EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Tecnológico Nacional de México

Centro Nacional de Investigación y Desarrollo Tecnológico

Tesis de Maestría

Planificador de rutas en ciudades utilizando GTFS

presentado por
Ing. Carlos Alberto Ruiz Gutiérrez

como requisito para la obtención del grado de
Maestro en Ciencias de la Computación

Director de tesis
Dr. Hugo Estrada Esquivel

Codirectora de tesis
Dra. Alicia Martínez Rebollar

Cuernavaca, Morelos, México. Diciembre de 2023.

cenidet
Centro Nacional de Investigación y Desarrollo Tecnológico

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Centro Nacional de Investigación
y Desarrollo Tecnológico
Departamento de Ciencias Computacionales

SEP TecNM CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
RECIBIDO
08 NOV 2023
SUBDIRECCIÓN ACADÉMICA

Cuernavaca, Mor., 02/octubre/2023

OFICIO No. DCC/178/2023
Asunto: Aceptación de documento de tesis
CENIDET-AC-004-M14-OFICIO

CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO
PRESENTE

Por este conducto, los integrantes de Comité Tutorial de CARLOS ALBERTO RUIZ GUTIÉRREZ con número de control M21CE066, de la Maestría en Ciencias de la Computación, le informamos que hemos revisado el trabajo de tesis de grado titulado **"PLANIFICADOR DE RUTAS EN CIUDADES UTILIZANDO GTFS"**, y hemos encontrado que se han atendido todas las observaciones que se le indicaron, por lo que hemos acordado aceptar el documento de tesis y le solicitamos la autorización de impresión definitiva.

HUGO ESTRADA ESQUIVEL
Director de tesis

ALICIA MARTÍNEZ REBOLLAR
Codirectora de tesis

JAVIER ORTIZ HERNÁNDEZ
Revisor 1

NIMROD GONZÁLEZ FRANCO
Revisor 2

EDUC
08 NOV 2023
CENTRO NAL. ... Y DESA... SERVIC...
RECIBIDO

C.c.p. Depto. Servicios Escolares.
Expediente / Estudiante

Cuernavaca, Mor., 13/noviembre/2023
No. De Oficio: SAC/174/2023
Asunto: Autorización de impresión de tesis

CARLOS ALBERTO RUIZ GUTIÉRREZ
CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS DE LA COMPUTACIÓN
P R E S E N T E

Por este conducto, tengo el agrado de comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado **"PLANIFICADOR DE RUTAS EN CIUDADES UTILIZANDO GTFS"**, ha informado a esta Subdirección Académica, que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

Esperando que el logro del mismo sea acorde con sus aspiraciones profesionales, reciba un cordial saludo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
"Conocimiento y tecnología al servicio de México"

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
SUBDIRECCIÓN ACADÉMICA

CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO

C. c. p. Departamento de Ciencias Computacionales
Departamento de Servicios Escolares

CMAZ/lmz

Dedicatoria

Con cariño dedico este trabajo a:

A mis padres, Gabriel Ruiz y María Gutiérrez,
por los esfuerzos que realizan cada día por mí,
por las enseñanzas que me han brindado,
por sus cansancios y desvelos,
por los regaños y el amor que me tienen;
Dios les pague todo lo que han hecho por mí.
¡Los amo!

A mis hermanos Lizethe, Ana, Mario y Melissa,
con quienes he compartido risas,
aventuras y momentos inolvidables,
he aprendido mucho de ustedes, los quiero mucho.

A mis seres queridos, quienes han estado a mi lado
en cada etapa de mi vida, quiero expresar mi gratitud
por su generosidad, comprensión y aliento.

Agradecimientos

Queridos padres, familia y seres queridos,

Hoy quiero expresar mi más profundo agradecimiento por el apoyo incondicional que me han brindado a lo largo de esta emocionante y desafiante etapa de mi vida profesional. Con su constante motivación, paciencia y cariño han sido mi motor para superar obstáculos y alcanzar este logro significativo en mi vida. Cada paso que he dado ha sido respaldado con su amor y confianza, y por eso, estoy eternamente agradecido.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) y al Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), por la oportunidad y facilidad brindada para la realización de esta tesis.

A mi director de tesis, el Dr. Hugo Estrada, quiero expresar mi sincero agradecimiento. Por brindarme el apoyo y la guía en esta etapa de mi formación profesional. Tener la paciencia y confianza en mí para la realización de este trabajo de investigación.

A mi codirectora de tesis, la Dra. Alicia Martínez, quiero agradecerle por su invalorable contribución a mi investigación. Con su presencia y apoyo han sido esenciales para alcanzar este logro académico.

A mis compañeros del área de Cómputo inteligente, quiero expresar mi gratitud por ser una fuente constante de inspiración y colaboración. Las reuniones han enriquecido mi entendimiento y han contribuido a la calidad de mi trabajo. Juntos hemos enfrentado desafíos y festejado éxitos. Estoy agradecido por tenerlos como colegas y amigos.

A mi comité revisor, el Dr. Javier Ortiz y el Dr. Nimrod González, agradezco profundamente su tiempo, conocimiento y valiosos comentarios. Sus sugerencias y críticas constructivas han mejorado significativamente mi tesis, y estoy agradecido por su dedicación a mi crecimiento académico.

Este logro no habría sido posible sin cada uno de ustedes. Su contribución ha sido invaluable y ha dejado una huella imborrable en mi camino. Gracias por creer en mí, por alentarme y por compartir este viaje conmigo.

Con gratitud sincera.

Carlos Ruiz

Resumen

En un mundo cada vez más urbano, la movilidad eficiente y sostenible se convierte en un desafío crítico para las ciudades. En esta tesis se aborda este desafío mediante el desarrollo de un planificador de rutas para el transporte público en la Ciudad de México, el cual utiliza tecnologías de código abierto y se basa en la especificación GTFS (*General Transit Feed Specification*).

El objetivo principal de este trabajo fue diseñar y crear un sistema capaz de encontrar la ruta más corta en distancia entre dos puntos dentro de la ciudad, considerando todos los sistemas de transporte público que se encuentran representados en la especificación GTFS de la CDMX, tales como el Metrobús, Servicio de Tren Ligero, Ferrocarriles Suburbanos, Red de Transporte de Pasajeros RTP y el Sistema de Transporte Colectivo Metro.

Para lograr este propósito, se propuso la construcción de un modelo de datos que permita consumir la información del transporte público y su integración con un algoritmo de planificación de rutas. Además, se desarrolló un algoritmo eficiente que calcula las rutas óptimas, y se implementó un prototipo funcional que garantice la reutilización y accesibilidad del código del planificador en otros proyectos de movilidad.

El sistema propuesto representa una solución versátil y de código abierto que busca promover la movilidad inteligente y facilitar el acceso de los ciudadanos a las opciones de transporte público.

Con este enfoque innovador y tecnológico, se pretende contribuir significativamente al desarrollo de soluciones de movilidad sostenible y eficiente en la ciudad, al tiempo que se proporciona una herramienta práctica y accesible para los ciudadanos en su búsqueda de rutas óptimas para alcanzar sus destinos en el transporte público.

Abstract

In an increasingly urban world, efficient and sustainable mobility becomes a critical challenge for cities. This thesis addresses this challenge by developing a route planner for public transportation in Mexico City, which uses open-source technologies and is based on the GTFS (General Transit Feed Specification).

The main objective of this work was to design and create a system capable of finding the shortest route in distance between two points within the city, considering all public transport systems that are represented in the GTFS specification of the CDMX, such as Metrobus, Light Rail Service, Suburban Railways, RTP Passenger Transport Network and the Metro Collective Transport System.

To achieve this purpose, we proposed the construction of a data model that allows the consumption of public transportation information and its integration with a route planning algorithm. In addition, an efficient algorithm that calculates optimal routes was developed, and a functional prototype was implemented to ensure the reusability and accessibility of the planner code in other mobility projects.

The proposed system represents a versatile and open-source solution that seeks to promote smart mobility and facilitate citizens' access to public transportation options.

With this innovative and technological approach, it aims to contribute significantly to the development of sustainable and efficient mobility solutions in the city, while providing a practical and accessible tool for citizens in their search for optimal routes to reach their destinations by public transport.

Contenido

Índice de figuras

Índice de tablas

Capítulo 1 Introducción

En este capítulo se presenta la problemática involucrada en el proyecto de tesis, el objetivo general y objetivos específicos de la tesis, así como los alcances y limitaciones del prototipo propuesto en este proyecto de investigación.

1.1 Introducción

Hoy en día, la complejidad de las vialidades de una ciudad y del sistema de transporte público y privado de las urbes modernas representa un gran desafío para cualquier ciudadano que pretenda determinar las rutas de transporte óptimas para alcanzar un destino en la ciudad, ya sea por la complejidad del sistema de transporte público o porque ocurran eventualidades tales como accidentes, la presencia de obras y/o de congestiones que dificultan e imposibilitan temporalmente la utilización de ciertos sectores de la ciudad.

El uso masivo del transporte público y la actual complejidad de las distintas rutas que operan en las ciudades ha ocasionado que los ciudadanos requieran de ayuda al momento de planificar una ruta que le ayude a ahorrar tiempo. Esta problemática es atendida por diversos sistemas de software que permiten que los ciudadanos puedan alcanzar un destino utilizando auto particular, como es el caso de *Waze* o utilizando algunos transportes públicos como es el caso de *Google Maps*. Si bien estos sistemas son de gran ayuda para los usuarios finales, tienen el inconveniente de ser sistemas cerrados propiedad de empresas, lo cual ocasiona que estos recursos de planificación de rutas no puedan ser reutilizados por desarrolladores que deseen crear aplicaciones para movilidad.

Con la finalidad de dar una solución a esta problemática, se propone el desarrollo de un prototipo de software que permita el consumo de los datos GTFS (Especificación general de tránsito), los cuales contienen la información de las agencias de tránsito que operan en la ciudad. Esta estructura de datos de transporte público será la entrada de un algoritmo que determinará la ruta más corta para alcanzar un destino en la ciudad. Una vez que genera la ruta, ésta será visualizada a través de mapa de tal forma que pueda ser visualizada la ruta que debe seguir el usuario.

En este contexto, este trabajo de tesis propone el desarrollo de un algoritmo de planificación de rutas de transporte público con tecnologías *open source*. Las rutas propuestas incluyen todos los sistemas de transporte público que aparecen en el estándar GTFS de la Ciudad de México.

1.2 Planteamiento del problema

El incremento de población mundial y el aumento de las ciudades ha generado un crecimiento desmesurado de las urbes modernas. La ciudadanía en general exige el constante rediseño en la gestión y administración de los servicios de transporte público. El servicio público, que moviliza masivamente a la población, intenta dar solución a las necesidades de desplazamiento de las personas (Celi, 2018).

Un promedio del 31.1% de la población mundial hace uso del transporte privado, el 40.8% hace uso del transporte público y un 31.6% utilizan el transporte no motorizado, ya sea a pie o bicicleta (Celi, 2018).

La Ciudad de México cuenta con una población de 9,209,944 habitantes y la Zona Metropolitana del Valle de México tiene 20,137,152 pobladores de acuerdo con el censo más reciente del INEGI (INEGI, 2021). La mayoría de estos ciudadanos utilizan a diario la red de transporte público para viajar desde sus hogares a los centros de trabajo, educativos o de entretenimiento (Peña, 2012).

El Instituto Nacional de Estadística y Geografía (INEGI) determinó que, en la Zona Metropolitana del Valle de México, se realizan 35 millones de viajes de lunes a viernes. De estos viajes, el 45% son a través del transporte público, siendo el microbús o las rutas de transporte público (RTP) el medio de transporte más utilizado (Notimex, 2018).

El sistema de transporte público en la Ciudad de México es una red compleja que integra diferentes tipos de transporte en diferentes rutas que recorren toda la ciudad. En esta red multimodal es muy complicado saber qué combinación de transportes públicos se debe tomar para llegar más rápido a un destino (López y otros, 2018). El uso masivo del transporte público y la actual complejidad de las distintas rutas que operan en las ciudades ha ocasionado que los ciudadanos requieran de ayudas para planificar rutas que le ayuden a ahorrar tiempo de viaje (Bull, 2003).

En este sentido han surgido diversos sistemas de software que permiten que los ciudadanos puedan alcanzar un destino utilizando auto particular (La Vanguardia, 2018), como es el caso de *Waze* (Waze, 2021) o utilizando algunos transportes públicos como es el caso de *Google Maps* (Google support, 2021). Si bien estos sistemas son de gran ayuda para los usuarios finales, tienen el inconveniente de ser sistemas cerrados propiedad de una empresa. Esto ocasiona que no sea posible reutilizar estos recursos por desarrolladores que deseen crear aplicaciones para movilidad (Brotton, 2014). Es importante mencionar que hoy en día no se puede pensar en aplicaciones de movilidad en ciudades que no integre, como parte básica de su funcionalidad, un planificador de rutas en la ciudad.

En esta tesis se propone el **desarrollo de un algoritmo y su correspondiente sistema de software basado en tecnologías *open source*, para la planificación de rutas de transporte público con especificaciones que le permiten su futura reutilización en proyectos de transporte inteligente**. Las rutas definidas por el sistema propuesto deben incluir todos los sistemas de transporte público que aparecen en el estándar GTFS para la Ciudad de México: Metrobús, Servicio de Tren Ligero, Ferrocarriles Suburbanos, Red de Transporte de Pasajero RTP, Sistema de Transporte Colectivo Metro. Las rutas generadas por el sistema podrán ser utilizadas por los ciudadanos de la Ciudad de México para alcanzar un destino.

1.3 Objetivo general

Desarrollar un planificador de rutas para transporte público utilizando especificaciones GTFS y tecnología de código abierto que permita encontrar la ruta más corta, en distancia, entre dos puntos de una ciudad.

1.3.1 Objetivos específicos

- Construir un modelo de datos para consumir los datos de trasporte público con el algoritmo planificador de rutas.
- Desarrollar un algoritmo que calcule la ruta más corta en distancia entre dos puntos de una ciudad utilizando los sistemas de transporte público que cuenten con representación en la especificación GTFS de la Ciudad de México.
- Construir un prototipo para la planificación de rutas de trasporte público utilizando mecanismos y tecnologías que permitan que el código del planificador sea abierto y reutilizable por otros proyectos de movilidad.
- Validar el correcto funcionamiento del algoritmo y del sistema de planificación de rutas.

1.4 Alcances

- Como resultado de la tesis se produjo un sistema que permite seleccionar dos puntos (inicio y fin) en la Ciudad de México utilizando un mapa que se despliega en un navegador.
- El sistema determina la ruta más corta (distancia) entre los dos puntos seleccionados por el usuario utilizando exclusivamente los sistemas de transporte público de la Ciudad de México registrados en GTFS.
- El sistema muestra la ruta determinada por el algoritmo utilizando un mapa gráfico que se despliega en un navegador.
- El planificador de rutas fue desarrollado como un componente independiente que pueda ser acoplado en otros desarrollos que requieran de un planificador de rutas.
- El planificador de rutas propuesto fue desarrollado únicamente con código abierto.

1.5 Limitaciones

- El caso de estudio de esta tesis se realizó tomando en cuenta solo las rutas de transporte colectivo de la Ciudad de México que cuenten con una especificación GTFS.
- Solo se consideraron los sistemas de transporte público incluidos en los datos GTFS de la Ciudad de México, por ejemplo, el sistema de transporte de Rutas Urbanas de la CDMX no cuenta con una especificación GTFS por lo cual no fue considerado en esta tesis.
- En el desarrollo del prototipo, se optó exclusivamente por el uso de tecnologías de código abierto.

Capítulo 2 Marco teórico

En este capítulo se presentan los fundamentos teóricos más relevantes para el proyecto de tesis.

2.1 Ciudades inteligentes

Una ciudad inteligente puede definirse como aquella ciudad que usa las tecnologías de la información y las comunicaciones para hacer que, tanto su infraestructura, como sus componentes y servicios públicos ofrecidos sean más interactivos, eficientes y los ciudadanos puedan ser más conscientes de ellos (Fundación Telefónica, 2011).

De manera descriptiva, una ciudad inteligente es un espacio urbano con infraestructuras, redes y plataformas inteligentes, con millones de sensores y actuadores, que incluye a las personas y a sus teléfonos móviles. Este espacio es capaz de escuchar y comprender lo que está pasando en la ciudad por lo que permite tomar mejores decisiones y proporcionar la información y los servicios adecuados a sus habitantes (Fundación Telefónica, 2011).

Las tecnologías TIC para ciudades inteligentes están llamadas a convertirse en una de las herramientas más potentes en los próximos años. La integración de tecnologías de la información y las comunicaciones en la evolución de una ciudad no solo va a suponer mejoras notables en la provisión de los servicios, sino que va a constituir en sí misma una vía sostenible para el desarrollo económico y social en las próximas décadas de la economía de las ciudades y, por lo tanto, de la economía de los países (Fundación Telefónica, 2011).

2.2 Movilidad urbana

El concepto de movilidad se refiere a la sostenibilidad, la seguridad y la eficiencia de las infraestructuras y sistemas de transporte, así como a la accesibilidad local, nacional e internacional en las ciudades (Fundación Telefónica, 2011).

El crecimiento de la población en las ciudades, así como los nuevos hábitos de vida están presionando los sistemas de transporte para que aumenten su capacidad y ofrezcan un servicio más enfocado a los ciudadanos. Por este motivo, están surgiendo soluciones especialmente relevantes que ayuden a gestionar las redes de trasporte público, mejorando su eficiencia, permitiendo predecir mejor la demanda para optimizar el uso, reduciendo los costos operacionales, aumentando la seguridad y en general, mejorando la experiencia de usuario (Fundación Telefónica, 2011).

2.2.1 Transporte público

El trasporte público es fundamental para el crecimiento económico, social y la accesibilidad a los servicios esenciales. El transporte público es una parte fundamental de la infraestructura urbana, proporcionando acceso y movilidad a la población (Ibáñez, 2022).

Cada vez son más las personas buscan un método de movilidad más sostenible y beneficioso para el medio ambiente. La solución principal para los desplazamientos sostenibles es el transporte público, dando la mejor opción para dar un servicio de movilidad que cumple tanto las necesidades de la ciudadanía como del planeta. El uso de trasporte público tiene múltiples beneficios para la población y las ciudades (Beneyto, 2022).

- Contamina menos que el vehículo privado. El transporte público es la alternativa más ecológica para los desplazamientos que se hacen en la ciudad. La emisión de gases es mucho menor.
- El uso masivo del vehículo privado colapsa las ciudades y las hace más sucias y más ruidosas. Por eso el transporte público es la alternativa idónea para mejorar la movilidad urbana y evitar en gran medida las retenciones y atascos.
- Usar el transporte público resulta más barato que el vehículo privado. Con el transporte público se evitan los gastos del automóvil, mantenimiento, seguros, aparcamiento y combustible.
- Ahorrar tiempo. Muchas veces el uso del automóvil conlleva la búsqueda de aparcamiento en zonas saturadas. Con el transporte público llegas a tu destino sin complicaciones y sin estrés.

2.2.2 Planificación de rutas

El abanico de modos de transporte público que contamos en tiempos recientes son diferentes alternativas para trasladarse de un punto a otro de la ciudad. En cuanto a transporte masivo en ferrocarril disponemos del metro, tren ligero y tren suburbano; tenemos también autobuses rápidos conocidos como metrobús, mexibús, trolebuses y autobuses. El medio de transporte predominante es el de pequeña capacidad conocido como Microbús, debido a su flexibilidad pues llega prácticamente a todas las zonas de la ciudad además de tener horarios extendidos y frecuencia suficiente (Negrete, 2018).

Las grandes ciudades cuentan con una amplia variedad de medios de trasporte público lo que dificulta de manera importante la planificación de rutas. En la planificación de rutas tenemos desafíos asociados como son la optimización de tiempo, la cobertura de áreas geográficas y la adaptabilidad a las demandas de transporte.

La importancia de planificar rutas en grandes ciudades tiene beneficios como la planificación eficiente de rutas, mejora de la experiencia del usuario, la reducción del impacto ambiental, la reducción de costos y tiempo al momento de planificar rutas (Pérez, 2017).

2.3 Especificación general de alimentación en tránsito (GTFS)

GTFS (*General Transit Feed Specification*) es un estándar que define un formato común para los horarios de transporte público y la información geográfica asociada a ellos. El uso de GTFS permite que las empresas de transporte publiquen sus datos de transporte y que los programadores escriban aplicaciones que los consuman (Colque et al., 2019).

Un *feed* GTFS se compone de una serie de archivos de texto recopilados en un archivo ZIP. Cada archivo modela un aspecto específico de la información de transporte público: paradas, rutas, viajes y otros datos relacionados con los horarios. La especificación GTFS se puede utilizar para impulsar a los planificadores de viajes, los editores de horarios y los programadores de diversas aplicaciones (Google, 2021).

En 2005, con el fin de crear un planificador de viajes de tránsito utilizando la aplicación web Google Maps, Google y TriMet en Portland formularon la Especificación General de Alimentación de Tránsito (GTFS) (Catala et al., 2011).

En 2007, Google publicó las especificaciones de alimentación de tránsito y animó a las agencias de tránsito a utilizar el formato GTFS para crear datos de tránsito y publicarlos en la web como fuentes abiertas para uso público. El formato GTFS es fácil de crear para las agencias, cómodo para los programadores y lo suficientemente completo para que el público entienda un sistema de tránsito (Google transit, 2021).

El formato GTFS ha sido adoptado por el sector del transporte público como estándar para el intercambio de datos de horarios, y cada vez son más las agencias de transporte público que suben datos GTFS a Google. En la actualidad, más de 170 agencias de transporte generan y publican sus horarios en formato GTFS (Wong et al., 2013).

2.4 Algoritmo de planificación de rutas

Un algoritmo es una secuencia de pasos preestablecidos que se encarga de transformar una entrada en una salida para cumplir con una tarea específica para la cual fue diseñado. El algoritmo debe tener al menos una entrada y cumplir un objetivo por medio de una serie de pasos finitos. Para el algoritmo debe contener instrucciones claras y ser efectivo (Cormen et al., 2022).

La planificación de rutas es una de las técnicas más importantes debido a su amplio uso en las diferentes áreas. La planificación se define como la búsqueda de una ruta libre de obstáculos desde una posición inicial hasta otra final. Esta operación se realiza mediante el uso de la información que se posee del entorno actualmente, la descripción de la tarea de navegación y algún tipo de metodología estratégica. En los problemas de planificación de rutas de acuerdo con un simple

objetivo, se tiende a seleccionar un único mejor camino desde un origen a un destino (Ortega et al., 2021).

2.4.1 Algoritmo Dijkstra

Para la planificación de rutas los algoritmos de Dijkstra (Dijkstra, 1959) son los más utilizados debido a su eficiencia y eficacia. El algoritmo de Dijkstra se suele utilizar preferentemente para obtener la solución al problema de la determinación del camino más corto entre dos nodos de un grafo conexo (Ortega et al., 2021).

El algoritmo de Dijkstra funciona dado un grafo a cuyos arcos se han asociado una serie de pesos, se define el camino de coste mínimo de un vértice u a otro v, como el camino donde la suma de los pesos de los arcos que lo forman es la más baja entre las de todos los caminos posibles de *u* a v (Sánchez & Lozano, 2001). Teniendo *X* como nodo inicial y fuente del grafo, un vector *D* de tamaño *N* que se encargará de acumular y guardar al final del algoritmo las distancias de *X* al resto de nodos. En la tabla 1, se muestra el algoritmo Dijkstra.

Tabla 1 Algoritmo Dijkstra

1. Primero se inicializa todas las distancias en el vector *D* con valor infinito, quedando (∞, -). Esto es así porque resultan desconocidas, menos la de x que debe ir en 0 a causa de que la distancia de *x* a *x* seria siempre 0 (0,-).
2. Sea ***a*** = ***x*** (se toma ***a*** como un nodo actual).
3. Se recorren todos los nodos adyacentes de ***a***, excepto los nodos visitados, llamaremos a estos nodos ***vi***.
4. Ahora bien, si la distancia desde ***x*** hasta el nodo adyacente ***vi*** guardada en ***D*** es mayor que la distancia de ***x*** hasta ***a*** sumada a la distancia de ***a*** hasta ***vi***; esta se sustituye con la segunda. Esto es: Si (Di > Da + d(a, v1)) entonces Di = Da + d(a, vi).
5. Marcamos como completo el nodo ***a*** que determinaste en el paso 2.

6. En este paso, tomamos como próximo nodo al de menor valor en ***D*** y vuelves al paso 3 mientras que haya nodos no marcados.
7. Una vez terminado el algoritmo, ***D*** estará completamente lleno.

Las aplicaciones del algoritmo de Dijkstra son muy diversas y de gran importancia en distintas áreas del conocimiento. Algunas aplicaciones son: Entrega de paquetería, aplicaciones de sistemas geográficos, planificadores de rutas entre otras (Sánchez & Lozano, 2001).

Capítulo 3 Estado del arte

En este capítulo se presenta la revisión del estado del arte con el objetivo de presentar un panorama general en los trabajos de investigación que se consideraron relevantes para el desarrollo de este trabajo de investigación.

3.1 Clasificación de los trabajos del estado del arte

Los trabajos relacionados con este proyecto de tesis han sido clasificados en dos categorías: **a) estructura, procesamiento y validación de los datos de transporte GTFS,** en la cual se presentan trabajos que hacen uso de datos GTFS para representar información de transporte público en ciudades. **b) Diseño e implementación de algoritmos de planificación de rutas,** donde se presentan trabajos que hacen uso de modelos que integran un algoritmo de planificación de rutas.

3.2 Aspectos descriptivos de los trabajos del estado del arte

Los trabajos del estado del arte han sido descritos utilizando un conjunto de criterios con la finalidad de realizar una descripción uniforme y consistente de todos y cada uno de los trabajos relacionados. Los criterios utilizados con los siguientes:

- Descripción: en este apartado se muestra una breve descripción del trabajo analizado.
- Objetivo: describe el objetivo principal que persigue el autor con su trabajo de investigación.
- Arquitectura o metodología de solución: describe el conjunto de procedimientos utilizados en el trabajo de investigación analizado.
- Pruebas realizadas: describe el conjunto de pruebas empleadas en el trabajo de investigación.
- Conclusiones: se describe las conclusiones del trabajo analizado

3.3 Estructura, procesamiento y validación de los datos de transporte GTFS (*General Transit Feed Specification*)

A continuación, se presentan los trabajos de investigación clasificados en esta categoría.

3.3.1 Aplicación de planificación de rutas y horarios de transporte público para dispositivos móviles

En este trabajo de investigación (Szincsák & Vágner, 2015) el autor propone una aplicación de planificación de rutas de transporte público para dispositivos móviles, donde se menciona la importancia de tener acceso a la información cuando se realiza un viaje o durante el mismo, ya que busca proporcionar dicho servicio, el cual pueda ser consultado de una manera práctica a través de una aplicación móvil la cual consume datos GTFS que le permite funcionar sin una conexión a internet.

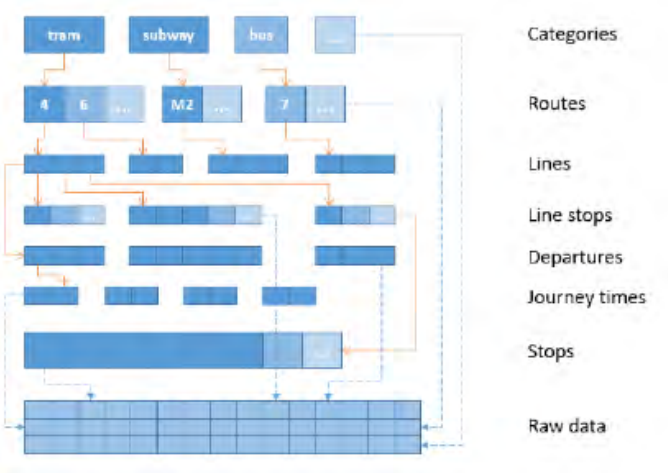

Figura 1 Estructura de datos lógica

El objetivo principal de este trabajo de investigación es proporcionar los servicios de movilidad sin una conexión a internet, que pueda ser consultado a través de una aplicación móvil, y que pueda ser utilizado en cualquier lugar del mundo donde se cuente con datos GTFS.

La metodología propuesta en este trabajo de investigación definió, como un primer paso, obtener los datos GTFS en un tipo de archivo ZIP que posteriormente crea una estructura de datos que les permite a los autores un mejor manejo de la información. La estructura de datos fue compuesta de dos partes. La parte lógica representa cómo están organizados y relacionados los datos y la parte física de la estructura representa cómo son almacenados y accedidos en memoria los datos.

Posteriormente, los autores proponen una conversión de los datos. Inicialmente se realizó una conversión en la que los datos eran procesados y extraídos de una computadora. Posteriormente se añadieron nuevos datos ya que eran de relevancia para la estructura diseñada. Finalmente, los datos fueron procesados en una aplicación hecha en C++ donde se crean las estructuras a partir de los datos.

Finalmente, el algoritmo es ejecutado en el dispositivo móvil. El algoritmo da como resultado una ruta con la hora de llegada más temprana, pero no es necesariamente lo mejor para el usuario. La aplicación contiene un mapa único basado en la API de Android de *Google Maps*. Con el objetivo de que la aplicación funcionará sin conexión a internet, los autores implementaron un mosaico personalizado que carga y renderiza mosaicos de una base de datos local.

Las pruebas realizadas en este trabajo de investigación fueron una representación física de los datos. En un primer método se mapeo todo el archivo de los datos GTFS en el espacio usando una dirección predefinida (que se almacena en la base de datos). Si el primer método no era posible se optaba a una segunda implementación que leía el archivo usando funciones de entrada y salida

normales. En el segundo método, era necesario ajustar los punteros para que coincidieran con las nuevas direcciones. Para que la base de datos mantuviese la compatibilidad binaria, en este trabajo se utilizó una clase envolvente en sistemas de 64 bits que tradujo las direcciones absolutas de 32 bits a punteros relativos.

Las conclusiones del trabajo de investigación de (Szincsák & Vágner, 2015) muestran como los datos GTFS formaron parte importante del proceso que se llevó a cabo. Las estructuras de datos fueron una parte importante dentro del trabajo de investigación, ya que con los datos GTFS eran los encargados de proveer la información necesaria al algoritmo para trazar la planeación de la ruta.

3.3.2 Algoritmo de Planificación de viajes para datos GTFS con estructura NoSQL para mejorar el rendimiento

En este trabajo de investigación (Alzaidi & Vagner, 2021) se propone un algoritmo para planificar viajes. El algoritmo pretende encontrar todas las rutas posibles entre dos puntos cualesquiera haciendo uso de los datos GTFS (*General Transit Feed Specification*). El algoritmo implementa dos tipos de búsqueda, uno a nivel de paradas y el otro a nivel de ruta. El autor (Alzaidi & Vagner, 2021), hace uso de una técnica para reducir la sobrecarga del servidor definiendo una estructura de datos en Redis para almacenar todos los posibles resultados de la búsqueda. Esta técnica se utilizó para evitar que se ejecute el algoritmo con cada solicitud liberando carga de procesamiento.

El objetivo principal de este trabajo de investigación fue reducir de manera considerable el tiempo de procesamiento al momento de generar todas las rutas posibles de un viaje utilizando datos GTFS.

La metodología propuesta en el trabajo de investigación de Alzaidi & Vagner define un modelo para el preprocesamiento de los datos GTFS. El primer paso de la metodología consiste en encontrar las rutas con las distintas paradas considerado en la trayectoria. Como segundo paso se crea una lista con todas estas paradas. En un tercer paso se calcula la distancia entre paradas, utilizando la fórmula *Haversine*. La fórmula calcula la distancia de cada parada con todas las demás.

Las pruebas realizadas en este trabajo de investigación muestran el planteamiento de dos casos concretos. Como primer caso es la implementación de Redis para la estructura de datos y su rendimiento con esta tecnología. Redis es una base de datos en memoria de código abierto y un sistema de almacenamiento en caché de alto rendimiento. Una prueba realizada fue implementar el algoritmo con Redis. Como segundo caso se implementó nuevamente el algoritmo, pero ahora sin Redis. En los casos de prueba se observa cuál era el desempeño en relación al tiempo que

duraba en ejecución el algoritmo. El caso uno resultó claramente con mejor desempeño en relación al caso número dos.

Las conclusiones del trabajo de investigación el autor hace referencia a la importancia que tienen los datos GTFS como un estándar que permita procesar y utilizar eficazmente los datos GTFS en la planificación de viajes ya que es una de las aplicaciones más demandadas por el usuario ya que puede ser de mucha utilidad para objetivos muy precisos, por ello la importancia de mejorar el rendimiento del tiempo necesario al reducir la complejidad de implementación del algoritmo de planificador de rutas.

3.3.3 Análisis de conformidad de las rutas del GTFS y de las trayectorias de los autobuses

En este trabajo de investigación (Queiroz et al., 2019) se realizó un análisis de conformidad sobre las rutas GTFS y los datos de posicionamiento de los autobuses en varias ciudades. Los datos GTFS se enfrentan a dos problemáticas principalmente, por un lado, versiones desactualizadas de los datos ya que algunas agencias de transporte no proporcionan GTFS con la misma frecuencia que cambia el tránsito. Por otro lado, ocurre una discrepancia con los datos de posicionamiento enviados por los autobuses.

El objetivo principal de este trabajo de investigación fue realizar un análisis de conformidad de las rutas GTFS y los datos de posicionamiento de autobuses en varias ciudades. Los autores reportan que se encontraron varias incoherencias en los datos relacionadas con las etiquetas de las rutas GPS y las rutas GTFS. Además de clasificar la conformidad de las trayectorias de los autobuses que aparecen en los datos GTFS. Se enumeraron las principales incoherencias encontradas en el análisis de los datos GTFS.

La metodología propuesta en el trabajo de investigación de Queiroz plantea la obtención de los datos GTFS y GPS de las ciudades a analizar, para posteriormente aplicar un proceso que integra los datos de posicionamiento del ruido con la red de carreteras para obtener posiciones mejoradas, a este proceso se conoce como *Map Matching*.

El segundo paso consiste en convertir los resultados del *Map Matching*. Este proceso genera vectores que representan las trayectorias de los autobuses y las rutas GTFS. Estos vectores mantienen la misma dimensión para permitir comparaciones entre los elementos utilizando métricas como el coseno.

El siguiente paso es la identificación de las rutas GTFS de cada autobús. Para ello, se calculó la similitud del coseno entre los vectores que representan las trayectorias de los autobuses y los

vectores que representan las rutas GTFS, con esto se obtiene una lista de trayectorias de autobuses que se asocia a su ruta más similar.

Como prueba del enfoque propuesto se generaron seis grupos. Cada grupo (representado por un recuadro) indica el nivel de conformidad de la trayectoria del autobús. El eje horizontal se refiere al grado de similitud, que va de 0 a 1. Las casillas situadas por encima del eje horizontal incluyen los casos en los que la etiqueta de la ruta del autobús es conforme a la ruta encontrada. Por otro lado, las casillas situadas por debajo del eje horizontal se refieren a los casos en los que la etiqueta de la ruta del autobús no es conforme con la ruta encontrada.

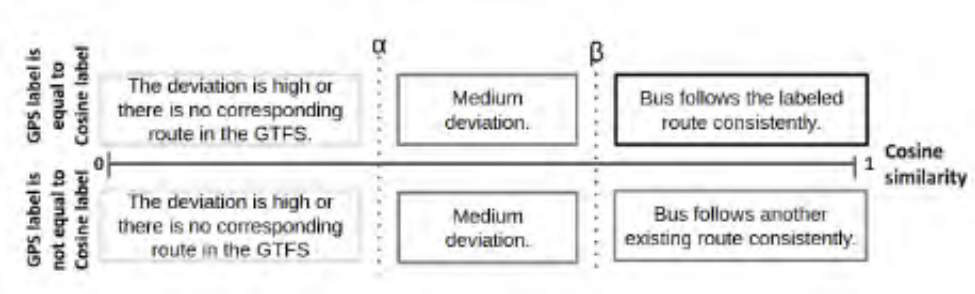

Figura 2 Niveles de conformidad de trayectorias según la similitud con GTFS

En conclusión, en este artículo se demostró que los autobuses que operan en diversas ciudades no siempre siguen la ruta programada, mostrando cierta incoherencia con la información indicada en GTFS. Algunas incoherencias son más graves. Una incoherencia fue la etiqueta de ruta GPS incorrecta. Otras incoherencias, como es el caso que el autobús siga más de una ruta o se desvíe de su ruta predefinida, podrían evitarse con un plan estratégico más eficaz en la creación de las GTFS y la supervisión en tiempo real de los datos del GPS del autobús.

3.3.4 Uso de los datos del GTFS para generar rutas de autobús con tiempo de viaje

En este artículo de investigación (McGlone, 2013) se propone una red de servicio especializada para seleccionar la hora del día y el día de la semana para generar áreas de servicio. Las áreas de servicio de transporte público se basaron en una red de tráfico que fuera factible utilizar. Después de algunas pruebas exhaustivas y el desarrollo de un flujo de trabajo en torno a la herramienta decidió que era demasiado complicado lo que trataba de hacer. El proyecto necesitaba un área de servicio más generalizada. Esta área de servicio requería una estimación de tiempo de viaje razonable.

El objetivo principal de este trabajo de investigación fue el desarrollo de una red de tránsito precisa para el análisis del área de servicio de Delaware haciendo uso de datos GTFS.

El proceso dio inicio descargando los datos GTFS proporcionados por la agencia de transporte. Los datos GTFS se conectaron a un conjunto de datos de la red de tránsito. Los datos GTFS los convirtió en un *shapefile* (un archivo con una forma definida) legible para la herramienta *ArcGIS Network Dataset*.

Una vez terminado, *ArcGIS* obtiene una polilínea para cada ruta de autobús individual. Una polilínea es un tipo de geometría utilizada para representar características lineales en un mapa o en un sistema de información geográfica (GIS). El análisis del área de servicio localiza una entidad de origen y crea un polígono alrededor de esa entidad atravesando el conjunto de datos de la red. El sistema solo toma una de las polilíneas e ignora las otras entidades. Las polilíneas del autobús coincidentes fueron aplanadas.

En un archivo de polilíneas se guardan las rutas de autobús con una mejor estimación de costo del tiempo. La línea en las paradas de autobús se dividió para que las áreas de servicio se generarán alrededor de las paradas. Por último, los tiempos de viaje de los autobuses que están en los datos GTFS se combinan con las rutas de los autobuses. Cada ruta de autobús se divide en viajes individuales. Los tiempos de viaje se generalizan un par de veces para obtener el resultado.

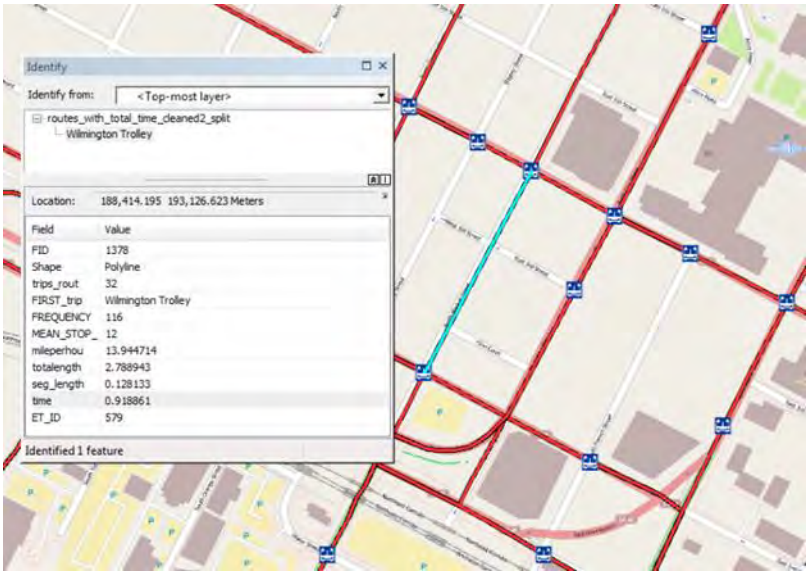

Figura 3 Herramienta que genera un shapefile de las rutas y lo convierte en un dataset con polígonos del área de servicio

Las pruebas del análisis del área de servicio localizan una entidad de origen y crea un polígono alrededor de esa entidad atravesando el conjunto de datos de la red. Cuando la entidad de origen para el área de servicio se encuentra a lo largo de una ruta con polilíneas de autobús coincidentes, solo toma una de las polilíneas e ignora las otras entidades. Esto crea polígonos gruesos que se

duplican en las áreas de servicio. Una alternativa a esto fue aplanar estas polilíneas de autobuses coincidentes y crear un archivo de las rutas para el análisis del área de servicio.

En conclusión, este trabajo de investigación de McGlone señala que su enfoque tiene algunas deficiencias ya que es muy generalizado el proceso. En primer lugar, los valores que proporciona del tiempo no son muy precisos. En segundo lugar, no existe la posibilidad de diferenciar qué ruta de autobús tomará el usuario al no introducirlo en el análisis del área de servicio.

3.3.5 Planificación de rutas en GTFS con Neo4j

En esta investigación (Vágner, 2021) se analizó como Neo4j, el cual funciona como un gestor de base de datos, puede dar soporte a los sistemas de planificación de rutas para el transporte público usando GTFS. En el trabajo no se consideró el nivel de presentación, es decir sólo se incluyen las capas de base de datos y el nivel lógico dentro de la investigación. Además, el autor examinó las herramientas y oportunidades del Neo4j para el almacenamiento y el algoritmo.

El objetivo fue realizar un análisis de Neo4j para dar soporte a los sistemas de planificación de rutas para el transporte público basados en fuentes GTFS.

En la metodología se describe un programa que funciona para la carga de los datos realizado en el lenguaje de programación de Python. Estos archivos se cargaron en la base de datos de forma que cada archivo lo convirtió en un nodo. Los valores de cada fila fueron convertidos en propiedades para los nodos. Para cada nodo se crearon relaciones basadas en la relación con los archivos. El autor cargó toda la información GTFS al Neo4j y como resultado obtuvo una estructura gráfica para probar las herramientas.

En la planificación de rutas el autor utilizó las funciones incorporadas por Neo4j el cual soporta algoritmos para la búsqueda de rutas. En este caso el autor utilizó un algoritmo de la ruta más corta como lo es Dijkstra. Esto para ver los nodos de la ruta entre el inicio y el destino. Con la estructura de datos el algoritmo encuentra la ruta entre dos paradas sin un cambio de línea de transporte.

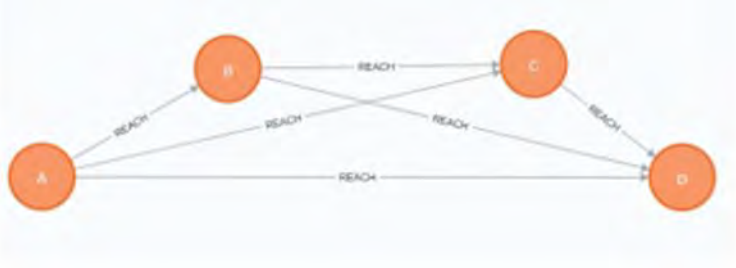

Figura 4 Relaciones entre las diferentes paradas

El autor realizó pruebas al planificador con los feeds GTFS de las ciudades de Debrecen, Budapest y Miskolc. Como resultado, el autor encontró que hay muchos datos GTFS que no contienen todos los archivos opcionales del estándar GTFS. Vágner probó múltiples algoritmos que devuelven una ruta con distintas características según el algoritmo. Resultó que los algoritmos no eran viables para implementarse en una aplicación ya que Neo4j limitaba los recursos. Algunos otros no eran compatibles con la estructura de datos generada.

En conclusión, Neo4 no soporta la planificación de rutas para el transporte público. Vágner encontró una solución a dicho problema, pero la función causa un error de memoria en algunos casos. Neo4j ofrece un navegador que se puede utilizar como una herramienta auxiliar con los datos GTFS, pero no para la planificación de rutas.

3.4 Diseño e implementación de algoritmos para la planeación de rutas

A continuación, se presentan los trabajos relacionados que pertenecen a esta categoría.

3.4.1 Planificación de rutas de transporte público: algoritmo de Dijkstra modificado

En este estudio, los autores (Bozyiğit et al., 2017) determinan las deficiencias del algoritmo de Dijkstra para el proceso de búsqueda del camino más corto. Dijkstra asume que los pesos de las aristas son constantes durante el proceso de búsqueda del camino más corto. Si los pesos de las aristas cambian durante la ejecución del algoritmo, este no reflejará esos cambios y podría producir resultados incorrectos. Para minimizar estas deficiencias, los autores definen un conjunto de reglas del sistema de sanciones. Siguiendo el conjunto de reglas dado, se desarrolla la función de penalización. Esta función de penalización se implementa en el algoritmo de Dijkstra. Por lo tanto, se propone un método de planificación de rutas de transporte público modificando el algoritmo de Dijkstra.

La metodología propuesta intenta evitar largas distancias a pie y múltiples transferencias en el camino. Para lograr esto se definió un conjunto de reglas en el algoritmo Dijkstra. Una primera regla consiste en que si el vértice adyacente del actual solo se accede caminando a él se penalizará en la suma del costo. Una segunda regla menciona que si llega al vértice actual caminando y el vértice adyacente es accesible por una línea de tránsito su penalización de transferencia se agrega al costo del vértice adyacente. Por último, en una tercera regla supone que si una línea de tránsito llega al vértice actual y su vértice adyacente sólo es accesible por una línea de tránsito diferente en este caso la penalización de transferencia se suma al costo del vértice adyacente. Estas reglas mencionadas son las que modifican el algoritmo Dijkstra agregando penalizaciones para reducir las distancias a pie.

Para cada ruta origen-destino, se calculan dos rutas (una para cada método). Además, se calculan los costos (la distancia de la ruta, el número de transferencias, la distancia a pie) de estas rutas. Estas pruebas se muestran en la tabla 2.

Tabla 2 Resultados promedio de los algoritmos

Algoritmo	Tiempo medio de funcionamiento	Distancia media de las rutas	Número medio de transferencias	Distancia media a pie
Algoritmo Dijkstra	242 ms	30.974 km	4.72	0.482 km
Algoritmo Dijkstra modificado	309 ms	33.598 km	2.48	0.396 km

El algoritmo de Dijkstra modificado se evaluó en la red de transporte de Izmir, Turquía y se compararon los resultados del algoritmo modificado contra los resultados del algoritmo original de Dijkstra. Los autores reportan que el algoritmo de Dijkstra modificado es significativamente mejor que el algoritmo de Dijkstra con respecto al número de transferencias y la distancia a pie.

En conclusión, se puede afirmar que, mediante el algoritmo de Dijkstra modificado, se propone una planificación de ruta ideal con respecto al camino más corto del usuario. Los tiempos de ejecución promedio de ambos algoritmos son lo suficientemente rápidos como para ser utilizados en las aplicaciones de transporte público. Aunque el algoritmo de Dijkstra modificado por los autores tiene el peor tiempo al momento de su ejecución.

3.4.2 Planificación de rutas de autobuses compartidos basados en el análisis de datos de transporte

En esta investigación (Kong et al., 2018) se propone un enfoque de dos etapas la cual se compone de la predicción de los requisitos de viaje y la planificación de rutas dinámicas. En primer lugar, se analizan los comportamientos de viaje de los usuarios para obtener características predictivas. En segundo lugar, se diseñó un algoritmo que genera rutas dinámicas y óptimas con destinos fijos en múltiples autobuses.

El análisis del comportamiento del viaje de los usuarios se utilizó para mejorar la predicción de las necesidades de viaje. Posteriormente, los resultados de la predicción y las propiedades de las estaciones se combinaron para obtener las rutas óptimas de los autobuses compartidos.

El objetivo que se propone en el trabajo fue un novedoso enfoque de planificación dinámica de rutas llamado *SubBus*. Este enfoque está pensado para que funcione como una especie de taxi colectivo donde prioriza la comodidad del usuario al optimizar una ruta a través de predicciones de rutas.

La metodología que propuso se compone de tres componentes principales: el preprocesamiento de datos, la predicción de las necesidades de viaje y la planificación dinámica de rutas. En el preprocesamiento de datos se incluyen la limpieza de datos, la organización de los datos y el mapeo de los datos. Se extrae información útil de los datos brutos y se eliminan los errores y valores atípicos.

La predicción de las necesidades de viaje se compone de tres operaciones: el análisis del comportamiento de viaje, la extracción de características y la predicción de requisitos. Para la planificación dinámica de rutas determinar los orígenes basándose en la información de la estación y en los requisitos de viaje de los pasajeros. Posteriormente se generó un conjunto de rutas candidatas. Por último, se diseñó un método para la planificación de rutas dinámicas de múltiples autobuses que operan al mismo tiempo. La metodología que uso se muestra en la figura 5.

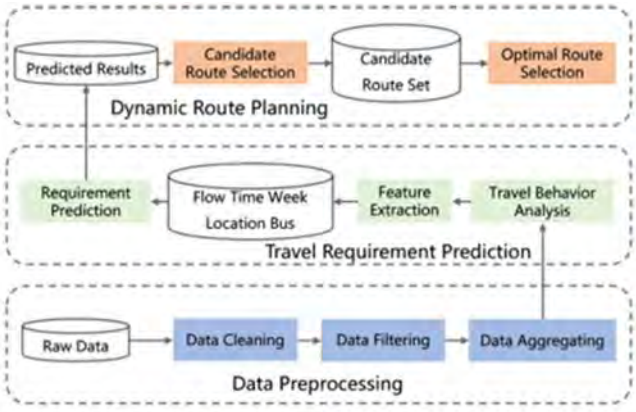

Figura 5 Modelo de Planificador de rutas en autobuses compartidos

Las pruebas las llevó a cabo con datos reales de autobuses compartidos de metro. Demuestra que SubBus supera a otros métodos en la planificación dinámica de rutas de transporte público tradicional. El estudio se llevó a cabo para demostrar la eficiencia y eficacia en el conjunto de datos de autobuses compartidos. Las pruebas arrojaron incrementos o disminución de usuarios

según los días, ya que no es lo mismo un lunes en un pico del día a un fin de semana con una baja demanda de usuarios.

En conclusión, los autores demuestran que su enfoque puede generar rutas efectivas para optimizar el estado de operación de los autobuses compartidos. Como resultado, se muestran rutas de planificación con una distancia de operación más corta y la comodidad del usuario en transportes compartidos.

3.4.3 Algoritmo de consulta de la ruta de tránsito del autobús basado en el algoritmo de la hormiga

En este estudio (Li et al., 2010) se propone un algoritmo de consulta de la ruta de tránsito del autobús basado en los tiempos de menor transferencia. Se basó en los algoritmos de Dijkstra y el de la hormiga para la consulta de las paradas del autobús. Este trabajo hace referencia a una analogía de la hormiga que busca comida deja una hormona en el camino que a su vez funciona como una señal guía para las hormigas que vienen detrás.

El objetivo fue la optimización de la selección de la ruta de viaje del autobús que le permita encontrar los menores tiempos de transferencia y paradas del autobús.

En cuanto a la metodología define 3 procesos para el algoritmo de consulta de una ruta en autobús. Como un primer proceso analiza la intensidad de las paradas de cada línea de autobús. Posteriormente selecciona el tamaño inicial de su población el cual lo representa como un número entero. Si una hormiga pasa el tiempo de transferencia 4 esa hormiga morirá.

Si las hormigas terminan su viaje pasarán a un segundo proceso. Después de que todas las hormigas hayan completado su recorrido, se actualiza la cantidad de feromonas en cada arista. Las aristas que forman parte de rutas más cortas obtienen una cantidad mayor de feromonas. Los tiempos de transferencia de las hormigas y el número de paradas se refresca según la intensidad de las feromonas de la hormiga en la línea. En un tercer proceso la última hormiga del grupo que busca comida (punto de destino) sigue la ruta de viaje con la consulta de las feromonas del grupo de hormigas que dejaron el rastro de la ruta (viaje en autobús) con un origen y un destino.

Las pruebas realizadas fueron hechas en China. Se utilizó el algoritmo de selección de rutas implementado en un sistema de elección de la ruta del autobús basado en la consulta de la parada. Para realizar la consulta de una ruta de viaje el usuario solo tiene que introducir el origen, el destino y los tiempos máximos de transferencia del autobús. Estas pruebas se realizaron con 28 líneas de autobús y un total de 205 paradas de autobuses. En la siguiente figura se muestra el resultado de la consulta de una ruta de viaje.

Figura 6 Sistema de elección de rutas de autobús

Los autores concluyen que el algoritmo reduce la demanda de espacio de memoria debido a su mejor capacidad de optimización de la ruta. Esto ajusta mejor la elección del viaje en autobús de los usuarios. Con un mecanismo de actualización en el algoritmo logra la optimización de la ruta en un autobús. La eficiencia y la precisión dependen del tamaño de la población inicial y de la tasa de desvío de la ruta.

3.5 Conclusiones del estado del arte

A continuación, se presenta una tabla comparativa que muestra el título de cada uno de los trabajos analizados, una descripción breve, el algoritmo utilizado por el trabajo de investigación y finalmente una breve descripción de la aportación de cada uno de los trabajos analizados.

Tabla 3 Conclusiones del estado del arte

TÍTULO	DESCRIPCIÓN	ALGORITMO	APORTACIÓN
Aplicación de planificación de rutas y horarios de transporte público para	Una aplicación de planificación de rutas de transporte público para dispositivos móviles que funcione sin una conexión a	Estructura de datos lógica y física	El sistema planteado logra planificar rutas con los datos almacenados en el dispositivo móvil.

dispositivos móviles (Szincsák & Vágner, 2015).	internet utilizando datos GTFS.	Conversión de los datos GTFS Se ejecuta el algoritmo en el dispositivo	
Algoritmo de Planificación de viajes para datos GTFS con estructura NoSQL para mejorar el rendimiento (Alzaidi & Vagner, 2021).	Un algoritmo para planificar viajes, que pretende encontrar todas las rutas posibles entre dos puntos, haciendo uso de los datos GTFS.	Estructura de datos con Redis Encontrar las rutas Listar rutas Calcula la distancia entre paradas	El algoritmo mejoró la velocidad de procesamiento de la información utilizando la herramienta Redis.
Análisis de conformidad de las rutas del GTFS y de las trayectorias de los autobuses (Queiroz et al., 2019).	Un análisis de conformidad de las rutas GTFS y los datos de posicionamiento de los autobuses	Obtención de los datos GTFS Y GPS Integra los datos de procesamiento con la red de Carreteras Genera los vectores de trayectorias	Realizó una clasificación de la conformación de rutas GTFS de trayectorias, con las principales incoherencias encontradas en las rutas.
Uso de los datos del GTFS para generar rutas de autobús con tiempo de viaje (McGlone, 2013).	La investigación fue sobre un área de servicio más generalizada. Esta área de servicio requería una estimación de tiempo de viaje razonable.	Estructura de datos Crea polígonos alrededor del área de origen. Selecciona una de las polilíneas que contiene la ruta de autobús. Los tiempos de viaje de los autobuses los combina con las rutas para obtener una ruta de menor tiempo.	Este trabajo realizó el desarrollo de una red de tránsito precisa para el análisis del área de servicio haciendo uso de datos GTFS.

Planificación de rutas en GTFS con Neo4j (Vágner, 2021).	Analizo cómo Neo4j que funciona como un gestor de base de datos da soporte a los sistemas de planificación de rutas para el transporte público usando GTFS.	Realizó una estructura de datos y una capa lógica donde implementó el algoritmo Dijkstra para una ruta más corta.	Realizó un análisis de Neo4j para dar soporte a los sistemas de planificación de rutas para el transporte público basados en fuentes GTFS.
Planificación de rutas de transporte público: algoritmo de Dijkstra modificado (Bozyiğit et al., 2017).	En este estudio se determinan las deficiencias del algoritmo de Dijkstra. Definen un conjunto de reglas del sistema de sanciones. Proponen un método de planificación de rutas de transporte público modificando el algoritmo de Dijkstra.	Se usó el algoritmo Dijkstra incluyendo algunas reglas. Estas reglas son penalizaciones en el coste del tiempo dependiendo si el acceso a una parada solo se accede a pie.	Proponen un sistema de planificación de rutas de transporte público modificando el algoritmo de Dijkstra. El algoritmo de Dijkstra modificado es significativamente mejor que el algoritmo de Dijkstra con respecto al número de transferencias y la distancia a pie.
Planificación de rutas de autobuses compartidos basados en el análisis de datos de transporte (Kong et al., 2018).	Propone un enfoque de dos etapas la cual se compone de la predicción de los requisitos de viaje y la planificación de rutas dinámicas. En primer lugar, analizar los comportamientos de viaje de los usuarios para obtener características predictivas. En segundo lugar, diseñó un algoritmo que genera rutas dinámicas y óptimas con destinos fijos en múltiples autobuses.	Se compone de tres componentes principales: El preprocesamiento de datos La predicción de las necesidades de viaje La planificación dinámica de rutas.	Es un enfoque de planificación dinámica de rutas llamado SubBus. Está pensado para que funcione como una especie de taxi colectivo donde prioriza la comodidad del usuario al optimizar una ruta a través de predicciones de rutas.
Algoritmo de consulta de la ruta de tránsito del autobús basado en el algoritmo de la hormiga (Li et al., 2010).	En este estudio se habla de un algoritmo para consultas de rutas de tránsito del autobús basado en los tiempos de menor transferencia.	Se basó en los algoritmos de Dijkstra y el de la hormiga para la consulta de las paradas del autobús.	El sistema propone una optimización de la selección de la ruta de viaje del autobús que le permita encontrar los menores tiempos de transferencia y paradas del autobús.

Desarrollo de un algoritmo de planificación de rutas en ciudades utilizando GTFS	En esta investigación se pretende el desarrollo de un planificador de rutas de transporte público con tecnologías de código libre.	Se planea resolver en tres fases: Diseño de una estructura de datos. Desarrollo del algoritmo de planeación de rutas. Despliegue de la ruta en un mapa.	Se pretende el desarrollo de un prototipo que permita la planificación de rutas de transporte público en ciudades utilizando especificaciones GTFS y tecnologías de código abierto que puedan incorporarse a proyectos de ciudades inteligentes.

Los datos GTFS pueden ser de mucha utilidad al momento de realizar desarrollos de movilidad. En el estado del arte podemos encontrar diferentes enfoques que utilizan GTFS para diversos propósitos. Estos enfoques nos brindan un panorama de lo que se puede realizar con los datos GTFS. En la revisión del estado del arte nos percatamos que es de utilidad un planificador de rutas basado en los datos GTFS que permita integrarse a proyectos de movilidad.

Capítulo 4 Metodología de solución

En este capítulo se presenta la metodología propuesta para el desarrollo de esta tesis. Este capítulo se divide en dos apartados importantes: metodología de solución y arquitectura del prototipo.

4.1 Metodología de solución

En esta tesis se desarrolló un prototipo que permite el consumo de datos GTFS de la Ciudad de México y utiliza esta información como entrada para un planificador de rutas, el cual determina la ruta más corta para alcanzar un destino en la ciudad utilizando transporte público. Los datos GTFS son archivos de texto que contienen la información de las agencias de tránsito de transporte público de una ciudad. En este contexto, se propone el desarrollo de un planificador de rutas de transporte público utilizando tecnologías *open source,* lo cual permite su reutilización en diversos proyectos de movilidad inteligente.

La figura 7 muestra la metodología propuesta para el desarrollo del prototipo. La metodología de solución consta de cuatro fases principales: la fase de modelado de datos, la fase del desarrollo del algoritmo planificador de rutas, la fase del desarrollo de la interfaz de usuario y la fase de la comunicación entre el algoritmo y la interfaz.

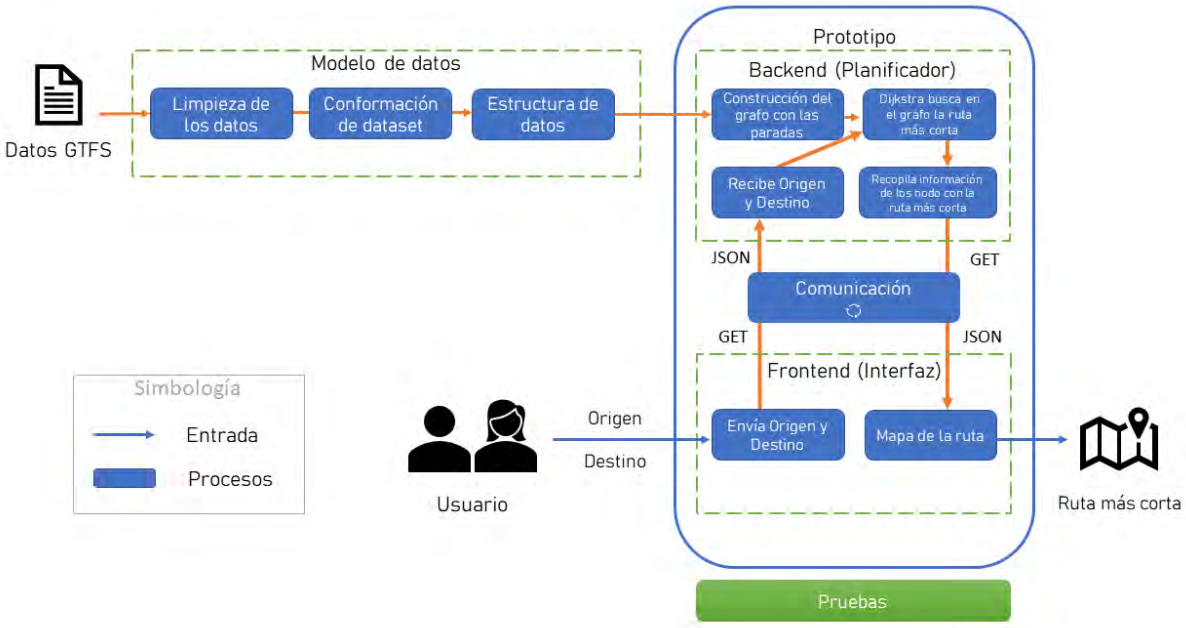

Figura 7 Metodología de solución

A continuación, se describe cada una de las fases que comprende la metodología de solución propuesta para generar un planificador de rutas en ciudades utilizando GTFS y tecnologías abiertas. Es importante comentar que cada una de las fases es explicada en detalle en el capítulo 5 de este documento de tesis.

4.1.1 Fase de modelado de datos

Un modelo de datos es una forma organizada de almacenar y manipular información. La importancia de un buen modelo de datos radica en la eficiencia y el rendimiento del planificador de rutas de transporte público. Un modelo de datos bien diseñado y optimizado permite realizar búsquedas rápidas y cálculos en menor tiempo.

La generación de un modelo de datos es de suma importancia para este proyecto, ya que los datos son la principal fuente que alimenta el algoritmo planificador de rutas. La fase de modelado de datos permitió analizar los datos GTFS con los que se trabajó y con este análisis determinar qué datos eran de utilidad para el objetivo de esta tesis. El modelo de datos permitió que la construcción de un grafo con la información GTFS sea fácil de administrar, ya que únicamente contendrá los datos necesarios para planificar rutas. Con esto nos aseguramos de no cargar información innecesaria o que no sea de utilidad en el grafo, lo cual permitirá mejorar los tiempos de respuesta.

4.1.2 Fase de desarrollo de la lógica del algoritmo planificador de rutas

En esta fase de la metodología de solución se desarrolló toda la lógica del prototipo que permite calcular la ruta más corta entre una parada de origen y una parada destino. El algoritmo de ruta más corta fue implementado en el lenguaje de Python. El algoritmo es alimentado con los datos GTFS para construir el grafo con la información de las paradas del transporte público. En este grafo se implementó un algoritmo de búsqueda Dijkstra, el cual tuvo que ser modificado para permitir la planificación de rutas de transporte público utilizando datos GTFS. Para implementar el algoritmo de Dijkstra en la planificación de rutas de transporte público utilizando datos GTFS, se necesitan algunas modificaciones y consideraciones específicas debido a la naturaleza de los datos. Una de las consideraciones es el modelo del grafo ya que con el caso de los datos GTFS, el grafo debe representar la red de transporte público. Donde los nodos del grafo representan las paradas del transporte público, y las aristas representan las rutas y conexiones entre paradas. Otra modificación realizada al algoritmo fue la incorporación de transbordos entre diferentes rutas o servicios de transporte público para llegar a un destino. El algoritmo debe considerar estas posibilidades y evaluar diferentes opciones de transbordo para encontrar la ruta más corta.

El algoritmo de búsqueda recibe el nodo origen y el nodo destino y recorre todos los posibles caminos hasta encontrar el camino más corto dentro del grafo. Posteriormente, el algoritmo realiza una lista con la colección de los nodos desde el nodo origen hasta el nodo destino. Es importante que el *backend* tuvo que cumplir con los siguientes requisitos: desarrollarse como un componente separado de la interfaz de usuario (*frontend*) para permitir que el planificador pueda ser utilizado en forma independiente por un desarrollador que desee utilizar el planificador de rutas. El *backend* encargado de planificar rutas de transporte público fue desarrollado con tecnologías *open source*, lo que garantiza la transparencia y accesibilidad del desarrollo. Al basarse en tecnologías de código

abierto, permite que cualquier persona interesada pueda acceder al código fuente, realizar modificaciones, implementar mejoras y contribuir al progreso del proyecto. Esta apertura en el desarrollo fomenta la innovación y promueve una mejora continua en la calidad de la planificación de rutas. La elección de tecnologías *open source* trae consigo varios beneficios. Por un lado, reduce significativamente los costos asociados con licencias de software, ya que beneficia en la gestión de recursos económicos para el desarrollo del prototipo. Por otro lado, al emplear tecnologías de código abierto, se favorece la libertad de uso y distribución del software, lo que permite que otros proyectos y sistemas puedan aprovechar y enriquecer la funcionalidad de la solución de planificación de rutas.

4.1.3 Fase de desarrollo de la interfaz de usuario

En esta fase de la metodología de solución se desarrolló la interfaz de usuario. Esta interfaz consiste en la visualización en un mapa de la ruta generada por el algoritmo. La interfaz está compuesta por un formulario que recibe un origen y un destino, un botón para generar la ruta y un mapa donde se visualiza la ruta sugerida por el planificador. Es importante que la interfaz tuvo que cumplir con los siguientes requisitos: desarrollarse como un componente separado de la lógica del algoritmo (*Backend*) para permitir que la interfaz pueda ser utilizada en forma separada por un desarrollador que desee utilizar su propio planificador de rutas. La interfaz fue desarrollada con el uso de tecnologías *open source* que garantiza la transparencia del desarrollo. Las tecnologías *open source* permiten que cualquiera pueda acceder al código fuente, realizar modificaciones, mejoras y contribuir a la evolución de la interfaz, lo que favorece la innovación y la continua mejora del proyecto. Además, al basarse en tecnologías de código abierto, se reducen los costos asociados con licencias y se promueve la libertad de uso y distribución del software.

4.1.4 Fase de comunicación entre el algoritmo y la interfaz

En esta fase de la metodología de solución se realizó la comunicación entre el algoritmo y la interfaz de usuario. La comunicación se realizó a través del protocolo HTTP y por medio de peticiones cliente-servidor. En estas peticiones el usuario solicita una ruta al algoritmo enviándole una petición. Esta petición contiene el origen y el destino de los puntos para los cuales se desea obtener la ruta más corta. El servidor a su vez recibe esta petición y la procesa devolviendo la lista de paradas que comprende la ruta más corta encontrada desde el origen al destino solicitado por el cliente.

4.2 Arquitectura del prototipo

La arquitectura implementada en el prototipo se basó en tecnologías *open source* para la planificación de rutas de transporte público utilizando datos GTFS. Esta elección aseguró la transparencia y accesibilidad del desarrollo, permitiendo que cualquier persona interesada pueda

acceder al código fuente, realizar modificaciones, implementar mejoras y contribuir al progreso del proyecto. La arquitectura se diseñó de manera modular y flexible, lo que facilita la integración de nuevas funcionalidades y la adaptabilidad a diferentes entornos de transporte público. Con el uso de esta arquitectura se cumple uno de los objetivos buscados en esta tesis: desarrollar un planificador de rutas con tecnologías de código abierto. Uno de los aspectos relevantes en la arquitectura es que se buscó la separación del *backend* del *frontend* de la aplicación, permitiendo que el planificador sea fácilmente exportable e integrado en proyectos de movilidad.

La arquitectura está compuesta por dos elementos importantes. En el *backend* se implementa la lógica del algoritmo desarrollado para planificar rutas. El desarrollo del *backend* fue programado en el lenguaje de Python y con ayuda del *Framework Flask*. El uso de Python y *Framework Flask* tuvo el objetivo de brindar una sólida y eficiente base tecnológica para el desarrollo del *backend* para el planificador de rutas de transporte público. Python, como lenguaje de programación, es reconocido por su legibilidad, facilidad de uso y versatilidad, lo que permitió un desarrollo rápido y claro del planificador de rutas. Por su parte, *Flask*, como un *framework* web de Python, proporcionó las herramientas necesarias para crear una arquitectura web ágil y escalable. Al ser un *framework* ligero, *Flask* se ajusta muy bien a proyectos que requieren un enfoque modular y sencillo, lo que encajó perfectamente con las necesidades del *backend* de planificación de rutas.

Por otra parte, tenemos el *frontend* que permite visualizar la ruta encontrada por el planificador. El desarrollo del *frontend* fue programado en el lenguaje de JavaScript y con ayuda del *Framework React*. El uso de JavaScript y *React* tuvo como objetivo proporcionar una experiencia de usuario interactiva y receptiva en el *frontend* para visualizar la ruta encontrada por el planificador de transporte público. Al emplear JavaScript, un lenguaje ampliamente utilizado en el desarrollo web, se logró la implementación de funciones dinámicas y un manejo ágil de la interfaz de usuario. Por su parte, *React*, es un moderno y popular *framework* de JavaScript que facilitó la creación de componentes reutilizables y una estructura modular en el *frontend*. Esto permitió una mayor eficiencia en el desarrollo, así como la capacidad de mantener y ampliar el sistema con mayor facilidad.

La comunicación entre ambos elementos se realiza a través del protocolo HTTP y en formato JSON lo que garantiza una interoperabilidad fluida y eficiente entre el *frontend* y el *backend*. Al utilizar el protocolo HTTP, se establece una comunicación basada en solicitudes y respuestas, permitiendo que el *frontend* pueda enviar peticiones al *backend* para obtener la información necesaria, como la ruta planificada, y recibir las respuestas correspondientes.

El formato JSON (*JavaScript Object Notation*) se emplea para estructurar y transmitir los datos entre el *frontend* y el *backend* de una manera ligera y sencilla de interpretar tanto para los sistemas como para los desarrolladores. JSON se ha convertido en un estándar para el intercambio de datos en aplicaciones web debido a su facilidad de lectura y escritura, así como su compatibilidad con

múltiples lenguajes de programación. Con HTTP y JSON, la comunicación entre el *frontend* y el *backend* es independiente del lenguaje o tecnología utilizada en cada componente, lo que brinda una mayor flexibilidad y facilita la integración con otros sistemas o servicios web.

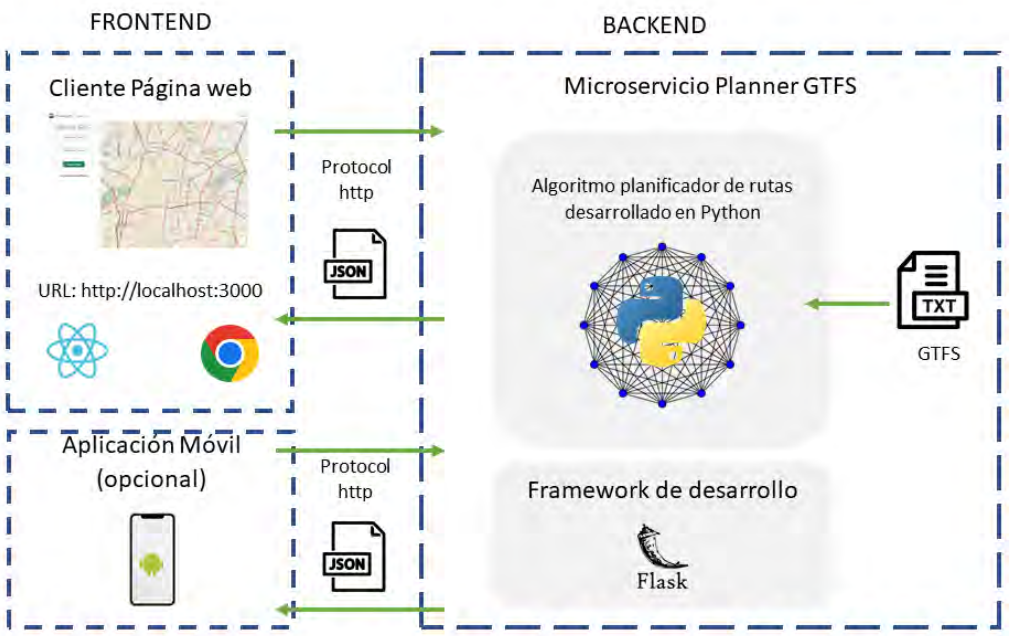

Figura 8 Arquitectura utilizada en el prototipo

A continuación, se detalla cada una de las partes que comprende la arquitectura utilizada para este proyecto de investigación.

4.2.1 Backend del prototipo

En este apartado de la tesis se describe el desarrollo del *backend* para el prototipo del planificador de rutas de transporte público utilizando datos GTFS. El objetivo principal de este proyecto es proporcionar a los usuarios una herramienta que les permita planificar rutas de transporte público de un punto de origen hasta un punto destino, utilizando la información disponible en los datos GTFS. El *backend* se encarga de procesar y analizar estos datos para generar una recomendación de la ruta más corta en relación a la distancia desde el origen hasta el destino. El *backend* del prototipo se basa en una arquitectura de servicio web que permite la comunicación con el *frontend* y el acceso a los datos GTFS.

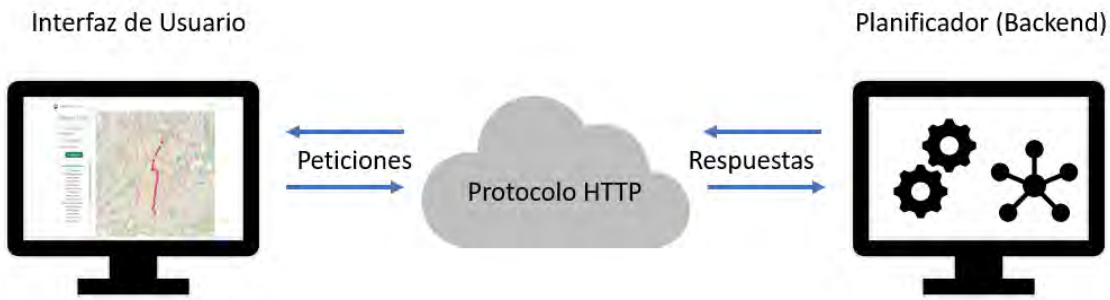

Figura 9 Conformación del backend del prototipo

4.2.2 Frontend del prototipo

Para el desarrollo del *frontend* del prototipo tiene como objetivo principal proporcionar a los usuarios una interfaz intuitiva y amigable que les permita interactuar con el sistema de planificación de rutas. El *frontend* fue desarrollado utilizando *React*, una biblioteca de JavaScript para construir interfaces de usuario, y se comunica con el *backend* a través del protocolo HTTP para obtener los datos necesarios y mostrar las recomendaciones de rutas al usuario.

Figura 10 Frontend del prototipo

4.2.3 Integración del planificador

La integración del planificador de rutas desarrollado con una arquitectura de microservicios permite reutilizar o incorporar este prototipo en otros desarrollos de movilidad. El planificador se puede integrar de dos formas. La primera forma de integrarlo a otro desarrollo es desplegando el planificador como lo conocemos y apuntar a la URL donde se desplegó para comenzar hacer

peticiones de rutas. La segunda forma es descargar el prototipo y realizar modificaciones al código para adaptarlo a las necesidades del colaborador añadiendo mejoras o funcionalidades específicas.

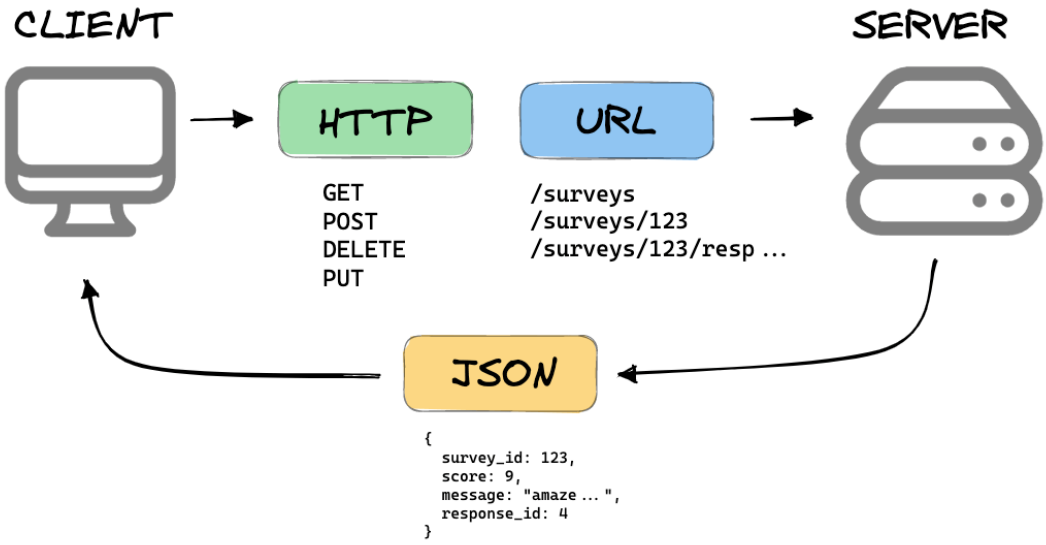

Figura 11 Integración entre el backend y el frontend

Capítulo 5 Desarrollo del Planner GTFS

En este capítulo se detallan las actividades realizadas durante el desarrollo del planificador de rutas de transporte público utilizando datos GTFS de la Ciudad de México.

5.1 Desarrollo del Planner GTFS

El prototipo desarrollado en esta tesis, denominado Planner GTFS fue desarrollado en cuatro fases. La fase 1 es el **modelado de datos** que fue de suma importancia para organizar, almacenar y manipular la información de los datos GTFS. La fase 2 es el **desarrollo e implementación del algoritmo** donde se construyó un grafo con la información GTFS haciendo uso del modelo de datos e implementando un algoritmo de búsqueda en el grafo. La fase 3 es el **desarrollo de la interfaz de usuario** que consistió en generar una interfaz con un mapa donde se visualizan las rutas encontradas por el algoritmo. Por último, para la fase 4 se realizó la **comunicación entre componentes** del algoritmo y la interfaz para completar el prototipo de esta tesis.

5.2 Implementación del modelado de datos

Para conformar el modelo de datos se realizaron las siguientes tareas: a) obtención de los datos GTFS, b) limpieza de los datos seleccionando información de utilidad, c) conformación de un nuevo *dataset* con los datos listos y finalmente, d) construcción de la estructura de datos para formar un grafo con la información GTFS.

a) Obtención de los datos

Los datos GTFS se obtuvieron a través de la página datos.cdmx.gob.mx por medio de la Secretaría de Movilidad (SEMOVI) que es una entidad pública del gobierno de la Ciudad de México. Los datos GTFS se actualizan constantemente por la propia institución de movilidad. Los datos GTFS se pueden descargar sin ninguna restricción y están disponibles para utilizarlos ya que son datos libres.

Figura 12 Descarga de datos GTFS

Los datos GTFS de la Secretaría de Movilidad (SEMOVI) están disponibles en un archivo con extensión .ZIP que en su interior contiene archivos de texto (.txt) con los datos de transporte público de la ciudad. Los archivos de texto contienen información como es el total de paradas de transporte público de la ciudad, las agencias de transporte público, las rutas que realizan cada trasporte público y con qué frecuencia se realiza cada viaje por mencionar algunos datos contenidos en los archivos de texto GTFS.

b) Limpieza de los datos

Para la limpieza de los datos se analizó el conjunto de archivos de texto (.txt) que previamente se obtuvo del sitio web de la Secretaría de Movilidad. Este conjunto de archivos está comprendido por ocho documentos y cada documento puede ser procesado como una base de datos, donde cada documento sería el equivalente a una base de datos. Estos documentos contienen la información de las agencias, el calendario laboral del transporte público, las frecuencias, los rutas y paradas de cada transporte público, los tiempos de paradas y los trayectos completos. A pesar de que GTFS es un estándar los datos cuentan con inconsistencias, por ello es importante realizar una limpieza de los datos para conocer la información que es de utilidad para esta tesis.

Como primer paso, se analizó cada uno de los archivos de texto GTFS para determinar la información que podría ser de utilidad para poder construir el grafo. Como se ha comentado, los archivos GTFS se componen de 8 documentos, los cuales contienen la información de las agencias, las rutas que existen, las paradas de cada una de las rutas, las trayectorias, los tiempos de paradas de los transportes, la información del calendario de los días laborables y todos los puntos geográficos por lo que transita el transporte público de la Ciudad de México.

Los archivos GTFS se relacionan entre sí, lo cual facilita el procesamiento de los archivos como tablas de una base de datos. Para el modelo de datos se identificó la estructura de los datos GTFS. Este modelo se recreó a través de las relaciones que existen entre las diferentes tablas de los datos GTFS. Para lograr esto se analizaron los tipos de datos con los que están conformados cada uno de los campos.

La Figura 13 muestra el modelo de datos y las relaciones encontradas entre sus documentos, esto es de gran utilidad al momento de realizar consultas al modelo. Algunas de las relaciones encontradas pueden ser de utilidad para el objetivo del planificador de rutas, como son las trayectorias, las rutas y las paradas del transporte público.

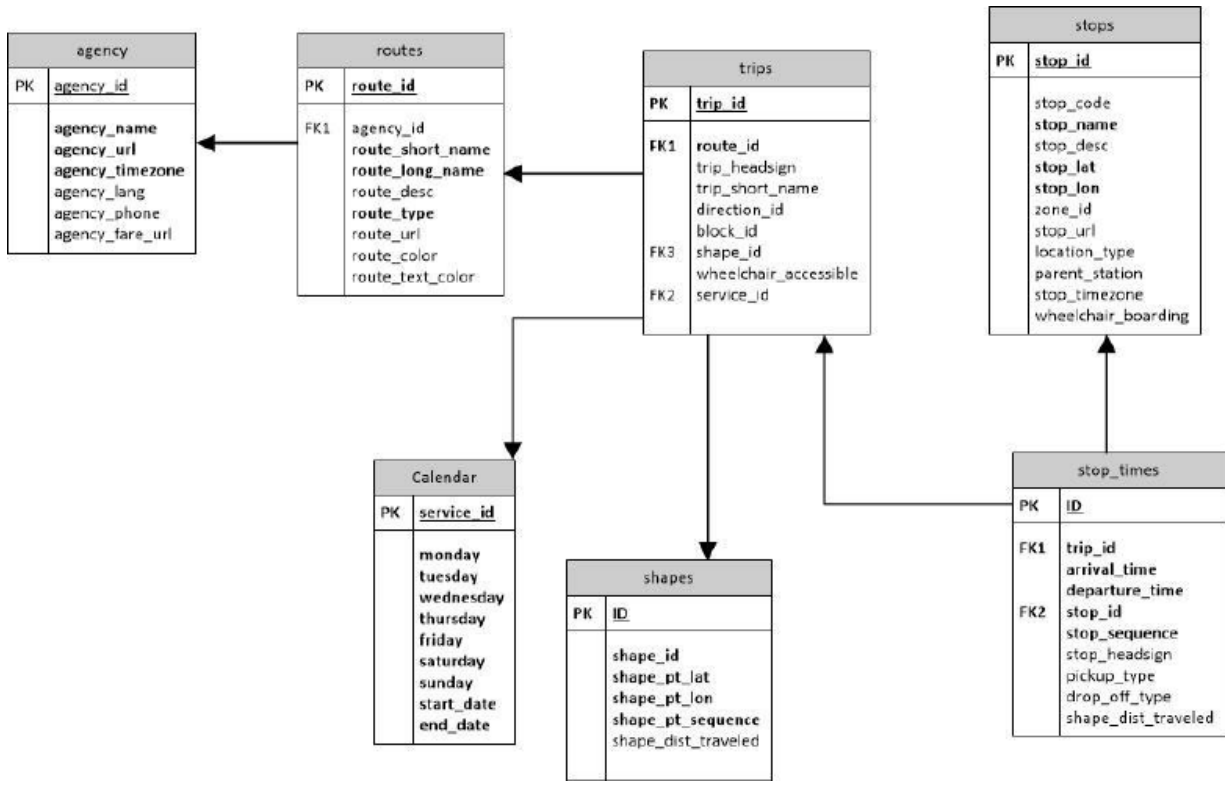

Figura 13 Diagrama de relación de los datos GTFS

GTFS es un estándar creado por Google, sin embargo, es necesario tener en cuenta que la información para instanciar los modelos GTFS es proporcionada por las agencias de tránsito que pueden introducir errores en su captura. De esta forma, los archivos GTFS pueden contener datos inconsistentes, datos duplicados o incluso datos faltantes. Por ello es importante realizar un análisis y/o limpieza de los datos con los que se trabaja.

En el análisis de los datos GTFS de la Ciudad de México nos percatamos que al hacer una consulta simple de alguna de las rutas se encontraban múltiples datos duplicados, por ejemplo, múltiples definiciones de una misma ruta de transporte. Por esta razón fue necesario realizar una limpieza de los datos para eliminar los datos duplicados en los datos GTFS. Para ello fue necesario la construcción de una consulta donde se seleccione los registros distintos de las tablas que se utilizarán para la construcción del grafo cumpliendo las condiciones de ambos campos como se muestra en la Figura 14.

```
SELECT DISTINCT
        stops.stop_id,
        stops.stop_name,
        stops.stop_lat,
        stops.stop_lon,
        routes.route_id,
        routes.route_long_name
FROM
        stops,
        stop_times,
        trips,
        routes
WHERE
        stops.stop_id = stop_times.stop_id
                AND trips.trip_id = stop_times.trip_id
                AND routes.route_id = trips.route_id;
```

Figura 14 Construcción de la consulta para eliminar registros duplicados

De esta forma, solo se conservan registros únicos de cada uno de los transportes, rutas y paradas. Esto es de utilidad ya que al construir el grafo no habrá repetición de nodos como sucedía con los datos GTFS sin filtrar. El siguiente paso es conformar un nuevo *dataset* que contenga toda la información que es necesaria para este trabajo de tesis.

c) Conformación del dataset

Una vez realizada la limpieza de los datos se conformó un nuevo *dataset* que contiene los datos necesarios para calcular una ruta entre dos puntos de la ciudad. Este nuevo conjunto de datos sirvió para extraer la información necesaria y poder generar un grafo con el cual implementar un algoritmo de búsqueda.

El *dataset* fue conformado con la información de los archivos de stops, stop_times, trips y routes. Con esta información se conformó un solo conjunto de datos donde que contiene la información necesaria para construir el grafo. La conformación del *dataset* se realizó en una base de datos como es SQLite por medio de consultas SQL.

La consulta SQL está conformada de cuatro partes importantes. La primera parte implica realizar una inserción dentro de una tabla que contendrá los siguientes campos: stop_id, stop_name, stop_lat, stop_lon, route_id, agency_id, route_long_name y route_color. La segunda parte de la consulta implica seleccionar únicamente los registros que sean distintos para no tener registros duplicados. La tercera parte de la consulta implica indicar que la información será obtenida de los archivos GTFS considerando los campos stops, stop_times, trips y routes. Por último, se indica que debe cumplir la condición que los IDs de los diferentes archivos sean iguales para asegurarse de que se trate del mismo registro. La consulta completa se puede observar en la figura 15.

```
INSERT INTO stops_routes (stop_id, stop_name, stop_lat, stop_lon, route_id, agency_id, route_long_name, route_color)
SELECT DISTINCT
        stops.stop_id,
        stops.stop_name,
        stops.stop_lat,
        stops.stop_lon,
        routes.route_id,
        routes.agency_id,
        routes.route_long_name,
        routes.route_color
FROM
        stops,
        stop_times,
        trips,
        routes
WHERE
        stops.stop_id = stop_times.stop_id
                AND trips.trip_id = stop_times.trip_id
                AND routes.route_id = trips.route_id;
```

Figura 15 Consulta para conformación del dataset

El resultado que se obtiene de ejecutar la consulta es un *dataset* que contiene todos los registros con la información en un solo documento lo cual facilita el acceso a la información al momento de la construcción de un grafo y hacer consultas de información. El resultado de la conformación del conjunto de datos se puede observar en la figura 16.

	stop_id	stop_name	stop_lat	stop_lon	route_id	agency_id	e_long_n	route_color
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
1	05SE12-CONTITUCION	CETRAM Constitución de 1917	19.34502	-99.06467	CMX05SE12	RTP	Constit...	76B729
2	05SE12-TLAHUACETRAM	CETRAM Tláhuac - Andén A	19.28588	-99.01453	CMX05SE12	RTP	Constit...	76B729
3	05SE12-CNOREALVERA1	Camino Real y Calle Veracruz	19.32474	-99.05026	CMX05SE12	RTP	Constit...	76B729
4	05SE12-CNOREALVERA0	Camino Real y Calle Veracruz	19.32473	-99.0501	CMX05SE12	RTP	Constit...	76B729
5	05SE12-CNOREALMIRON1	Camino Real y Salvador Díaz Mirón	19.31317	-99.03803	CMX05SE12	RTP	Constit...	76B729

Figura 16 Dataset con la información GTFS

d) Construcción de la estructura de datos para formar el grafo

Se construyó la estructura de datos para proveer al algoritmo con la información del transporte público. Esta estructura de datos permite la construcción del grafo con todas las paradas y los diferentes transportes públicos que se encuentran dentro de GTFS.

En el conjunto de datos se cuenta con información como el id de las paradas, el nombre de las paradas, latitud, longitud, id de la ruta, la agencia a la que pertenece, el nombre de la ruta y el color de la misma. Para la estructura de datos del grafo solo es necesario el nombre de las paradas con su latitud y longitud. Esto sirve para calcular la distancia entre cada una de las paradas de GTFS. En la figura 17 se muestra un ejemplo del *dataset* con los datos necesarios para conformar el grafo.

	stop_id	stop_name	stop_lat	stop_lon
0	0300L3-TENAYUCA	Tenayuca	19.52862	-99.17011
1	0300L3-SANJOSEESCA	San José de la Escalera	19.52272	-99.16556
2	0300L3-PROGRESONA	Progreso Nacional	19.51969	-99.16377
3	0300L3-TRESANEGAS	Tres Anegas	19.51551	-99.16193
4	0300L6-COLBACH1	Colegio de Bachilleres 1	19.51164	-99.19731

Figura 17 Datos para generar el grafo

Para la estructura de datos se generó un subconjunto de datos del *dataset* original que contiene únicamente el origen de la parada y su destino siguiente con su respectiva distancia entre ambas estaciones. Con ayuda de la librería Pandas se manipuló el *dataset* de forma que solo tengamos el origen de la parada y el destino con su respectiva distancia entre ambas paradas de transporte público. Para determinar las distancias entre las paradas se utilizó la librería de *Math*, y más específicamente la función *haversine*. La fórmula del *haversine* es utilizada para conocer distancias geográficas entre diferentes puntos ya que toma en cuenta el radio de la tierra. Se define una función calcular distancia que toma las latitudes y longitudes de dos puntos y devuelve la distancia en kilómetros utilizando la fórmula del *haversine*. Posteriormente se va iterando con cada uno de los datos que conforman el *dataset* y se van guardando en orden para no perder los pares origen y destino.

Las latitudes y longitudes deben estar en grados decimales. Además, es importante tener en cuenta que la fórmula del *haversine* proporciona una distancia en línea recta entre los dos puntos en la superficie de una esfera (la Tierra), por lo que puede haber discrepancias con distancias reales en terreno si los puntos están muy alejados.

De esta forma, podemos implementar esta estructura de datos para la construcción del grafo con los datos GTFS, donde el origen de cada registro será un nodo del grafo y la conexión de este nodo es el destino con su peso asignado entre cada par de nodos. Al asignarle un peso a cada par de nodos se puede conocer cual tienen la menor distancia al realizar la suma total de los pesos. El resultado final del *dataset* para construir el grafo con las paradas del transporte público se muestra en la figura 18.

	Origen	Destino	Distancia
0	01METRO-Pantitlan	01METRO-Zaragoza	1320
1	01METRO-Zaragoza	01METRO-GómezFarías	762
2	01METRO-GómezFarías	01METRO-BoulevardPuertoAéreo	611
3	01METRO-BoulevardPuertoAéreo	01METRO-Balbuena	595
4	01METRO-Balbuena	01METRO-Moctezuma	703

Figura 18 Dataset resultante para la construcción del grafo

5.3 Implementación del algoritmo Dijkstra para datos GTFS

En esta sección se describe el módulo que implementa el algoritmo de planificación de rutas. El algoritmo realiza la búsqueda del camino más corto utilizando la estrategia Dijkstra. Este algoritmo utiliza la información obtenida del modelo de datos GTFS para generar recomendaciones de rutas más cortas en relación a la distancia de un punto a otro punto de la ciudad.

Para el desarrollo del algoritmo se utilizó el *framework Flask* de Python que permite crear entornos virtuales para crear aplicaciones web. Para usar Flask es necesario tener instalado el sistema Python (https://www.python.org). Una vez instalado Python se requiere abrir una terminal de comandos y ejecutar el comando 'python -m venv nombre-Prototipo', el cual crea un entorno virtual y una carpeta con el mismo nombre donde contiene todos los archivos del entorno virtual. Para activar el entorno virtual se realiza ejecutando el comando 'nombre-carpeta\Scripts\activate'. Una vez activado el entorno virtual la instalación del *framework Flask* se realiza con el comando 'pip install flask'. Este comando descargará e instalará todas las librerías necesarias para usar *Flask* como se muestra en la figura 19.

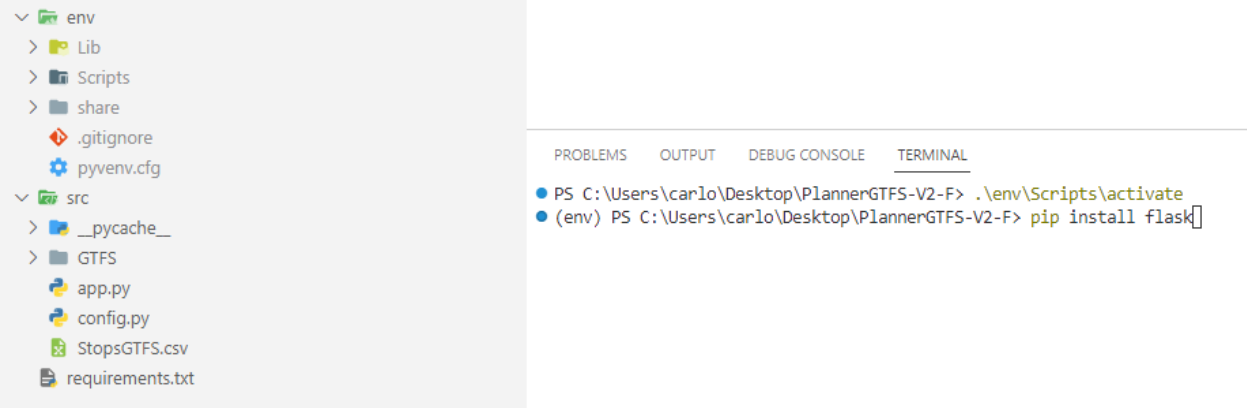

Figura 19 Preparación del entorno virtual del backend

Una vez que se tiene el entorno de trabajo se importan las librerías requeridas para este proyecto de tesis: Pandas para manipulación de conjunto de datos, Networkx para la creación de grafos y Jsonify para generar un resultado en formato JSON.

Una vez importado las librerías necesarias como es *Pandas* que se utiliza para leer el conjunto de datos. El conjunto de datos contiene todas y cada una las paradas que se encuentran dentro de los datos GTFS y sus distancias con los que conecta cada una de las paradas. Posteriormente se procedió a crear un grafo vacío para ir agregando los nodos con la información GTFS. Para esto se utiliza la librería de Networkx y más específicamente la función nx.DiGraph() la cual crea un objeto de tipo multígrafo que posteriormente se agregan nodos desde el conjunto de datos que previamente se ha cargado en memoria.

Después de generar el grafo se agregan los nodos para construir el grafo de la ciudad. Cada nodo corresponde a una parada del transporte público y está parada a su vez conecta con otra parada de transporte. Entre estas paradas hay una distancia. Esta distancia es la que se utilizó para determinar cuál será el camino más corto desde una parada X a una parada Z. Con ayuda de la librería Networkx y la función 'from_pandas_edgelist()' nos permite agregar los nodos desde un archivo. En la función se indican cuatro argumentos: el primer argumento indica el conjunto de datos de donde tomará la información, seguido del segundo argumento que será cada uno de los nodos. El tercer argumento deberá tener la información con la que conecta ese nodo. Por último, el cuarto argumento deberá contener las distancias entre los pares de cada una de las paradas que anteriormente se indicó. En la figura 20 se muestra la implementación de la función from_pandas_edgelist donde se indican los parámetros necesarios para construir el grafo. El parámetro '*df_stops*' representa el conjunto de datos, de este conjunto toma la información. El siguiente parámetro es '*source*' el cual indica el nombre de la columna origen dentro del *dataset* que toma como referencia. El tercer parámetro es '*target*' donde se indican las estaciones destino que conectan con la estación origen y esta columna se encuentra dentro de nuestro *dataset*. Como

último parámetro se indica la columna que contendrá las distancias entre cada par de paradas. Estos datos son los requeridos para construir un multígrafo y poder implementar algoritmos de búsqueda.

```
# 1. Leer los archivos GTFS utilizando Pandas
df_stops = pd.read_csv(r'C:/Users/carlo/Desktop/PlannerGTFS-V2-F/src/GTFS/Graphgtfs-V1.csv')
stopinfo = pd.read_csv(r'C:/Users/carlo/Desktop/PlannerGTFS-V2-F/src/GTFS/stop.csv')

GRAPH = nx.DiGraph()
GRAPH = nx.from_pandas_edgelist(df_stops, source='Origen', target='Destino', edge_attr='Distancia')
```

Figura 20 Código para la construcción del grafo con cada una de las paradas de GTFS

En el framework Flask son muy utilizados los decoradores, estos decoradores sirven para asociar una URL con una función específica para el manejo de una ruta dentro del proyecto. Para este prototipo se implementó un decorador para mostrar todas las paradas que contiene los datos GTFS y por medio del método GET poder obtener un listado de todas estas paradas.

En este caso, la ruta asociada es '/paradas' y se especifica que solo aceptará solicitudes de tipo GET (consultas para obtener información). Esto significa que cuando un cliente realice una solicitud GET a la URL '/paradas', *Flask* ejecutará la función de manejo de ruta correspondiente. Indica los métodos de solicitud HTTP que la función de manejo de ruta aceptará. En este caso, solo aceptará solicitudes GET, que se utilizan para obtener información del servidor como se muestra en la figura 21.

La función está construida para que al momento de recibir alguna petición GET muestre todas las paradas que contiene los datos GTFS de igual forma si solo se requiere consultar una parada especifica la petición deberá ser enviada con el ID o nombre de la parada. La respuesta a estas peticiones se devuelve en formato JSON con ayuda de la librería Jsonify. El formato JSON (*JavaScript Object Notation*) es un formato abierto utilizado como alternativa al XML para la transferencia de datos estructurados entre un servidor de Web y una aplicación Web. El formato ganó popularidad en servicios de la Web, como clientes de correo electrónico y páginas de compras, dado que consigue transmitir una gran cantidad de información entre el cliente y el servidor utilizando una menor cantidad de caracteres (IBM, 2023).

```python
@app.route('/paradasgtfs', methods=['GET'])
def paradasgtfs():
    stops_info = stops[['stop_id', 'stop_name']]
    stops_info['stop_id'] = stops_info['stop_id'].map(unique_stop_ids)
    stops_list = []
    for index, row in stops_info.iterrows():
        stop_info = {
            'id': row['stop_id'],
            'name': row['stop_name']
        }
        stops_list.append(stop_info)

    resultadoo = {
        'stops': stops_list
    }

    return jsonify(resultadoo)
```

Figura 21 Código de la función que enlista todas las paradas que están dentro de los datos GTFS

Para la función que realiza el cálculo de la ruta más corta también se utilizó un decorador para el manejo de la ruta. Como primer paso se lee el archivo que contiene información de todas las paradas y sus distancias y son cargados en un *DataFrame*. Posteriormente se crea un objeto de tipo grafo dirigido utilizando la clase DiGraph de la biblioteca de Networkx. Este grafo es utilizado para representar las conexiones entre cada una de las paradas y sus respectivas distancias. En la aplicación se define el nombre de la URL a la cual se comunica para realizar peticiones al servidor y este a su vez realizar el cálculo de la ruta más corta y devolver la respuesta con la ruta más corta según el origen y el destino solicitados. La ruta definida es: 'Nombre-del-dominio/short-route' esta dirección es la que escucha las peticiones del cliente al momento de solicitar una ruta.

La función short-route maneja las solicitudes GET. Estas solicitudes deben incluir los parámetros de origen y destino importantes para poder generar la ruta más corta. Estos parámetros representan las paradas de origen y destino para encontrar la ruta más corta. Una vez que se reciben estos parámetros y con el grafo previamente construido con la información GTFS se implementa el algoritmo de búsqueda como es Dijkstra para encontrar el camino más corto en relación a distancia dentro del grafo. El algoritmo realiza una comprobación de los nodos más cercanos con el origen para conocer cuál es el camino más corto realizando una suma total de las aristas desde el origen hasta el destino. La librería de Networkx facilita la implementación del algoritmo la cual se aplicó al grafo previamente generado.

Por último, una vez que el algoritmo encuentra la ruta más corta entre el origen y el destino se filtra la información de nuestro DataFrame con cada una de las paradas que conforman la ruta sugerida por el algoritmo. Esta información es utilizada para obtener información adicional de cada una de las paradas como lo es el nombre completo, su latitud, longitud, color de la ruta y tipo de trasporte público que pertenece. Toda esta información de la ruta sugerida es almacenada en un diccionario de forma ordenada en formato JSON para ser enviada cada vez que un cliente realice una petición de una ruta más corta. A continuación, se muestra la función completa de la implementación del planificador de rutas de transporte público en la figura 22.

```python
@app.route('/short-route', methods=['GET'])
def shortroute():

    origen = request.args.get('origen')
    destino = request.args.get('destino')

    start_time = int(round(time.time() * 1000000)) # Inicio del temporizador

    path = nx.dijkstra_path(GRAPH, source=origen, target=destino, weight='Distancia')

    end_time = int(round(time.time() * 1000000)) # Fin del temporizador
    duration = end_time - start_time # Cálculo de la duración en microsegundos

    print('La distancia son: ' + str(nx.dijkstra_path_length(GRAPH, source=origen, target=destino, weight='Distancia'))

    stops_info = stopinfo[['stop_id', 'stop_name', 'stop_lat', 'stop_lon', 'color', 'route_line']]
    stops_in_path = stops_info[stops_info['stop_id'].isin(path)]

    path_info = []
    for stop_id in path:
        stop_info = stops_in_path[stops_in_path['stop_id'] == stop_id].iloc[0]
        stop_info_dict = {
            'name': stop_info['stop_name'],
            'latitude': stop_info['stop_lat'],
            'longitude': stop_info['stop_lon'],
            'color': stop_info['color'],
            'route_line': stop_info['route_line']
        }
        path_info.append(stop_info_dict)

    result = {
        'path': path,
        'stops': path_info
    }

    return jsonify(result)
```

Figura 22 Código de la función que recibe los parámetros y genera la respuesta con la ruta encontrada

5.4 Implementación de la interfaz de usuario

Para el desarrollo de la interfaz de usuario se priorizó la utilización de tecnologías de código abierto siguiendo la tendencia del planificador de rutas. Para ello se utilizó herramientas como son el

framework React para el desarrollo del cliente web. Para los estilos del sitio web se utilizó Bootstrap que es un *framework* de estilos de software libre.

La interfaz de usuario está conformada por cinco elementos principales. El objetivo de que la interfaz tenga pocos elementos es para que sea intuitiva y fácil de usar. El primer elemento es una barra de navegación que contiene el icono del Planner, el botón de inicio. El segundo elemento este situado en la parte izquierda del sitio web el cual es un formulario donde indicamos cual es el origen y el destino de la ruta que se desea conocer. El tercer elemento es el botón que realiza las peticiones al planificador de rutas. El cuarto elemento es información adicional de la ruta el cual está situado debajo del botón de calcular ruta. Por último, el quinto elemento es la incorporación de un mapa gráfico que permite visualizar la ruta generada por el algoritmo planificador de rutas. Este mapa es un desarrollo *open source* de OpenStreetMap que cuenta con la cartografía de la mayoría de las ciudades. La interfaz completa se muestra en la figura 23.

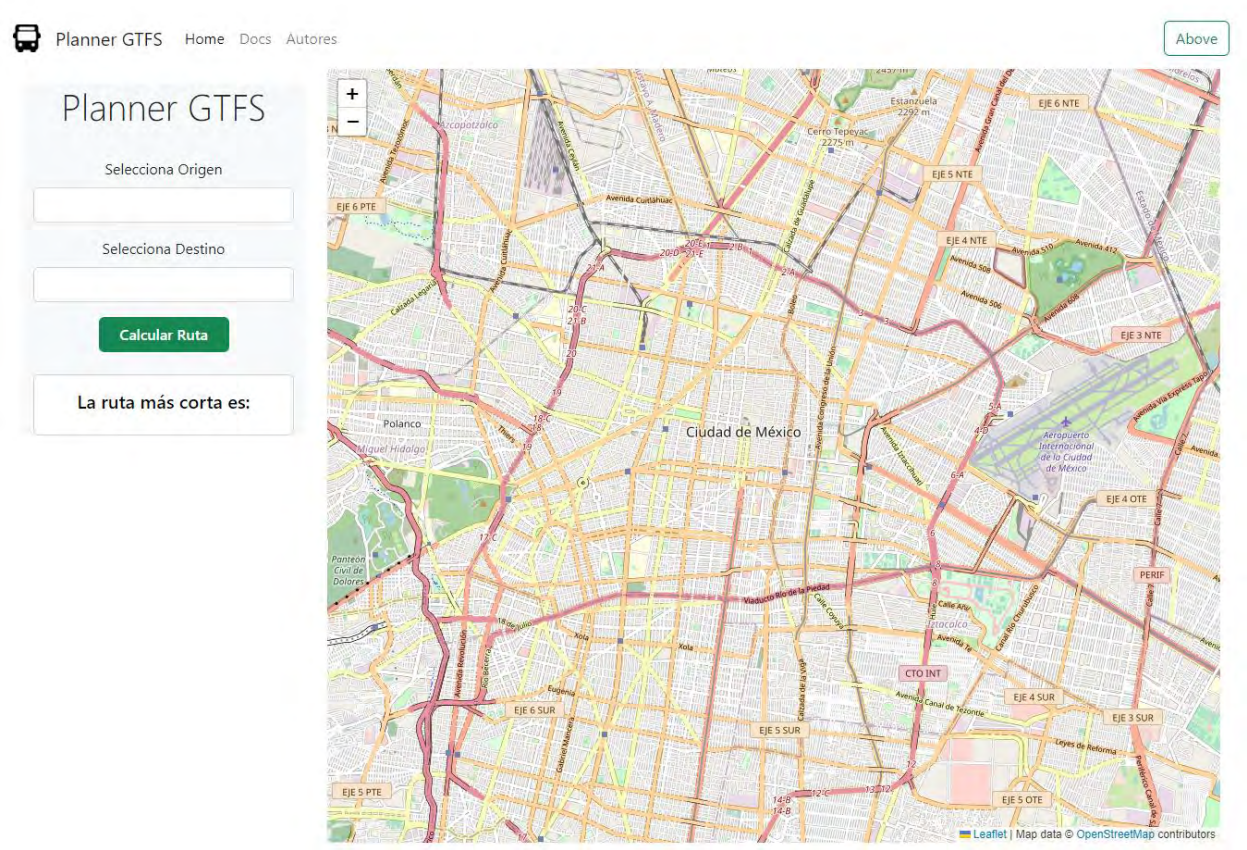

Figura 23 Interfaz de usuario del planificador

El funcionamiento de la interfaz es intuitivo y fácil de utilizar. Para calcular una ruta solo debemos ingresar un origen y un destino que aparezcan dentro de los datos GTFS. Para elegir un origen es muy sencillo dar clic en dentro del primer campo del formulario y despliega una lista con todas las

paradas existentes de GTFS o bien también cuenta con la opción de autocompletado escribiendo el inicio del nombre de la estación y las coincidencias de los nombres de las paradas se reducen a las que coinciden con lo escrito en el campo. Lo mismo sucede para el campo de destino si no se desea buscar dentro de la inmensa lista de paradas que hay dentro de GTFS.

Adicional a la información proporcionada por el planificador de rutas en el mapa gráfico se muestra la ruta con marcadores en cada una de las paradas correspondientes de la ruta sugerida. Estos marcadores muestran información como el nombre de la estación, el medio de transporte y con que estaciones conectan como se muestra en la figura 24.

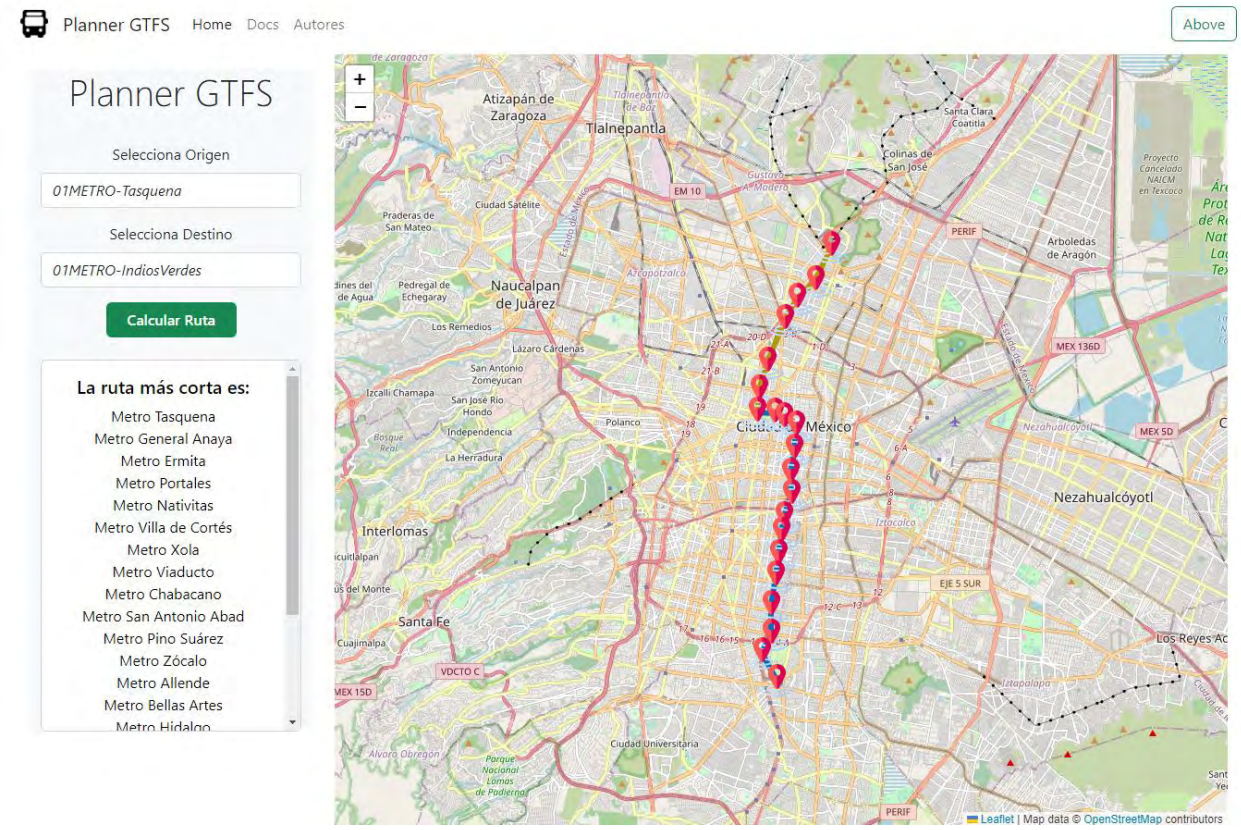

Figura 24 Funciones de la interfaz de usuario

5.5 Comunicación entre componentes

La comunicación entre componentes se realiza a través del protocolo HTTP por medio de peticiones tipo GET. Una petición GET solicita al servidor una información o recurso concreto. Cuando te conectas a un sitio web, tu navegador suele enviar varias peticiones GET para recibir los datos que necesitas. Esta comunicación se establece tanto en el *backend* como en el *frontend*.

La configuración que se establece del lado del *backend* es la asignación de la URL a la que se debe apuntar para realizar peticiones. Para este caso el prototipo se configuró con el puerto por default que es el 5000. La URL en el lado del *backend* está compuesta por tres elementos indispensables para poder generar la comunicación con el planificador. El primer elemento de la URL es el dominio o la dirección donde está ejecutándose el *backend*. Este dominio se puede modificar según cada sea el dominio del planificador en este caso se dejó por default con la dirección 127.0.0.1:5000/. El segundo elemento es la función que se requiere consultar para este caso se nombró como /ruta-corta. Por último, el tercer elemento que compone la URL son los parámetros de origen y destino que se pasan por medio de la URL con la palabra origen seguido del ID de la parada. La petición GET completa se muestra en la siguiente figura.

```
#URL ejemplo:
#http://127.0.0.1:5000/ruta-corta?origen=Metro%20Polit%C3%A9cnico%20_0&destino=Eje%20Central%20y%20Poniente%20152_0
```

Figura 25 Ejemplo de la URL para realizar peticiones al backend

La configuración que se establece del lado del *frontend* permite configurar como se debe enviar las peticiones y que parámetros debe contener para obtener una respuesta exitosa. En React se utilizan hooks para realizar efectos secundarios en un componente funcional, como realizar llamadas a una API o suscribirse a eventos. En este caso, se ejecutará una función llamada allStops que toma setStops como argumento. Esta función realiza una llamada a la API para obtener todas las paradas disponibles.

La función llamada handleCalcularRuta se define como una función asíncrona (async) que toma un argumento event. Esta función maneja el evento de calcular una ruta cuando se dispara un evento. Este evento es una petición programada en el botón de 'calcular ruta' que se encuentra dentro del formulario un botón o un formulario. En el botón de 'calcular ruta' se validan que ambos campos de 'origen' y 'destino' sean campos válidos y que esas paradas existan dentro de los datos antes de enviar la petición. En este caso, evita que se envíe un formulario vacío y se recargue la página. Si alguna de las variables es falsa, se muestra una alerta que indica que se debe seleccionar el origen y el destino.

En la función se utiliza el método map para extraer los stop_id de cada objeto en el arreglo stops. Luego utiliza el método includes para verificar si origen y destino están incluidos en el arreglo de stop_id. Si alguna de las paradas no existe en el arreglo de stops, se muestra una alerta indicando que la parada seleccionada no existe, se restablecen los valores de origen y destino, y la función realiza el (*return*). Para la petición se utiliza la biblioteca Axios para realizar una solicitud GET a una URL específica. La URL incluye los parámetros origen y destino con sus respectivos valores. La respuesta de la solicitud se asigna a una constante.

Esta función de actualización del estado (setRutaMasCorta) es utilizada para actualizar el estado de RutaMasCorta con el valor de peticion.data.stops. Esto actualiza el estado del componente con la información de la ruta más corta obtenida como respuesta de la solicitud. Se captura cualquier error que ocurra durante la solicitud y muestra una alerta indicando que se ha producido un error y que se debe intentar nuevamente más tarde. Esta línea se ejecutará sin importar si la solicitud fue exitosa o si se produjo un error. Esto se realiza con el objetivo de encapsular cualquier error que pueda surgir al momento de que un servicio deje de funcionar o no esté disponible.

En una constante es guardada la URL a la que se solicitan las peticiones de las rutas. Esta URL es la misma que se configuró del lado del *backend* esto garantiza que tanto cliente como servidor están en el mismo canal. Al capturar los parámetros origen y destino la URL estará completa para iniciar una petición y quedará en espera para la respuesta posterior. Esta respuesta es recibida en formato JSON y visualizada en la interfaz de forma gráfica. El fragmento del código se muestra en la figura 26.

```
function MapView() {
  const [stops, setStops] = useState([])
  const [origen, setOrigen] = useState('');
  const [destino, setDestino] = useState('');
  const [rutaMasCorta, setRutaMasCorta] = useState([]);


  useEffect(() => {
    allStops(setStops)
  }, [])


  const handleCalcularRuta = async (event) => {
    event.preventDefault();
    if (!origen || !destino) {
      alert('Selecciona origen y destino');
      return;
    }
    try {
      if (!stops.map(stop => stop.stop_id).includes(origen) || !stops.map(stop => stop.stop_id).includes(destino)) {
        alert('La parada seleccionada no existe');
        setOrigen('');
        setDestino('');
        return;
      }
      const peticion = await axios.get(`http://127.0.0.1:5000/short-route?origen=${origen}&destino=${destino}`)
      setRutaMasCorta(peticion.data.stops)
    } catch (error) {
      alert('Se ha producido un error. Inténtelo de nuevo más tarde.');
    } finally {
      setOrigen('');
      setDestino('');
    }

  }
```

Figura 26 Código de la comunicación entre el cliente y el servidor

Capítulo 6 Pruebas y resultados

En este capítulo se muestran las pruebas realizadas al prototipo con el objetivo de validar el correcto funcionamiento del planificador.

6.1 Experimentación

La experimentación tiene como objetivo realizar pruebas del prototipo desarrollado, donde su principal funcionalidad es la planificación de rutas para calcular la ruta más corta en términos de distancia entre un origen y un destino. Las pruebas evalúan la calidad y congruencia en la generación de rutas más cortas para transporte público de la Ciudad de México.

Para lograr este objetivo, se deben realizar pruebas exhaustivas del sistema, que incluya diferentes escenarios, así como diferentes parámetros y variables que afecten la calidad del servicio. Estas pruebas deben proporcionar datos de utilidad sobre la precisión de las rutas recomendadas, la velocidad de procesamiento del algoritmo y la satisfacción del usuario.

El objetivo final de la experimentación es garantizar que el planificador de rutas pueda proporcionar información confiable y útil para los usuarios. Si el planificador es capaz de lograr producir información confiable y útil, esto permitirá que los usuarios planifiquen sus viajes de manera eficiente y maximicen su satisfacción y comodidad durante el viaje en términos de distancia recorrida.

La experimentación se realizó con tres tipos de pruebas:

I. Pruebas de obtención de la ruta más corta: En esta fase se realizaron pruebas para comprobar que nuestro algoritmo sea capaz de encontrar la ruta más corta dentro de un grafo. Para ello se implementó una función donde se extraen todas las posibles rutas que existen desde el punto de origen al punto destino. Todas esas rutas posibles fueron registradas en una tabla con su peso correspondiente. Posteriormente se consultó la misma ruta (mismo origen y destino) con nuestro algoritmo y se verificó que el algoritmo encuentre la ruta más corta. Para esta fase se usó un grafo más pequeño con un solo tipo de medio de transporte, esto para realizar pruebas más controladas de las posibles combinaciones que encuentre el algoritmo.
II. Pruebas de funcionalidad: Las pruebas de funcionalidad son un tipo de prueba que se realiza para verificar el correcto funcionamiento de un sistema o aplicación. En el caso específico de un planificador de rutas, estas pruebas buscan confirmar que el sistema es capaz de calcular una ruta entre dos puntos de manera precisa y que la ruta generada sea factible, es decir que exista una ruta que utilice diversos sistemas de transporte público para llegar del origen al destino. Para realizar estas pruebas, se les pidió a diferentes usuarios que realizarán la consulta de al menos una ruta. Para esas rutas se registró información de relevancia como: El tiempo que tarda en realizar la consulta, cuántos medios de transporte público se utilizaron, el total de las estaciones, el origen, el destino y la distancia total de la ruta. Con esa información se realizó la comparación con un planificador bien establecido como es el caso de Google Maps.

III. Pruebas de comparación con *Google Map*s: Para la fase final de esta experimentación se comparó cada una de las rutas que consultaron los usuarios. Para ello se extrajo la distancia final de cada ruta calculada y se registró en una tabla para posteriormente esa distancia compararla con la distancia que da un planificador bien establecido como lo es *Google Map*s. Idealmente se busca que el planificador desarrollado en esta tesis tenga una eficacia similar a la de un planificador comercial.

6.2 Procedimiento de la experimentación

En esta sección se describen brevemente las pruebas realizadas para evaluar el algoritmo de planificación de rutas de transporte público. El objetivo principal de las pruebas es verificar el correcto funcionamiento del algoritmo. Las pruebas se realizaron en tres ámbitos diferentes: pruebas de la ruta más corta, pruebas de funcionalidad y pruebas de comparación.

6.2.1 Pruebas de obtención de la ruta más corta

En esta fase inicial de la prueba se busca demostrar la efectividad del algoritmo que se está utilizando para encontrar la ruta más corta dentro de un grafo. Para lograr esto, se ha implementado una función que extrae todas las posibles rutas entre el origen y el destino. Estas rutas posibles se han registrado en una tabla con su respectivo peso cada una de ellas. Posteriormente, se ha utilizado el algoritmo desarrollado en esta tesis para buscar la misma ruta (con el mismo origen y destino) para comprobar que efectivamente se encuentre la ruta más corta.

6.2.2 Pruebas de funcionalidad

Pruebas de funcionalidad, se realizaron pruebas para verificar que el planificador de rutas puede calcular una ruta entre dos puntos correctamente y que la ruta generada es factible con la información del mapa.

Para la realización de las pruebas de funcionalidad se tomó en cuenta la norma ISO-9126. En forma más específica se utilizó la característica de funcionalidad de dicho estándar. De acuerdo con la característica de funcionalidad dentro de la norma ISO-9126, ésta permite evaluar un producto de software que maneje de forma adecuada las funciones que satisfagan las necesidades para las cuales fueron creadas. La característica de funcionalidad evalúa cinco aspectos de un desarrollo de software los cuales son: Adecuación, Exactitud, Interoperabilidad, Conformidad y Seguridad.

A continuación, se describen los aspectos evaluados con la norma ISO-9126 para la característica de funcionalidad:

Adecuación: se confirma que el prototipo cumple de forma correcta con el objetivo de planificar rutas de transporte público de la Ciudad de México.

Exactitud: El prototipo muestra rutas precisas en diferentes escenarios.

Interoperabilidad: El prototipo cumple con este aspecto al poder interactuar con otros sistemas gracias al diseño de su arquitectura basada en microservicios. El sistema admite la comunicación de diferentes clientes por medio de peticiones HTTP.

Conformidad: El sistema cumple con este aspecto ya que trabaja únicamente con datos libres y bajo un estándar como lo es GTFS. Estos datos son abiertos para cualquier persona y publicados por la Secretaría de Movilidad de la Ciudad de México.

Seguridad: Al no solicitar información de ningún tipo al usuario, la seguridad no es un punto fuerte del planificador. El acceso a los datos GTFS es libre y esto no representa un problema.

Para evaluar la funcionalidad del software de acuerdo con la norma ISO-9126 se siguieron los siguientes pasos:

- Identificación de los objetivos y requerimientos
    - El software debe permitir a los usuarios planificar sus rutas en transporte público.
    - El software debe mostrar la ruta disponible
    - El software debe ser fácil de usar y accesible para todos los usuarios
    - El software debe ser capaz de mostrar información actualizada sobre el transporte público
- Establecimiento de los criterios de evaluación
    - El software debe ser capaz de mostrar al menos una opción de ruta para llegar de un lugar a otro.
    - El software debe ser fácil de usar para usuarios de todas las edades y niveles de experiencia en el uso de la tecnología.
    - El software debe ser capaz de actualizar la información al buscar una ruta nueva
- Definición de las pruebas
    - Para evaluar la capacidad del software para mostrar rutas, se pide a los usuarios que planifiquen una ruta entre dos puntos y se registra la ruta con toda la información que muestra.
    - Para evaluar la facilidad de uso, se realizó pruebas de usabilidad con usuarios de diferentes edades y niveles de experiencia, registrando su capacidad para completar la planificación de rutas.
    - Para evaluar la capacidad del software para mostrar información actualizada sobre el transporte público y las rutas se comparó con un planificador bien establecido como es *Google Maps*.
- Ejecución de las pruebas

    - Se pide a los usuarios que planifiquen rutas y se registra toda información generada por la ruta mostrada.
    - Los usuarios seleccionan una ruta y se comparan las distancias mostradas con las distancias reales.
    - Los usuarios realizan tareas específicas en el software y se registra su capacidad para completarlas con éxito.
- Documentación de resultados
    - Se documentan los resultados de la evaluación de la funcionalidad del software.

6.2.3 Pruebas de comparación con *Google Maps*

Se realizaron pruebas comparando los resultados del planificador de rutas (prototipo) con el planificador de rutas *Google Maps*. Esto se realizó para asegurarse que el planificador de rutas está proporcionando la mejor ruta posible y comparar la distancia que proporciona cada planificador.

6.3 Resultados obtenidos

En esta sección se muestran los resultados obtenidos tras someter al planificador de rutas a pruebas. Como evidencia de las pruebas se obtuvieron los siguientes resultados.

6.3.1 Resultados de las pruebas de obtención de la ruta más corta

Para esta prueba se seleccionó un conjunto aleatorio de rutas de un solo medio de transporte (Metro de la CDMX). Esto para controlar de mejor forma la cantidad de posibles combinaciones desde un origen a un destino. Estas pruebas no fueron realizadas por usuarios externos sino por el grupo de asesores de tesis y un servidor. Se establecieron puntos de origen y destino dentro del conjunto seleccionado. Estos puntos se eligieron al azar para simular escenarios de navegación de usuarios.

A continuación, se muestra el registro de las diferentes rutas consultadas por el grupo de asesores de esta tesis. En esta prueba se registra información como: el origen, el destino, todas las posibles rutas que existen desde el origen al destino con sus respectivas distancias y se compara con la ruta que se obtuvo con nuestro algoritmo. Para esta prueba se realizaron 5 escenarios con diferentes origen y destino para la prueba más corta.

La primera prueba se eligió la ruta 1 que tiene como origen la estación de Guerrero y como destino la estación Salto del Agua. La tabla 4 muestra todas las rutas posibles que existen entre esa ruta, la cuales fueron obtenidas por la función path_all que se encuentra en la librería de Networkx. Esta librería de Python nos permite obtener todas las posibles rutas alternativas entre dos nodos de un grafo.

En la tabla 4 se indica también cuál es la ruta más corta que fue elegida por nuestro sistema. En este primer experimento, se nuestra de color verde la ruta encontrada por nuestro algoritmo. El algoritmo obtuvo como ruta más corta la ruta que también tiene la menor distancia en km por la función path_all. Por lo cual se considera que nuestro sistema efectivamente logró obtener la ruta más corta.

Tabla 4 Prueba 1 de la fase de la ruta más corta

Ruta 1: **Guerrero – Salto del Agua**		
Posibles Rutas con la función *path_all* de *Networkx*	Distancias Totales	Ruta generada por el algoritmo
Alternativa 1: Guerrero-Hidalgo-BellasArtes-Allende-Zocalo-PinoSuarez-IsabellaCatolica-SaltodelAgua	3898 m	X
Alternativa 2: Guerrero-Hidalgo-BellasArtes-Allende-Zocalo-PinoSuarez-SanAntonioAbad-Chabacano-Obrera-Doctores-SaltodelAgua	7362 m	X
Alternativa 3: Guerrero-Hidalgo-BellasArtes-SanJuandeLetran-SaltodelAgua	2298 m	SI
Alternativa 4: Guerrero-Hidalgo-Juarez-Balderas-SaltodelAgua	2372 m	X
Alternativa 5: Guerrero-Garibaldi-BellasArtes-Hidalgo-Juarez-Balderas-SaltodelAgua	3508 m	X
Alternativa 6: Guerrero-Garibaldi-BellasArtes-Allende-Zocalo-PinoSuarez-IsabellaCatolica-SaltodelAgua	4140 m	X
Alternativa 7: Guerrero-Garibaldi-BellasArtes-Allende-Zocalo-PinoSuarez-SanAntonioAbad-Chabacano-Obrera-Doctores-SaltodelAgua	7604 m	X

Alternativa 8: Guerrero-Garibaldi-BellasArtes-SanJuandeLetran-SaltodelAgua	2540 m	X

A continuación, se muestra la ruta proporcionada por el prototipo Planner GTFS. La ruta más corta encontrada entre el punto de origen y el punto de destino tiene una distancia total de 2.29 km. Esta ruta está comprendida por 5 paradas. La ruta encontrada se muestra de color rojo, marcando el origen y el destino con un punto amarillo dentro como se muestra en la figura 27.

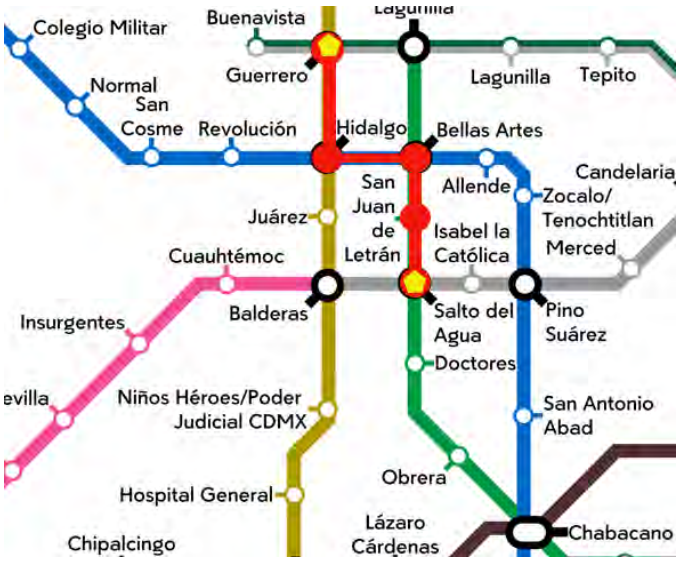

Figura 27 Ruta 1 encontrada por el algoritmo

La segunda prueba fue elegir la ruta 2 que tiene como origen la estación de Garibaldi y como destino la estación Revolución. La tabla 5 muestra todas las rutas posibles que existen entre esa ruta, la cuales fueron obtenidas por la función path_all que se encuentra en la librería de Networkx, la cual nos permite obtener todas las posibles rutas alternativas entre dos nodos de un grafo. En la tabla 5 se indica también cuál es la ruta más corta que fue elegida por nuestro sistema. En esta segunda prueba, nuestro algoritmo obtuvo como ruta más corta la ruta que también tiene la menor distancia en km por la función path_all. Por lo cual se considera que nuestro sistema efectivamente logró obtener la ruta más corta.

Tabla 5 Prueba 2 de la primera fase de pruebas

Ruta 2: **Garibaldi - Revolución**

Posibles Rutas con la función path_all de Networkx	Distancias Totales (metros)	Ruta generada por el algoritmo
Alternativa 1: Garibaldi-BellasArtes-Hidalgo-Revolucion	1668 m	SI
Alternativa 2: Garibaldi-BellasArtes-Allende-Zocalo-PinoSuarez-IsabellaCatolica-SaltodelAgua-Balderas-Juarez-Hidalgo-Revolucion	5640 m	X
Alternativa 3: Garibaldi-BellasArtes-Allende-Zocalo-PinoSuarez-SanAntonioAbad-Chabacano-Obrera-Doctores-SaltodelAgua-Balderas-Juarez-Hidalgo-Revolucion	9104 m	X
Alternativa 4: Garibaldi-BellasArtes-SanJuandeLetran-SaltodelAgua-Balderas-Juarez-Hidalgo-Revolucion	4040 m	X
Alternativa 5: Garibaldi-Guerrero-Hidalgo-Revolucion	2046 m	X

En la figura 28 se muestra la ruta encontrada por el algoritmo. Esta ruta comprende de cuatro estaciones y una distancia total de 1.6 km. Esta ruta coincide con una de las alternativas que existen para llegar al destino. La ruta encontrada fue resaltada en color rojo marcando con un punto amarillo el origen y el destino.

Figura 28 Ruta 2 encontrada por el algoritmo

La tercera prueba fue elegir la ruta 3 que tiene como origen la estación de Pino Suárez y como destino la estación Obrera. La tabla 6 muestra todas las rutas posibles que existen entre esa ruta. En la tabla 6 se indica también cuál es la ruta más corta que fue elegida por nuestro sistema. En esta tercera prueba, nuestro algoritmo obtuvo como ruta más corta la ruta que también tiene la menor distancia en km por la función path_all. Por lo cual se considera que nuestro sistema efectivamente logró obtener la ruta más corta.

Tabla 6 Prueba 3 de la primera fase de pruebas

Ruta 3: Pino Suárez - Obrera		
Posibles Rutas con la función path_all de Networkx	Distancias Totales (metros)	Ruta generada por el algoritmo
Alternativa 1: PinoSuarez-IsabellaCatolica-SaltodelAgua-Doctores-Obrera	2340 m	SI
Alternativa 2: PinoSuarez-Zocalo-Allende-BellasArtes-Hidalgo-Juarez-Balderas-SaltodelAgua-Doctores-Obrera	5176 m	X

Alternativa 3: PinoSuarez-Zocalo-Allende-BellasArtes-Garibaldi-Guerrero-Hidalgo-Juarez-Balderas-SaltodelAgua-Doctores-Obrera	6822 m	X
Alternativa 4: PinoSuarez-Zocalo-Allende-BellasArtes-SanJuandeLetran-SaltodelAgua-Doctores-Obrera	4208 m	X
Alternativa 5: PinoSuarez-SanAntonioAbad-Chabacano-Obrera	3154 m	X

En la figura 29 se muestra la ruta encontrada por el algoritmo. Esta ruta comprende de cinco estaciones y una distancia total de 2.3 km. Esta ruta coincide con una de las alternativas que existen para llegar al destino. Se comparó con las alternativas y efectivamente resultó ser la de menor distancia. La ruta encontrada fue resaltada en color rojo marcando con un punto amarillo el origen y el destino.

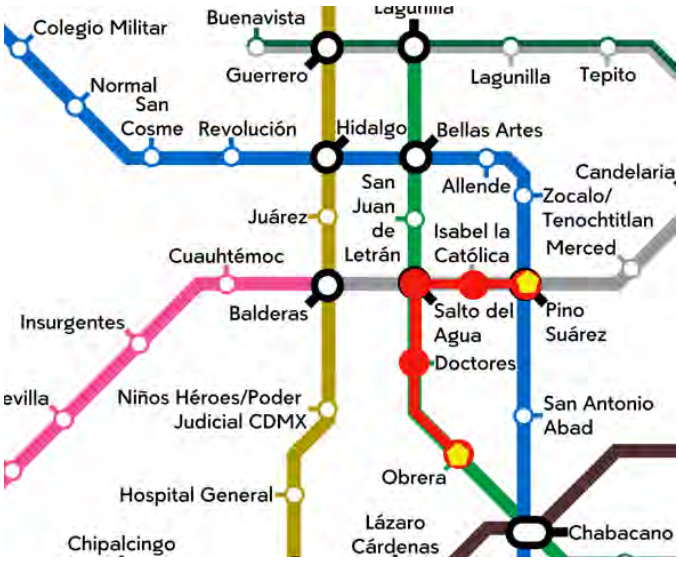

Figura 29 Ruta 3 encontrada por el algoritmo

A continuación, se muestra un resumen con los resultados obtenidos en la prueba de la ruta más corta. Se tomó un pequeño número de nodos en un subgrafo con al menos 20 paradas diferentes para realizar esta prueba. Para la prueba se establecieron diferentes orígenes y destinos dentro del subgrafo y se extrajeron todas las posibles rutas desde el origen hasta el destino.

Una vez extraído las posibles rutas para llegar desde el origen al destino se implementó el algoritmo con los mismos origen y destino de cada uno de los intentos realizados y comprobar que

se obtenga la ruta más corta. Como resultados de esta prueba se realizaron un total de 5 diferentes escenarios (origen y destino), donde al implementar el algoritmo y comparar con todas las posibles rutas se obtuvo un 100% de efectividad en la prueba de la ruta más corta.

6.3.2 Resultados de las pruebas de funcionalidad

Se realizaron pruebas de funcionalidad, donde se verificó que el planificador calcule una ruta entre dos puntos (origen y destino) y que la ruta generada sea factible con la información que se muestra en el mapa proporcionado por un planificador bien establecido como es Google Maps. Para estas pruebas se establecieron los mismos puntos de origen y destino en ambos planificadores.

En esta fase de las pruebas los participantes que probaron el prototipo realizaron el cálculo de algunas rutas. Estas rutas se registraron en una tabla. Esta tabla contiene información importante de la ruta. La forma correcta de interpretar la información dentro de las tablas es la siguiente.

En la primera tabla de cada una de las rutas se muestran tres columnas. La primera columna hace referencia a donde inicia y donde termina la ruta encontrada. La segunda columna de la tabla indica que sistema de transporte pertenece la parada de la ruta. En la tercera columna se indica el nombre de cada una de las paradas que comprenden la ruta encontrada por el Planner GTFS.

Tabla 7 Ruta 1 de la fase de funcionalidad proporcionada por el Planner GTFS

Ruta 1: Planner GTFS		
Estaciones recorridas	Sistema de transporte	Nombre de la parada
1 Origen	Metro	Indios verdes
2	Metro	Deportivo 18 de marzo
3	Metro	Potrero
4	Metro	La raza
5	Metro	Tlatelolco
6	Metro	Guerrero
7	Metro	Hidalgo
8	Metro	Bellas Artes
9	Metro	Allende
10	Metro	Zócalo
11	Metro	Pino Suárez
12	Metro	San Antonio Abad
13	Metro	Chabacano
14	Metro	Viaducto
15	Metro	Xola
16	Metro	Villa de Cortés
17	Metro	Nativitas
18	Metro	Portales
19	Metro	Ermita

20	Metro	General Anaya
21 Destino	Metro	Tasqueña

En seguida de la tabla de la ruta encontrada por el prototipo (Planner GTFS) se muestra un resumen de los datos generados por el Planner GTFS. Esta tabla está compuesta por cuatro columnas principales. La primera columna muestra el total de estaciones recorridas por el algoritmo (desde el origen hasta el destino). La segunda columna muestra el tiempo que tarda el prototipo en encontrar la ruta más corta. La tercera columna muestra los sistemas de transporte públicos necesarios para llegar desde el origen al destino. La cuarta columna indica la distancia total de la ruta generada por el prototipo.

Tabla 8 Ruta 1 de la segunda fase proporcionada por el Planner GTFS

Ruta 1: **Datos de la ruta (Planner GTFS)**			
Total de estaciones recorridas: **21**	Tiempo transcurrido por el algoritmo: **1000** µs	Sistemas de Transporte Público utilizados: **1**	Distancia Total: **19.1** km

A continuación, se muestra la primera tabla de cada una de las rutas que Google Maps sugirió. La tabla está compuesta por tres columnas principales. La primera columna hace referencia a donde inicia y donde termina la ruta encontrada. La segunda columna de la tabla indica que sistema de transporte pertenece la parada de la ruta. En la tercera columna se indica el nombre de cada una de las paradas que comprenden la ruta encontrada por el planificador de *Google Maps*.

Como se muestra en la tabla del planificador de *Google Maps* esta primera ruta es muy similar a la proporcionada por el prototipo. Las únicas variaciones que se observan son en referencia a la distancia de la ruta.

Tabla 9 Ruta 1 de la segunda fase proporcionada por Google Maps

Ruta 1: Google Maps		
Estaciones recorridas	Sistema de transporte	Nombre de la parada
1 Origen	Metro	Indios verdes
2	Metro	Deportivo 18 de marzo
3	Metro	Potrero
4	Metro	La raza
5	Metro	Tlatelolco
6	Metro	Guerrero
7	Metro	Hidalgo
8	Metro	Bellas Artes
9	Metro	Allende
10	Metro	Zócalo

11	Metro	Pino Suárez
12	Metro	San Antonio Abad
13	Metro	Chabacano
14	Metro	Viaducto
15	Metro	Xola
16	Metro	Villa de Cortés
17	Metro	Nativitas
18	Metro	Portales
19	Metro	Ermita
20	Metro	General Anaya
21 Destino	Metro	Tasqueña

En seguida de la tabla la ruta encontrada por *Google Maps* se muestra un resumen de los datos generados por el Planner GTFS. Esta tabla está compuesta por tres columnas principales. La primera columna muestra el total de estaciones recorridas por el algoritmo (desde el origen hasta el destino). La segunda columna muestra los sistemas de transporte públicos necesarios para llegar desde el origen al destino. La tercera columna indica la distancia total de la ruta generada por el prototipo.

Tabla 10 Resumen de la ruta de Google Maps

Ruta 1: **Datos de la ruta (Google Maps)**		
Total de estaciones recorridas: **21**	Sistemas de Transporte Público utilizado: **1**	Distancia Total: **19.23** km

A continuación, se muestran las rutas generadas por cada planificador de rutas. En el lado derecho se muestra la ruta generada por nuestro planificador, el Planner GTFS. En la parte izquierda del documento se muestra la ruta generada por *Google Maps*. Como se observa en esta primera comparación ambas rutas son muy similares y contienen el mismo número de paradas.

Ruta Sugerida por *Google Maps*

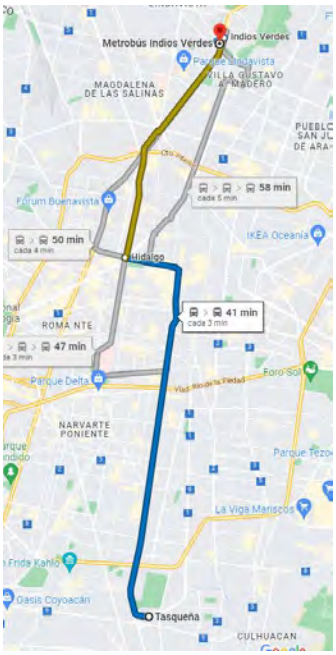

Figura 30 Ruta 1 Google Maps

Ruta sugerida por Planner GTFS

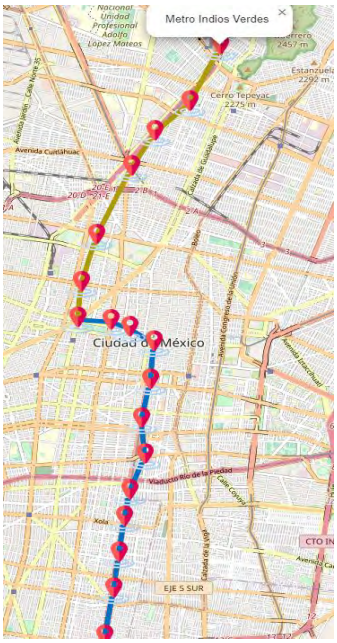

Figura 31 Ruta 1 Planner GTFS

Otra de las rutas proporcionadas por los usuarios que probaron el planificador se muestra en la tabla 11. Esta ruta tiene como origen el Metrobús Insurgentes y como destino el Tren ligero de la Noria. Esta ruta fue encontrada por el prototipo donde está conformada por 36 estaciones y utiliza tres sistemas de transporte público diferentes para llegar desde el origen hasta el destino.

Tabla 11 Ruta 2 de la fase de funcionalidad proporcionada por el Planner GTFS

Ruta 2: Planner GTFS		
Estaciones recorridas	Sistema de transporte	Nombre de la parada
1 Origen	Metrobús	Glorieta Insurgentes
2	Metrobús	Durango
3	Metrobús	Álvaro Obregón
4	Metrobús	Sonora
5	Metrobús	Campeche
6	Metrobús	Chilpancingo
7	Metrobús	Nuevo León
8	Metrobús	Viaducto
9	Metrobús	Amores

10	Metrobús	Etiopía
11	Metrobús	Doctor Vértiz
12	Metrobús	Centro SCOP
13	Metrobús	Álamos
14	Metrobús	Xola
15	Metro	Xola
16	Metro	Villa de Cortés
17	Metro	Nativitas
18	Metro	Portales
19	Metro	Ermita
20	Metro	General Anaya
21	Metro	Tasqueña
22	Tren Ligero	Tasqueña
23	Tren Ligero	Las Torres
24	Tren Ligero	Ciudad Jardín
25	Tren Ligero	La Virgen
26	Tren Ligero	Xotepingo
27	Tren Ligero	Nezahualpilli
28	Tren Ligero	Registro Federal
29	Tren Ligero	Textitlán
30	Tren Ligero	El Vergel
31	Tren Ligero	Estadio Azteca
32	Tren Ligero	Huipulco
33	Tren Ligero	Xomali
34	Tren Ligero	Periférico Participación Ciudadana
35	Tren Ligero	Tepepan
36 Destino	Tren Ligero	La Noria

Esta ruta se puede apreciar que es una distancia mayor a las anteriores registradas y también tiene un número significativo de paradas con un total de 36 como se muestra en la tabla 12. Además, el tiempo de encontrar la ruta fue mayor en comparación con rutas más cortas.

Tabla 12 Resumen de la ruta 2 del Planner GTFS

Ruta 2: **Datos de la ruta (Planner GTFS)**			
Total de estaciones recorridas: **36**	Tiempo transcurrido por el algoritmo: **6333** µs	Sistemas de Transporte Público utilizado: **3**	Distancia Total: **22.7** km

En cuanto a la ruta proporcionada por el planificador de *Google Maps* que se muestra en la tabla 13, hay un cambio al inicio de la ruta en comparación con el prototipo Planner GTFS. Este ligero cambio se muestra al inicio de la ruta donde cambian las primeras estaciones de esta ruta. Una

observación que se notó es que, aunque el inicio de la ruta cambie el número de estación es igual al del prototipo.

Tabla 13 Ruta 2 de la fase de funcionalidad proporcionada por Google Maps

Ruta 2: Google Maps		
Estaciones recorridas	Sistema de transporte	Nombre de la parada
1 Origen	Metrobús	Glorieta Insurgentes
2	Metrobús	Hamburgo
3	Metrobús	Reforma
4	Metrobús	Plaza de la república
5	Metrobús	Revolución
6	Metro	Revolución
7	Metro	Hidalgo
8	Metro	Bellas Artes
9	Metro	Allende
10	Metro	Zócalo
11	Metro	Pino Suárez
12	Metro	San Antonio Abad
13	Metro	Chabacano
14	Metro	Viaducto
15	Metro	Xola
16	Metro	Villa de Cortés
17	Metro	Nativitas
18	Metro	Portales
19	Metro	Ermita
20	Metro	General Anaya
21	Metro	Tasqueña
22	Tren Ligero	Tasqueña
23	Tren Ligero	Las Torres
24	Tren Ligero	Ciudad Jardín
25	Tren Ligero	La Virgen
26	Tren Ligero	Xotepingo
27	Tren Ligero	Nezahualpilli
28	Tren Ligero	Registro Federal
29	Tren Ligero	Textitlán
30	Tren Ligero	El Vergel
31	Tren Ligero	Estadio Azteca
32	Tren Ligero	Huipulco
33	Tren Ligero	Xomali
34	Tren Ligero	Periférico Participación Ciudadana
35	Tren Ligero	Tepepan

36 Destino	Tren Ligero	La Noria

También se puede observar que en relación a distancia en esta comparación es más notoria entre ambos planificadores. En esta ruta es donde se encontró la mayor diferencia entre ambos planificadores con una diferencia de al menos 2.5 km. De igual manera se utilizó el mismo número de transportes públicos con un total de 3 como se muestra en la tabla 14.

Tabla 14 Resumen de la ruta 2 de Google Maps

Ruta 2: **Datos de la ruta (Google Maps)**		
Total de estaciones recorridas: **36**	Sistemas de Transporte Público utilizado: **3**	Distancia Total: **25.2** km

A continuación, se muestra la comparación entre ambos planificadores donde se observa que ambas rutas tienen una precisión en la ruta propuesta. El cambio más significativo de esta ruta sucede al principio de la ruta como se puede observar en la figura 32 y 33.

Ruta Sugerida por *Google Maps*

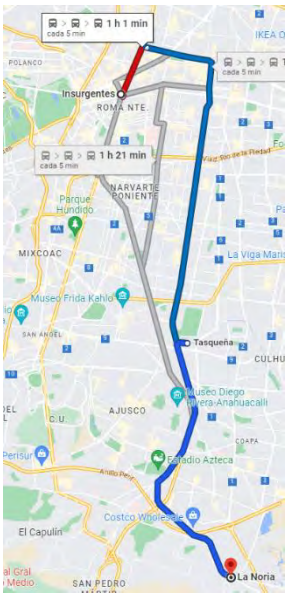

Figura 32 Ruta 2 Google Maps

Ruta sugerida por *Planner GTFS*

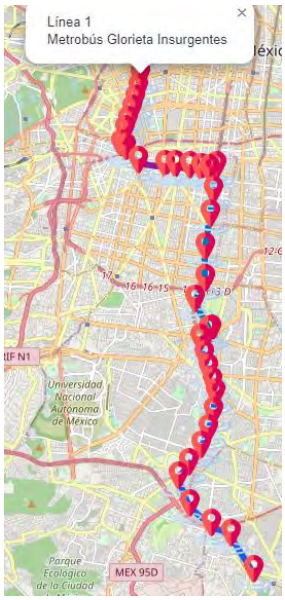

Figura 33 Ruta 2 Planner GTFS

En una sexta ruta que comprende de Constitución de 1917 y con destino a Metrobús Nápoles cuenta con un total de 24 estaciones y una distancia de 16.3 km. La ruta encontrada por el Planner GTFS involucra 3 sistemas de transporte público diferentes como se muestra en la siguiente tabla.

Tabla 15 Ruta 3 de la fase de funcionalidad proporcionada por Planner GTFS

Ruta 3: Planner GTFS		
Estaciones recorridas	Sistema de transporte	Nombre de la parada
1 Origen	Cablebus	Constitución de 1917
2	Metro	Constitución de 1917
3	Metro	UAM-I
4	Metro	Cerro de la Estrella
5	Metro	Iztapalapa
6	Metro	Atlalilco
7	Metro	Escuadrón 201
8	Metro	Aculco
9	Metro	Apatlaco
10	Metro	Iztacalco
11	Metro	Coyuya
12	Metrobús	Coyuya
13	Metrobús	La Viga
14	Metrobús	Andrés Molina
15	Metrobús	Las Américas
16	Metrobús	Xola
17	Metrobús	Álamos
18	Metrobús	Centro SCOP
19	Metrobús	Doctor Vértiz
20	Metrobús	Etiopía
21	Metrobús	Amores
22	Metrobús	Viaducto
23	Metrobús	Nuevo León
24	Metrobús	La Piedad
25	Metrobús	Poliforum
26 Destino	Metrobús	Nápoles

El tiempo en encontrar la ruta fue relativamente rápido en comparación con otras rutas con tan solo 1100 microsegundos. Una distancia total de 16.3 km y tres sistemas de transporte diferentes.

Tabla 16 Resumen de la ruta 3 del Planner GTFS

Ruta 3: **Datos de la ruta (Planner GTFS)**			
Total de estaciones recorridas: **26**	Tiempo transcurrido por el algoritmo: **1100** µs	Sistemas de Transporte Público utilizado: **3**	Distancia Total: **16.3** km

En la tabla 17 del planificador de Google Maps esta ruta es muy similar a la proporcionada por el prototipo Planner GTFS. Las estaciones sugeridas por Google Maps son las mismas que sugirió el prototipo.

Tabla 17 Ruta 3 de la fase de funcionalidad proporcionada por Google Maps

Ruta 3: Google Maps		
Estaciones recorridas	Sistema de transporte	Nombre de la parada
1 Origen	Cablebus	Constitución de 1917
2	Metro	Constitución de 1917
3	Metro	UAM-I
4	Metro	Cerro de la Estrella
5	Metro	Iztapalapa
6	Metro	Atlalilco
7	Metro	Escuadrón 201
8	Metro	Aculco
9	Metro	Apatlaco
10	Metro	Iztacalco
11	Metro	Coyuya
12	Metrobús	Coyuya
13	Metrobús	La Viga
14	Metrobús	Andrés Molina
15	Metrobús	Las Américas
16	Metrobús	Xola
17	Metrobús	Álamos
18	Metrobús	Centro SCOP
19	Metrobús	Doctor Vértiz
20	Metrobús	Etiopía
21	Metrobús	Amores
22	Metrobús	Viaducto
23	Metrobús	Nuevo León
24	Metrobús	La Piedad
25	Metrobús	Poliforum
26 Destino	Metrobús	Nápoles

Las únicas variaciones que se observan son en referencia a la distancia de la ruta entre ambos planificadores tienen al menos unos 700 m de diferencia entre ambos planificadores.

Tabla 18 Resumen de la ruta 3 de Google Maps

Ruta 3: **Datos de la ruta (Google Maps)**

Total de estaciones recorridas: **26**	Sistemas de Transporte Público utilizado: **3**	Distancia Total: **17** km

En la comparación entre ambas rutas no hay una diferencia significativa. La ruta proporcionada por el Planner GTFS es preciso con el número de estaciones y la representación en el mapa como se muestra en las figuras 34 y 35.

Ruta Sugerida por *Google Maps*

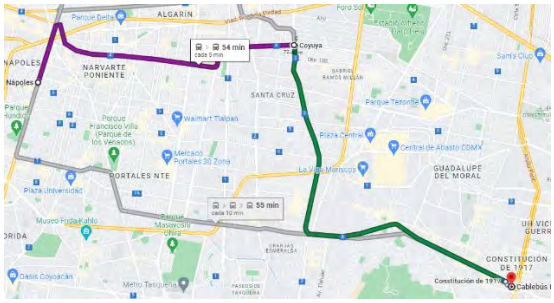

Figura 34 Ruta 3 de Google Maps

Ruta sugerida por *Planner GTFS*

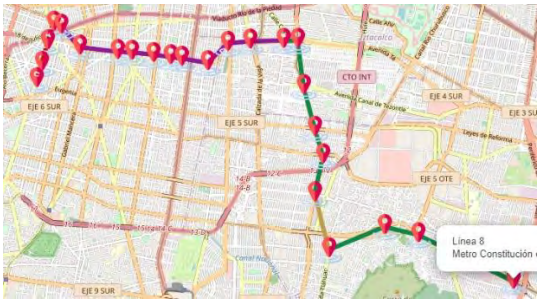

Figura 35 Ruta 3 del Planner GTFS

Los aspectos evaluados en la comparación de las rutas entre ambos planificadores arrojaron las siguientes conclusiones:

La exactitud en las rutas generadas por el prototipo Planner GTFS en comparación con las rutas proporcionadas por *Google Maps* fueron altamente precisas en términos de la secuencia de estaciones y paradas. La exactitud se refiere a qué tan precisas y confiables son las rutas sugeridas por el planificador en términos de las estaciones y paradas de transporte público involucradas.

También se observó una alta interoperabilidad entre el Planificador GTFS y *Google Maps*. Ambos planificadores generaron rutas similares, lo que sugiere que el Planificador GTFS puede funcionar de manera coherente como un planificador establecido como *Google Maps*. Además, al estar diseñado con una arquitectura de microservicios permite interactuar con otros proyectos de software.

La conformidad de los resultados indicó que el Planner GTFS cumplía con los estándares de datos GTFS al utilizar eficazmente la información de rutas y paradas proporcionada. Además, las rutas generadas por el Planner GTFS se alineaban con las expectativas de los usuarios. La conformidad se refiere a la capacidad del planificador para cumplir con estándares y especificaciones, en este caso, los datos GTFS.

6.3.3 Resultados de las pruebas de comparación con *Google Maps*

Se realizaron pruebas en las cuales se tomaron los resultados del planificador GTFS y se compararon con un planificador muy establecido como lo es *Google Maps*. En esta comparación se consultó la misma ruta para ambos planificadores, con el mismo origen y destino.

Para estas pruebas se hizo uso de diferentes métricas que facilitara la comparación entre ambos planificadores. La métrica del Error Absoluto Medio (MAE) mide cuánto se desvía en promedio la distancia calculada por el Planner de las distancias del planificador referencia como lo es *Google Maps*. La métrica del Error Porcentual Medio (MAPE) mide el error promedio para evaluar la precisión relativa. Otra métrica utilizada es la del Error Máximo (EM) que indica la mayor diferencia entre ambos planificadores. Esta métrica ayuda a identificar las rutas con el mayor caso de desviación.

Otra métrica utilizada fue el Coeficiente de Correlación que indica la relación lineal entre las distancias de los planificadores. Un coeficiente cercano a 1 nos indica una fuerte relación lineal positiva. Esto se traduce en que tan parecidas son las rutas generadas por el Planner con referencia a *Google Maps*. Por último, también se sometió a la métrica de Razón de Error Cuadrático Medio (RMSE) que mide el error promedio en términos cuadráticos afectando los errores más grandes. RMSE se encuentra calculando la raíz cuadrada del Error Cuadrático Medio, que a su vez se calcula como la raíz cuadrada del promedio de los errores al cuadrado entre dos conjuntos de datos.

En la tabla 19 se muestran los resultados que arrojó el planificador de esta tesis y se comparó con los resultados obtenidos con el planificador de *Google Maps*.

Tabla 19 Resumen de prueba de comparación entre el Planner GTFS y Google Maps

Núm.	Ruta	Distancia Planner GTFS	Distancia *Google Maps*	MAE	MAPE
1	Metro Indios Verdes – Metro Tasqueña	19.1 km	19.23 km	130 m	1%
2	Metro Tasqueña – Metro Terminal aérea	16.5 km	17.1 km	600 m	4%
3	Metrobús Glorieta Insurgentes – Tren ligero La Noria	22.7 km	25.2 km	2.5 km	10%
4	Metrobús Ciudad Universitaria – Metro Aculco	17.7 km	18.2 km	500 m	3%
5	Tren Ligero Estadio Azteca – Metro Nezahualcóyotl	24.6 km	24.9 km	300 m	2%
6	Cablebus Constitución de 1917 - Metrobús Nápoles	16.3 km	17 km	700 m	5%
7	Suburbano Lechería - Cablebus Santa Marta	37.2 km	37.8 km	600 m	2%
8	Metrobús Villa Olímpica - Metro Ciudad Azteca	34.4 km	37.3 km	2.8 km	7.7%
9	Metro Olivos - Metro San Joaquín	23.3 km	26.6 km	3.3 km	12.4%

Los resultados de la comparación indican que existe una ligera diferencia entre nuestro planificador y *Google Maps*. En términos generales las rutas generadas por el prototipo tienen un promedio del 96 % de similitud con respecto a las sugeridas por un planificador bien establecido como lo es *Google Maps*.

En un balance general las pruebas de comparación fueron favorables al medir los resultados obtenidos con el Planner GTFS y ser comparados con un planificador referente como lo es *Google Maps*. En la tabla 20 se muestran los resultados en términos generales. Con el objetivo de intentar igualar los resultados que proporciona *Google Maps* con su planificador, pero para el caso de esta tesis solamente se usaron tecnologías de código abierto.

Tabla 20 Resultados generales de las pruebas de comparación

MAE	MAPE	Error Máximo	Coeficiente de Correlación	RMSE
1.2 km	**4.8 %**	**3.3 km**	**0.98**	**1.7 km**

Capítulo 7 Conclusiones y trabajos futuros

En este capítulo se presentan las conclusiones derivadas del desarrollo del planificador de rutas de transporte público. Además, se exploran oportunidades para el avance futuro de la movilidad urbana e identificamos áreas clave de investigación y desarrollo.

7.1 Conclusiones

Las diferentes evaluaciones del planificador de rutas desarrollado en esta tesis han permitido resaltar su efectividad en el cálculo de rutas y la capacidad de proporcionar opciones de viaje convenientes y eficientes para los usuarios. Las pruebas de comparación permitieron determinar que nuestro planificador tiene una efectividad muy semejante al de un planificador de uso muy amplio como es *Google Map.*

Una de las conclusiones que se obtuvieron es que el planificador de rutas de transporte público basado en GTFS y tecnologías de código abierto tiene el potencial de mejorar la accesibilidad y la movilidad urbana al proporcionar a los usuarios una herramienta fácil de usar y una visión completa de las opciones de transporte disponibles.

Se pueden destacar las ventajas de utilizar tecnologías de código abierto en el desarrollo del planificador de rutas, como la disponibilidad de recursos y herramientas gratuitas, la flexibilidad para personalizar y adaptar el sistema, y la posibilidad de colaboración y contribución de la comunidad de desarrolladores. Tanto el *Frontend* como el *Backend* están orientados a su reutilización al ser desarrollados como módulos totalmente independientes.

Se puede concluir que la integración de datos GTFS en el planificador de rutas es crucial para garantizar la precisión y la calidad de las rutas generadas. Los datos GTFS proporcionan información detallada sobre las redes de transporte público.

El planificador puede dar resultados en forma muy eficiente, ya que obtiene la ruta más corta en un promedio de 20 milisegundos aun cuando el grafo sea relativamente grande debido a los múltiples sistemas de transporte y a las múltiples paradas de cada sistema de transporte público.

Se puede concluir que el trabajo de tesis ha realizado una contribución al campo académico y a la comunidad en términos de investigación y desarrollo de soluciones de planificación de rutas de transporte público. El planificador de rutas basado en GTFS y tecnologías de código abierto puede ser utilizado como base para futuros estudios y proyectos relacionados con la movilidad urbana.

7.2 Trabajos futuros

Uno de los trabajos futuros es ampliar el planificador de rutas para tener en cuenta factores adicionales, como el tráfico en tiempo real, las condiciones meteorológicas o la disponibilidad de transporte en diferentes momentos del día. Esto permitiría generar rutas más precisas y adaptadas a las condiciones en tiempo real.

Otro de los trabajos futuros es la adaptación del planificador de rutas para ofrecer opciones de rutas que incluyan múltiples modos de transporte, como autobuses, trenes, tranvías y bicicletas compartidas siempre que se cuenten con datos GTFS o equivalentes a estos datos. Esto proporcionaría a los usuarios una visión integral y completa de las opciones de transporte disponibles.

Uno de los trabajos a realizar en las próximas etapas es agregar la capacidad de considerar las restricciones y preferencias del usuario al calcular las rutas. Por ejemplo, permitir que los usuarios especifiquen horarios preferidos, preferencias de transporte (como evitar caminar largas distancias) o restricciones de accesibilidad. Esto podría incluir la implementación de nuevas características, como la consideración de factores adicionales, la integración de múltiples modos de transporte y la optimización del rendimiento del sistema.

Finalmente, otro de los trabajos futuros es crear una aplicación móvil que integre el planificador de rutas y brinde a los usuarios una experiencia de viaje completa en sus dispositivos móviles. Esto permitiría a los usuarios acceder al planificador de rutas en movimiento y recibir notificaciones y actualizaciones en tiempo real sobre su viaje.

Referencias

A continuación, se muestran las referencias utilizadas en la realización de esta tesis. Las referencias fueron consultadas en diferentes fuentes de información.

Alzaidi, M., & Vagner, A. (31 de mayo de 2021). Trip planning algorithm for GTFS data with NoSQL structure to improve the performance. *Jatit, Vol. 99*, 10. Obtenido de http://www.jatit.org/volumes/Vol99No10/10Vol99No10.pdf

Beneyto, R. (19 de Mayo de 2022). *www.imbric.com*. Recuperado el 24 de Abril de 2023, de https://www.imbric.com/la-importancia-del-transporte-publico-en-las-ciudades/

Bozyiğit, A., Alankuş, G., & Nasiboğlu, E. (1 de octubre de 2017). Public transport route planning: Modified dijkstra's algorithm. *International Conference on Computer Science and Engineering (UBMK)*, 502-505. https://doi.org/10.1109/UBMK.2017.8093444.

Brotton, J. (2014). *Historia del mundo en 12 mapas.* España: Penguin Random House.

Bull, A. (2003). *Congestión de tránsito el problema y cómo enfrentarlo.* CEPAL.

Catala, M., Downing, S., & Hayward, D. (15 de noviembre de 2011). Expanding the Google Transit Feed Specification to Support Operation and Planning. *National Center for Transit Research*. Obtenido de https://rosap.ntl.bts.gov/view/dot/39780

Celi, S. F. (20 de enero de 2018). Análisis del transporte público a nivel mundial. *Espacios, Vol. 39*, 10. Obtenido de https://www.revistaespacios.com/a18v39n18/a18v39n18p10.pdf

Colque, A., Valdivia, R., Navarrete, M., & Aracena, S. (29 de noviembre de 2019). Un sistema de información geográfico para el transporte público basado en el estandar GTFS realtime. *Scielo, 29*(1), 51-62. Obtenido de https://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-33052021000100051

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). Introduction to Algorithms. En F. Edition (Ed.), *Introduction to Algorithms* (págs. 5-12). Cambridge, Massachusetts: The MIT Press. https://doi.org/ISBN 9780262046305

Fundación Telefónica. (19 de diciembre de 2011). Smart Cities: un primer paso hacia la internet de las cosas. 13-16.

Google. (6 de octubre de 2021). *API de Google Transit*. Obtenido de https://developers.google.com/transit/gtfs?hl=es

Google support. (22 de octubre de 2021). *support.google.com*. Obtenido de https://support.google.com/maps/answer/144339?hl=es&co=GENIE.Platform%3DDesktop

Google transit. (10 de septiembre de 2021). *developers.google.com*. (Google) Obtenido de https://developers.google.com/transit/gtfs/guides

Ibáñez, M. O. (29 de Septiembre de 2022). *Banco mundial*. Recuperado el 24 de Abril de 2023, de https://www.bancomundial.org/es/topic/transport/overview#:~:text=El%20transporte%20es%20fundamental%20para,estos%20beneficios%20no%20se%20materializan.

IBM. (1 de junio de 2023). *https://cloud.ibm.com*. Recuperado el 20 de junio de 2023, de https://cloud.ibm.com/docs/assistant?topic=assistant-dialog-responses-json&locale=es#:~:text=El%20formato%20JSON%20gen%C3%A9rico%20de,por%20una%20aplicaci%C3%B3n%20cliente%20personalizada.

INEGI. (29 de enero de 2021). En la Ciudad de México somos 9 209 944 habitantes: Censo de población y vivienda 2020. *censo 2020*, 1-6. Obtenido de https://www.inegi.org.mx/contenidos/saladeprensa/boletines/2021/EstSociodemo/ResultCenso2020_CdMx.pdf

Kong, X., Li, M., Tang, T., Tian, K., Moreira-Matias, L., & Xia, F. (11 de septiembre de 2018). Shared Subway Shuttle Bus Route Planning Based on Transport Data Analytics. *IEEE Transactions on Automation Science and Engineering, Vol. 15*(4), 1507-1520. https://doi.org/10.1109/TASE.2018.2865494

La Vanguardia. (9 de abril de 2018). *www.lavanguardia.com*. Obtenido de https://www.lavanguardia.com/ocio/viajes/20180409/442244212659/aplicaciones-planificar-viaje.html

Li, W., Chen, X., & Yang, B. (2 de febrero de 2010). Bus Travel Transit Path Query Algorithm Based on Ant Algorithm. *Third International Conference on Genetic and Evolutionary Computing*, 665-669. https://doi.org/10.1109/WGEC.2009.13

López, D., Lozano, A., González, H., Guzmán, A., & Maldonado, F. (2018). Hiperpuma: Sistema Multimodal de Información al Viajero. En P. Fernández, M. Suárez, & H. Quiroz, *La movilidad en la Ciudad de México Impactos, conflictos y oportunidades* (págs. 119-151). Ciudad de México: IG.

McGlone, D. (13 de agosto de 2013). *Using GTFS Data to Generate Bus Routes with Travel Time.* Obtenido de azavea.com: https://www.azavea.com/blog/2013/08/13/using-gtfs-data-to-generate-bus-routes-with-travel-time/

Negrete, M. E. (1 de marzo de 2018). El desafío de la movilidad y el transporte urbano y metropolitano. *Foro Consultivo Científico y Tecnológico, AC*, 9 - 11. Recuperado el 24 de Abril de 2023, de http://www.foroconsultivo.org.mx/proyectos_estrategicos/img/8/23.pdf

Notimex. (19 de febrero de 2018). *https://unamglobal.unam.mx*. Obtenido de https://unamglobal.unam.mx/microbus-o-combi-transporte-publico-mas-utilizado-en-el-valle-de-mexico/

Ortega, F. A., Marseglia, G., Mesa, J. A., & Piedra de la Cuadra, R. (2021). Un algoritmo para planificar rutas más rápidas con arcos dependientes del tiempo en redes urbanas. En R.-E.

e. transporte. Sevilla, España: Universidad de Burgos. Servicio de Publicaciones e Imagen Institucional. https://doi.org/10.36443/10259/6927

Peña, J. A. (3 de septiembre de 2012). Sistemas de transporte en México: un análisis de centralidad en teoría de redes. *EMALCA, Vol. 3*, 3. Obtenido de https://rde.inegi.org.mx/index.php/2012/09/10/sistemas-de-transporte-en-mexico-un-analisis-de-centralidad-en-teoria-de-redes/

Pérez, A. (17 de Agosto de 2017). *bixpe.com*. Recuperado el 25 de Abril de 2023, de https://www.bixpe.com/blog/beneficios-de-la-planificacion-de-rutas/

Queiroz, A. R., Santos, V., Nascimento, D., & Pires, C. E. (26 de noviembre de 2019). Conformity Analysis of GTFS Routes and Bus Trajectories. *UFCG*, 1-6. Obtenido de https://sol.sbc.org.br/index.php/sbbd/article/view/8823/8724

Sánchez, G., & Lozano, V. M. (16 de julio de 2001). Algoritmo de Dijkstra. Un Tutorial Interactivo. *VII Jornadas de Enseñanza Universitaria de la Informática, JENUI*, 1-5. Obtenido de http://rua.ua.es/dspace/handle/10045/128198

Szincsák, T., & Vágner, A. (29 de enero de 2015). Public transit schedule and route planner application for mobile devices. *Eger, Vol. 2*, 153-161. https://doi.org/10.14794/ICAI.9.2014.2.153

Vágner, A. (10 de julio de 2021). Route planning on GTFS using Neo4j∗. *Annales Mathematicae et Informaticae*, 1-17. https://doi.org/10.33039/ami.2021.07.001

Waze. (22 de abril de 2021). *medium.com*. Obtenido de https://medium.com/waze/how-to-improve-your-mobility-habits-for-a-greener-future-e07c01a8082e

Wong, J., Landon, R., Kari, W., & Regan, H. (13 de enero de 2013). Open Transit Data: State of the Practice and Experiences from Participating Agencies in the United States. *Transportation Research Broad*. Obtenido de https://www.semanticscholar.org/paper/Open-Transit-Data%3A-State-of-the-Practice-and-from-Wong-Reed/a2b17e6f4fe7b738b4c3b8982cff9f446df1c398