



Programación de un sistema de rutas mínimas asociado al metro de Santiago de Chile

Autores: Oscar Horta, David Huichacura, Ignacio Jara, Catalina Ledesma, Miguel Jaime.

15.11.2021



Profesor: Mauricio Fernando Hidalgo Barrientos

Asignatura: Algoritmos

Número de Grupo: 4

Índice

Glosario de términos	3
Resumen	4
Contexto	5
Análisis del problema desde el punto de vista de la ingeniería	5
Árbol de problemas y objetivos	7
Figure 1: Árbol de objetivos del proyecto.	7
Figure 2: Árbol de problemas del proyecto.	8
Trabajo Previo	9
Objetivo General	10
Objetivos Específicos	10
Enfoque Solucion	11
Figure 3: Enfoque de la solución del proyecto.	11
Método de evaluación de la solución	12
Figure 4: Ruta de ejemplo.	13
Marco Teórico	13
Figure 5: Ejemplo de grafo dirigido y no dirigido.	15
Figure 6: Ejemplo de un grafo ponderado.	15
Figure 7: Ejemplo de grafo simple.	16
Figure 8: Ejemplo de grafo inconexo y conexo.	16
Figure 9: Ejemplo de grafo completo.	16
WBS	17
Figure 10: Diagrama del Work Breakdown Structure	17
Descripción de roles	18
Matriz 1 : Asignación de Siglas a cada integrante del grupo.	18
Matriz 2 : Descripción de Roles requeridos en el desarrollo del Proyecto	18
Matriz 3: Asignación de roles a cada integrante del grupo con respecto a cada actividad a desarrollar.	19
Estimación de tiempo y esfuerzo	20
Matriz 4: “Estimación de tiempo de esfuerzo”.	20

Carta Gantt	21
Matriz 5: “Carta Gantt”.	21
Construcción de la solución propuesta (MJ Y DH)	22
Código 1: Código que demuestra la obtención de resultados.	22
Código 2: “Algoritmo de búsqueda de distancia en una api”.	23
Figure 11: Construcción del grafo.	24
Figure 12: Base de datos expuesto en el excel.	24
Código 3: “grafo parte 1 agregado a excel y creación de grafo”	25
Código 4: “grafo parte 2 agregado a excel y creación de grafo”.	25
Código 5: Integración de la base de datos en el código.	26
Código 6: Construcción del traslado 1:1.	28
Código 7: Construcción del traslado 1:3.	28
Código 8: Construcción del login.	29
Figure 13: Aspecto de la ventana de ingreso.	30
Código 9: Construcción de la ventana principal.	31
Figure 14: Aspecto de la ventana principal.	32
Código 10: Construcción de la ventada de traslados.	33
Figure 15: Ventana de la elección del tipo de traslado.	33
Código 11: Ventana del traslado 1:1.	34
Código 12: Funciones de la ruta con combobox.	35
Código 13: Construcción de entrega de resultados al usuario.	35
Código 14: Creación del botón que entrega los valores.	35
Figure 16: Ventana para el ingreso de datos para el usuario.	36
Código 15: Construcción de la ventana del traslado 1:3.	36
Código 16: Construcción de la rotación de locales y distancias.	37
Código 17: Creación del botón que calcula la ruta.	37
Figure 17: Aspecto del resultado que entrega el programa.	38
Explicación del proceso de construcción de la solución (MJ Y DH)	38
Evaluación de efectividad de la solución	39
Prueba 1: “Traslado 1:1”.	39
Prueba 2: “Traslado 1:1”.	40
Prueba 1: “Traslado 1:3”.	41
Prueba 2: “Traslado 1:3”.	42
Resultados y Discusión	43
Conclusiones	44

Referencias

44

Glosario de términos

Nodos: Son elementos con características, al menos uno de estos funciona como un punto de referencia para otro nodo, además están incrustados dentro de una lista enlazada, un árbol o un grafo.

Rutas: Es usado en la programación para indicar la localización de un archivo, mediante una cadena de caracteres concreta.

API: Las siglas significan “Interfaz de programación de aplicaciones”, es un conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizada por otro software.

Algoritmos de dijkstra: También llamado algoritmo de caminos mínimos, es usado para encontrar la ruta más corta – de menor peso – en un grafo desde un vértice A hasta un vértice B pasando por “n” vértices intermedios, donde $n > 0$, y cada arista tiene un “peso” asociado $p > 0$. Las aristas pueden ser “dirigidas”.

Programación Dinámica: En informática, la programación dinámica es un método para reducir el tiempo de ejecución de un algoritmo, que consiste en resolver los subproblemas una sola vez, guardando sus soluciones (por ejemplo en un vector) para una futura utilización.

Minimización: Reducir considerablemente o al mínimo, ya sea material o no, como situaciones o valor de algo.

Optimizar: Es utilizado para determinar los valores de las variables que intervienen en un proceso o sistema para que el resultado que se obtenga sea el mejor posible.

Gestión de rutas: Se realiza antes de la ejecución de rutas y consiste en la preparación de los recorridos que llevarán a cabo los distintos procesos logísticos.

Teoría de grafos: La teoría busca representar de forma visual conjuntos de datos abstractos en formas de nodos o vértices y la unión o relaciones que estas pueden tener con otros nodos a través de aristas.

Frontend: Es la parte de un sitio web que interactúa con los usuarios, ya que está del lado del cliente.

Backend: Es la parte que se conecta con la base de datos y el servidor que utiliza dicho sitio web, ya que el backend corre del lado del servidor.

Resumen

Dado que la empresa de cartas *Magic* llamada “MHB Shortest Jobs Spa”, ha decidido expandir su negocio a locales físicos, los cuales se ubicaran a las afueras de las estaciones del metro de Santiago de Chile. Surge una nueva problemática; mantener el flujo de mercadería en todos los locales para satisfacer la demanda de los clientes, para así tener un buen servicio y experiencia de venta agradable.

Nace la necesidad de reponer el stock de cada local en el tiempo y ruta más óptimo, por ello se realizó un sistema que mantenga el flujo de la mercadería suficiente en cada local y así poder satisfacer la demanda de los clientes. El sistema consiste en la búsqueda de rutas con la inserción de la librería de búsqueda “request” dentro del API, que permite la búsqueda por medio de google maps para obtener la distancia en kilómetros, que en conjunto con la aplicación del algoritmo de dijkstra, que señalar la ruta mínima dentro del grafo, señala el recorrido que se debe hacer con la ruta mínima. Con ello se obtienen las rutas y distancia para optimizar la bencina gastada por kilómetro, dentro del API y el Save Excel se descubrió que al momento de hacer las consultas para la construcción del grafo, se tomaba en promedio 10 minutos para la construcción, por ende en primera instancia (primera consulta) es ineficiente.

Para la mejora del sistema se debe plantear el remover las consultas por internet, dado que hace que el programa se demore en dar una respuesta, por ende se podría utilizar otros métodos como la aproximación de distancias mediante valores promedios.

Contexto

Las empresas inevitablemente tienen diversas necesidades y requisitos, que al pasar el tiempo van en aumento, una de ellas son los espacios físicos, la cual está dentro de las siete estrategias de marketing que aumentan las ventas de las empresas, es así como los establecimientos en donde se expone el producto o servicios al público por lo general tiene una proyección positiva.

Un caso en particular es la empresa “MHB Shortest Jobs Spa”, ya que al expandirse con sus locales a la afueras del metro, necesita un sistema que optimice la distancia recorrida por los reponedores desde un local a otro sea la mínima posible, dado que permite reducir los costos en bencina y tiempo. Por ello se investiga la creación de un programa con python, que permita la optimización del negocio, por ende se plantea la posibilidad de la utilización de algoritmos que minimicen la distancia, se implementaran los nodos para que realice la búsqueda recorrigendolos y la búsqueda de alguna interfaz que permita un manejo fácil para el usuario. Dada esta problemática, se inició una investigación para la búsqueda de conceptos y material que permita desarrollar una solución óptima, con lo cual se hallaron un abanico de opciones, se optó por la utilización de grafos, consultas y el algoritmos de dijkstra en conjunto. Además de la búsqueda de funciones de la librería del tkinter para Python.

Análisis del problema desde el punto de vista de la ingeniería

El principal factor qué provocó el requerimiento de software capaz de calcular las rutas mínimas entre las estaciones de metro de la región metropolitana, fue un sistema propuesto por el CEO de le empresa *MHB Shortest Jobs SpA*, el cual consta de implementar permita mantener un constante flujo de movimiento de mercadería, y la , es la carencia de eficiencia que desencadena no tener un programa que realice la minimización de rutas de repartidores de mercadería en dicha empresa.

Primeramente, la satisfacción del cliente es fundamental, el mismo CEO de la empresa nombrada con anterioridad lo reconoce, aclarando que una de sus finalidades es satisfacer la demanda de sus clientes, y si el programa propuesto no llegara en ningún momento. Las consecuencias de esto muy posiblemente serían perder clientes, perder ingresos y utilidades, no poder renovar stock por la falta de compras, y mantener un sistema de rutas de transporte que no necesariamente siga la ruta que favorece la eficiencia y eficacia deseada.

Desde otro punto de vista, las circunstancias nombradas anteriormente poseen diferentes ramas de la ingeniería. El transporte, la reposición y el inventario de stock, es posible tratarla con la Ingeniería Civil en Logística, cuya profesión se encarga de los temas anteriores, como también del área de tecnologías de información, calidad, gestión de almacenes, etc. El no contar con alguien especializado en esta área también podría llevar a la empresa a una caída en términos de ingresos, eficiencia y eficacia, haciendo que los procesos de la misma no sean medibles, por ende, incontrolables.

Para la solución propuesta por el dueño de la empresa, es totalmente necesario contar con un Ingeniero en Informática, especialmente uno que esté enfocado en el área del desarrollo de software, debido a que realizar este programa en el cual se deban optimizar y considerar variables de distancia y recorrido, significa que es de suma importancia poseer conocimientos en lenguajes de programación y etiquetado, optimización, algoritmos, teoría de grafos, estructuras de dato, bases de datos y posiblemente UML o casos de uso. Sólo para facilitar la comunicación con el cliente, ya que podría tener un conocimiento nulo en el tema de la informática enfocada en el área de desarrollo; La carencia de este profesional, sólo provocaría que la empresa mantendría sus sistemas de ruta para la entrega de mercancía, que posiblemente no estén digitalizados de forma óptima, y esto sólo puede generar una demora en los tiempos de entrega de productos, y por consiguiente, una mayor espera a los clientes por aquellos productos que hayan solicitado.

Árbol de problemas y objetivos

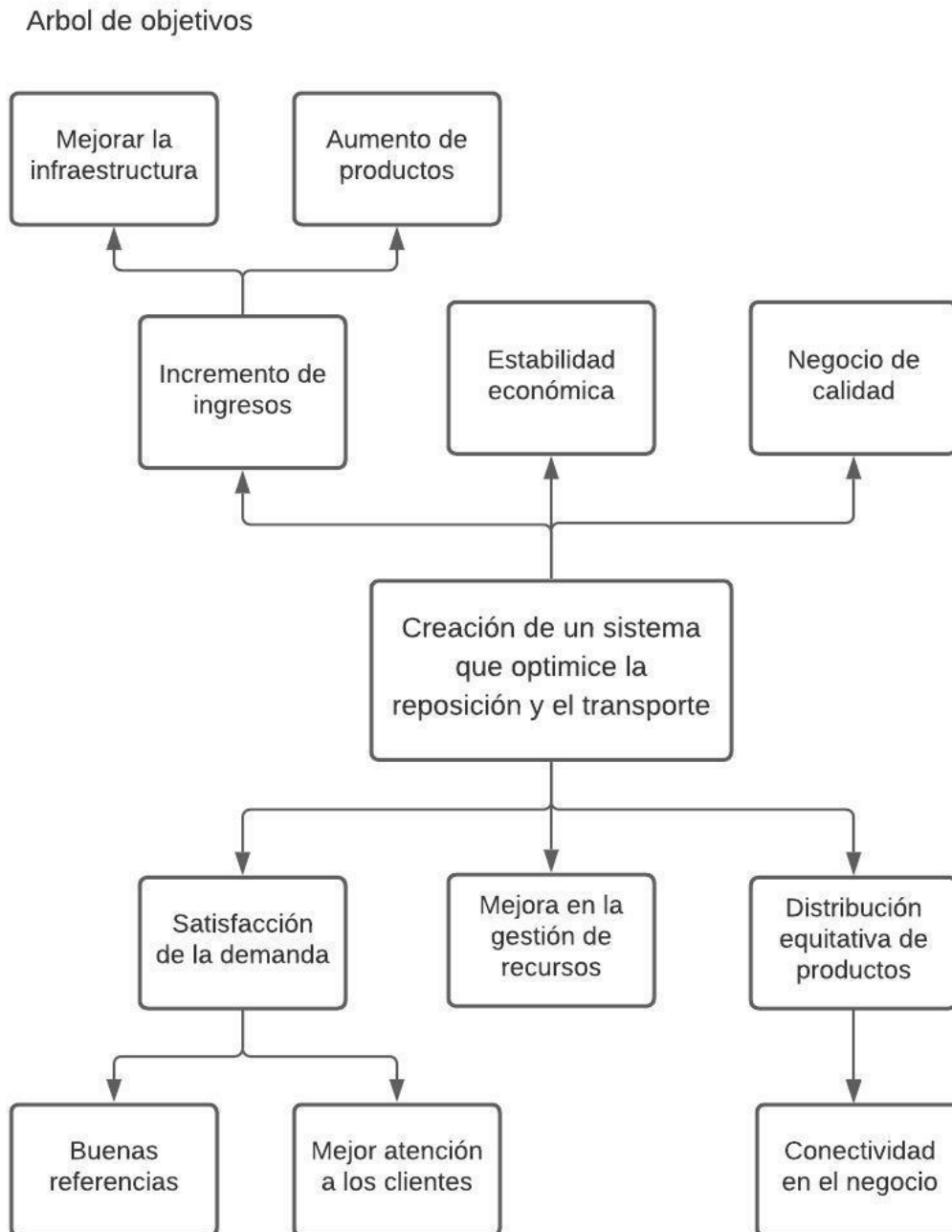


Figure 1: Árbol de objetivos del proyecto.

Arbol de problemas

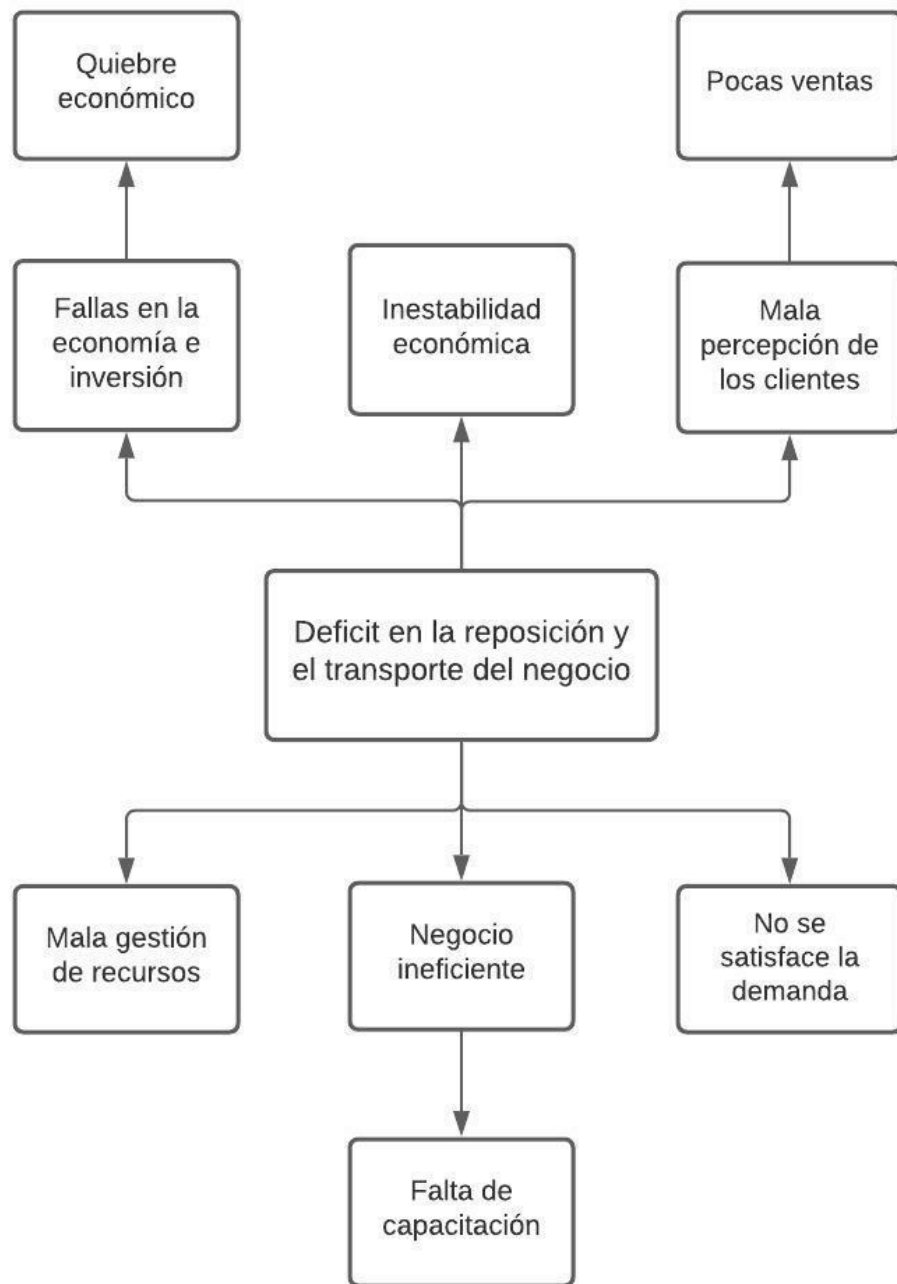


Figure 2: Árbol de problemas del proyecto.

Trabajo Previo

La acción que se requiere tomar en el presente proyecto es en respuesta ante la necesidad de una solución eficaz y eficiente para el sistema de rutas del transporte de la empresa en cuestión, por lo cual es de suma importancia el conocimiento de otros proyectos hechos con anterioridad, debido a que esta mención a otros trabajos prácticos que tienen algunas semejanzas con el actual. Servirá como una retroalimentación de parámetros que podrían pasar por alto.

El caso de la empresa *Semacaf Máquinas de Café S.L(1)*, es un claro ejemplo de gestión de rutas siguiendo conocimientos informáticos anteriormente nombrados, la cual es la Teoría de grafos, basados principalmente en el algoritmo del problema del viajero(2), cuyo objetivo de este algoritmo es principalmente encontrar un recorrido entero que conecte todos los nodos de una red, pero con la condición de que estos sólo puedan ser visitados una vez, y además, la salida de este algoritmo debe ser la ruta óptima, es decir, que minimice la distancia total de esta, incluso, el tiempo de recorrido; el objetivo de basarse en aquel algoritmo y realizar una metodología en base a la teoría de grafos. Es minimizar los costes de la empresa, ya que poseían elevados costes de carburantes, mano de obra, etc.

Jeanpier Flores, estudiante de la Universidad César Vallejo, en su tesis, estableció un modelo heurístico, con el objetivo de minimizar los costos operativos del Servicio de transporte de la ruta de la empresa Brandom S.A.C, año 2018(3). Esto lo logró implementando el algoritmo de los ahorros(4), cuya materia se centra en la planificación de rutas, cuya base es posible implementar en sistemas informáticos, ya que esto combina soluciones de dos rutas distintas para formar otra ruta, en donde se autentifican los ahorros.

Objetivo General

- Optimizar el flujo de mercadería en locales de la empresa *MHB Shortest Jobs SPA* por medio de un sistema de rutas mínimas.

Objetivos Específicos

- Conocer los métodos y algoritmos para el desarrollo del software.
- Conocer implementaciones de algoritmos y técnicas en un lenguaje de programación.
- Utilizar diferentes herramientas para el modelado, diseño y construcción de software.
- Especificar la solución implementada por diferentes medios de comunicación.

Enfoque Solucion

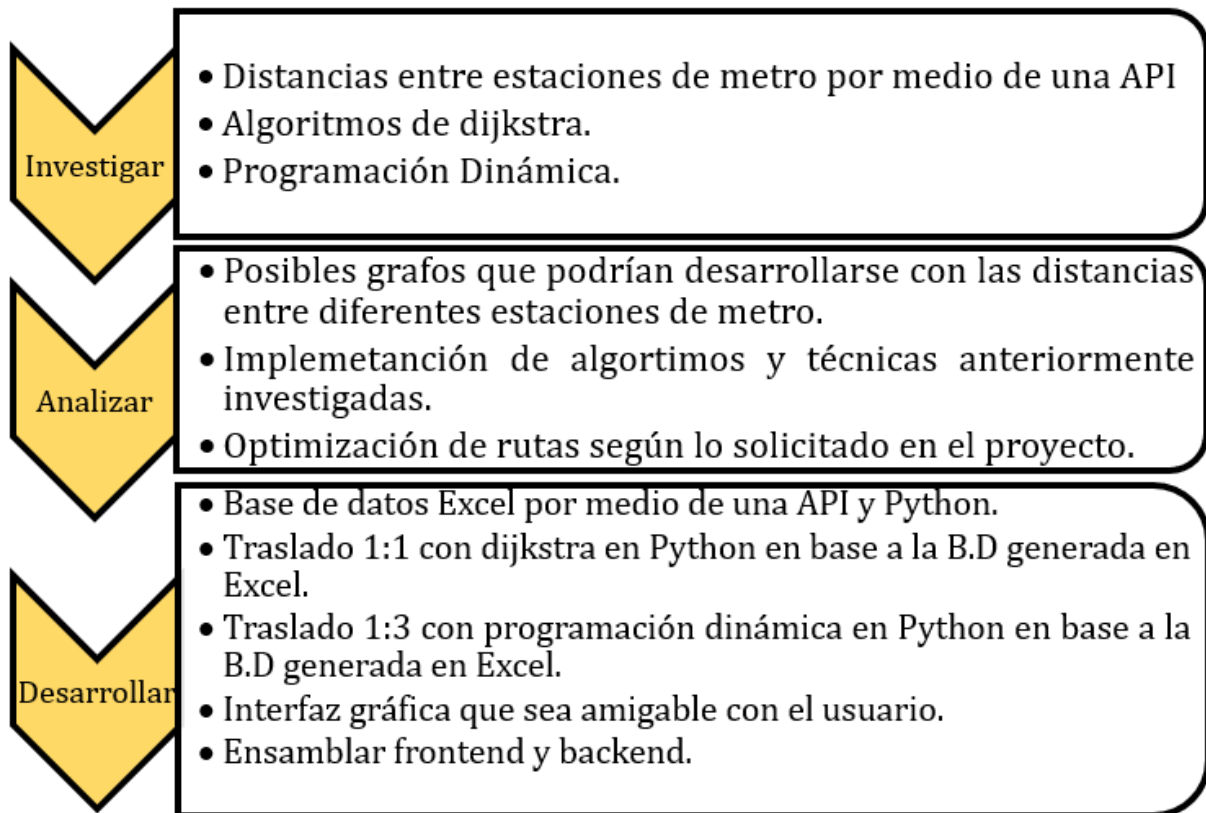


Figure 3: Enfoque de la solución del proyecto.

Para un funcionamiento clave del programa. Se espera que cuente con múltiples herramientas, para que pueda llevar a cabo la optimización de rutas mínimas, es decir, se debe recabar información necesaria para la posterior implementación en el análisis de esta, y por último en el desarrollo.

Considerar las distancias entre estaciones de metro, es fundamental para el proyecto, teniendo en cuenta que estas serán distancias las cuales deberán recorrer los vehículos designados de la empresa para realizar la ruta en la cual deberá entregar la mercadería y reponer stock; y para la factibilidad del programa se usará el algoritmo de dijkstra, el cual tendrá la finalidad de evaluar las rutas mínimas, en base a la distancia entre las diferentes estaciones de metro que existen, a su vez, la técnica que es de suma importancia para el desarrollo. Es la programación dinámica, puesto que ayudará a optimizar la solución del Traslado 1:3.

Por medio de la información recolectada, se deben considerar las estaciones como nodos de un grafo, y las calles entre estaciones como las aristas de este, a su vez, los pesos de este grafo serán las distancias dichas anteriormente, teniendo eso mente, es posible implementar el algoritmo y técnica dicha con anterioridad y optimizar las rutas de acuerdo a lo propuesto.

La fase de desarrollo posee un gran nivel dificultad, debido a que se debe obtener una API para consultar cuáles son las distancias entre todos los metros existentes, y teniendo en cuenta que las estaciones del metro de Santiago son 118, esta es una matriz de 118^2 . Es debido a eso que se deben hacer consultas directamente a la API, para luego configurarla y que esta deposite los resultados en una matriz creada en Excel; luego de esto, viene la parte en la cual se aplica el algoritmo y la técnica, en donde se realizan los ítems necesarios para posteriormente unir la parte funcional con la parte visual del programa.

Método de evaluación de la solución

Con la finalidad de controlar el proceso, se necesitará un estándar el cual se debe cumplir, este será el encargado de medir la solución programada, y con esta medición será posible concluir si el proceso creado se controla, y por consecuencia, será posible obtener la productividad de este.

Es necesario tener en cuenta que los locales de la empresa, están ubicados a las afueras de las estaciones de metro, por lo tanto, la ruta que deberá seguir el conductor. Será la de pasar por diferentes locales de metro de acuerdo a la menor distancia posible que ofrece el grafo creado, es decir, si este desea ir de Plaza de Puente alto a Hospital Sótero del Río, se deberán considerar como ruta las estaciones que estén dentro del recorrido de menor distancia, esto quiere decir que la ruta sería igual a la siguiente imagen:



Figure 4: Ruta de ejemplo.

Es por eso que el estándar a considerar a la hora de probar el programa serán variables obtenidas gracias a Google Maps. Teniendo aquello en mente, se realizarán dos pruebas, en las cuales se analizará la ruta que arroja el sistema satelital en conjunto a la distancia, comparando los resultados obtenidos con los del programa creado. Sin embargo, para el traslado 1:3, se ocupará como estándar las mismas distancias de Google Maps para medir la eficiencia, para comparar las distancias de las tres estaciones más cercanas del software realizado. Para aquella prueba de calidad se harán dos testeos, al igual que en el traslado 1:1.

Al momento de considerar los parámetros anteriormente mencionados, es posible que el resultado sea distinto a lo esperado, y si llega a ser de esta manera, indicaría una falla en el mismo programa, en consecuencia, no podría ser implementado para la empresa. En caso contrario, ambos ítems cumplirían el estándar de calidad, de modo que sería 100% eficaz y productivo aplicar este programa a la empresa.

Marco Teórico

Dentro de la programación hay bastantes variables y funciones que facilita la resolución de problemas, algunos de ellos son los siguientes; los vértices son los elementos que compone un grafo, este puede estar asociado a un valor N que represente el grado de aquel elemento. A la vez los grados dentro de los grafos, son utilizados para conocer la cantidad de conexiones que posee un vértice, por otro lado, lo que permite las conexiones dentro del grafo son las llamadas "aristas", encargadas de unir los vértices de un grafo, puede ser dirigidas (de origen a destino) o no dirigida y existen 3 tipos:

- Aristas adyacentes: Aristas que convergen en el mismo vértice.
- Aristas paralelas: Aristas cuyo si el vértice inicial y final son el mismo.
- Arista cíclicas: Comienza y termina en el mismo vértice.

En programación, también se encuentran los “caminos” que consisten en el conjunto de vértices interconectados a través de una “ruta” de aristas, es decir, el recorrido que se hace para llegar de un nodo X a un nodo Y en un grafo. La cantidad de vértices intermedios determinan “el largo” del camino (distancia entre dos vértices específicos). Dentro de las estructuras básicas de la programación, se encuentra la recursividad que corresponde a ecuaciones en que el valor de la función para un n-ésimo valor determinado está directamente relacionado con uno o más valores inmediatamente anteriores y se dividen en dos tipos:

- Ecuación de primer orden: Son aquellas en que el valor de $T(n)$ depende, principalmente, del valor de $T(n-1)$ – sin que intervengan otros valores de $T(n-x)$.

$$T(n) = aT(n-1) + b_n \quad (1)$$

Siendo $a > 0$ y b_n una función lineal de n .

- Ecuaciones lineales con coeficientes constantes: Son aquellas en que el valor de $T(n)$ depende de varios valores de $T(n-x)$.

$$f_n = Af_{n-1} + Bf_{n-2} + Cf_{n-3} + \dots$$

Donde A, B y C son constantes. Además, se sabe que estas ecuaciones siempre tienen soluciones de la forma: (2)

$$f_n = \lambda^n$$

También existen los ciclos dentro de los grafos y se definen como una sucesión de aristas adyacentes, donde no se recorre dos veces la misma arista, y se regresa al punto inicial. También está el ciclo Hamiltoniano, que es aquel que recorre todos los vértices del grafo exactamente una vez.

Una de las estructuras más conocidas dentro de la informática son los grafos, aquellas contienen una estructura de tipo $G=(V,E)$ y corresponde a una pareja ordenada en la que V es un conjunto no vacío de vértices y E es un conjunto de aristas. Una forma de representar los grafos es mediante un arreglo de listas, donde cada lista corresponde a un nodo V y sus

elementos son los nodos adyacentes a V , para así comenzar una búsqueda en amplitud y los caminos “cortos”, aplicando la recursividad en los vecinos de los nodos con el número 1 y se marca con $k+1$ los nodos no marcados y que sean vecinos de vértices k .

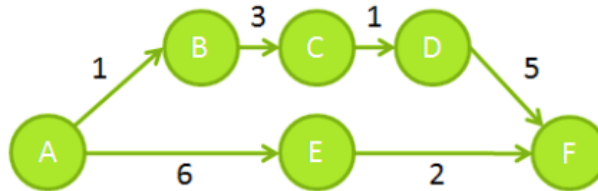
Por otro lado la llamada “Teoría de los grafos” pertenece a una rama de las matemáticas y las ciencias de la computación que se dedica a estudiar las propiedades de los grafos, es así como se distinguen distintos tipos de grafos:

- Grafo no dirigido: Es aquel grafo donde las aristas no tienen un sentido marcado de punto de inicio y punto de fin.
- Grafo dirigido o Digrafo: Es un grafo con dirección asignada en sus aristas.



Figure 5: Ejemplo de grafo dirigido y no dirigido.

- Grafo ponderados: Es aquel que posee un número en cada arista, llamado valuación, ponderación o coste según el contexto, y sirve para determinar “algún aspecto” de estudio sobre el grafo. Formalmente, es un grafo con una función $v: A \rightarrow \mathbb{R}^+$.



Peso del camino A,C: $W_{ab} + W_{bc} = 4$

Camino más corto: A, F

Mayor Peso camino más corto: $W_{ab} + W_{bc} + W_{cd} + W_{df} = 10$

Menor Peso camino más corto : $W_{ae} + W_{ef} = 8$

Figure 6: Ejemplo de un grafo ponderado.

- Grafo simple: Es aquel donde al menos una arista une dos vértices cualesquiera. No existe “loop” ni más de una arista entre dos vértices.

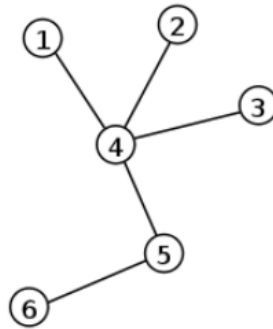


Figure 7: Ejemplo de grafo simple.

- Grafo conexo: Es aquel donde cada par de vértices está conectado por un camino.

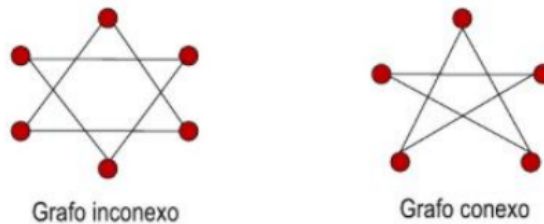


Figure 8: Ejemplo de grafo inconexo y conexo.

- Grafo completo: Es aquel donde todos los pares de vértices tienen una arista que los une. Posee $n(n-1)/2$ aristas en función de sus n vértices.

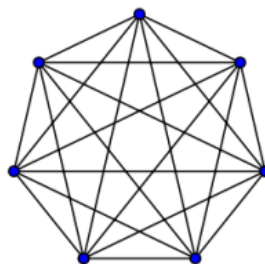


Figure 9: Ejemplo de grafo completo.

Una de las funciones que permite optimizar la programación de soluciones, son los algoritmos, ya que son un conjunto ordenado de operaciones sistemáticas que permite hacer un cálculo y hallar la solución de un tipo de problemas. Existen diversos tipos de algoritmos, pero para hallar la solución de la problemática se desarrollará el algoritmo de Dijkstra o también llamado algoritmo de caminos mínimos, dada su función de encontrar la ruta más corta – de menor peso – en un grafo desde un vértice A hasta un vértice B pasando por “n” vértices intermedios, donde $n > 0$, y cada arista tiene un “peso” asociado $p > 0$. Las aristas pueden ser “dirigidas” (digrafo).

Metodológicamente dentro de la informática, se halla la programación dinámica, usado principalmente para reducir el tiempo de ejecución de un algoritmo mediante la resolución de los subproblemas una sola vez y se guardan las soluciones (por ejemplo en un vector) para una futura utilización.

Por último, las librerías son una parte importante del código debido a que desde allí se obtienen más codificaciones, que podrían ser utilizadas para la solución, algunas de ellas son:

- request: Permite hacer consultas por medio de un url.
- tkinter: Permite la proyección en pantalla, para el ingreso de información y así comprobar el funcionamiento (interfaz).
- xlrd: Permite leer el excel ingresado.
- xlwt: Permite escribir en el excel ingresado.

WBS

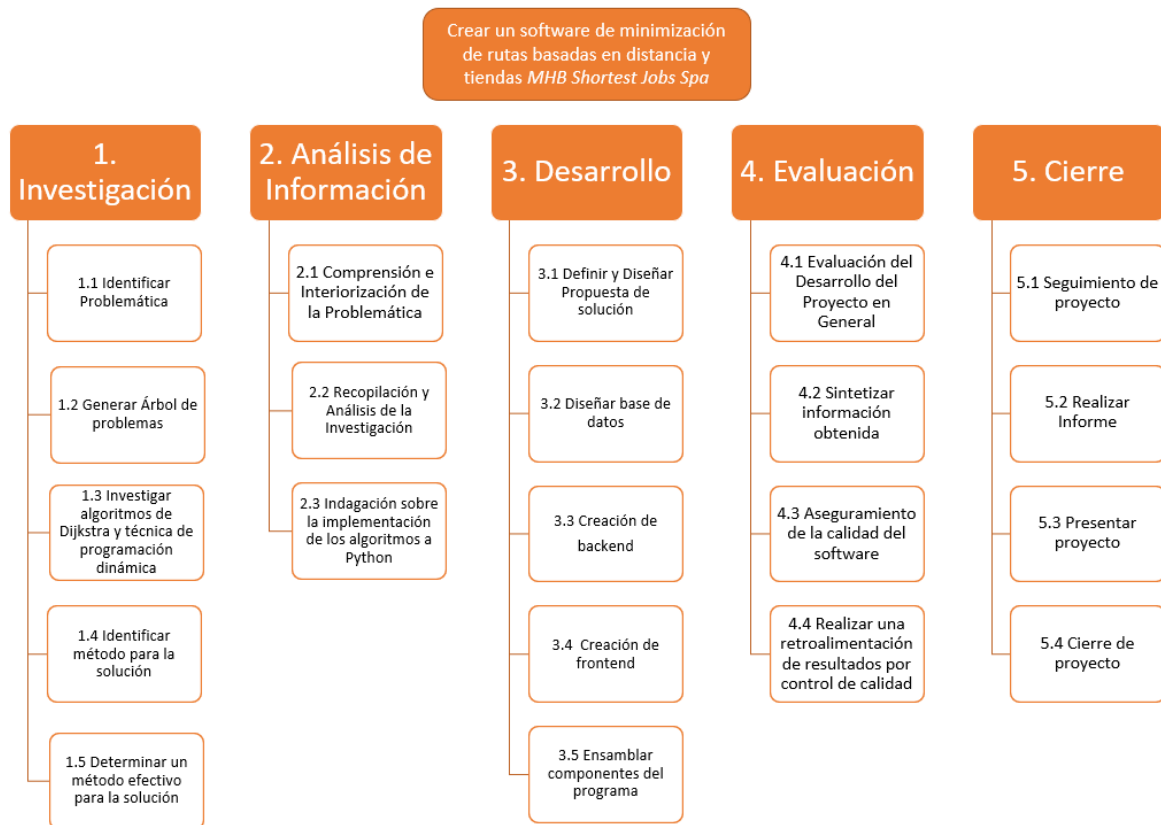


Figure 10: Diagrama del Work Breakdown Structure

Descripción de roles

Para un desempeño correcto del proyecto, se necesita distribuir y asignar roles a cada participante con el fin de optimizar tiempo en relación a las tareas a cumplir, para ello se utilizará una Matriz, en la cual se realizará la correspondiente asignación de roles con respecto a cada una de las tareas a realizar.

En la siguiente matriz (1) se describe la sigla que identifica a cada uno de los integrantes del grupo:

Nombre del integrante	Sigla
Miguel Jaime	M.J
Ignacio Jara	I.J
Catalina Ledezma	C.L
David Huichacura	D.H
Oscar Horta	O.H

Matriz 1 : Asignación de Siglas a cada integrante del grupo.

Ahora, en la matriz (2), se describe la función que se le asigna a cada uno de los roles que se involucraran en el desarrollo del presente proyecto:

Sigla	Rol	Descripción
R	Responsable	Será el responsable de que todo el proyecto se ejecute de manera correcta y eficiente.
E	Encargado	Será el encargado de que la actividad se realice según lo solicitado.
C	Consultado	Será el que proporciona consejos o información para la actividad solicitada.
A	Apoyo	Será el que brinde apoyo en cualquier circunstancia que lo amerite.
I	Informado	Será el que deba estar informado sobre el progreso que se lleva en la actividad y de las decisiones que se tomen durante el proceso.

Matriz 2 : Descripción de Roles requeridos en el desarrollo del Proyecto

Asignación de roles

En la matriz (3) se le asigna un respectivo rol a cada integrante del grupo en torno a una actividad específica, esta asignación se realiza de una manera equitativa y considerando las habilidades de cada uno de los integrantes del grupo. Todo esto con el fin de llevar adelante de manera eficiente y optimizada el desarrollo del presente Proyecto:

ACTIVIDAD	M.J	I.J	C.L	D.H	O.H
1.Investigación	R				
1.1 Identificar Problemática.	E	C	E	A	I
1.2 Generar Árbol de problemas.	E	R	C	I	E
1.3 Investigar algoritmos de Dijkstra y técnica de programación dinámica.	A	I	R	E	R
1.4 Identificar método para la solución.	C	E	C	A	I
1.5 Determinar un método efectivo para la solución.	E	E	R	I	I
2.Análisis de Información		R			
2.1 Comprensión e Interiorización de la Problemática	E	E	C	A	I
2.2 Recopilación y Análisis de la Investigación	R	C	E	I	E
2.3 Indagación sobre la implementación de los algoritmos a Python.	A	I	E	R	E
3.Desarrollo					R
3.1 Definir y Diseñar Propuesta de solución	R	E	C	I	I
3.2 Diseñar base de datos.	A	I	C	R	E
3.3 Creación de backend.	R	E	I	A	E
3.4 Creación frontend.	A	E	E	C	R
3.5 Ensamblar componentes del programa.					
4.Evaluación				R	
4.1 Evaluación del Desarrollo del Proyecto en General.	I	E	A	C	I
4.2 Sintetizar la información obtenida.	A	R	E	I	I
4.3 Aseguramiento de la calidad del software.	E	C	C	R	E
4.4 Realizar una retroalimentación de resultados por control de calidad.	E	C	E	C	E
5.Cierre			R		
5.1 Seguimiento de proyecto.	C	A	C	R	A
5.2 Realizar informe.	A	C	C	E	R
5.3 Presentar proyecto.	E	C	E	I	E
5.4 Cierre proyecto.	R	I	I	C	E

Matriz 3: Asignación de roles a cada integrante del grupo con respecto a cada actividad a desarrollar.

Estimación de tiempo y esfuerzo

Para poder lograr un trabajo eficaz, se realizará una asignación de horas trabajadas por día en HH durante los meses que se desarrolle el proyecto. Esto conlleva a que el equipo de ingenieros logre realizar un trabajo equitativo, aproveche los tiempos y logre un resultado de trabajo óptimo.

FASE	ACTIVIDAD	Horas de Trabajo por Día
1.Investigación	1.1 Identificar Problemática.	2 horas
	1.2 Generar Árbol de problemas.	2 horas
	1.3 Investigar algoritmos de Dijkstra y técnica de programación dinámica.	2 horas
	1.4 Identificar método para la solución.	1 hora
	1.5 Determinar un método efectivo para la solución.	1 hora
2.Análisis	2.1 Comprensión e Interiorización de la Problemática.	3 horas
	2.2 Recopilación y Análisis de la Investigación	3 horas
	2.3 Indagación sobre la implementación de los algoritmos a Python.	2 horas
3.Desarrollo	3.1 Definir y Diseñar Propuesta de solución	2 horas
	3.2 Diseñar base de datos	5 horas
	3.3 Creación de backend.	4 horas
	3.4 Creación frontend.	3 horas
	3.5 Ensamblar componentes del programa.	3 horas
4.Evaluación	4.1 Evaluación del Desarrollo del Proyecto en General.	2 horas
	4.2 Sintetizar la información obtenida.	3 horas
	4.3 Aseguramiento de la calidad del software.	3 horas
	4.4 Realizar una retroalimentación de resultados por control de calidad.	4 horas
5.Cierre	5.1 Seguimiento de proyecto	3 horas
	5.2 Realizar informe.	7 horas
	5.3 Presentar proyecto.	2 horas
	5.4 Cierre proyecto.	2 horas

Matriz 4: "Estimación de tiempo de esfuerzo".

Carta Gantt

En la presente Carta Gantt se detallan los tiempos planificados en que se van a desarrollar cada una de las actividades del proyecto.

Optimizar el flujo de mercadería en locales de la empresa MHB Shortest Jobs SPA por medio de un sistema de rutas mínimas.			MES	AGOSTO				SEPTIEMBRE				OCTUBRE				NOVIEMBRE			
			SEMANA	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
ACTIVIDAD	Fecha Inicio	Fecha Termino																	
1. Investigación																			
1.1 Identificar Problemática.	15-08-2021	19-08-2021				X													
1.2 Generar Árbol de problemas.	16-08-2021	20-08-2021				X													
1.3 Investigar algoritmos de Dijkstra y técnica de programación dinámica.	16-08-2021	21-08-2021				X													
1.4 Identificar método para la solución.	22-08-2021	25-08-2021					X												
1.5 Determinar un método efectivo para la solución.	25-08-2021	28-08-2021					X												
2. Análisis de Información																			
2.1 Comprensión e Interiorización de la Problemática	31-08-2021	04-09-2021					X	X											
2.2 Recopilación y Análisis de la Investigación	01-09-2021	05-09-2021						X											
2.3 Indagación sobre la implementación de los algoritmos a Python.	02-09-2021	05-09-2021						X											
3. Desarrollo																			
3.1 Definir y Diseñar: Propuesta de solución	12-09-2021	18-09-2021							X	X									
3.2 Diseñar base de datos.	13-09-2021	19-09-2021							X	X									
3.3 Creación de backend.	01-10-2021	14-10-2021										X	X						
3.4 Creación frontend.	01-10-2021	14-10-2021										X	X						
3.5 Ensamblar componentes del programa.	15-10-2021	31-10-2021												X	X				
4. Evaluación																			
4.1 Evaluación del Desarrollo del Proyecto en General.	01-11-2021	07-11-2021														X			
4.2 Sintetizar la información obtenida.	02-11-2021	07-11-2021														X			
4.3 Aseguramiento de la calidad del software.	03-11-2021	07-11-2021														X			
4.4 Realizar una retroalimentación de resultados por control de calidad.	01-11-2021	14-11-2021														X	X		
5. Cierre																			
5.1 Seguimiento de proyecto.	01-09-2021	07-11-2021						X				X				X			
5.2 Realizar informe.	01-11-2021	14-11-2021														X	X		
5.3 Presentar proyecto.	15-11-2021	15-11-2021																X	
5.4 Cierre proyecto.	15-11-2021	15-11-2021																X	

Matriz 5: "Carta Gantt".

Construcción de la solución propuesta (MJ Y DH)

Aplica correctamente los conceptos explicados en el marco teórico, desarrollando la solución propuesta en la Introducción. Entrega el producto/prototipo o entregable asociado al proyecto.

para el correcto desarrollo de la solución propuesta, en primer lugar se definió que la estructura la cual representa las distancias entre puntos, será un grafo. para la obtención de este grafo se realizó una base de datos en excel la cual es una matriz que su contenido serán las distancias mínimas entre dos puntos. para el correcto llenado de esta matriz y obtención del grafo se realizó lo siguiente:

Se utilizó la api de google maps para obtener la distancia de un punto origen a un punto destino como se muestra en la figura.

```
Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)]
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\david\Documents\Proyecto Algoritmo\proyecto-algoritmos\proyecto\A
{
  "destination_addresses" : [ "Neptuno, Lo Prado, Región Metropolitana, Chile" ],
  "origin_addresses" : [ "San Pablo, Lo Prado, Región Metropolitana, Chile" ],
  "rows" : [
    {
      "elements" : [
        {
          "distance" : {
            "text" : "0.8 km",
            "value" : 830
          },
          "duration" : {
            "text" : "2 mins",
            "value" : 120
          },
          "status" : "OK"
        }
      ]
    }
  ],
  "status" : "OK"
}
```

Código 1: Código que demuestra la obtención de resultados.

Como se ve en la figura sale un row que contiene sus elementos de la consulta teniendo consigo la distancia y duración para armar el grafo, sin embargo solo se necesita la distancia que está contenida en el índice distance que contiene un índice text que contiene su distancia en kilómetros. Para ello se transforma esa iteración en un diccionario con el

comando "json_response" y preguntando sus llaves "row","elements","distance","text", cada uno está la posición 0 del uno del otro. Su iteración es la siguiente:

```
distancia = json_response["rows"][0]["elements"][0]["distance"]["text"]
```

Por otro lado en un archivo de texto llamando "estaciones.txt" se encuentra todas las estaciones de metro en la cual se recoge las estaciones y se transforma en una lista con el comando read() y split() haciendo que los saltos de líneas sean una coma.

Dado a lo anterior, se recorre la lista de estaciones con doble for para llevar a cabo las consultas de api donde un for recorre en "i" y el otro en "j" y se obtiene la distancia de cada iteración de nxn según el largo de la lista y se lleva a una función "Grafo" del archivo "SaveExce.py" que entra origen (i) y destino (j) con su respectiva distancia. Se muestra en la siguiente imagen el algoritmo utilizado:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
##proyecto inicio 05-11-2021

import requests
import SaveExcel as SE

archivo = open("estaciones.txt",encoding ="utf8")

datos = archivo.read().replace(" ","+")
datos = datos.lower().split("\n")

for i in range(0,len(datos)):
    for j in range(0,len(datos)):
        if datos[i]==datos[j]:#en el caso que las estaciones sean iguales se ingrese cero
            SE.Grafo(datos[i],datos[j],0)
        else:#en el caso que sean distintas
            url = ("https://maps.googleapis.com/maps/api/distancematrix/json?origins=Metro%metro+"
                  +str(datos[i])+",region+metropolitana+"&destinations=Metro%metro+"
                  +str(datos[j])+",region+metropolitana+"&key=AIzaSyD_VKKZ2ZyZIJ9DaBto4IpK2Z8lj2HInJ0")
            response = requests.get(url)
            json_response = response.json()
            distancia = json_response["rows"][0]["elements"][0]["distance"]["text"]
            distancia = distancia.replace(" km","")
            SE.Grafo(datos[i],datos[j],distancia)
```

Código 2: "Algoritmo de búsqueda de distancia en una api".

En lo consiguiente, el proceso que se lleva a cabo es la creación de un excel que contenga las distancia con sus respectivos nodos que son las estaciones del metro, donde se recoge sus datos de cada línea a una lista y esta se agrega a una lista llamada grafo, en la cual se utiliza para almacenar las información y recibir los datos entregados por la api y devolverlos al archivo "Grafo.xls" el proceso se muestra en la siguiente figura:

```

mante','santa+isabel','rodrigo+de+araya','carlos+valdivinos','camino+agrigola','santa+joaquin','pedrero+','mirador+','bellavista+de+la+florida','cerrillos','lo+valledor','pres
idente+pedro+aguirre+cerda','bio+bio','estadio+nacional','ines+de+suarez','ñuble']]
[[['NA','sant+pablo','neptuno','pajaritos','las+rejas','ecuador','santalberto+hurtado','universidad+de+santiago','estacion+central','u.l.a.','republica','los+heroes','lamoned
a','universidad+de+chile','santa+lucia','universidad+catolica','baquedano','salvador','manuel+montt','pedro+de+valdivia','los+leones','tobalaba','el+golf','alcantara','escu
ela+militar','manguhue','fernando+de+maggallanes','los+dominicos','la+cisterna','el+parron','lo+valle','ciudad+del+niño','departamental','lo+via','san+miguel','el+lilano','
franklin','parque+ohiggins','toesca','santa+ana','puente+cal+y+canto','patronato','cerro+blanco','cementerios','einstein','dorsal','zapadores','vespucio+morte
','los+libertadores','cardenal+caro','vivaceta','conchali','plaza+chacabuco','hospitales','plazade+armas','parque+almagro','matta','irarrazaval','monseñor+y+zaguirre','Run
oa','chile+españa','villa+frei','plaza+egaña','fernando+castillo+velasco','cristobal+colon','francisco+bilbao','principe+de+gales','simon+bollivar','los+orientales','greCIA','
los+presidentes','quillin','las+torres','macul','vicuña+mackenna','vicente+valdes','rojas+maggallanes','trinidad','sant+jose+de+la+estrella','los+guillaves','elisa+correa','ho
spital+sotero+del+rio','protectora+de+la+infancia','plaza+mercedes','plaza+de+pueblo+alto','santa+julia','la+granja','santa+rosa','san+ramon','plaza+de+mapiu','santiago+puertas',
'del+sol','monte+tabor','las+parcelas','laguna+sur','barrancas','pudahuel','lo+prado','blanqueado','gruta+de+lourdes','quinta+normal','cumming','bellas+artes','parque+busta
mante','santa+isabel','rodrigo+de+araya','carlos+valdivinos','camino+agrigola','santa+joaquin','pedrero+','mirador+','bellavista+de+la+florida','cerrillos','lo+valledor','pres
idente+pedro+aguirre+cerda','bio+bio','estadio+nacional','ines+de+suarez','ñuble','sant+pablo',0.0,'0.0','1.9','2.9','3.6','4.3','4.9','5.5','5.4','6.1','6.5','7.2',
'7.6','8.2','8.9','9.4','9.5','14.2','13.8','14.3','15.0','16.4','18.4','19.2','19.0','19.9','23.0','17.0','18.5','13.7','12.8','12.3','12.9','12.1','10.6',
'9.3','8.5','7.4','6.4','9.9','8.2','9.1','10.2','11.3','12.6','18.2','16.8','12.6','13.3','12.1','10.1','9.8','8.5','7.4','8.2','10.0','10.6','12.4','18.2',
'18.0','33.4','32.6','33.8','16.2','17.2','21.1','21.9','31.5','33.0','29.7','28.6','27.7','26.1','25.0','24.5','24.6','27.0','27.9','26.7','29.6','28.8','24.3','
'31.2','31.7','24.2','23.2','19.9','19.1','16.0','14.7','12.6','6.4','5.5','3.9','3.3','2.0','1.7','1.6','3.1','4.7','5.7','8.0','10.1','11.1','14.8','14.3','14',
'.7','15.5','26.5','25.6','24.7','6.2','9.1','9.8','12.4','15.3','15.8','12.3','7.7']]
[[['NA','sant+pablo','neptuno','pajaritos','las+rejas','ecuador','santalberto+hurtado','universidad+de+santiago','estacion+central','u.l.a.','republica','los+heroes','lamoned

```

Figure 11: Construcción del grafo.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																												
1	NA	sant+pablo	neptuno	pajaritos	las+rejas	ecuador	santalberto+hurtado	universidad+de+santiago	estacion+central	u.l.a.	republica	los+heroes	lamoned	a	universidad+de+chile	santa+lucia	universidad+catolica	baquedano	salvador	manuel+montt	pedro+de+valdivia	los+leones	tobalaba	el+golf	alcantara	escuela+militar	manguhue	fernando+de+maggallanes	los+dominicos	la+cisterna	el+parron	lo+valle	ciudad+del+niño	departamental	lo+via	san+miguel	el+lilano	franklin	parque+ohiggins	toesca	santa+ana	puente+cal+y+canto	patronato	cerro+blanco	cementerios	einstein	dorsal	zapadores	vespucio+morte	los+libertadores	cardenal+caro	vivaceta	conchali	plaza+chacabuco	hospitales	plazade+armas	parque+almagro	matta	irarrazaval	monseñor+y+zaguirre	Run	oa	chile+españa	villa+frei	plaza+egaña	fernando+castillo+velasco	cristobal+colon	francisco+bilbao	principe+de+gales	simon+bollivar	los+orientales	greCIA	los+presidentes	quillin	las+torres	macul	vicuña+mackenna	vicente+valdes	rojas+maggallanes	trinidad	sant+jose+de+la+estrella	los+guillaves	elisa+correa	hospital+sotero+del+rio	protectora+de+la+infancia	plaza+mercedes	plaza+de+pueblo+alto	santa+julia	la+granja	santa+rosa	san+ramon	plaza+de+mapiu	santiago+puertas	del+sol	monte+tabor	las+parcelas	laguna+sur	barrancas	pudahuel	lo+prado	blanqueado	gruta+de+lourdes	quinta+normal	cumming	bellas+artes	parque+bustamante	santa+isabel	rodrigo+de+araya	carlos+valdivinos	camino+agrigola	santa+joaquin	pedrero	mirador	bellavista+de+la+florida	cerrillos	lo+valledor	presidente+pedro+aguirre+cerda	bio+bio	estadio+nacional	ines+de+suarez	ñuble	sant+pablo	0.0	0.0	1.9	2.9	3.6	4.3	4.9	5.5	5.4	6.1	6.5	7.2	7.6	8.2	8.9	9.4	9.5	14.2	13.8	14.3	15.0	16.4	18.4	19.2	19.0	19.9	23.0	17.0	18.5	13.7	12.8	12.3	12.9	12.1	10.6	9.3	8.5	7.4	6.4	9.9	8.2	9.1	10.2	11.3	12.6	18.2	16.8	12.6	13.3	12.1	10.1	9.8	8.5	7.4	8.2	10.0	10.6	12.4	18.2	18.0	33.4	32.6	33.8	16.2	17.2	21.1	21.9	31.5	33.0	29.7	28.6	27.7	26.1	25.0	24.5	24.6	27.0	27.9	26.7	29.6	28.8	24.3	31.2	31.7	24.2	23.2	19.9	19.1	16.0	14.7	12.6	6.4	5.5	3.9	3.3	2.0	1.7	1.6	3.1	4.7	5.7	8.0	10.1	11.1	14.8	14.3	14.7	15.5	26.5	25.6	24.7	6.2	9.1	9.8	12.4	15.3	15.8	12.3	7.7																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									
2	sant+pablo	0	0.8	1.9	2.9	3.6	4.3	4.9	5.5	6.4	6.1	6.5	7.2	7.6	8.2	8.9	9.4	9.5	14.2	13.8	14.3	15.0	16.4	18.4	19.2	19.0	19.9	23.0	17.0	18.5	13.7	12.8	12.3	12.9	12.1	10.6	9.3	8.5	7.4	6.4	9.9	8.2	9.1	10.2	11.3	12.6	18.2	16.8	12.6	13.3	12.1	10.1	9.8	8.5	7.4	8.2	10.0	10.6	12.4	18.2	18.0	33.4	32.6	33.8	16.2	17.2	21.1	21.9	31.5	33.0	29.7	28.6	27.7	26.1	25.0	24.5	24.6	27.0	27.9	26.7	29.6	28.8	24.3	31.2	31.7	24.2	23.2	19.9	19.1	16.0	14.7	12.6	6.4	5.5	3.9	3.3	2.0	1.7	1.6	3.1	4.7	5.7	8.0	10.1	11.1	14.8	14.3	14.7	15.5	26.5	25.6	24.7	6.2	9.1	9.8	12.4	15.3	15.8	12.3	7.7																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																		
3	neptuno	1.9	0	1.1	2.1	2.7	3.4	4.0	4.7	5.3	5.9	6.4	7.1	7.5	8.0	8.5	9.3	10.4	14.9	14.5	15.1	15.7	17.1	18.1	18.9	19.7	20.6	23.7	21.6	23.1	17.6	16.1	16.6	17.2	17.8	18.3	18.8	19.3	19.8	20.3	20.8	21.3	21.8	22.3	22.8	23.3	23.8	24.3	24.8	25.3	25.8	26.3	26.8	27.3	27.8	28.3	28.8	29.3	29.8	30.3	30.8	31.3	31.8	32.3	32.8	33.3	33.8	34.3	34.8	35.3	35.8	36.3	36.8	37.3	37.8	38.3	38.8	39.3	39.8	40.3	40.8	41.3	41.8	42.3	42.8	43.3	43.8	44.3	44.8	45.3	45.8	46.3	46.8	47.3	47.8	48.3	48.8	49.3	49.8	50.3	50.8	51.3	51.8	52.3	52.8	53.3	53.8	54.3	54.8	55.3	55.8	56.3	56.8	57.3	57.8	58.3	58.8	59.3	59.8	60.3	60.8	61.3	61.8	62.3	62.8	63.3	63.8	64.3	64.8	65.3	65.8	66.3	66.8	67.3	67.8	68.3	68.8	69.3	69.8	70.3	70.8	71.3	71.8	72.3	72.8	73.3	73.8	74.3	74.8	75.3	75.8	76.3	76.8	77.3	77.8	78.3	78.8	79.3	79.8	80.3	80.8	81.3	81.8	82.3	82.8	83.3	83.8	84.3	84.8	85.3	85.8	86.3	86.8	87.3	87.8	88.3	88.8	89.3	89.8	90.3	90.8	91.3	91.8	92.3	92.8	93.3	93.8	94.3	94.8	95.3	95.8	96.3	96.8	97.3	97.8	98.3	98.8	99.3	99.8	100.3	100.8	101.3	101.8	102.3	102.8	103.3	103.8	104.3	104.8	105.3	105.8	106.3	106.8	107.3	107.8	108.3	108.8	109.3	109.8	110.3	110.8	111.3	111.8	112.3	112.8	113.3	113.8	114.3	114.8	115.3	115.8	116.3	116.8	117.3	117.8	118.3	118.8	119.3	119.8	120.3	120.8	121.3	121.8	122.3	122.8	123.3	123.8	124.3	124.8	125.3	125.8	126.3	126.8	127.3	127.8	128.3	128.8	129.3	129.8	130.3	130.8	131.3	131.8	132.3	132.8	133.3	133.8	134.3	134.8	135.3	135.8	136.3	136.8	137.3	137.8	138.3	138.8	139.3	139.8	140.3	140.8	141.3	141.8	142.3	142.8	143.3	143.8	144.3	144.8	145.3	145.8	146.3	146.8	147.3	147.8	148.3	148.8	149.3	149.8	150.3	150.8	151.3	151.8	152.3	152.8	153.3	153.8	154.3	154.8	155.3	155.8	156.3	156.8	157.3	157.8	158.3	158.8	159.3	159.8	160.3	160.8	161.3	161.8	162.3	162.8	163.3	163.8	164.3	164.8	165.3	165.8	166.3	166.8	167.3	167.8	168.3	168.8	169.3	169.8	170.3	170.8	171.3	171.8	172.3	172.8	173.3	173.8	174.3	174.8	175.3	175.8	176.3	176.8	177.3	177.8	178.3	178.8	179.3	179.8	180.3	180.8	181.3	181.8	182.3	182.8	183.3	183.8	184.3	184.8	185.3	185.8	186.3	186.8	187.3	187.8	188.3	188.8	189.3	189.8	190.3	190.8	191.3	191.8	192.3	192.8	193.3	193.8	194.3	194.8	195.3	195.8	196.3	196.8	197.3	197.8	198.3	198.8	199.3	199.8	200.3	200.8	201.3	201.8	202.3	202.8	203.3	203.8	204.3	204.8	205.3	205.8	206.3	206.8	207.3	207.8	208.3	208.8	209.3	209.8	210.3	210.8	211.3	211.8	212.3	212.8	213.3	213.8	214.3	214.8	215.3	215.8	216.3	216.8	217.3	217.8	218.3	218.8	219.3	219.8	220.3	220.8	221.3	221.8	222.3	222.8	223.3	223.8	224.3	224.8	225.3	225.8	226.3	226.8	227.3	227.8	228.3	228.8	229.3	229.8	230.3	230.8	231.3	231.8	232.3	232.8	233.3	233.8	234.3	234.8	235.3	235.8	236.3	236.8	237.3	237.8	238.3	238.8	239.3	239.8	240.3	240.8	241.3	241.8	242.3	242.8	243.3	243.8	244.3	244.8	245.3	245.8	246.3	246.8	247.3	247.8	248.3	248.8	249.3	249.8	250.3	250.8	251.3	251.8	252.3	252.8	253.3	253.8	254.3	254.8	255.3	255.8	256.3	256.8	257.3	257.8	258.3	258.8	259.3	259.8	260.3	260.8	261.3	261.8	262.3	262.8	263.3	263.8	264.3	264.8	265.3	265.8	266.3	266.8	267.3	267.8	268.3	268.8	269.3	269.8	270.3	270.8	271.3	271.8	272.3	272.8	273.3	273.8	274.3	274.8	275.3	275.8	276.3	276.8	277.3	277.8	278.3	278.8	279.3	279.8	280.3	280.8	281.3	281.8	282.3	282.8	283.3	283.8	284.3	284.8	285.3	285.8	286.3	286.8	287.3	287.8	288.3	288.8	289.3	289.8	290.3	290.8	291.3	291.8	292.3	292.8	293.3	293.8	294.3	294.8	295.3	295.8	296.3	296.8	297.3	297.8	298.3	298.8	299.3	299.8	300.3	300.8	301.3	301.8	302.3	302.8	303.3	303.8	304.3	304.8	305.3	305.8	306.3	306.8	307.3	307.8	308.3	308.8	309.3	309.8	310.3	310.8	311.3	311.8	312.3	312.8	313.3	313.8	314.3	314.8	315.3	315.8	316.3	316.8	317.3	317.8	318.3	318.8	319.3	319.8	320.3	320.8	321.3	321.8	322.3	322.8	323.3	323.8	324.3	324.8	325.3	325.8	326.3	326.8	327.3	327.8	328.3	328.8	329.3	329.8	330.3	330.8	331.3	331.8	332.3	332.8	333.3	333.8	334.3	334.8	335.3	335.8	336.3	336.8	337.3	337.8	338.3	338.8	339.3	339.8	340.3	340.8	341.3	341.8	342.3	342.8	343.3	343.8	344.3	344.8	345.3	345.8	346.3	346.8	347.3	347.8	348.3	348.8	349.3	349.8	350.3	350.8	351.3	351.8	352.3	352.8	353.3	353.8	354.3	354.8	355.3	355.8	356.3	356.8	357.3	357.8	358.3	358.8	359.3	359.8	360.3	360.8	361.3	361.8	362.3	362.8	363.3	363.8	364.3	364.8	365.3	365.8	366.3	366.8	367.3	367.8	368.3	368.8	369.3	369.8	370.3	370.8	371.3	371.8	372.3	372.8	373.3	373.8	374.3	374.8	375.3	375.8	376.3	376.8	377.3	377.8	378.3	378.8	379.3	379.8	380.3	380.8	381.3	381.8	382.3	382.8	383.3	383.8	384.3	384.8	385.3	385.8	386.3	386.8	387.3	387.8	388.3	388.8	389.3	389.8	390.3	390.8	391.3	391.8	392.3	392.8	393.3	393.8	394.3	394.8	395.3	395.8	396.3	396.8	397.3	397.8	398.3	398.8	399.3	399.8	400.3	400.8	401.3	401.8	402.3	402.8	403.3	403.8	404.3	404.8	405.3	405.8	406.3	406.8	407.3	407.8	408.3	408.8	409.3	409.8	410.3	410.8	411.3	411.8	412.3	412.8	413.3	413.8	414.3	414.8	415.3	415.8	416.3	416.8	417.3	417.8	418.3	418.8	419.3	419.8	420.3	420.8	421.3	421.8	422.3	422.8	423.3	423.8	424.3	424.8	425.3	425.8	426.3	426.8	427.3	427.8	428.3	428.8	429.3	429.8	430.3	430.8	431.3	431.8	432.3	432.8	433.3	433.8	434.3	434.8	435.3	435.8	436.3	436.8	437.3	437.8	438.3	438.8	439.3	439.8	440.3	440.8	441.3	441.8	442.3	442.8	443.3	443.8	444.3	444.8	445.3	445.8	446.3	446.8	447.3	447.8	448.3	448.8	449.3	449.8	450.3	450.8	451.3	451.8	452.3	452.8	453.3	453.8	454.3	454.8	455.3	455.8	456.3	456.8	457.3	457.8	458.3	458.8	459.3	459.8	460.3	460.8	

```

import xlwt
import xlrd

def Grafo(origen,destino,distancia):
    documento = xlwt.Workbook(encoding="uft-8") #se lee el doc
    sheet = documento.add_sheet("Distancia",cell_overwrite_ok=True) #se toma la hija distancia para escribir
    style = xlwt.XFStyle()

    libro = xlrd.open_workbook("Grafo.xls") #se abre el excel para leer
    sh = libro.sheet_by_name("Distancia") #se toma la hoja distancia para leer
    nrows = sh.nrows #largo filas
    ncols = sh.ncols #largo columna
    vertice = [] #este para las filas
    grafo = [] #este es el grafo

    if nrows==0 and ncols==0: #si el largo de la fila y columna es 0
        if origen==destino: #en el caso que el origen y el destino sean iguales
            sheet.write(nrows,nrows,"NA") #Se agrega un NA valor nulo a la celda 0,0
            sheet.write(nrows+1,nrows,origen) #columna
            sheet.write(nrows,nrows+1,destino) #Fila
            sheet.write(nrows+1,nrows+1,distancia) #Agrega la distancia
            documento.save("Grafo.xls") #se guarda

        else:
            for i in range(0,nrows): #se recorre hasta el largo de la fila
                for j in range(0,ncols): #hasta el largo de la columna
                    vertice.append(sh.cell_value(i,j)) #se extrae los valores con sus vertices para ser agregadas al grafo
                    if "" in vertice: #en el caso que no exista un valor en la celda y es agregada a la lista
                        vertice.remove("") #esta se borra
            grafo.append(vertice) #Toma la lista vertice y lo agrega al grafo
            vertice = [] #Se resetea el vertice para agrega los siguientes
            print(grafo)

```

Código 3: “grafo parte 1 agregado a excel y creación de grafo”

```

n = 0
seguir = True
if origen!=destino: #en el caso que sean distintos el origen y el destino
    largo = len(grafo) #se toma el largo del grafo
    while seguir: #seguir hasta que sea Falso
        n +=1 #contador
        if origen in grafo[n] and destino in grafo[0]: #si el origen existe en el grafo y el destino en la posicion 0
            grafo[n].append(distancia) #se agrega al grafo su distancia
            seguir = False
        elif origen in grafo[n] and destino not in grafo[0]: #en el caso que no exista el destino en la posicion 0 y si el ori
            grafo[0].append(destino) #se agrega el destino
            grafo[n].append(distancia) #y su distancia
            seguir = False
        elif origen not in grafo[largo-1]: #en el caso que el origen no exista en el grafo en la ultima posicion en la cual se
            grafo.append([origen]) #se agrega el origen
            grafo[largo].append(distancia) #y su distancia ya que el destino ya existe
            seguir = False
    else: #en la caso que el destino y el origen sean iguales
        seguir = True
        while seguir:
            if origen in grafo[n]: #se busca el origen el grafo
                grafo[n].append(distancia) #y se agraga su distancia ya que el destino ya existe
                seguir = False

for i in range(0,len(grafo)): #se recorre el grafo
    for j in range(0,len(grafo[i])):
        sheet.write(i,j,grafo[i][j]) #y se agrega al archivo excel
documento.save("Grafo.xls")

```

Código 4: “grafo parte 2 agregado a excel y creación de grafo”.

Una vez con la base de datos lista, se procede a pasar los datos a python, quedando de la siguiente forma:

```
l1=pd.read_excel("kk.xlsx")

asd1=l1.columns.array

xx=[]#Nombre de las estaciones

for j in asd1:
    j = j
    xx.append(j)

xx.pop(0)

grafo=[]#Grafo que contiene las distancias minimas
for k in xx:
    t1=l1[k]
    t2=[]
    for j in t1:
        t2.append(j)
    grafo.append(t2)
```

Código 5: Integración de la base de datos en el código.

Para la lectura del archivo excel se ocupó la librería pandas junto con su función `.columns.array` la cual selecciona las columnas que contienen las distancias y las convierte en una lista. ya pasando por los dos for del código 5 la lista “xx” contiene el nombre de las tiendas y “grafo” las distancias de estas hacia las demás.

En cuanto al desarrollo del traslado 1:1 se realizó con el algoritmo de dijkstra, ya que este entrega la distancia mínima entre dos puntos en un grafo, el cual queda de la siguiente forma en código:

```
def dijkstra(graf, src, op2):
    d1=len(graf)
    d2=len(graf[0])
    infi=999999999
    dist=[float(infi)]*d1
    punt=[-1]*d1
    dist[src]=0
    cola=[]
    for i in range(d1):
        cola.append(i)
    while cola:
        min=float(infi)
        min_index=-1
        for i in range(len(dist)):
            if(dist[i]<min and i in cola):
                min=dist[i]
                min_index=i
        u=min_index
        cola.remove(u)
        for i in range(d2):
            if graf[u][i] and i in cola:
                if (dist[u] + graf[u][i]<dist[i]):
                    dist[i]=dist[u]+graf[u][i]
                    punt[i]=u
    ll=[]
    p=[]
    c1=[]
    def mostrarcamino(punt,j,xx):
        if(punt[j]==-1):
            ll.append(xx[j])
            return
        mostrarcamino(punt,punt[j],xx)
        ll.append(xx[j])
    mostrarcamino(punt,op2,xx)
    ñ9=round(dist[op2],2)
    p.append(ñ9)
    c1.append(ll)
    c1.append(p)
    return c1
```

Código 6: Construcción del traslado 1:1.

El resultado de la función `dij`, será la lista “`c1`” la cual contiene los locales por los que debe pasar y la distancia mínima de recorrido. Por otro lado para el traslado 1:3 se ocupó la técnica de programación dinámica, la cual queda de la siguiente forma:

```
#Algoritmo para resolver traslado 1:3
def unotres(u):
    v=0
    for i in range(0,len(xx)):
        if(u==xx[i]):
            v=i
    return grafo[v]
def tresp(ini):
    dic={}
    for i in range(0,len(ini)):
        dic[i]=ini[i]

    dicor = sorted(dic.items(), key=operator.itemgetter(1))
    m1=dicor[1]
    m2=dicor[2]
    m3=dicor[3]
    return m1,m2,m3
```

Código 7: Construcción del traslado 1:3.

La función “`unotres`” recibe como parámetro el local de inicio y entrega la lista de las distancias asociadas a ese local. para la función “`tresp`” esta recibe el resultado de “`unotres`” y crea un diccionario en donde la llave será índice “`i`” el cual representa el nombre de la tienda y su contenido será la distancia mínima desde el local de inicio “`u`” hasta la tienda del índice “`i`”. La variable “`dicor`” representa el diccionario ordenado de menor a mayor en base a los ítems del diccionario que en este caso serán las distancias. Como el traslado 1:3 pide las 3 distancias más cercanas, se almacenaron estas en las variables `m1,m2,m3` y se retornaron(`dicor[0]` representa la diagonal de la matriz, por eso no se toma en cuenta).

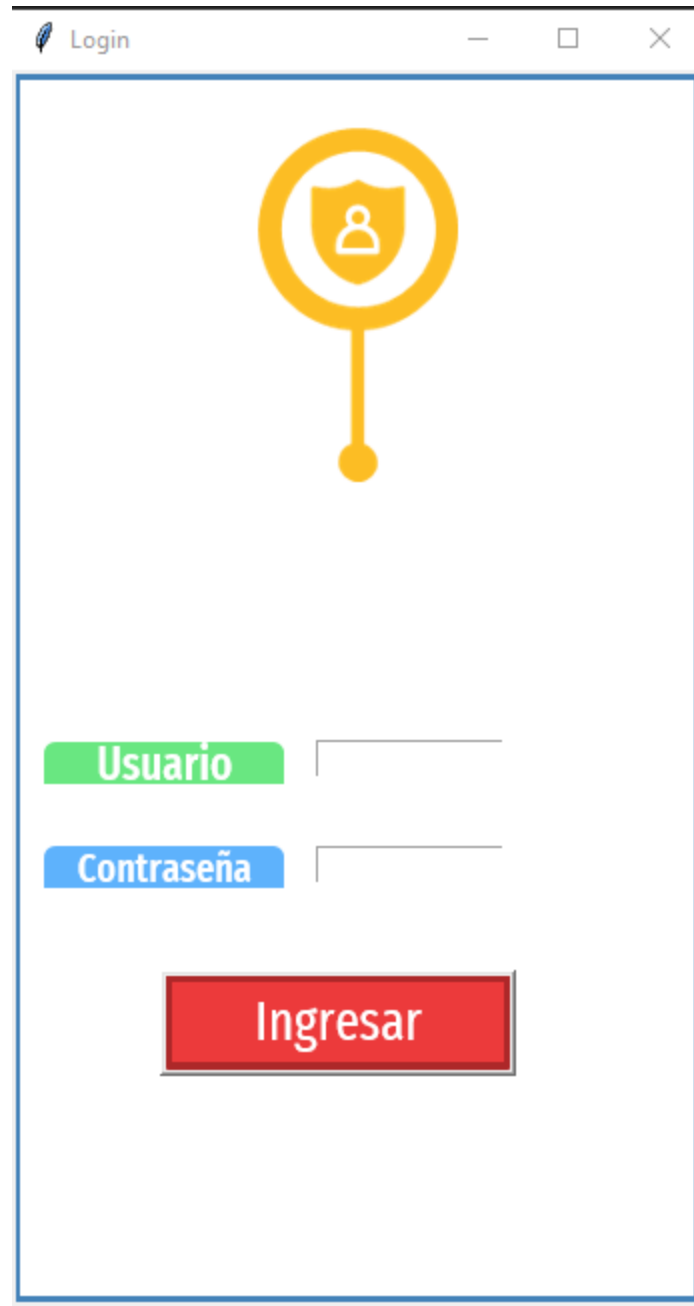
Los resultados de los traslados 1:1 y 1:3 se mostrarán al usuario mediante una interfaz gráfica la cual se compone de la siguiente manera:

Ventana login:

```
#Se crea la ventana login
v0=Tk()
im0=PhotoImage(file="login.png")
im01=PhotoImage(file="ingresar.png")
im02=PhotoImage(file="contra.png")
v0.title("Login")
v0.geometry("347x619")
fondo0=Label(image=im0,text="imagen 5.0 de fonde",bd=0)
fondo0.place(x = 0, y = 0, relwidth = 1, relheight = 1)
usuario=StringVar()
usu=Entry(v0, width=15, textvariable=usuario).place(x=152,y=335)
contraseña=StringVar()
con=Entry(v0, width=15, textvariable=contraseña,show="*").place(x=152,y=3
def pas():
    if(usuario.get()=="mauricio"and contraseña.get()=="hidalgo"):
        v0.destroy()
        principal()
    else:
        inco=Label(v0,text="Contraseña incorrecta").place(x=100,y=423)
bot=tkinter.Button(v0,image=im01,command=lambda: pas())
bot.place(x=74,y=450)
v0.mainloop()
```

Código 8: Construcción del login.

Esta ventana contiene el inicio de sesión del usuario, también se crea la función “pas” la cual sirve para validar si el usuario y contraseña son correctos, si ese es el caso, destruye la ventana y llama a la función “principal”, si no es el caso y el usuario se equivoca le muestra el mensaje de “contraseña incorrecta”.



The image shows a login window titled "Login" with standard window controls (minimize, maximize, close). The window contains a large yellow circular icon with a shield and a person silhouette inside, hanging from a vertical line. Below this icon are two input fields: the first is labeled "Usuario" in a green box, and the second is labeled "Contraseña" in a blue box. At the bottom center is a red button labeled "Ingresar".

Figure 13: Aspecto de la ventana de ingreso.

Ventana principal:

```
def principal():  
    v=Tk()#Se define la ventana principal  
    im=PhotoImage(file="press.png")  
    im1=PhotoImage(file="tr.png")  
    im2=PhotoImage(file="boton.png")  
    im3=PhotoImage(file="boton1.png")  
    im4=PhotoImage(file="UNO.png")  
    im5=PhotoImage(file="unotres.png")  
    #Elementos de la ventana  
    v.title("Minimal Routes")  
    v.geometry("1010x753")  
    im=PhotoImage(file="press.png")  
    fondo=Label(image=im,text="imagen 5.0 de fonde",bd=0)  
    fondo.place(x = 0, y = 0, relwidth = 1, relheight = 1)
```

Código 9: Construcción de la ventana principal.

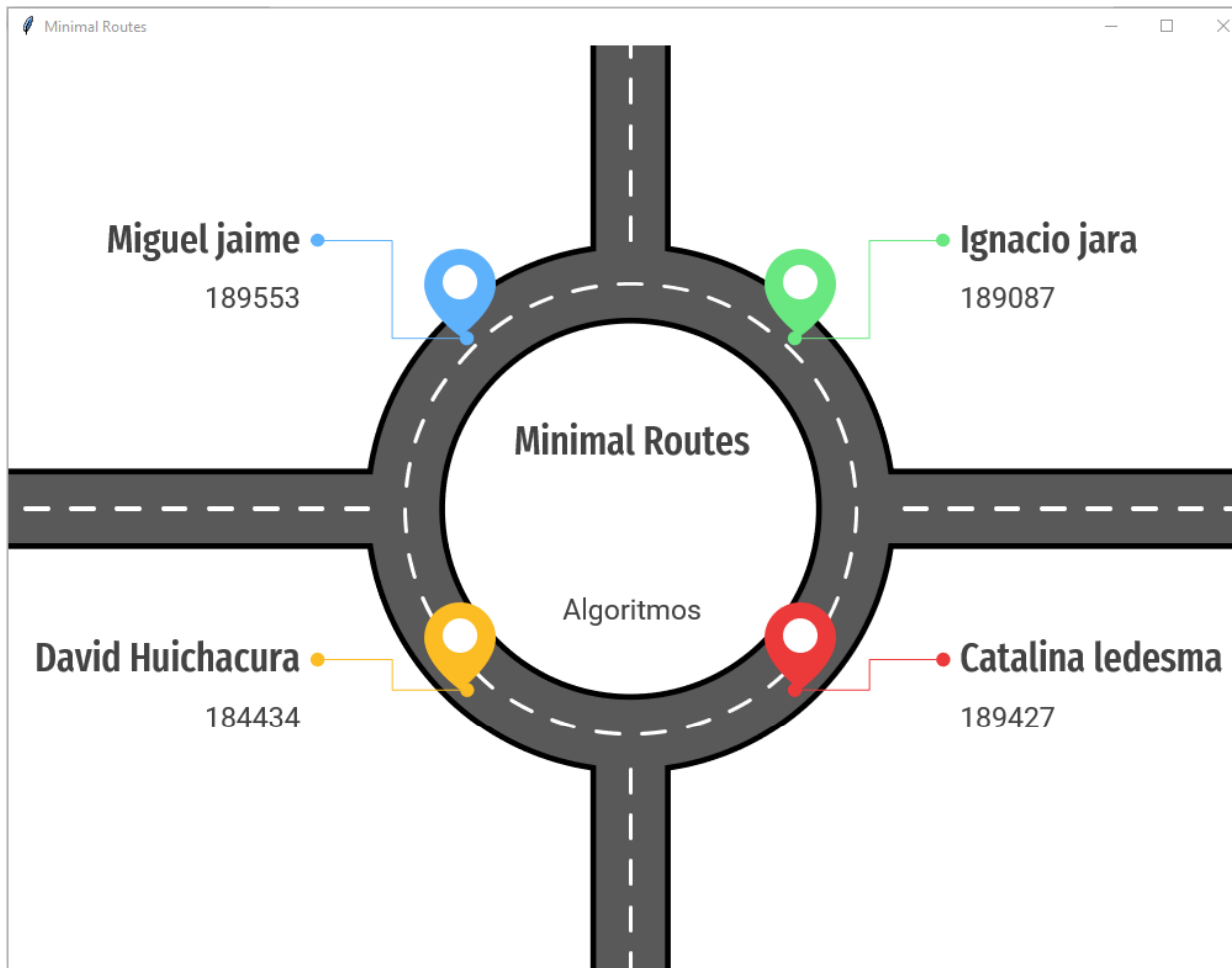


Figure 14: Aspecto de la ventana principal.

Al llamar a la función “principal” esta crea una nueva ventana la cual no es interactiva y contiene la información del programa.

Ventana traslados:

```
v2 = Toplevel(v)
v2.geometry("392x568")
fondo1=Label(v2,image=im1,text="imagen 5.0 de fonde",bd=0)
fondo1.place(x = 0, y = 0, relwidth = 1, relheight = 1)
v2.title("Traslados")
b1=tkinter.Button(v2,image=im2,command=uno)
b1.place(x=5,y=120)
b2=tkinter.Button(v2,image=im3,command=dos)
```

Código 10: Construcción de la ventana de traslados.

En esta ventana es una sub-ventana de la ventana principal, la cual sirve para que el usuario elija que traslado desea realizar.

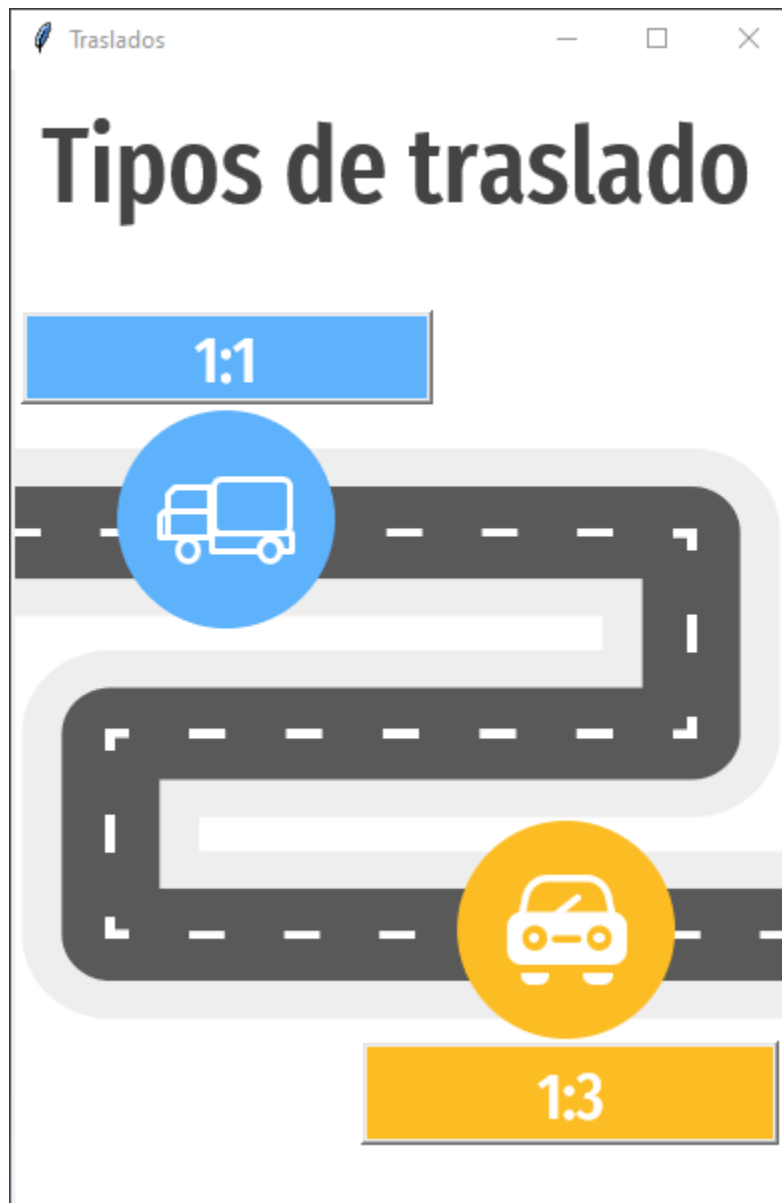


Figure 15: Ventana de la elección del tipo de traslado.

Ventana 1:1:

```
def uno():  
    if(b1.getboolean(s=TRUE)):  
        #Elementos de la sub-ventana  
        v3=Toplevel(v2)  
        v3.geometry("863x404")  
        v3.title("Traslado 1:1")  
        fondo1=Label(v3,image=im4,text="imagen 5.0 de fonde",bd=0)  
        fondo1.place(x = 0, y = 0, relwidth = 1, relheight = 1)  
        f2=Label(v3,wraplength=250)  
        f3=Label(v3)  
        n = tk.StringVar()  
        n2 = tk.StringVar()  
        list = ttk.Combobox(v3, textvariable = n)  
        list1 = ttk.Combobox(v3, textvariable = n2)  
        list1["values"]=(xx)  
        list['values'] = (xx)
```

Código 11: Ventana del traslado 1:1.

Al accionar el botón 1:1 de la ventana de traslados se llama a la función “uno”, la cual crea una sub-ventana con sus elementos correspondientes. Gracias al combobox el usuario puede elegir un local de inicio y destino, pero como el algoritmo ya programado de dijkstrak, recibe valores de los índices del grafo se crearon las siguientes dos funciones:

```

##Funcion para obtener inicio de la ruta en base a .get() de combobox
def klk(xx):
    c1=0
    for i in range(0,len(xx)):
        if(list.get()==xx[i]):
            c1=i
            break
    return c1
##Funcion para obtener inicio de la ruta en base a .get() de combobox
def klk1(xx):
    c1=0
    for i in range(0,len(xx)):
        if(list1.get()==xx[i]):
            c1=i
            break
    return c1

```

Código 12:Funciones de la ruta con combobox.

Retornando los valores de “c1”, los cuales serán el índice en donde se encuentra esa tienda en el grafo. Ya con los valores de inicio y destino se llamó a la siguiente función:

```

##Funcion para mostrar resultados al usuario
def fff():
    mm=dij(grafo,klk(xx),klk1(xx))
    tt=str(mm[0])
    tt1=str(mm[1])
    f2["text"]="Los locales visitados son: "+tt
    f3["text"]="Con una distancia de: "+tt1+"KM"
    f2.place(x=521,y=230)
    f3.place(x=521,y=200)

```

Código 13: Construcción de entrega de resultados al usuario.

Por último se creó el botón para mostrar los valores en la ventana:

```

b2=tkinter.Button(v3,text="Calcular ruta",command=lambda: fff())
b2.place(x=554,y=95)

```

Código 14: Creación del botón que entrega los valores.



Figure 16: Ventana para el ingreso de datos para el usuario.

ventana 1:3:

```
def dos():
    if(b2.getboolean(s=TRUE)):
        #Especificaciones de la sub-ventana
        v3=Toplevel(v2)
        v3.geometry("1063x552")
        v3.title("Traslado 1:3")
        fondo1=Label(v3,image=im5,text="imagen 5.0 de fonde",bd=0)
        fondo1.place(x = 0, y = 0, relwidth = 1, relheight = 1)
        n = tk.StringVar()
        list = ttk.Combobox(v3, textvariable = n)
        list["values"]=(xx)
        list.place(x=36,y=320)
        d1=Label(v3)#,anchor=CENTER)
        d2=Label(v3)#,anchor=CENTER)
        d3=Label(v3)#,anchor=CENTER)
        salida=tkinter.Button(v3,text="Salir",command=lambda: v3.destroy())
        salida.place(x=990,y=503)
```

Código 15: Construcción de la ventana del traslado 1:3.

Al accionar el botón 1:3 de la ventana de traslados se llama a la función “dos”, la cual crea una sub-ventana con sus elementos correspondientes.

Para mostrar los resultados al usuario se utilizó el algoritmo en base a programación dinámica del código 7, como ya se mencionó en ese apartado para lograr la rotación de los locales y distancias se empleó el siguiente código:

```
def p2(u):
    ini=unotres(u)
    cer=tresp(ini)
    li=[]
    for i in cer:
        li.append(xx[i[0]])
    d1["text"]=str(xx[cer[0][0]])+"\n Distancia: "+str(cer[0][1])+"KM"
    d2["text"]=str(xx[cer[1][0]])+"\n Distancia: "+str(cer[1][1])+"KM"
    d3["text"]=str(xx[cer[2][0]])+"\n Distancia: "+str(cer[2][1])+"KM"
    d2.place(x=545,y=53)
    d1.place(x=372,y=428)
    d3.place(x=900,y=220)
    list["values"]=li
```

Código 16: Construcción de la rotación de locales y distancias.

Ya con las funciones “unotres” y “tresp” definidas se creará la función “p2” la cual mostrará los resultados al usuario, en donde la variable “li” contiene las 3 tiendas con sus respectivas 3 distancias mínimas. para ir actualizando el valor de las 3 tiendas en base a la decisión del usuario se cambian los valores del combobox por los valores de la lista “li”, con el fin de que cada vez que se llame a esta función sus valores sean actualizados.

```
b3=tkinter.Button(v3,text="Calcular ruta",command=lambda: p2(list.get()))
b3.place(x=69,y=350)
```

Código 17: Creación del botón que calcula la ruta.

Por último se crea el boton “calcular ruta” el cual al ser presionado manda el valor del combobox a la función “p2”

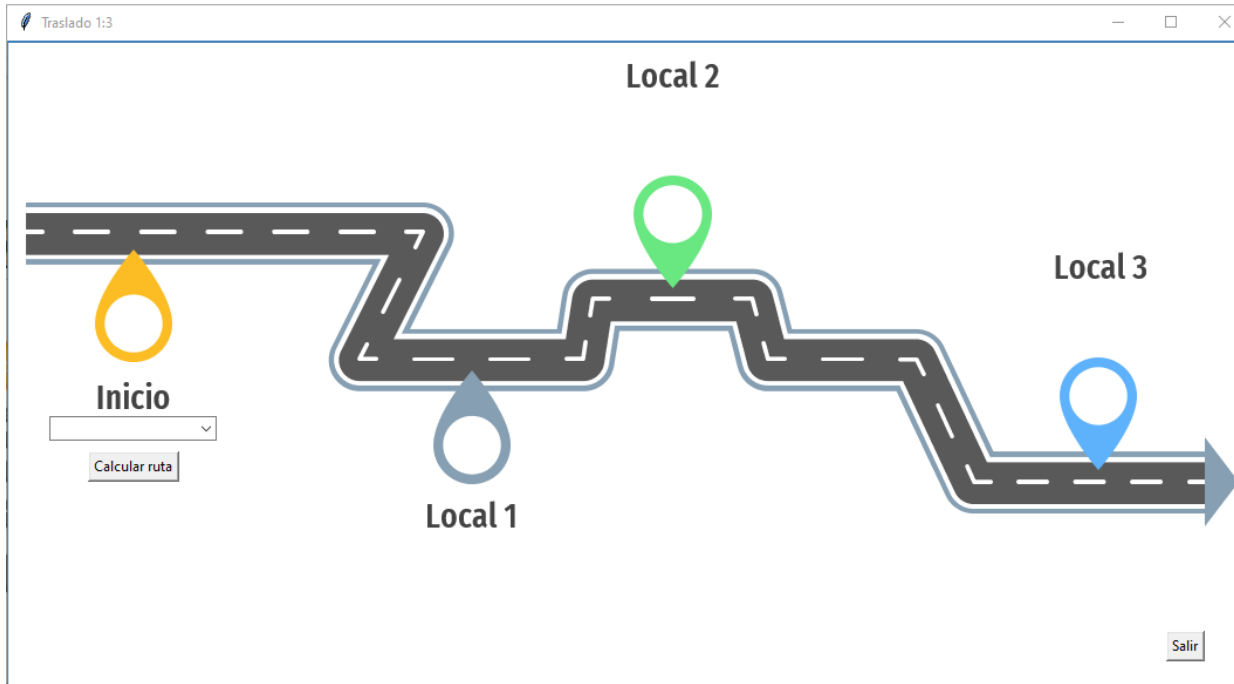


Figure 17: Aspecto del resultado que entrega el programa.

Explicación del proceso de construcción de la solución (MJ Y DH)

1.-desarrollo de la base de datos a ocupar

se investigó cual estructura era más óptima para el desarrollo de la problemática llegando a la conclusión de una matriz de distancias realizada en excel. esto debido a que para conocer la distancia mínima entre un punto y otro junto con el camino que se recorre, el algoritmo de dijkstra, es uno de los más eficientes.

2.-desarrollo back-end

desarrollo de traslado 1:1

se investigó la implementación de dijkstra

desarrollo de traslado 1:3

se investigó la implementación de la técnica de programación dinámica

3.-desarrollo front-end

para el desarrollo del interfaz se investigó y implementó la librería tkinter

Evaluación de efectividad de la solución

Anteriormente, se dejó en claro que la efectividad de la solución hecha iba a ser por medio de la comparativa de distancias usando Google Maps, y la API de Google Cloud Distance Matrix API. Usando como estándar Google Maps.

Para el traslado 1:1 se realizarán dos pruebas, y en base a estas dos comparativas se sacará una media porcentual qué indicará la eficiencia en base al estándar.

En la primera prueba el local de inicio será Universidad de Santiago y el local de destino será Elisa Correa. Tal como se puede apreciar en la siguiente imagen:



Prueba 1: "Traslado 1:1".

La distancia obtenida en el programa creado es de 18.7 km, mientras que la ruta más óptima de google maps es de 25.9 km. Por consiguiente, cambiarían totalmente las estaciones de ruta, es decir, que si 25.9 km es el 100% de efectividad, el programa realizado en esta prueba posee un 72.2% de efectividad.

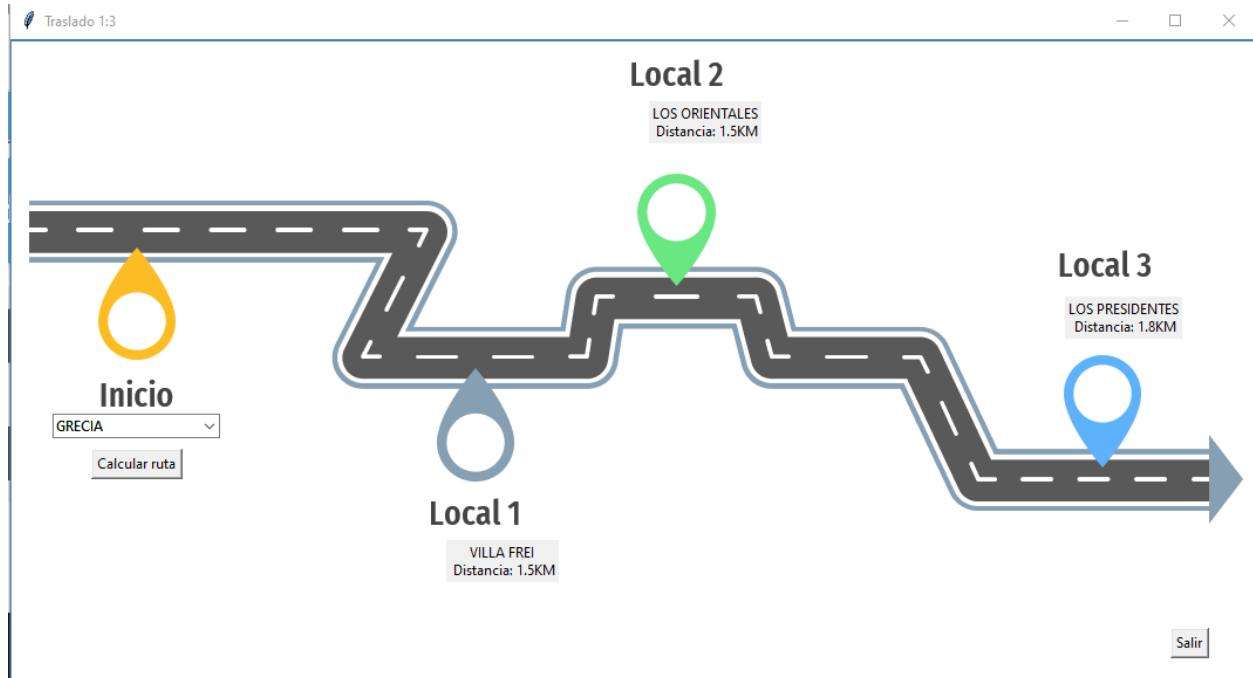
En esta ocasión el local de inicio será Vespucio Norte, mientras que el local de destino será Estadio Nacional, donde se pueden apreciar los resultados en la siguiente imagen:



Prueba 2: "Traslado 1:1".

Es posible observar que la distancia en esta ocasión es de 11.8 km, y si estas mismas entradas de inicio y destino se ingresan a google maps. Ésta entregará una distancia de 17.1 km, esto entregaría una eficiencia en base al estándar de un 69%.

Para el traslado 1:3, se van a comparar las distancias de las tres estaciones más cercanas según la ruta, siguiendo la misma comparativa y estándar, es decir. Se tomará como estándar Google Maps, y se sacará un porcentaje en base a las distancias que arroje el mismo.

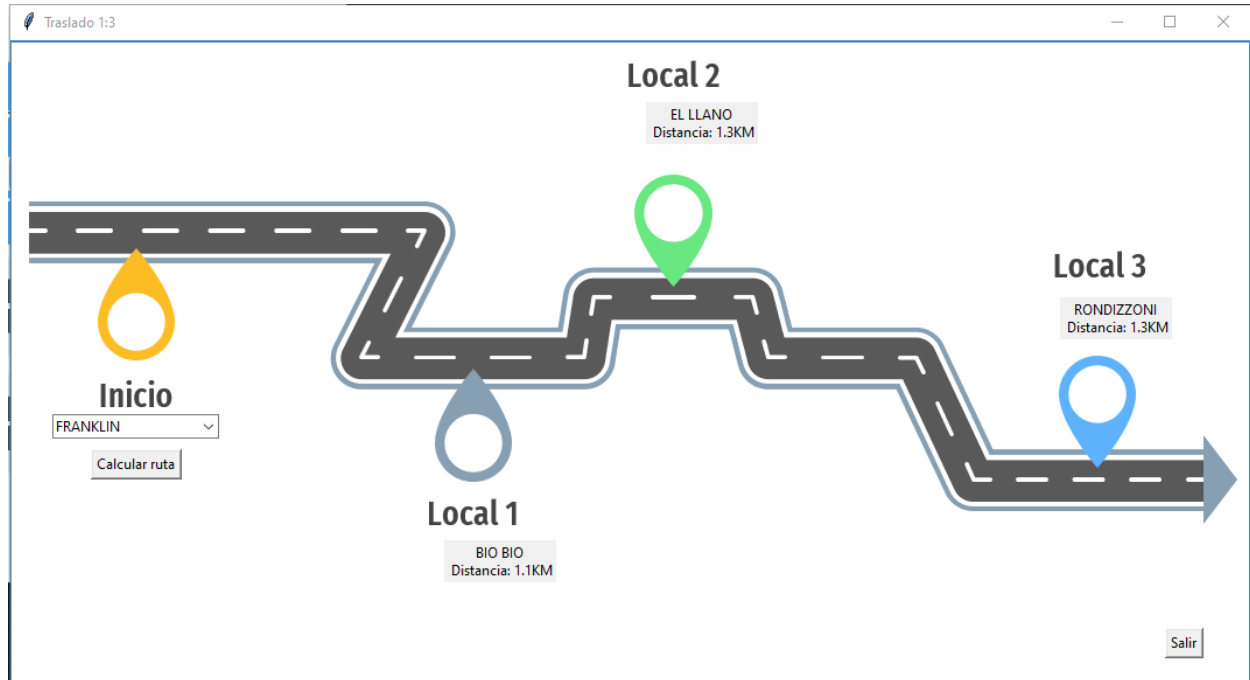


Prueba 1: "Traslado 1:3".

Para la primera prueba, se usó la estación Grecia como punto de partida, y a continuación las tres estaciones más cercanas son Villa Frei, cuyo recorrido en Google maps posee una distancia óptima de 2.4 km, que en términos porcentuales el programa logró una eficiencia del 63%. Mientras que, los orientales posee una distancia de 1.2 km, por lo tanto, es una eficiencia del 80%, por último los presidentes posee una distancia de ruta de 1.3 km, funcionando en un 72%.

Ya que son tres distancias y porcentajes diferentes, el promedio de la eficiencia de la prueba uno en el traslado 1:3 es del 71.6% aproximadamente.

Para la segunda prueba del traslado 1:3, el local de inicio es Franklin, tal como se puede apreciar en la imagen siguiente:



Prueba 2: "Traslado 1:3".

De Franklin al Bio bio en el programa son 1.1 km, pero en Google Maps posee una distancia óptima de 2 km, significando que posee una eficiencia de un 55%. Entre tanto, el recorrido hasta El Llano, tiene una distancia de 1.3 km, en Maps sujeta 2.1 km. Obteniendo una eficiencia del 62% aproximadamente, finalmente en Rondizzoni fuera del programa hay un trecho de 1.6 km, sacando un 81% de exactitud; El promedio de estos tres porcentajes es de 66%.

Finalmente para poder medir la eficiencia que el programa posee en base al estándar declarado, se sacará el promedio de los porcentajes obtenidos en cada prueba teniendo en consideración qué los resultados obtenidos son:

Traslado 1:1 → Prueba 1: 72.2%
Prueba 2: 69%

Traslado 1:3 → Prueba 1: 71.6%
Prueba 2: 66%

Por lo tanto, la eficiencia promedio del programa generado en base a la propuesta es de un 69.7%, siendo esta la efectividad medida en base a las pruebas realizadas.

Resultados y Discusión

Empezando por los resultados de la efectividad de la solución, cabe destacar que estos resultados se obtuvieron solo con dos pruebas de todas las combinaciones posibles que pudieron haber sido, pero aún así la eficiencia bajo el estándar que se declaró es de un 69.7%, el cual es un porcentaje bajísimo como para implementar este programa en una empresa, ya que se debe considerar que de 100 camiones que transportan mercadería para stock. 30 posiblemente se pierdan, y esto provocaría una evidente falla en el sistema logístico de la empresa, llevándola al inevitable fracaso.

Lo que se aplicó para el desarrollo de la solución del proyecto, fue inicialmente los grafos y la búsqueda de rutas mediante el algoritmo de dijkstra en los archivos de traslado para optimizar el código, todo con el objetivo de obtener el camino más corto dentro del traslado. Luego se implementó la búsqueda en google maps, mediante la librería llamada “request”, con ello se consultaban las rutas entre cada estación de metro y se añadían al excel de la base de datos (grafo), para posteriormente calcularlas distancias y la bencina. Por último se implementa la interfaz con ventanas emergentes y combobox, para la interacción del usuario.

Dentro del código se halló una inconsistencia en el API, dado que al momento de construir la base de datos, las consultas que lo conforman tardaban alrededor de unos 10 minutos en cargarse, ya que se hacen varias consultas y se cargan al mismo tiempo. Por ende para facilitar y agilizar el uso del programa, se podría usar valores promediados en las distancias, teniendo en cuenta que no otorgara una respuesta exacta sino que un aproximado de gastos.

Conclusiones

Evalúa el nivel de logro de los objetivos, explicando con claridad las razones de aquellos logros.

En conclusión, se lograron los objetivos propuestos en el proyecto. Sin embargo, aunque el desarrollo fue eficaz, no fue eficiente, ya que se cumplieron los objetivos, pero no de la mejor forma posible, debido a la diferencia de distancias que habían en la API trabajada y en Google Maps. Es sabido que los servicios API son de paga, por lo tanto, para una mejora del proyecto, se tendría que invertir en él.

Finalmente, el software cumplió con los requisitos solicitados, creando una interfaz que sea amigable para el usuario, y un backend que cumpliera de forma óptima los procesos requeridos para el sistema de minimización de rutas que necesitaba la empresa.

Referencias

- (1) Puchades Cortés, Vanesa, & Mula Bru, Josefa, & Rodríguez Villalobos, Alejandro (2008). Aplicación de la Teoría de Grafos para mejorar la planificación de rutas de trabajo de una empresa del sector de la distribución automática. *Revista de Métodos Cuantitativos para la Economía y la Empresa*, 6(), 7-22. [fecha de Consulta 15 de Noviembre de 2021]. ISSN: . Disponible en: <https://www.redalyc.org/articulo.oa?id=233117226002>
- (2) López, B. S. (2020, 16 febrero). *Problema del agente viajero – TSP*. Ingeniería Industrial Online. <https://www.ingenieriaindustrialonline.com/investigacion-de-operaciones/problema-del-agente-viajero-tsp/>
- (3) Salazar, P. J. F. (2019, 16 julio). *Modelo heurístico de asignación de rutas para minimizar los costos operativos del Servicio de Transporte de Ruta de la empresa Brandom S.A.C, 2018*. Red de Repositorios LATAM. <https://repositorioslatinoamericanos.uchile.cl/handle/2250/3243180>
- (4) *Tema 43*. (s. f.). TEMA 43 - FREESERVERS. <http://darna.freesevers.com/t43a.htm>