INSTITUTO MEXICANO DEL TRANSPORTE

Desarrollo de mapas resultado del asistente automático para el diseño de rutas de distribución.

José Alejandro Ascencio Laguna
Agustín Bustos Rosales
Alma Rosa Zamora Domínguez
José Alfonso Balbuena Cruz

Publicación Técnica No. 675
Sanfandila, Qro.
2022

ISSN 0188-7297

Esta investigación fue realizada en la Coordinación de Transporte Integrado y Logística del Instituto Mexicano del Transporte, por el Mtro. José Alejandro Ascencio Laguna, el Dr. Agustín Bustos Rosales, la Lic. Alma Rosa Zamora Domínguez y el Mtro. José Alfonso Balbuena Cruz.

Esta investigación es el producto final del proyecto de investigación interna **TI 06/21 Implementación del módulo de presentación de resultados y despliegue en producción del asistente automático para el diseño de rutas de distribución**.

Contenido

Página

Índice de figuras

Índice de tablas

Sinopsis

Dada la demanda de la automatización de los procesos, el acceso a grandes cantidades de datos y la disponibilidad de tecnologías orientadas a la gestión de la operación del transporte de carga, es de suma importancia el desarrollo de aplicaciones que le permitan al usuario final la visualización dinámica de los resultados generados por este tipo de soluciones, de tal forma que se logre la abstracción de la complejidad que estas conllevan.

El principal objetivo de la abstracción en términos de tecnología, consiste en encarar las características esenciales de un objeto, sin tomar en cuenta los aspectos no relevantes para otro objeto que hace uso de él. En este sentido y en el contexto de este estudio, se busca que el usuario final interactúe con los mínimos elementos posibles para generar los resultados esperados para que pueda minimizar los esfuerzos por visualizar e interpretar los datos resultado provenientes de procesos complejos.

Con base en lo anterior, esta publicación describe la metodología usada para la presentación de resultados en formato de mapa geográfico que muestre las soluciones de optimización de rutas de distribución en la aplicación denominada “Asistente automático para el diseño de rutas de distribución” del Instituto Mexicano del Transporte.

Abstract

Given the demand for process automation, access to large amounts of data, and the availability of technologies aimed at managing freight transport operations, it is extremely important to develop applications that allow the end user to visualize dynamics of the results generated by this type of solutions, in such a way that the abstraction of the complexity that these entail is achieved.

The main objective of abstraction in terms of technology is to deal with the essential characteristics of an object, without taking into account aspects that are not relevant to another object that makes use of it. In this sense and in the context of this study, it is sought that the end user interacts with the minimum possible elements to generate the expected results so that he can minimize the efforts to visualize and interpret the data resulting from complex processes.

Based on the above, this publication describes the methodology used for the presentation of results in geographic map format that shows the distribution route optimization solutions in the application called "Automatic assistant for the design of distribution routes" of the Mexican Institute of Transportation.

Resumen ejecutivo

En el presente estudio se describe el desarrollo del módulo de presentación de resultados en formato de mapas geográficos por ruta de vehículo, dicha fase de proyecto pretende proporcionar al usuario final una herramienta para visualizar los resultados de optimización en rutas de distribución entre el proveedor y sus clientes.

El *API JavaScript* de *Google Maps* permite visualizar y publicar mapas en la Web, de manera estática y dinámica a través de imágenes *Street View*, dicha herramienta facilita la presentación de rutas con marcadores de tipo inicio y fin de ruta, puntos intermedios y ventanas emergentes de etiquetación de posición geográfica, la comunicación entre los mapas resultado y el "Asistente automático para el diseño de rutas de distribución" del Instituto Mexicano del Transporte se da a través de un objeto codificado en esta nueva fase que proporciona la ruta por vehículo, latitud y longitud de los centros de distribución y los clientes.

Con la información generada por el objeto que permite la comunicación con el *API JavaScript* es posible formar los mapas resultado por vehículo, definiendo el origen como centro de distribución, el destino como último nodo de ruta optimizada o último cliente, y como puntos intermedios los clientes restantes, cada ubicación con su respectiva etiqueta y línea de seguimiento para visualizar el orden de entrega.

La propuesta en este estudio complementa la funcionalidad del "Asistente automático para el diseño de rutas de distribución" del Instituto Mexicano del Transporte al crear las bases para fungir como una aplicación para la adecuación de nuevos modelos de ruteo. Se proponen como trabajos a futuro el uso de la inteligencia artificial con sus más importantes técnicas como lo son el *Machine Learning*, *Redes Neuronales Artificiales*, *Genéticos* y *Heurísticos*.

Introducción

Uno de los principales centros de atención de las grandes empresas son la logística y el transporte, debido a que a través de estos elementos es posible reducir los gastos de operación, se refiere a cómo se gestiona el tiempo y la distancia de manera eficiente para llegar a un destino con el objetivo de satisfacer la necesidad de desplazamiento y comunicación de personas y/o mercancía (Castelli, 2019).

La clave del éxito en las empresas (Castelli, 2019) se enfoca en la administración de la cadena de suministro, abastecimiento, producción, almacenaje y optimización del transporte entre los distintos modos. Cabe señalar que se ha demostrado que la priorización y el correcto manejo de estos factores generarán ventajas competitivas.

Desde un contexto en la aplicación del método JIT (*Just In Time*) en las empresas, el cual busca reducir los recursos que son necesarios para alcanzar los mismos objetivos con una cadena sin *stock*, se requiere que el flujo de entregas sea igualado con el flujo de consumos (Cisneros, 2019) y (Cisneros, 2020). Por lo tanto, uno de los principales insumos de información será la capacidad de la flota vehicular y sus tiempos de respuesta por la parte distribuidora, y la cantidad de producto por la parte demandante, con esto es posible diseñar una logística con las mismas reglas y el mismo objetivo compartido por las dos partes.

En el 2018 el IMT desarrolla el “Asistente automático para el diseño de rutas de distribución”, obteniendo como resultados una aplicación en un entorno web que da solución de diseño de rutas de distribución a través del algoritmo heurístico de tipo CVRP con el framework OR-Tolls de Google, dicho software permite la agregación, modificación y eliminación de puntos geográficos, tanto en centros de distribución como en clientes. También da la posibilidad de especificar capacidades vehiculares homogéneas y las demandas de los clientes para de manera automática generar el árbol de solución.

La línea de investigación de Sistemas inteligentes o nuevas tecnologías del transporte del Instituto Mexicano del transporte (IMT), tiene como principal objetivo combinar la informática y las telecomunicaciones para dar solución a los problemas actuales del transporte. Los estudios más destacados en dicha línea son la implantación de tecnología ITS, evaluación de nuevas tecnologías, simulación e información en tiempo real.

“La gestión logística se ha convertido en el elemento de carácter estratégico en el mundo empresarial de la actualidad. Dentro de la misma se destaca, por su impacto en los clientes e importancia económica, el subsistema de distribución” (Reyes Chavez, Tamayo Garcia, & Leyva Zaldívar, 2011). Con base a dicho contexto, el aprovisionamiento, la distribución y el diseño de rutas optimas generan ventajas

competitivas, además de motivar a los investigadores a proponer nuevos modelos de ruteo para mejorar el rendimiento logístico

Con base en lo anterior y para atender una de las estrategias prioritarias del Programa Sectorial de Comunicaciones y Transportes 2020-2024, específicamente la estrategia 2.6.5, “Fomentar la implementación de sistemas inteligentes de transporte”, alineado al quinto objetivo del Programa para un Gobierno Cercano y Moderno (PGCM), “Establecer una Estrategia Digital Nacional que acelere la inserción de México en la Sociedad de la Información y del Conocimiento” (Secretaría de Comunicaciones y Transportes, 2020), este proyecto ha mejorado la interfaz gráfica del “Asistente automático para el diseño de rutas de distribución”, y también ha desarrollado la funcionalidad para presentar los resultados con mapas que permiten la visualización de la planeación optimizada de la distribución de mercancías.

1. Arquitecturas y lenguajes de programación

Este apartado describe la infraestructura de solución propuesta, los lenguajes de programación y las librerías utilizadas, además se presentan las proyecciones gráficas de los módulos, métodos de programación, servicios Web utilizados y sus relaciones que han permitido cumplir con el objetivo de estudio.

1.1 Arquitectura de tres capas

Es de suma importancia que esta ampliación al “Asistente automático para el diseño de rutas de distribución” se alinea a la misma arquitectura, un modelo de tres capas permite separar las partes que componen una aplicación, esto con el objetivo de facilitar el desarrollo, el mantenimiento y la escalabilidad.

Es importante mencionar que la visión de esta aplicación pretende resolver a futuro otros problemas relacionados con las tareas del transporte de carga y distribución de mercancías, implementando nuevos módulos de optimización y de gestión de la operación en tiempo real, con base a esto, la arquitectura de tres capas permite separa la interfaz de usuario de la lógica de los modelos, con esto se consigue posibilitar el uso de más de un lenguaje de programación y no limitar el uso de algoritmos ya desarrollados en el mercado (Ascencio et al. 2018).

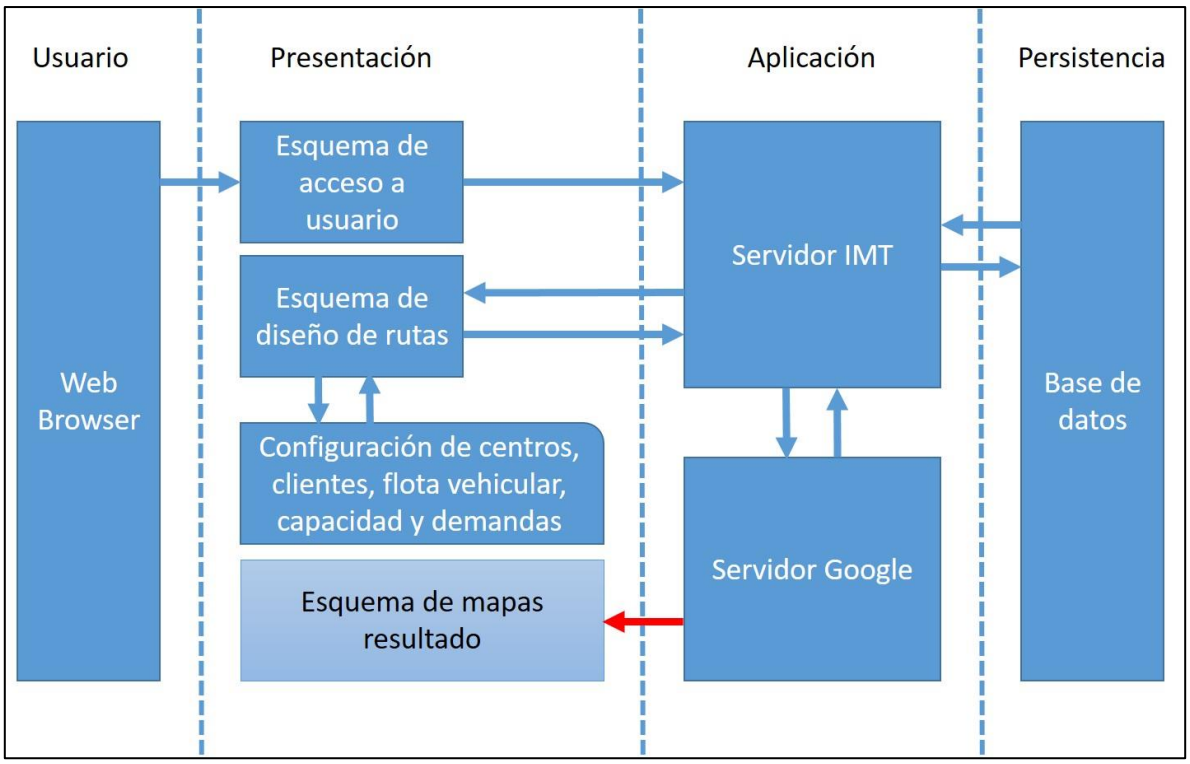

Figura 1.1 Diagrama de uso complementado con el de (Ascencio et al., 2018).

Se puede observar en la Figura 1.1 que el esquema de resultados en formato de mapas depende de la interacción con la capa de presentación y el Servidor IMT en la capa de aplicación, donde el usuario define y configura geográficamente los centros de distribución, clientes, flota vehicular, capacidades y demandas. El Servidor IMT hace una petición en formato de servicio Web al API de Google, que a diferencia del modelo de optimización, entrega un objeto visual con la representación de una ruta con origen-destino y un conjunto de puntos intermedios.

Cabe mencionar que el objeto resultado que entrega el API de Google es mapeado para considera al origen como el centro de distribución, el destino el último cliente a entregar y los puntos intermedios las entregas a los clientes restantes.

1.2 Arquitectura Cliente-Servidor

En la arquitectura cliente-servidor el procesamiento de información es distribuido, el acceso a los recursos es trasparente, quiere decir que el cliente desconoce la ubicación física de los servidores, permite la ejecución multiplataforma y la dependencia especifica de hardware (Ruiz Aranda, 2013), por lo tanto, es indispensable seguir dicho esquema para conseguir la alineación con la primera fase del “Asistente automático para el diseño de rutas de distribución”.

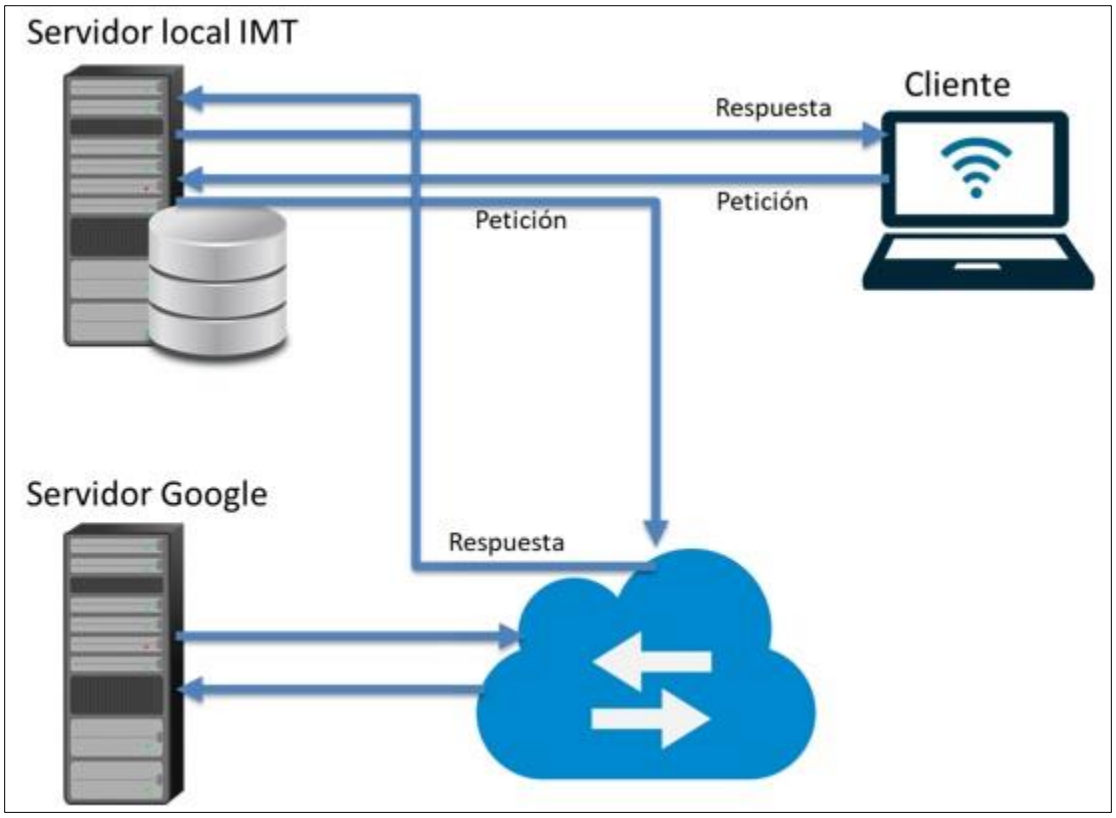

Figura 1.2 Diagrama Cliente-Servidor extraído de (Ascencio et al., 2018).

En la Figura 1.2 se representa la relación que existe entre los servidores, la nube y el cliente en la que se tiene una conexión local entre el servicio que entrega las rutas optimizadas (Servidor IMT) y la interfaz gráfica para usuario final (Cliente), así como una conexión por internet con el algoritmo de optimización y entrega de objeto de mapa resultado (Servidor Google).

Cabe mencionar que el módulo de mapas resultante, principal objetivo de este estudio, no implica una modificación a la estructura original en el contexto de este apartado.

1.3 Lenguajes de programación

Un lenguaje de programación es el idioma estandarizado que permite la comunicación entre la computadora y el desarrollador de software, sitios web, scripts, instrucciones de ejecución, etc. (Wild code school, 2021), los lenguajes de programación consisten en un conjunto de palabras clave y sintaxis para organizar las instrucciones de un algoritmo que se ejecuta a nivel máquina para resolver un problema en específico.

En la actualidad existe un sinfín de librerías de programación, las cuales son implementaciones con cierta funcionalidad y estructura capaces de ser invocadas de una manera sencilla desde otro programa, a través de sentencias específicas que implican una entrada que usualmente permiten la adaptación al procesamiento y por consecuente a una salida que resuelve problemas particulares.

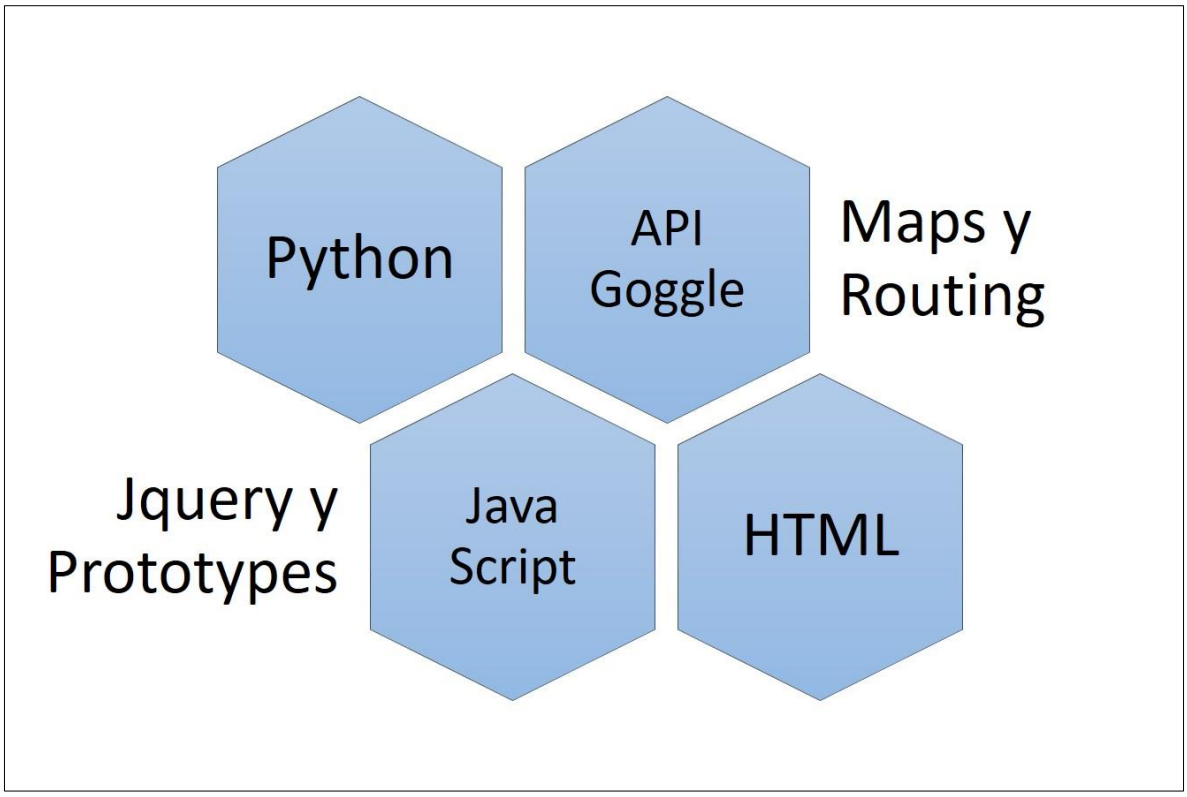

Figura 1.3 Lenguajes de programación.

En este módulo de ampliación al “Asistente automático para el diseño de rutas de distribución” se utilizaron cuatro lenguajes de programación, dos liberáis y dos sintaxis específicas, ver definición en Figura 1.3 y descripción en Tabla 1.1.

Tabla 1.1 Lenguajes y librerías

Lenguaje	Librería	Sintaxis	Usabilidad
Python	Ninguna en especifico	Programación Orientada a Objetos.	Modificación al objeto de optimización de rutas para la alineación a proceso geográfico.
Multilenguaje	API Google	Objeto Clase de tipo Mapa	Configuración de mapa resultado
HTML	Ninguna en especifico	Etiquetado para navegador Web	Crear página web para embeber mapa resultado
Java Script	Jquery	Prototype	Embeber mapa resultado, consumir servicio API Google y dar funcionalidad dinámica a los mapa resultado por vehículo.

Nota. Los lenguajes y librerías son las usadas en la fase uno del Asistente automático para el diseño de rutas de distribución, adicionando para este módulo una sintaxis del API de Google para la publicación de mapas resultado.

Fuente: Especificación realizada por el IMT

Cabe mencionar que todas las herramientas utilizadas en este estudio son de tipo *freeware* (distribución gratuita), lo cual permite el desarrollo ágil, de bajo costo y usualmente con mayor soporte gratuito. A continuación, la clasificación de dichos elementos a nivel de capa:

- Presentación. El uso de *Html (HyperText Markup Language)* para la declaración de objetos de interfaz y el lenguaje de programación *Java Script* con la librería *Jquery* para manipulación funcional de los elementos de interacción con el usuario.

- Aplicación. El lenguaje de programación *Python* como lenguaje principal para modificar el resultado de optimización a objeto geográfico y lograr la comunicación con las librerías externas, las *API's* de Google para la configuración de rutas por vehículo, además del uso *Prototypes* de *Java Script* para la codificación y manipulación de mapas resultado.

- Persistencia. En este módulo no existen procesos de gestión de base de datos.

2. Ruteo inteligente

El enrutamiento en las empresas de transporte de carga es crucial para una gestión de la flota y operación que permita tiempos de entrega eficiente y satisfacción al cliente, la primera meta busca reducir los costos y la segunda mejorar la posición competitiva. Cabe mencionar que adicionalmente, una correcta administración del ruteo permite incrementar la vida útil de los vehículos y reducir los gastos de combustible y de mantenimiento (Sanz, 2020a).

Según (Sanz, 2020a), el protocolo de ruteo perfecto considera lo siguiente:

- Minimizar el tiempo total de transporte.
- Lograr una distancia de recorrido lo menor posible.
- Disminuir tiempos de espera.
- Reducir el uso de los vehículos.

En resumen, un proceso de ruteo eficiente calibra la utilización de los recursos necesario de la operación, por consecuente reduce los costos totales y maximiza los beneficios, tanto al cliente como al proveedor.

2.1 Ventajas del ruteo inteligente

El principal objetivo del ruteo inteligente consiste en el ahorro del tiempo y la simplificación del proceso de distribución, es cierto que un sistema de este tipo requiere dispositivos inteligentes basados en geolocalización, un algoritmo eficiente de optimización y otro de predicción del tráfico, es indispensable tener en cuenta que una distancia corta no significa un tiempo de recorrido menor, pues este depende de la congestión vehicular actual y a futuro.

A continuación, se presentan los beneficios de un ruteo inteligente según (SITRACK, 2022):

- Se minimiza el tiempo de planeación y por consecuente mejora la capacidad de reacción.
- Se reducen tiempos logísticos en el proceso de la generación de rutas.
- Es posible implementar la cartera de clientes, no sólo por la reducción de los tiempos de recorrido, por consecuente la capacidad de entrega, sino que también se conoce la disponibilidad de la flota a futuro.
- Se optimiza el servicio ya que el detalle de las distancias está planificado.
- Se minimizan los errores de planificación, un humano puede pasar por alto alguno de los factores que deben ser considerados.

- Se generan ahorros en combustible y en gastos de mantenimiento, de echo es posible restringir vehículos específicos a distancias largas, permitiendo planificar las fechas de taller.
- Control en entrega de mercancías y evitar pérdidas de cargamento, monitorear la operación en tiempo real es un potencial altamente considerable.
- Es posible considerar trayectos adecuados, previniendo rutas peligrosas o prohibidas.

2.2 Inteligencia artificial en el ruteo

El auge de la inteligencia artificial (IA) fue consecuencia del cómputo en la nube, aunque se contaba con metodologías poderosas, no se contaba con el poder computacional actual, específicamente con la posibilidad de contar con GPU y TPU desde internet a un bajo costo, la cantidad de información disponible y acumulada en la web, y la capacidad de generar procesos en paralelo y en un entorno de clúster (BIG Data). Cabe mencionar que paradigmas como IoT (Internet of Things) también ha sido importante en este contexto, ya que ha sido la principal herramienta para recabar información de distintos dispositivos en tiempo real y en entera comunicación.

En la actualidad la inteligencia artificial puede ser aplicada por cualquier empresa, no sólo por los motivos mencionados en el párrafo anterior, sino que también debido a la existencia de organizaciones con mucha experiencia en este contexto y que uno de sus principales giros es precisamente proveer de plataformas que permiten generar modelos de optimización y de predicción de una manera muy simple sin la necesidad de ser un experto matemático, por ejemplo se puede mencionar AWS de AI y ML de Amazon, y Microsoft AI.

Cabe mencionar que la principal dificultad para generar un adecuado protocolo de ruteo inteligente a través de inteligencia artificial, es la disponibilidad de un gran volumen de datos de alta cálida, por ejemplo, se requieren las posiciones actuales y en tiempo real de la flota vehicular, los históricos de ruteo planeado y el real, los registros de tráfico y los detalles de distribución. Por otro lado, se sabe que una de las principales fortalezas de la IA es la predicción y la optimización de procesos, lo cual requiere de un tiempo considerable para su despliegue en producción, es necesario probar varios modelos, calibrarlos, evaluarlos y compararlos, según (Sanz, 2020b), muchas veces los clientes tienden a desesperarse por no haber resultados inmediatos tangibles, sobre todo cuando las empresas no son conscientes de lo que este tipo de proyectos conlleva.

Según (Sanz, 2020b), en la optimización de procesos logísticos en la gestión de la flota vehicular la IA permite:

- Identificación automática de vehículos.
- Identificación automática de choferes.
- Vehículos autónomos.

- Predicción de tráfico.
- Predicción de zonas conflictivas o de alta criminalidad.
- Optimización de rutas.
- Optimización e tiempos de espera.
- Entre muchas otras más.

Una de las principales ventajas del sector logístico es que la mayoría de las empresas transportistas en México y en todo el mundo cuentan con dispositivos de conectividad, lo que permite recolectar información bastante útil para realizar analíticas robustas.

2.3 Heurísticos para la optimización de rutas

Los algoritmos heurísticos permiten generan soluciones de buena calidad sin garantizar el alcance de la óptima, su principal ventaja es la obtención de resultados en tiempos de ejecución reducidos (Escudero, 2017).

Los tipos de heurísticas usadas para el problema de ruteo son:

- Constructivas. Técnicas iterativas que gradualmente van creando una solución, donde en cada paso se agrega un nuevo elemento que se evalúa y se deshecha cuando no mejor que otro. Un claro ejemplo es el método de *Ahorros de Clarke & Wright* (Alinaghian, Kaviani, & Khaledan, 2015).
- De mejora. Técnica que parte de una solución factible, a través de intra-ruta y extra-ruta, en la primera con heurísticas haciendo variaciones entre los arcos y la segunda haciendo heurísticas entre dos o más rutas, Un claro ejemplo es el método Swap (Lum et al., 2015).
- De relajación. Técnica que *"descompone un modelo lineal entero en dos conjuntos de restricciones, difíciles y fáciles. Las primeras son relajadas al incluirlas en la función objetivo multiplicándolas por un factor de penalización, tal y como se hace en el método de los multiplicadores de Lagrange, sirviendo esto para acotar el problema original, lo que reduce el tiempo empleado en el proceso de resolución"* (Escudero, 2017).
- Metaheurísticos. Técnica que hace la combinación de distintas técnicas heurísticas para realizar la exploración del dominio de búsqueda de una manera más eficiente, se consideran seis tipos de metaheurísticas: recosido simulado, genéticos, redes neuronales, búsqueda tabú, colonia de hormigas y *Greedy Randomized Adaptative Search Procedure (GRASP)* (Toth & Vigo, 2002).

Ya que el ruteo es un problema de búsqueda de espacios de soluciones locales para encontrar óptimos en tiempos razonables y evitar el estancamiento en el universo de posibilidades, éste se considera de tipo *NP-Hard*, por lo tanto, se justifica el uso de algoritmos heurísticos, *"es posible expresar que en problemas de espacios de búsqueda muy grandes no sólo es práctico usar este tipo de técnicas, sino que es esencial, pues dicha formulación discrimina rutas de búsqueda no prometedoras, lo cual, si se plantea de manera objetiva, las soluciones siempre*

serán óptimas, cabe mencionar que dichos modelos sacrifican la validación del universo de soluciones y la detección de la solución más óptima, para generar una en un tiempo razonable" (Ascencio et al., 2018).

3. API de Cloud

El API de Google Cloud es una herramienta que permite automatizar un sinfín de tareas a través de los distintos lenguajes de programación, la manera de comunicarse con sus funcionalidades es con llamadas tipo REST, las cuales son interfaces de codificación que permite la interacción con servicios en la Web, dichas peticiones están definidas por protocolos que buscan la integración con el software de aplicación, como si fuera un contrato entre consumidor (el que hace la llamada) y el productor (quien entrega la respuesta), suelen permitir el envío de parámetros que permiten personalizar el procesamiento y la salida.

3.1 API de Direcciones (*Directions API*)

El API de direcciones es un servicio web de Google que a través del protocolo HTTP permite solicitudes que devuelven indicaciones de cómo llegar de un origen a un destino, el formato respuesta de este recurso es JSON o XML, además puede recibir parámetros de entrada que permiten personalizar la salida a distintos modos de transporte, tales como: transporte público, a pie, en bicicleta o conducción.

La respuesta de *Directions API* responde en tiempo real y a petición del usuario, la respuesta puede resultar en un documento de indicaciones o a través de un mapa, sin embargo, para este último es necesario usar *Maps JavaScript API*.

Los lenguajes de programación permitidos para consumir este servicio son:

- Del lado del servidor.
    - Pthon
    - Go
    - Node.js
    - Java
    - Ruby
- Del lado del cliente.
    - Javascript.

Si se requiere conocer los detalles de los parámetros de las peticiones, restricciones, políticas y objeto de respuesta, es posible revisar la publicación técnica No. 583 de (Ascencio et al., 2018), este documento se concentrará en los aspectos del módulo de presentación de resultados en mapas digitales.

3.2 Maps JavaScript API

El API de *JavaScript* permite la creación de mapas personalizados en páginas web y dispositivos móviles, puede ser en formato de ruta, satélite, hibrido y de terreno. Es posible configurar sus capas de mapa, estilos, controles, eventos, etc. (Google, 2021a).

Google te permite a través de esta API usar sus controles visuales de mapa, por ejemplos el objeto para modificar el tipo de mapa, modo pantalla completa, arrastre de hombrecito y, acercar y alejar.

Figura 3.1 Mapa ejemplo con API JavaScript.

Para poder crear el mapa de la Figura 3.1 con *JavaScript* se requiere:

- Crear un *script HTML5*.
- Agregar una etiqueta *HTML5* de tipo *div* con atributo *name* “map”.
- Definir la función JavaScript de creación de mapa en el *div.*
- Cargar mapa JavaScript usando el evento *load*.

La principal restricción de este tipo de mapas es que cada proyecto debe contener los créditos de Google a nivel visual y de programación, para controlar dicho suceso, los servicios deben contener como parámetro una llave que se configura al crear la cuenta en *Google Platform*, esta llave puede o no tener un costo, Google regala una cantidad determinada de peticiones de mapa gratuitas por día, sino se excede, la cantidad a pagar es nula.

A continuación, un ejemplo de carga de mapa con JavaScript:

```
// The location of Uluru
const uluru = { lat: -25.344, lng: 131.036 };
// The map, centered at Uluru
const map = new google.maps.Map(document.getElementById("map"), {
  zoom: 4,
  center: uluru,
});
```

Figura 3.2 Carga de mapa con API Google Maps.

Se observa en la Figura 3.2 que se requiere crear un objeto de tipo *Google Maps*, donde se define el objeto web donde se incrustará, en este caso “map”, el acercamiento por default *zoom = 4* y un centro de vista que con posición geográfica latitud y longitud queda configurada.

3.3 Directions Requests

Directions Request es el objeto donde se encuentran los parámetros de entrada a las solicitudes de *Directions API*, para obtener los resultados de este servicio se requiere crear un objeto *DirectionsService* y llamar al método *route()*, para iniciar la solicitud se le pasa como argumento el objeto denominado *DirectionsRequest*.

Los atributos del objeto *DirectionsRequest* son los siguientes:

```
{
  origin: LatLng | String | google.maps.Place,
  destination: LatLng | String | google.maps.Place,
  travelMode: TravelMode,
  transitOptions: TransitOptions,
  drivingOptions: DrivingOptions,
  unitSystem: UnitSystem,
  waypoints[]: DirectionsWaypoint,
  optimizeWaypoints: Boolean,
  provideRouteAlternatives: Boolean,
  avoidFerries: Boolean,
  avoidHighways: Boolean,
  avoidTolls: Boolean,
  region: String
}
```

Figura 3.3 Argumentos *DirectionsRequest*.

La definición de cada uno de los atributos presentados en la Figura 3.3 consiste en:

- *Origin*. Latitud y longitud de la posición origen, es un parámetro obligatorio.

- *Destination*. Latitud y longitud de la posición destino, es un parámetro obligatorio.
- *travelMode.* Específica el modo de transporte para usarlo en el cálculo de tiempos y distancias, es un parámetro obligatorio.
- *transitOptions.* Especifica las opciones de tránsito, están sujetas al modo de viaje, por ejemplo, evitar autopistas, peajes, puntos intermedios, etc., es un parámetro opcional.
- *drivingOptions.* Especifica los valores de manejo, por ejemplo, la fecha y hora de salida, y especificaciones de tráfico como uso de tránsito con predicción o actuales. Es un parámetro opcional.
- *unitSystem.* Especifica el tipo de unidades, puede ser en millas o enkilometros. Es un parámetro opcional.
- *Waypoints,* Es un objeto latitud-longitud multidimensional que especifica los puntos intermedios en una ruta. Es un parámetro opcional.
- *optimizerWaypoints. Especifica cómo deben tratarse los puntos intermedios, si esta con valor verdadero realiza un orden eficiente y genera un objeto adicional a la respuesta, de lo contrario, usa el orden sin modificación.* Es un parámetro opcional.
- *provideRouteAlternatives.* Si esta en valor verdadero permite agregar rutas alternativas, sin embargo, esta opción es disponible cuando no se usan puntos intermedios, esto debido al tiempo de respuesta que se vuelve poco práctico. Es un parámetro opcional.
- *avoidFerries.* Si esta en valor verdadero indica que debe evitar transbordadores. Es un parámetro opcional.
- *avoidHighways.* Si esta en valor verdadero indica que debe evitar autopistas principales. Es un parámetro opcional.
- *avoidTolls.* Si esta en valor verdadero indica que debe evitar peajes. Es un parámetro opcional.
- *region.* Específica el código de región.

Esta publicación no tiene como objetivo presentar un tutorial para crear mapas con Google, sin embargo, se considera necesario conocer los parámetros de entrada a las solicitudes del servicio, ya que esto permitirá conocer la funcionalidad completa y el límite de personalización en la salida resultado.

3.4 CalculateAndDisplayRoute

CalculateAndDisplayRoute es el objeto que permitirá la comunicación entre el objeto *DirectionsService* y el objeto *google.maps*, este elemento de codificación contiene la definición de los *waypoints* a través de un arreglo de latitudes y longitudes de los puntos intermedios de una ruta, la cual es especificada en *DirectiosnRequest*.

Para que un objeto tipo ruta sea asociado a un mapa gráfico de *Google Maps JavaScript,* se requiere agregar a la definición del evento *load* (ver en Figura 3.2) el método *CalculateAndDisplayRoute*.

```
function initMap() {
  const directionsService = new google.maps.DirectionsService();
  const directionsRenderer = new google.maps.DirectionsRenderer();
  const map = new google.maps.Map(document.getElementById("map"), {
    zoom: 6,
    center: { lat: 41.85, lng: -87.65 },
  });

  directionsRenderer.setMap(map);
  document.getElementById("submit").addEventListener("click", () => {
    calculateAndDisplayRoute(directionsService, directionsRenderer);
  });
}
```

Figura 3.4 Load Map con calculateAndDisplayRoute (Google, 2021b).

Se puede observar en la Figura 3.4 que dentro de las llaves de codificación del método *initMap()* se encuentra el evento *click* de un botón con identificador único *submit*, que indica que al ser presionado el mapa se actualizara a una ruta especificada en la función *directionsService*.

A través de la configuración definida anteriormente es posible publicar mapas de ruta con puntos intermedios, tal como se presenta en la Figura 3.5, donde **A** es el origen, **C** es el destino y **B** es un punto intermedio, normalmente los puntos intermedios o *waypoints* se usan para indicarle al tuteador la restricción de paso por carrera.

Figura 3.5 Mapa con puntos intermedios (Google, 2021b).

Google Maps tiene la capacidad para generar muchas otras configuraciones de personalización en los mapas, funcionalidades y restricciones, sin embargo, como ya se mencionó anteriormente, no es objetivo de esta publicación presentar un tutorial, pero si dar una idea general de cómo se ha empleado *Maps JavaScript* al problema de distribución de mercancías con la publicación de mapa resultado.

4. Desarrollo del módulo de presentación de resultados

En este apartado se describen los principales objetivos del desarrollo del módulo de presentación de resultados del Asistente automático de distribución de rutas, los requerimientos detectados, las restricciones y los problemas a los que se enfrentaron en la fase de codificación, al final se muestran las interfaces de usuario que ejemplifican el funcionamiento final de la plataforma desarrollada.

4.1 Objetivos

El principal objetico es la generación de mapas que permita visualizar el resultado generado por el “Asistente automático para el diseño de rutas de distribución” y generar los módulos necesarios para la publicación en un servidor web.

Objetivos específicos.

- Reingeniería del objeto de resultados de distribución para alinearlo al objeto de mapa *Maps JavaScript*.
- Desarrollo del módulo de presentación de resultados con mapas de visualización.
- Pruebas, mantenimiento y despliegue de plataforma.

4.2 Requerimientos

Los requerimientos detectados fueron los siguientes:

- Requerimientos del negocio.
    - Una herramienta que permita al investigador la visualización de mapas con la ruta de distribución de mercancías,
    - Infraestructura con alta capacidad de escalabilidad para la adecuación de nuevos modelos de solución.
- Requerimientos de usuario.
    - Interfaz gráfica intuitiva y amigable.
    - Interfaz que permita visualizar las rutas óptimas al conjunto de centros de distribución.
    - Interfaz que permita la navegación de rutas de distribución por vehículo.
- Requerimientos de sistema.
    - Organizar la información geográfica de las configuraciones del problema de ruteo y establecer una estructura de datos que pueda manipularse con eficiencia.

  - Modificar el objeto resultado de distribución de rutas para que tenga la estructura que *Maps JavaScript* requiere para generar los mapas.
  - Generar las clases (objetos de programación) que le den el formato a los datos de tal manera que sean reconocidos por las librerías de ruteo.
  - Permitir el acceso multiusuario a la herramienta en un entorno Web.
- Requerimientos de software.
  - Funcionales.
    - Permitir la navegación entre mapas de ruta por vehículo.
    - Presentar marcadores de mapa por cliente y centro de distribución.
    - Permitir expandir y reducir mapa para la visualización exacta de la ubicación,
    - Visualizar el resultado del algoritmo de ruteo sobre *OpenStreetMap*.
  - No funcionales.
    - Configuración de un servidor Web con acceso local.
- Requerimientos de hardware.
  - Un servidor con sistema operativo Windows Server de 64 bits dedicado a la aplicación.
  - Acceso a internet con al menos 4 Mb. de ancho de banda.

4.3 Alcances y limitaciones

A continuación, la detección de los alcances y limitaciones:

- Alcances.
  - El módulo de publicación de mapas presenta una navegación por vehículo y ruta optimizada por el Asistente automático de distribución de rutas.
  - Permite reducir y expandir el mapa.
  - Permite mover desde el *mouse* de la computadora el mapa para la personalización visual.
  - Permite conocer el nombre de los clientes y centros de distribución definidos en el Asistente automático de distribución de rutas.
- Limitaciones.
  - Un solo centro de distribución.
  - Capacidades homogéneas para la flota vehicular.
  - Supone que los centros de distribución soportan la demanda de los clientes.
  - Las carreteras prohibidas no son consideradas.
  - Permite sólo veinticinco puntos intermedios.
  - El mapa es generado por ruta y vehículos, no se visualiza la operación completa en un solo mapa.
  - Considera un IP homologada con salida a Internet, pero requiere del acceso a la INTRANET del Instituto Mexicano del Transporte.

4.4 Complicaciones en el desarrollo

A continuación, se describen algunas de las complicaciones que se lograron resolver en el desarrollo del módulo y la adecuación en las librerías y herramientas mencionadas anteriormente:

- El algoritmo generaba como resultado un texto de las rutas por vehículo, por lo tanto, fue necesario formatearlo para adicionar los datos geográficos y la asignación de los puntos intermedios como clientes.
- Fue necesario mapear el funcionamiento de *DirectionsResponse* para que considere el origen como centro de distribución, el destino como último cliente y los puntos intermedios como el resto de los clientes.
- Se formateo el resultado de optimización para fragmentar la respuesta por ruta de vehículo, con esto fue posible desarrollar la navegación de la operación desde interfaz gráfica.

4.5 Resultados del desarrollo

A continuación, se muestran los resultados gráficos obtenidos en el desarrollo, agregando en orden la navegación completa de la plataforma, desde el acceso a usuario, definición de centros de distribución, clientes, flota vehicular, capacidades, demandas y, por último, el sistema de navegación mapa resultado:

Figura 4.1 Acceso a usuarios.

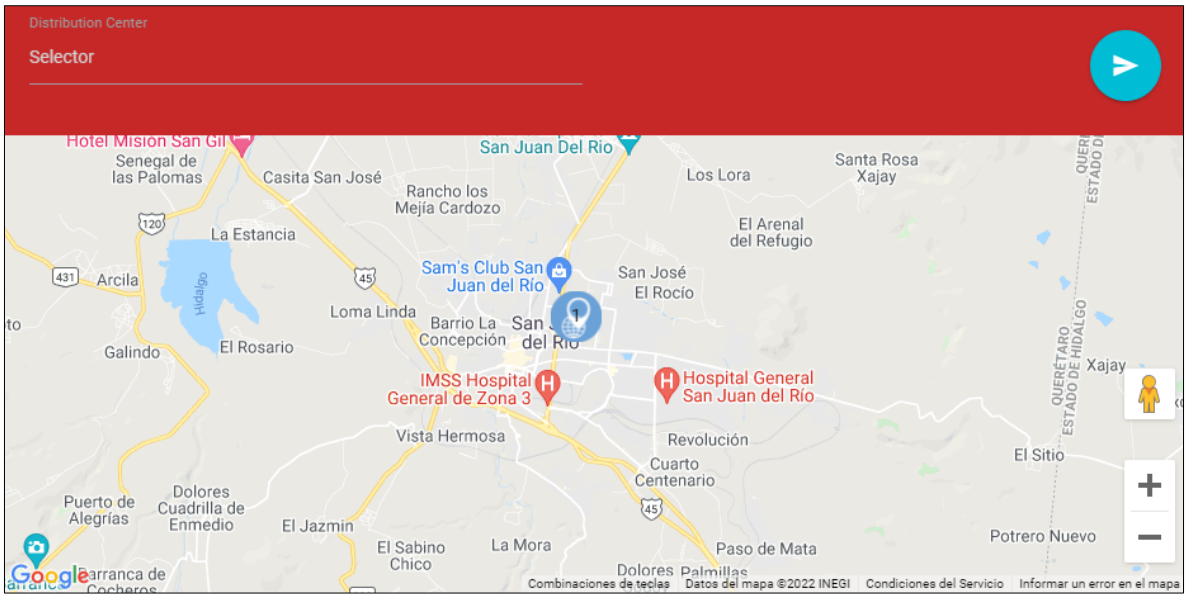

Figura 4.2 Definición de ubicación y nombre del centro de distribución.

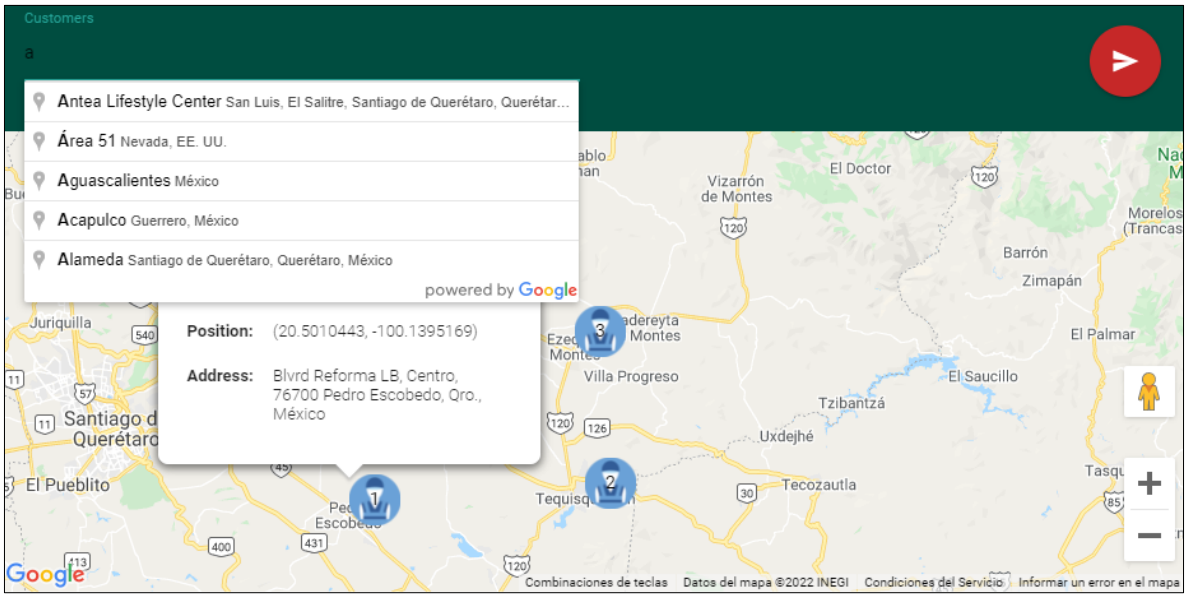

Figura 4.3 Definición de ubicación y nombre de clientes.

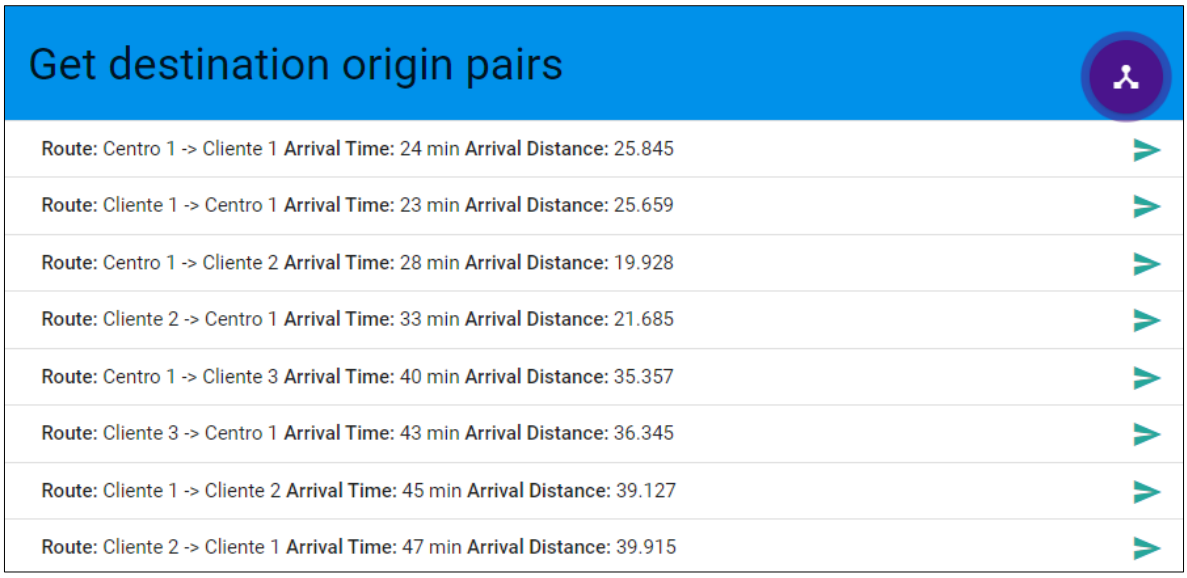

Figura 4.4 Generación de matriz de distancias y arribos todos vs todos.

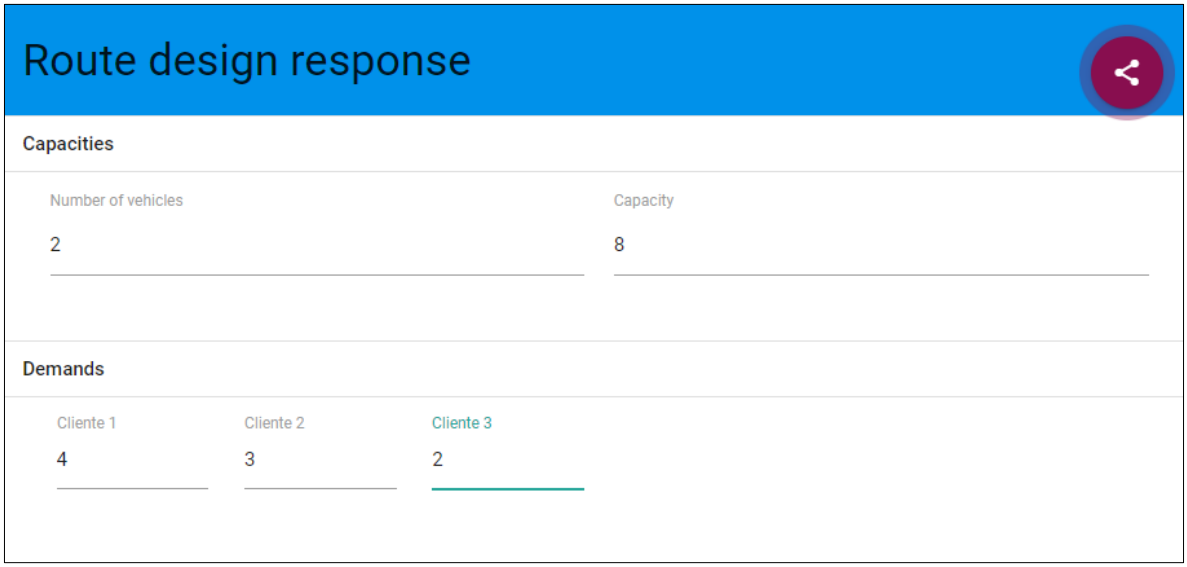

Figura 4.5 Configuración de flota, capacidades y demandas.

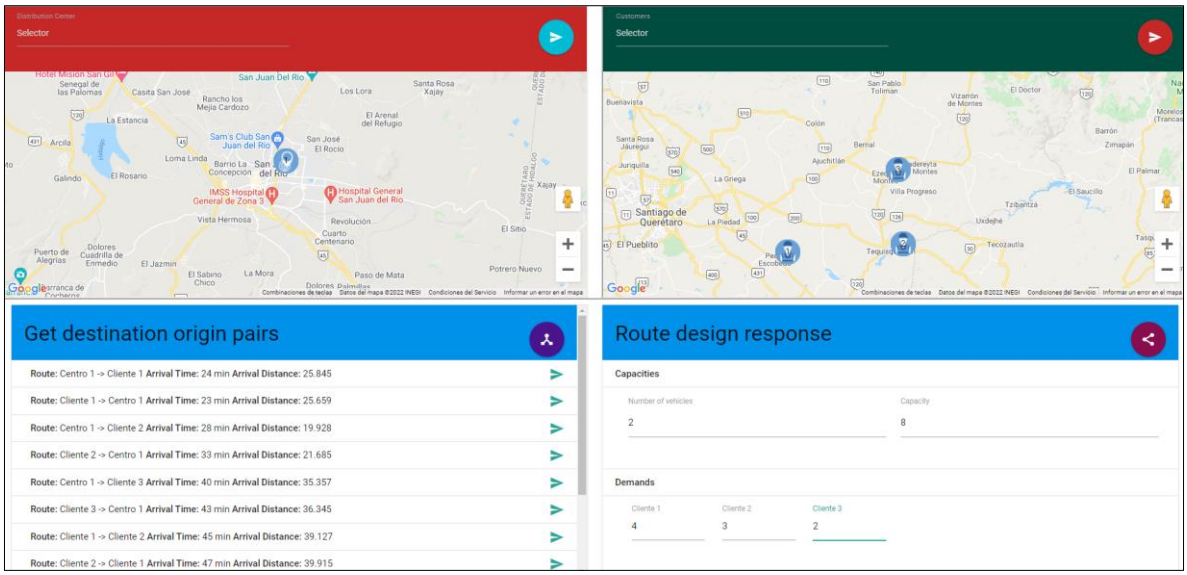

Figura 4.6 Sistema integral de configuración de problemas.

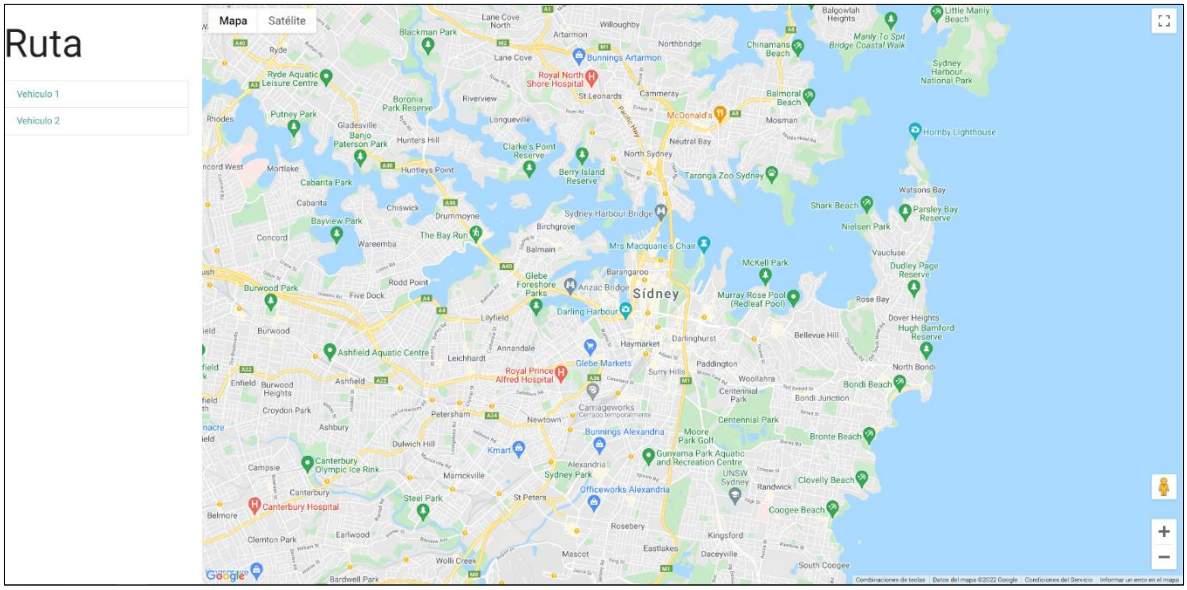

Figura 4.7 Sistema integral de configuración de problemas.

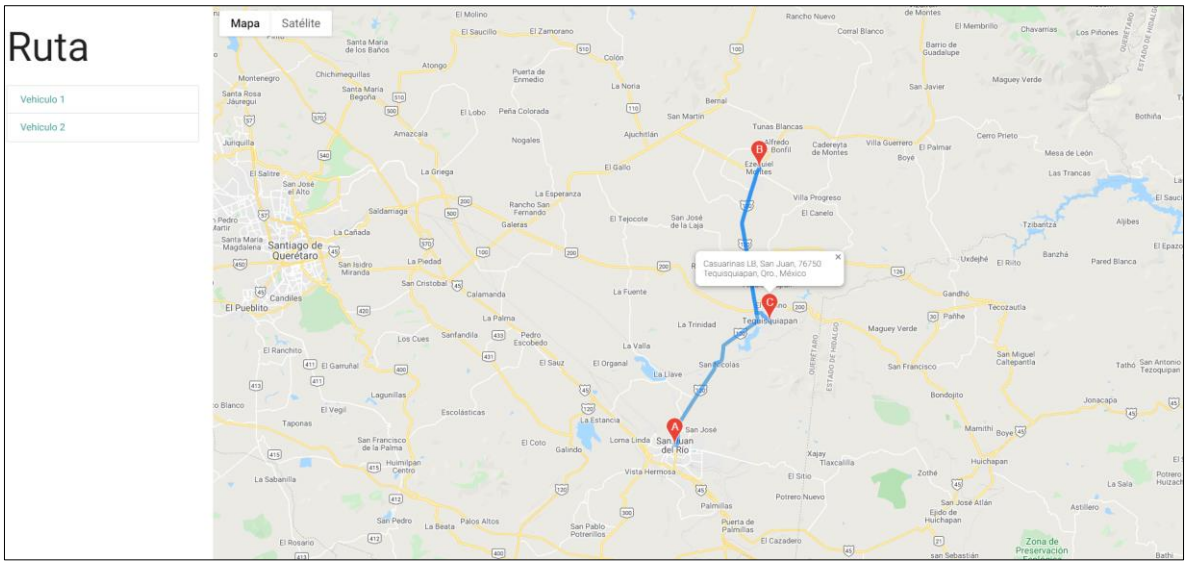

Figura 4.8 Ejemplo de visualización de ruta con puntos intermedios.

Conclusiones

Definitivamente incorporar el ruteo inteligente en empresas de transporte de carga, genera ventajas competitivas, no solo en cuanto a calidad del servicio y eficiencia en las rutas de distribución, sino que también permite un ahorro considerable en tiempo y dinero.

Es posible concluir que las empresas que tienen por giro la gestión logística de la operación de distribución de mercancías conseguirán mejorar posición competitiva conforme incremente su madurez en cuanto a sus protocolos de ruteo inteligente, actualmente los clientes desean eficiencia en sus entregas y seguridad en la entrega, al igual que los socios, maximizar sus ingresos y seguridad en sus trayectos.

Los algoritmos heurísticos generan soluciones aceptables sacrificando encontrar el óptimo, pero en tiempos prácticos de ejecución.

Maps JavaScript permite la publicación de mapas de ruteo con puntos intermedios, los cuales en este estudio son considerados como clientes, origen el centro de distribución y destino la última milla o el último cliente.

A continuación, se presentan los trabajos a futuro.

- Adicionar ventanas de tiempo al heurístico.
- Desarrollar un heurístico que considere rutas prohibidas, de alta accidentabilidad e inseguras.
- Una propuesta de infraestructura IoT para un cluster de transportistas y la generación de los requerimientos para una ciudad inteligente en el transporte de carga.
- Considerar más de un centro de distribución en el CVRP.
- Permitir como primera instancia la definición de rutas prohibidas y zonas de alta accidentabilidad, validando la factibilidad de calcularlo u obtenerlo a través de un servicio web. Como segunda instancia es desarrollar modelos de *Machine Learning* para la detección y pronóstico de rutas prohibidas y zonas inseguras.
- Proponer un modelo de predicción de tráfico a través de *Deep Learning* alimentado por la posición actual de los vehículos asignados a una ruta en específica.

Bibliografía

Alinaghian, M., Kaviani, Z., & Khaledan, S. (2015). A novel heuristic algorithm based on Clark and Wright Algorithm for Green Vehicle Routing Problem. *International Journal of Supply and Operations Management*, *2*(2), 784–797.

Ascencio, A., Bustos, A., Jiménez, E., Balbuena, J., & Zamora, A. (2018). *Asistente automático para diseño de rutas de distribución.* Querétaro, México. Retrieved from https://imt.mx/archivos/Publicaciones/PublicacionTecnica/pt538.pdf

Castelli, F. (2019). Logística y transporte, optimizan tiempo y espacio en las empresas. Retrieved February 8, 2022, from https://www.multi-packing.com.co/logistica-y-transporte

Cisneros, J. (2019). Jut In Time (JIT) o Stock Cero. Retrieved February 8, 2022, from https://www.datadec.es/blog/just-in-time-jit-o-stock-cero

Cisneros, J. (2020). Seis consejos para reducir el tiempo de la cadena logística. Retrieved February 8, 2022, from https://www.datadec.es/blog/6-consejos-reducir-tiempo-de-cadena-logistica

Escudero, A. (2017). *Resolución heurística de un problema de rutado con aplicaciones para el comercio electrónico*. Universidad de Sevilla.

Google. (2021a). Maps JavaScript API. Retrieved February 11, 2022, from https://developers.google.com/maps/documentation/javascript/overview?hl=es-419

Google. (2021b). Waypoints in Directions. Retrieved February 14, 2022, from https://developers.google.com/maps/documentation/javascript/examples/directions-waypoints?hl=es-419

Lum, O., Chen, P., Wang, X., Golden, B., & Wasil, E. (2015). A Heuristic Approach for the Swap-Body Vehicle Routing Problem. *14th INFORMS Computing Society Conference*, *11*(13), 172–187.

Reyes Chavez, E., Tamayo Garcia, Y., & Leyva Zaldívar, M. (2011). Procedimiento para el diseño de redes de distribución logística. Retrieved from http://www.eumed.net/ce/2011b/cgz.htm

Ruiz Aranda, P. (2013). Arquitectura cliente/servidor. Retrieved from http://somebooks.es/arquitectura-clienteservidor/

Sanz, J. Á. (2020a). En busca del recorrido perfecto. Retrieved February 10, 2022, from https://blog.getpulpo.com/blog/gestión-de-flotas-y-movilidad-cómo-planificar-la-postpandemia-0

Sanz, J. Á. (2020b). Inteligencia Artificial en la gestión de flotas. Retrieved February 11, 2022, from https://blog.getpulpo.com/blog/inteligencia-artificial-en-la-gestión-de-flotas-con-luis-pintado

Secretaría de Comunicaciones y Transportes. (2020). Programa Sectorial de Comunicaciones y Transportes 2020-2024. Retrieved November 16, 2021, from https://www.dof.gob.mx/nota_detalle.php?codigo=5596042&fecha=02/07/2020

SITRACK. (2022). Ventajas del ruteo inteligente. Retrieved February 10, 2022, from http://blog.sitrack.com/ventajas-del-ruteo-inteligente

Toth, P., & Vigo, D. (2002). *The Vehicle Routing Problem*. (2002 Society for Industrial and Applied Mathematics, Ed.) (Ilustrada).

Wild code school. (2021). Tipos de Lenguajes de Programación Más Usados en el 2021. Retrieved February 10, 2022, from https://www.wildcodeschool.com/es-ES/blog/tipos-de-lenguajes-de-programacion

COMUNICACIONES
SECRETARÍA DE COMUNICACIONES Y TRANSPORTES

INSTITUTO MEXICANO DEL TRANSPORTE

Km 12+000 Carretera Estatal 431 “El Colorado Galindo”
Parque Tecnológico San Fandila, Mpio. Pedro Escobedo,
Querétaro, México. C.P. 76703
Tel: +52 (442) 216 97 77 ext. 2610
Fax: +52 (442) 216 9671

publicaciones@imt.mx

http://www.imt.mx/