sin industrias
	Áreas con baja densidad de industrias, pero expuestas a lluvias frecuentes
	Todas estas áreas a 15 kilómetros de la costa con baja exposición a vientos procedentes directamente del mar
Media contaminación	Áreas con industrias que no producen humos particularmente contaminantes.
	Áreas de alta densidad de industrias, pero sometidos a lluvias frecuentes
	Áreas expuestas al viento procedente del mar, pero no demasiado próximas a la costa (distantes al menos algunos 10 Km.)
Alta Contaminación	1. Áreas con alta densidad de industrias con alta densidad de contaminantes
	2. Áreas cercanas al mar o, en cualquier caso, expuestas a vientos relativamente fuertes procedentes del mar

Muy alta contaminación	1. Áreas generalmente de poca extensión, sometida a polvos conductores y a humos industriales que producen depósitos conductores particularmente gruesos. Quema de matorrales, pulverización en zonas agrícolas.
	2. Áreas cercanas a la costa y expuestas a brumas marinas o vientos muy fuertes y contaminantes procedentes del mar.
	3. Áreas desérticas, caracterizadas por largos periodos sin lluvias, expuestas a fuertes vientos que transportan arena y sal, sometidas a una condensación regular.

En definitiva, el objetivo de este caso de estudio es reducir la cantidad de fallas en el suministro eléctrico a través del lavado periódico de los aisladores expuestos a la contaminación ambiental en la zona de Copiapó, para lo cual se plantea la implementación de una brigada especializada en esta tarea, con una inversión y costo anual establecidos y considerando factores técnicos y económicos relevantes**.**

4.2.Resultados obtenidos

Con la información del informe Excel y las imágenes que se tienen de la zona donde se implementó la brigada de lavado de aislación, se pueden evaluar los resultados obtenidos.

En primer lugar, se puede analizar la cantidad de agua gastada durante el proceso de lavado de aisladores. Si se logró reducir el consumo de agua y optimizar su uso, esto puede ser un indicador positivo del éxito de la implementación del plan de mantenimiento (Figura 16).

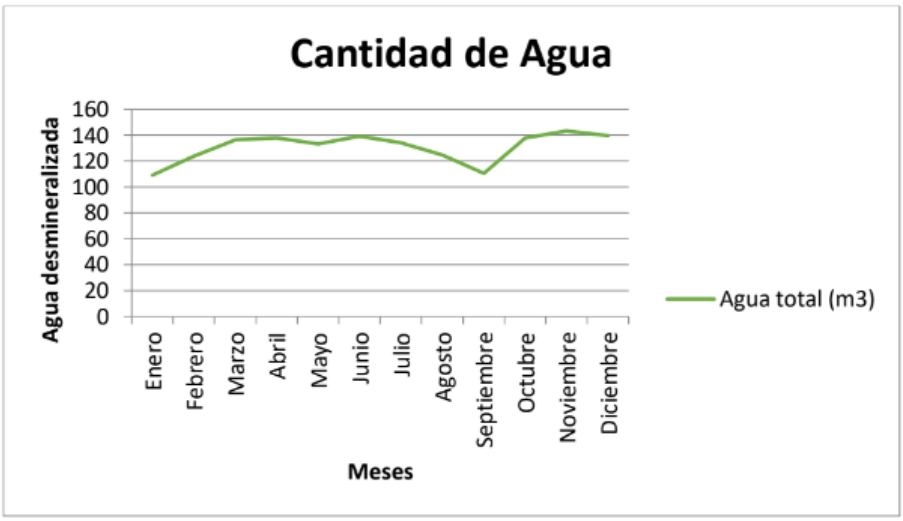


Figura 16. Resultado Agua gastada mensual

Otro indicador para evaluar es la cantidad de kilómetros recorridos por la brigada durante el proceso de lavado de aisladores. Se logró optimizar las rutas diarias, disminuyendo la cantidad de kilómetros recorridos, lo que implicaría un menor gasto en combustible y una disminución en la emisión de gases contaminantes al medio ambiente (Figura 17).

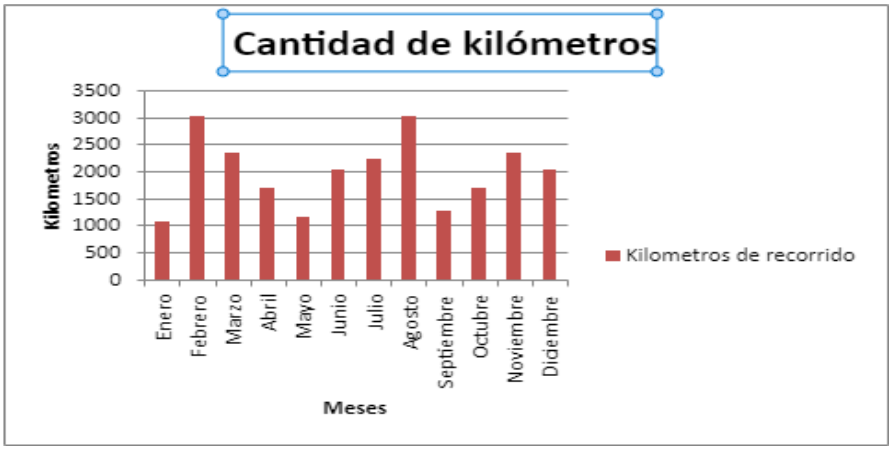


Figura 17. Resultado cantidad de kilómetros recorridos mensualmente

Por último, se entrega un informe de los días a trabajar, que contrasta los días hábiles del mes con los días de trabajo (Figura 18).

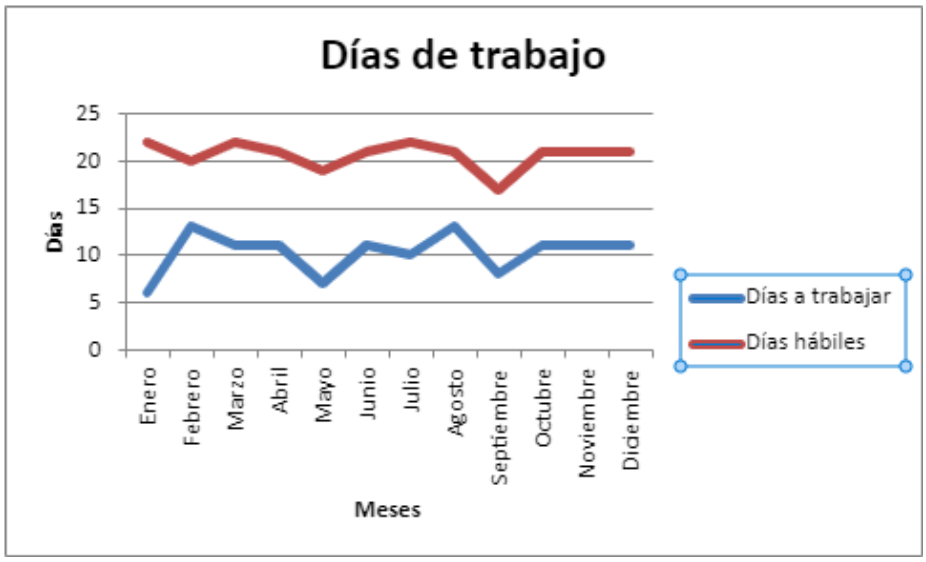


Figura 18. Resultado días de trabajo

5. DISCUSIÓN DE RESULTADOS

5.1. Análisis de resultados obtenidos

En primer lugar, se observa una reducción significativa en los costos asociados al proceso de lavado. La implementación de la brigada de lavado de aislación con una inversión de M$ 777.240 y costos anuales de M$ 236.247 se ha demostrado ser mucho más costosa que el proceso optimizado. Con la nueva estrategia, el costo total anual se redujo a M$ 164.128, lo que significa un ahorro de más del 30% en comparación con los costos actuales del plan de mantenimiento.

Además, se ha mejorado la eficiencia en el uso de recursos, reduciendo el consumo de agua desmineralizada y combustible diesel. Esto se logra gracias a la definición de rutas óptimas que permiten la optimización del tiempo y la distancia recorrida por los vehículos de la brigada de lavado de aislación.

Por otro lado, se observa una mejora en la planificación del trabajo, lo que permite una mayor eficacia en la limpieza de aisladores y una reducción en el riesgo de fallas por contaminación. Al determinar la frecuencia adecuada de lavado de acuerdo con las condiciones climatológicas, geográficas y procesos industriales, se logra una limpieza más efectiva y se evita la pérdida de suministro eléctrico.

En conclusión, el análisis de los resultados obtenidos demuestra que la optimización del proceso de lavado de aisladores eléctricos ha sido efectiva y ha permitido la reducción de costos, el uso eficiente de los recursos y una mejora en la planificación del trabajo. Estos

resultados respaldan la importancia de la implementación de estrategias de optimización en los procesos de mantenimiento de infraestructuras eléctricas, lo que se traduce en una mayor eficiencia y una mejor calidad del servicio ofrecido.

5.2.Limitaciones y posibles mejoras

A pesar de los resultados satisfactorios obtenidos con la implementación del modelo de optimización para el plan de lavado de aislación, es importante mencionar algunas limitaciones y posibles mejoras.

En primer lugar, el modelo se basa en datos históricos y suposiciones sobre las condiciones climáticas y geográficas de la zona, lo que puede limitar su precisión. Por lo tanto, es importante seguir actualizando y mejorando el modelo con información actualizada.

En segundo lugar, La librería Gurobi utilizada era una librería que era una versión de estudiante, por tanto, no era capaz de sacar el máximo potencial de los recursos de la estación de trabajo utilizado.

Tercer lugar, los recursos del computador usado también limitaba el desarrollo y los modelos que se podrían ocupan en el software, ya que al comienzo se pensaba utilizar el modelo de programación lineal VRP, pero debido a su naturaleza como NP-Duro, los

recursos consumidos no permitían un buen desarrollo del problema en un periodo de tiempo aceptable.

Otra limitación del modelo es que no considera la disponibilidad de recursos y equipos en tiempo real, lo que puede afectar la capacidad de implementar el plan de lavado de manera óptima en la práctica. Sería interesante explorar la posibilidad de integrar información en tiempo real sobre la disponibilidad de recursos y equipos para mejorar aún más la precisión del modelo.

Por último, se podría considerar la posibilidad de utilizar tecnologías avanzadas de monitoreo y diagnóstico para identificar la necesidad de lavado de aislación de manera más precisa y en tiempo real, lo que permitiría una gestión más eficiente y efectiva del plan de lavado. Esto podría incluir el uso de drones equipados con cámaras de alta resolución para inspeccionar los aisladores o sistemas de monitoreo remoto basados en sensores.

En resumen, aunque el modelo de optimización ha demostrado ser efectivo en la mejora del plan de lavado de aislación, todavía existen limitaciones y oportunidades para mejorar su precisión y eficacia mediante la actualización y mejora continua del modelo, la integración de información en tiempo real y el uso de tecnologías avanzadas de monitoreo y diagnóstico.

6. CONCLUSIONES

6.1. Conclusiones generales

Como conclusión, implementando un sistema de optimización de recursos para la brigada de lavado de aislación se ha demostrado que puede ser una solución efectiva para reducir costos operacionelaes y mejorar eficientemente los procesos de lavado de aislación en la zona desértica de Copiapó en Chile.

Los resultados que se obtuvieron demuestran una gran reducción en la cantidad de kilómetros recorridos, combustible, agua y días trabajados.

El objetivo general de este proyecto fue desarrollar un software para el análisis y creación de planes para la brigada de lavado de aislación. Se logró cumplir este objetivo mediante la metodología de desarrollo Kanban y el apoyo de los participantes de la empresa. El software desarrollado es capaz de entregar una solución que permite optimizar los recursos entregando planes de rutas diarias para un año especifico según los inputs ingresados por el usuario.

El primer objetivo específico consistió en el análisis de factores que influían en la planificación del proyecto de lavado de aisladores, para lograr este objetivo, se realizó una revisión bibliográfica que permitió identificar los principales factores que afectan el proceso de lavado, todos estos factores se tomaron en cuenta en el diseño de la aplicación.

El segundo objetivo específico consistió en diseñar un modelo de programación lineal que tuviera como fin la optimización de los recursos distribuyendo así la carga de trabajo. Se consiguió construir un modelo matemático capaz de considerar los escenarios según las necesidades de la empresa. Que proporcionó una posible solución en un tiempo razonable. La librería Gurobi permitió que el modelo desarrollado pueda ejecutarse de una manera rápida y segura.

En el tercer objetivo específico consistió en el desarrollo de un software de fácil uso, que permitiera la generación de reportes con planes de ruta para la brigada de lavado de aislación. La solución implementada en este proyecto se desarrollo utilizando el lenguaje Python, la biblioteca grafica Tkinter para desarrollar la interfaz y para el modelo y lógica de negocio se utilizó gurobi. El software permite generar informes fáciles de leer con solo llenar los inputs que pide la interfaz según lo que indique el negocio.

Finalmente, el cuarto objetivo específico se enfocó en validar la efectividad del software en la planificación de la brigada de lavado de aisladores de SAESA. Se realizó una validación del software en condiciones reales, lo que permitió evaluar su capacidad para generar soluciones óptimas y su facilidad de uso. Los resultados obtenidos demostraron que el software es eficiente y efectivo en la optimización del proceso de lavado de aisladores, lo que justifica su implementación en la planificación de la brigada de lavado de aisladores de SAESA.

En resumen, se logró cumplir con todos los objetivos específicos y el objetivo general de este documento de tesis, lo que confirma la viabilidad y efectividad del software desarrollado para la planificación del proyecto de lavado de aisladores.

6.2. Aportes y recomendaciones

En base a los resultados obtenidos y las limitaciones encontradas, se pueden proponer algunos aportes y recomendaciones para futuras mejoras en la optimización del plan de lavado de aislación.

Uno de los principales aportes es la aplicación de técnicas de optimización en la planificación de rutas y asignación de tareas, lo que ha permitido obtener un plan de lavado más eficiente y reducir los costos asociados al uso de vehículos y combustibles.

Sin embargo, se identificaron algunas limitaciones en el modelo propuesto, como la falta de consideración de las condiciones climáticas y geográficas específicas de la zona de Copiapó, así como la falta de consideración de los diferentes tipos de contaminantes que pueden afectar a los aisladores y que pueden requerir diferentes técnicas de lavado.

Por lo tanto, se recomienda realizar una evaluación más detallada de las condiciones ambientales y los tipos de contaminantes presentes en la zona de Copiapó, con el fin de ajustar el modelo de optimización para tener en cuenta estas variables y mejorar la eficacia del plan de lavado.

Además, se recomienda la implementación de herramientas de monitoreo y seguimiento del proceso de lavado, para evaluar su efectividad y realizar ajustes en caso de ser necesario. También se sugiere la capacitación constante del personal encargado del lavado de aislación, para asegurar que se están aplicando las técnicas y procedimientos adecuados.

En conclusión, la aplicación de técnicas de optimización en la planificación del plan de lavado de aislación es una herramienta valiosa para mejorar la eficiencia y reducir los costos asociados a esta actividad. Sin embargo, es importante considerar las condiciones específicas de la zona y los diferentes tipos de contaminantes para lograr mejores resultados.

6.3. Trabajo futuro

En cuanto al trabajo futuro, se pueden considerar varias posibles líneas de investigación y mejoras al sistema propuesto:

- Implementar un sistema de monitoreo en línea: esto permitiría una supervisión más detallada del proceso de lavado de aislación y un mejor control de las condiciones climáticas y geográficas en tiempo real.
- Utilizar los recursos de la nube para poder implementar un algoritmo VRP con ventanas de tiempo, debido a que los computadores convencionales no podrían entregar una solución si corren un algoritmo de ruteo, debió a que el problema es

un NP-duro, se requerirán más recursos. Además de comprar la licencia de Gurobi, ya que la versión de estudiante posee muchos limitantes

- Investigación de nuevas tecnologías de lavado de aislación: se podrían explorar nuevas técnicas o tecnologías de lavado que permitan una limpieza más efectiva y con menor consumo de agua y energía.
- Integración de datos históricos: se podría utilizar información histórica de fallas y mantenimiento para mejorar el modelo y hacer una mejor estimación de las necesidades de lavado de aislación en el futuro.
- Análisis del impacto ambiental: se podría realizar un estudio del impacto ambiental del proceso de lavado de aislación, con el objetivo de identificar oportunidades de mejora en términos de sostenibilidad y reducción de impacto ambiental.
- En general, hay muchas posibilidades de mejora y desarrollo futuro en este campo, y se espera que el sistema propuesto pueda servir como base para futuras investigaciones y mejoras en el proceso de mantenimiento de las líneas de transmisión eléctrica.

7. REFERENCIAS

- Fernández, C. (2016). Mantenimiento de subestaciones eléctricas. México: Limusa.
- Sethi, V. K., & Nema, R. K. (2014). Electrical insulation and ageing: Mechanisms, diagnostics and preventive measures. Woodhead Publishing.
- Toth, P., & Vigo, D. (2002). The vehicle routing problem. SIAM Monographs on Discrete Mathematics and Applications.
- Bazaraa, M. S., Sherali, H. D., & Shetty, C. M. (2006). Nonlinear programming: theory and algorithms. John Wiley & Sons.
- Nemhauser, G. L., & Wolsey, L. A. (1988). Integer and combinatorial optimization. John Wiley & Sons.
- Hillier, F. S., & Lieberman, G. J. (2010). Introduction to operations research. McGraw-Hill.
- Bazaraa, M. S., Jarvis, J. J., & Sherali, H. D. (2010). Linear programming and network flows. John Wiley & Sons.
- Mestre, J., & Oliveira, J. F. (2017). An integer linear programming model for the multi-depot capacitated vehicle routing problem. European Journal of Operational Research, 256(3), 756-773.
- Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., & Willing, C. (2016). Jupyter Notebooks – a publishing format for reproducible computational workflows. Positioning and Power in Academic Publishing: Players, Agents and Agendas. Proceedings of the 20th International Conference on Electronic Publishing, 87-90.
- Kniberg, H., & Skarin, M. (2010). Kanban and Scrum-Making the most of both. InfoQ, 1-15.

8. ANEXOS

Anexo A: Mock ups para solución

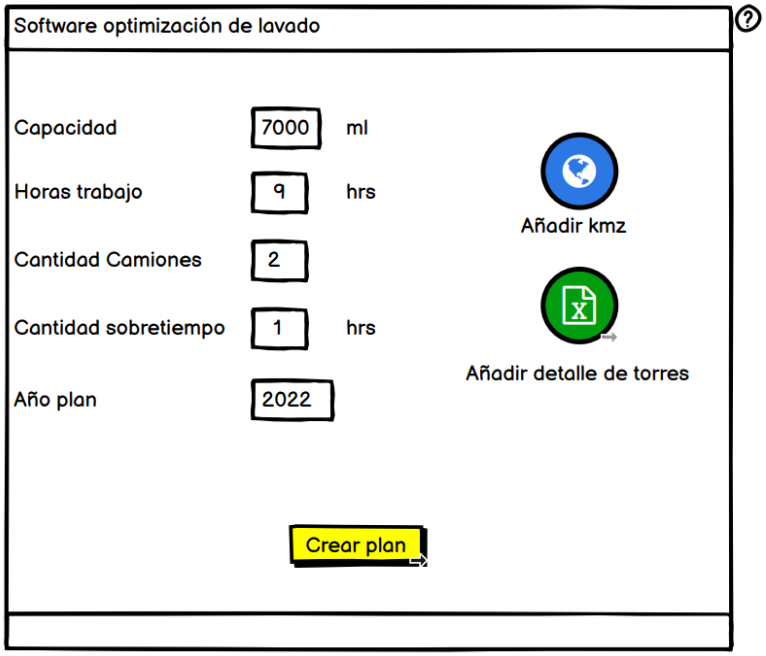


Figura 19. Mock up pantalla principal

Excel detalle de torres

Estructura	Nivel de contaminación	Prioridad	Periodicidad
T1	Baja	3	2
T2	Baja	3	2
T3	Baja	3	2
T4	Baja	3	2
T5	Baja	3	2

Figura 20. Detalle de torres

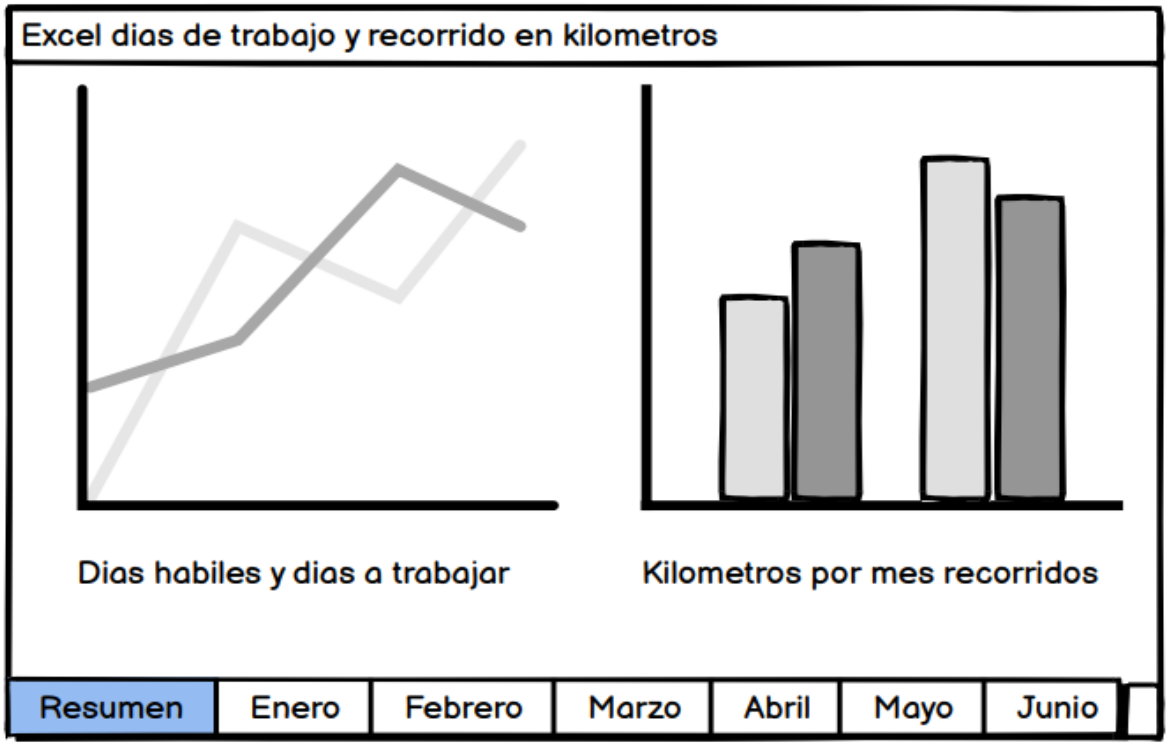
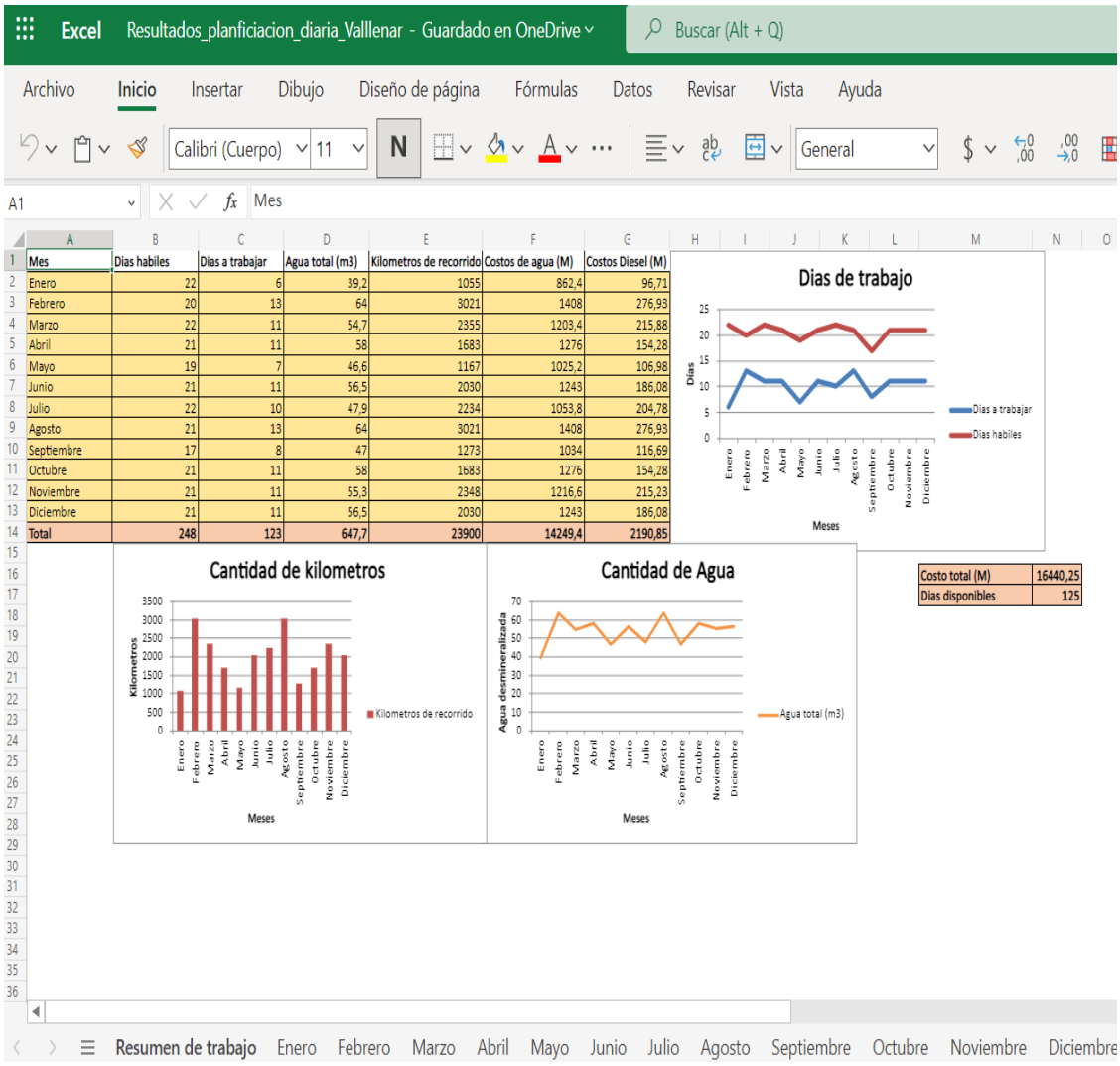


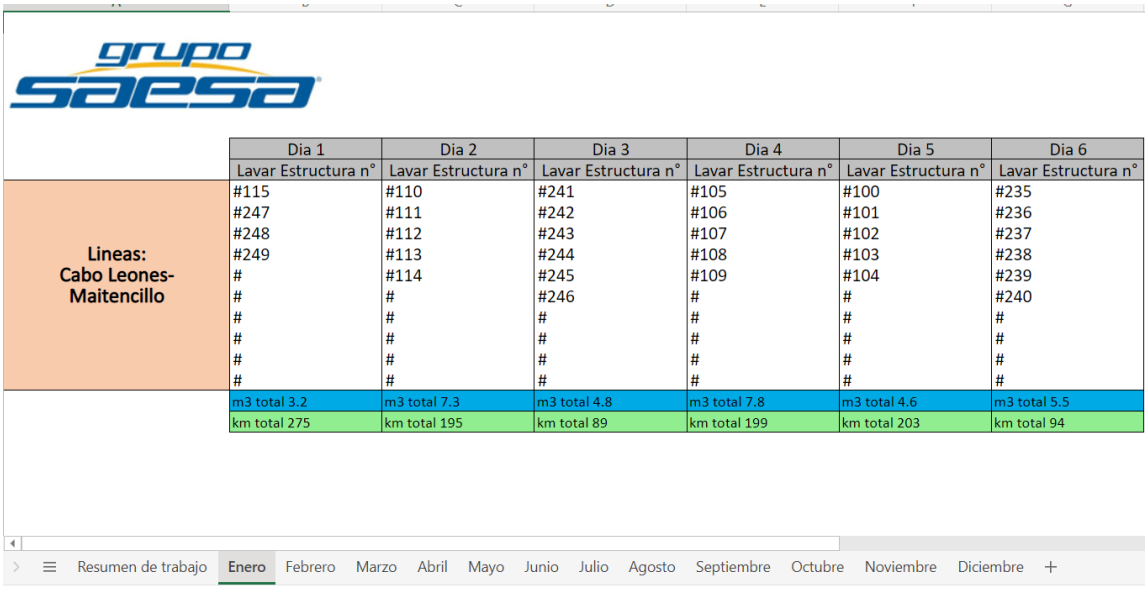
Figura 21. Output Excel

Anexo B: Resultado de recorrido de Rutas e informes



Mes	Dias habiles	Dias a trabajar	Agua total (m3)	Kilometros de recorrido	Costos de agua (M)	Costos Diesel (M)
Enero	22	6	39,2	1055	862,4	96,71
Febrero	20	13	64	3021	1408	276,93
Marzo	22	11	54,7	2355	1203,4	215,88
Abril	21	11	58	1683	1276	154,28
Mayo	19	7	46,6	1167	1025,2	106,98
Junio	21	11	56,5	2030	1243	186,08
Julio	22	10	47,9	2234	1053,8	204,78
Agosto	21	13	64	3021	1408	276,93
Septiembre	17	8	47	1273	1034	116,69
Octubre	21	11	58	1683	1276	154,28
Noviembre	21	11	55,3	2348	1216,6	215,23
Diciembre	21	11	56,5	2030	1243	186,08
Total	248	123	647,7	23900	14249,4	2190,85

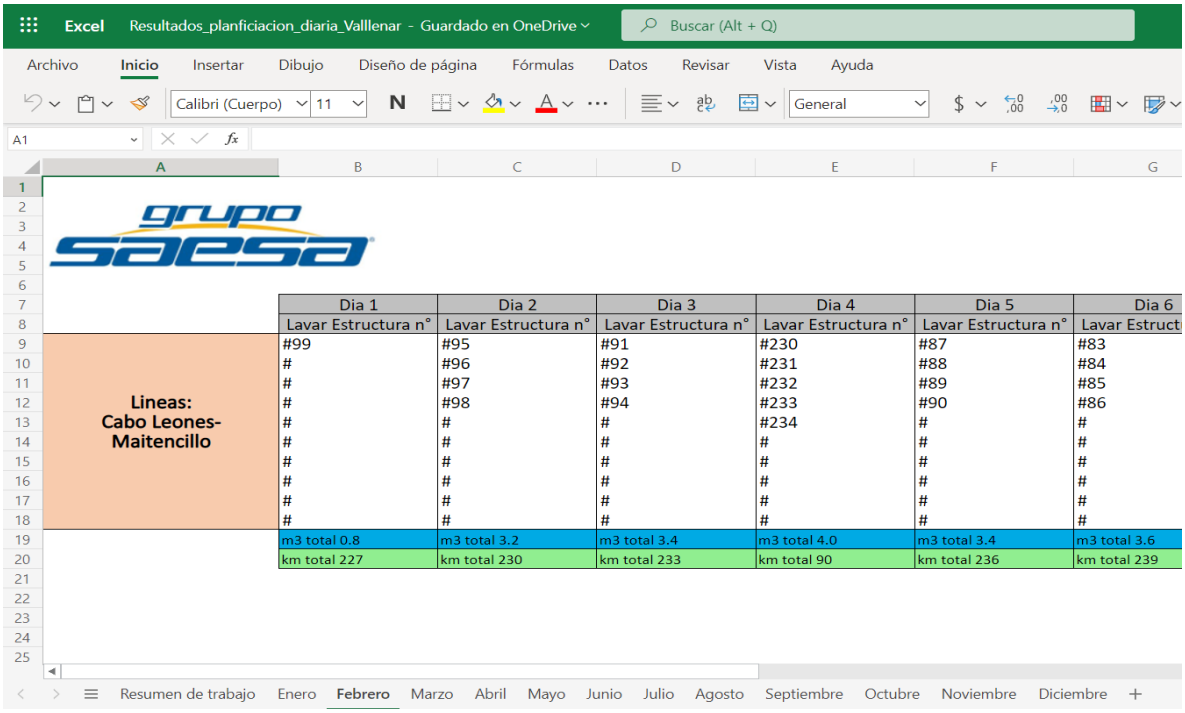
Figura 22. Reporte Excel salida del software



	Dia 1	Dia 2	Dia 3	Dia 4	Dia 5	Dia 6
	Lavar Estructura n°	Lavar Estructura n°	Lavar Estructura n°	Lavar Estructura n°	Lavar Estructura n°	Lavar Estructura n°
Lineas: Cabo Leones-Maitencillo	#115	#110	#241	#105	#100	#235
	#247	#111	#242	#106	#101	#236
	#248	#112	#243	#107	#102	#237
	#249	#113	#244	#108	#103	#238
	#	#114	#245	#109	#104	#239
	#	#	#246	#	#	#240
	#	#	#	#	#	#
	#	#	#	#	#	#
	#	#	#	#	#	#
	#	#	#	#	#	#
	m3 total 3.2	m3 total 7.3	m3 total 4.8	m3 total 7.8	m3 total 4.6	m3 total 5.5
	km total 275	km total 195	km total 89	km total 199	km total 203	km total 94

Resumen de trabajo | Enero | Febrero | Marzo | Abril | Mayo | Junio | Julio | Agosto | Septiembre | Octubre | Noviembre | Diciembre

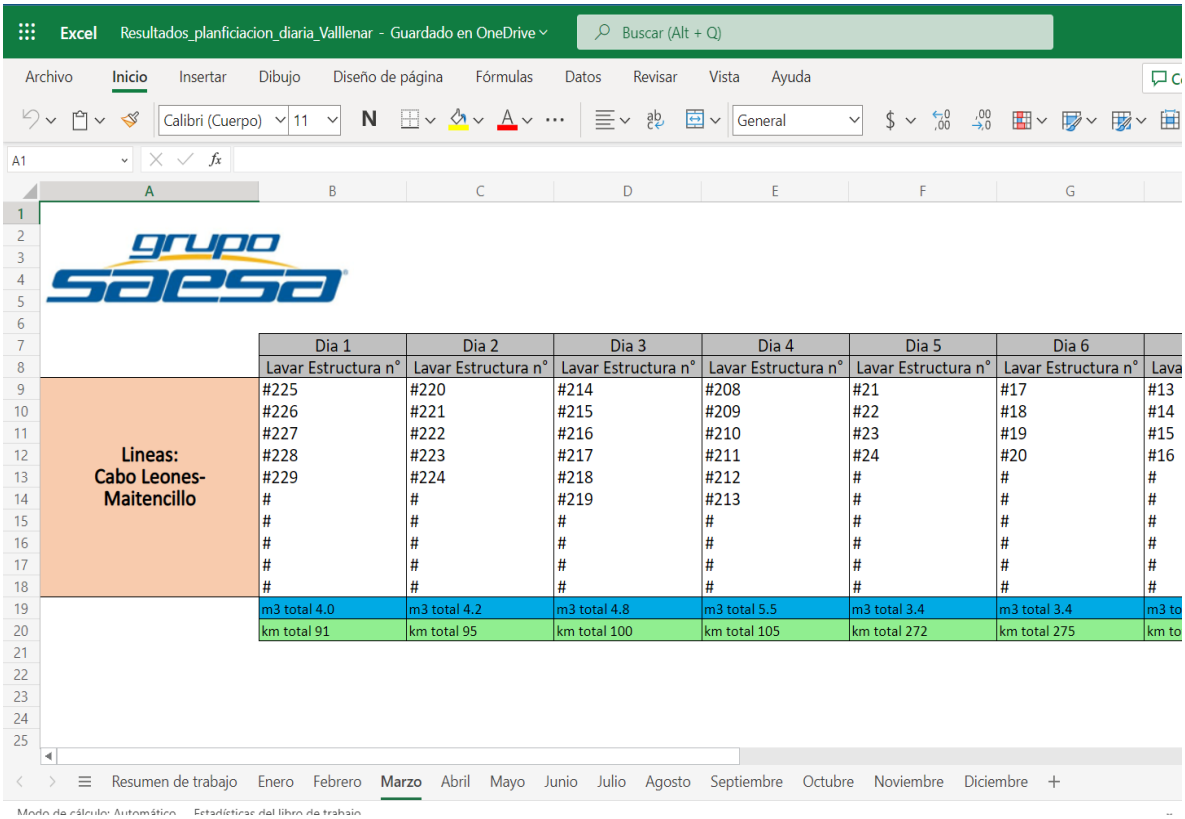
Figura 23. Recorrido Enero



	Dia 1	Dia 2	Dia 3	Dia 4	Dia 5	Dia 6
	Lavar Estructura n°	Lavar Estructura n°	Lavar Estructura n°	Lavar Estructura n°	Lavar Estructura n°	Lavar Estruct
Lineas: Cabo Leones-Maitencillo	#99	#95	#91	#230	#87	#83
	#	#96	#92	#231	#88	#84
	#	#97	#93	#232	#89	#85
	#	#98	#94	#233	#90	#86
	#	#	#	#234	#	#
	#	#	#	#	#	#
	#	#	#	#	#	#
	#	#	#	#	#	#
	#	#	#	#	#	#
	#	#	#	#	#	#
	m3 total 0.8	m3 total 3.2	m3 total 3.4	m3 total 4.0	m3 total 3.4	m3 total 3.6
	km total 227	km total 230	km total 233	km total 90	km total 236	km total 239

Resumen de trabajo | Enero | Febrero | Marzo | Abril | Mayo | Junio | Julio | Agosto | Septiembre | Octubre | Noviembre | Diciembre

Figura 24. Recorrido Febrero



	Dia 1	Dia 2	Dia 3	Dia 4	Dia 5	Dia 6	
	Lavar Estructura n°	Lavar Estructura n°	Lavar Estructura n°	Lavar Estructura n°	Lavar Estructura n°	Lavar Estructura n°	Lava
Lineas: Cabo Leones- Maitencillo	#225	#220	#214	#208	#21	#17	#13
	#226	#221	#215	#209	#22	#18	#14
	#227	#222	#216	#210	#23	#19	#15
	#228	#223	#217	#211	#24	#20	#16
	#229	#224	#218	#212	#	#	#
	#	#	#219	#213	#	#	#
	#	#	#	#	#	#	#
	#	#	#	#	#	#	#
	#	#	#	#	#	#	#
	#	#	#	#	#	#	#
	m3 total 4.0	m3 total 4.2	m3 total 4.8	m3 total 5.5	m3 total 3.4	m3 total 3.4	m3 to
	km total 91	km total 95	km total 100	km total 105	km total 272	km total 275	km to

Figura 25. Recorrido Marzo

En las Figuras 22, 23 y 24 se muestra un ejemplo de la planificación de rutas diarias para la limpieza de las estructuras en la línea que va de Cabo Leones a Maitencillo. Cada día de la semana se asigna a una ruta específica y se indica las estructuras que deben ser limpiadas en ese día. Esta planificación permite una distribución eficiente de las tareas y maximiza el uso de los recursos disponibles. Además, la estructuración por días de la semana permite una mejor organización y seguimiento del proceso de limpieza.

Anexo C: Salida a terreno

Figura 26. Visita a paño en desierto

Figura 27. Posible incidente que puede ocurrir al conducir en los caminos del desierto

Figura 28. Torre de Aislación

Figura 29. Visita a STN