ESCUELA SUPERIOR POLITECNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

“ANÁLISIS Y OPTIMIZACIÓN DE SISTEMAS DE ENRUTAMIENTO DE MEDIOS DE TRANSPORTE BASADO EN DATA DE TRÁFICO PARA APLICACIONES DE LOGÍSTICA, UTILIZANDO MÉTODOS HEURÍSTICOS.”

INFORME DE MATERIA INTEGRADORA

Previo a la obtención del título de

INGENIERO EN CIENCIAS COMPUTACIONALES
ESPECIALIZACIÓN MULTIMEDIA

ALVARO JAVIER ORTIZ SUAREZ

GUAYAQUIL – ECUADOR

AÑO: 2016

TRIBUNAL DE EVALUACIÓN

Ph.D. Lorena Carlo Unda	**Ph.D. Xavier Ochoa Chehab**
PROFESOR EVALUADOR	PROFESOR EVALUADOR

DECLARACIÓN EXPRESA

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, me corresponde exclusivamente; y doy mi consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

Álvaro Ortiz Suárez

RESUMEN

El objetivo principal del proyecto es resolver las complicaciones que enfrentan las empresas que requieren del transporte de bienes, y/o productos como parte fundamental de sus procesos de negocios.

Todo negocio que calce la previa descripción desea vender. Para vender, debe movilizar productos en su zona de efecto a diversos clientes. Para poder movilizar sus productos necesita flotas de vehículos, y personas que se encarguen de recorrer las calles para llegar a los clientes. Mientras más demanda tenga la empresa, más vehículos deberá poder coordinar para abastecer sus pedidos.

Llega un punto en que simple coordinación por parte de humanos deja de ser suficiente, e incluso se vuelve perjudicial por la introducción de errores al flujo. Estos errores pueden causar grandes pérdidas a los negocios no solo de forma financiera, sino también en credibilidad ante sus clientes. Estos altercados pueden traer, si ocurren de forma constante, grandes consecuencias que pueden causar el cierre de negocios [1].

El servicio desarrollado en este proyecto busca aliviar la carga de las empresas al momento de realizar sus entregas y pedidos, no solo cumpliendo con los horarios, pero también reduciendo la magnitud de recursos necesarios para abastecer un número creciente de clientes.

Se orientó el proyecto a crear una plataforma de código libre que sea capaz de resolver estos problemas en un tiempo aceptable y con resultados muy cercanos a la solución óptima. Se utilizó un entorno en Java debido a dependencia de las herramientas que se utilizaron como base del servicio de planificación de rutas: jsprit, y graphhopper. A su vez, el servicio es fácilmente extensible e incluso puede ser expuesto a aplicaciones a través de un servicio web.

La razón por la cual se concluyó no utilizar como centro, APIs más maduros como los de múltiples empresas ya existentes fue debido a dos razones:

- La mayoría de soluciones son de código cerrado, para proteger la propiedad intelectual de las empresas que lo distribuyen. Por consecuente, no puede ser adaptado cuando la funcionalidad proveída por los dueños del código fuente de estos servicios de enrutamiento no es suficiente para satisfacer las

necesidades funcionales y/o de magnitud de quienes necesitan generar rutas complejas. Esta responsabilidad suele caer en los departamentos de operaciones quienes están encargados de esta organización.

- Al no poder personalizar la funcionalidad interna al software de los proveedores como Google, y Waze, se suele trabajar alrededor de lo ya establecido, lo cual no siempre suele ser eficiente. Como ejemplo se encuentra el sistema de enrutamiento de Google, el cual sabemos que puede crear rutas muy complejas con múltiples puntos de acción pero es limitado hasta 8 en su plan gratis. Al no poder acceder al código fuente, el cual es ofuscado, no es posible utilizar el contenido para satisfacer las necesidades especificas de la compañía que requiere de este servicio.
- La presencia de limitaciones impuestas por los distribuidores a los planes gratis que acompañan sus servicios.

Al final del proyecto se pudo concluir que el servicio de planificación y enrutamiento, denominado de ahora en adelante como “Proyecto Carrozza”, es una buena solución para resolver problemas de planificación de rutas para pequeñas y medianas empresas que puedan incorporar el servicio en sus sistemas y no puedan costear un framework dedicado de gama alta como Google Maps for Work.

INDÍCE GENERAL

CAPÍTULO 1

1 PLANIFICACIÓN DE RUTAS EFECTIVAS PARA ENTREGA DE ENVÍOS

El problema que se trata de resolver es el enrutamiento efectivo en situaciones reales entre distintos puntos geográficos. Esto quiere decir que un individuo pueda moverse en un área geográfica alcanzando todos los puntos preestablecidos por el problema utilizando la menor cantidad de recursos, comúnmente tiempo o distancia.

1.1 Descripción del Proyecto

Los problemas que caen en la previa descripción son conocidos como problemas Vehicle Routing Problems (VRP) & Rich Traveling Salesman Problem (RTSP) [2]. Estos problemas son una causa de gran frustración para múltiples negocios e e-commerces que realizan entregas y/o pedidos a domicilio o que dependen de la movilización de sus transportes para entregas de paquetes.

Esto se debe a la complejidad de coordinar los recursos disponibles (vehículos) con la demanda (envíos) y sus respectivas necesidades o características (localización, ventanas de tiempo disponible, etc.) al momento de planear las rutas que se fuesen a efectuar.

En los últimos años, varias compañías han optado por construir sus propios sistemas de enrutamiento a través de técnicas como crowdsourcing [3] [4] y big data analysis. Algunas de estas, como Google [5], o Waze [6], dieron un paso más allá y expandieron sus servicios para el consumo de múltiples clientes.

Estas soluciones han permitido a muchos negocios, que dependen de realizar entregas, poder garantizar buenas rutas de navegación. Estas rutas representan conjuntos de jornadas laborales, asignadas a transportistas para realizar los pedidos y entregas a distintos puntos de una zona geográfica.

Mientras que estos sistemas proveen soluciones óptimas a problemas de alta dificultad, debido a ser código cerrado tienden a tener limitaciones o formas de pago asociadas a su servicio.

Otra desventaja es no poder extender la funcionalidad de los servicios necesarios para ajustarse a las políticas y necesidades de sus usuarios mas allá de los proveídos por defecto, debido a que las compañías buscan proteger tanto su propiedad intelectual, como las integridad de su código ya sea por razones técnicas o financieras. Esta limitación es entendible ya que sus servicios están optimizados para uso genérico, y así abarcar un mercado más grande.

1.2 Justificación del Proyecto

Al ser una parte crucial del proceso de logística de las empresas, la planificación de rutas se vuelve en un punto de dolor para estas mismas empresas las cuales, en su mayor parte, no están equipadas tecnológicamente para manejar sus recursos efectivamente.

El propósito de este proyecto es crear un servicio base capaz de abstraer las necesidades tecnológicas del flujo logístico de negocios relacionadas a planificación, enrutamiento, asignación, y navegación. El servicio debe ser capaz de extender su funcionalidad base mediante la exposición de webhooks que interactúan con la parte central del sistema (core), pudiendo así modificar el comportamiento interno del mismo sin necesidad de alterar el código.

Al exponer una plataforma con estas especificaciones, aliviamos el punto de dolor al cual se enfrentan comúnmente las empresas, y lo transformamos en una fortaleza que garantiza la correcta ejecución de entregas a sus clientes de manera óptima.

1.3 Soluciones Existentes o Disponibles

Entre las soluciones existentes para los negocios se encuentran:

- Realizar y asignar envíos individuales a los conductores a través de estimación humana. Como ejemplo se puede dar el caso de un operador que conozca la ciudad, el cual pueda proveer una ruta generada únicamente a través de su experiencia y conocimiento empírico.
- Realizar rutas que no tengan grandes grados de dificultad con herramientas y servicios en líneas como Google Maps' Directions Service API o Waze API.
- Realizar rutas de alto contenido y nivel de dificultad a través de Google Maps for Work API.

- Tercerizar el proceso de logística a compañías capacitadas en el manejo de flujo de recursos móviles. Hay actualmente, varias empresas que se encargan de proveer el soporte al flujo de logística tales como: DHL, UPS, Exel plc.

CAPÍTULO 2

2 DESARROLLO TÉCNICO

En este capítulo se analizan las técnicas de enrutamiento de vehículos, los parámetros que se van a medir y las técnicas a utilizar para filtrar la información adecuada en tiempo real.

Se presenta una descripción de las plataformas existentes que proveen servicios similares y se hace una comparación de las mismas. Luego se presentan las capacidades de la plataforma expuesta y se exponen las razones que la convierten en la mejor opción al escoger una herramienta desde el punto de vista logístico.

2.1 Introducción Técnica

El modelamiento de la data es importante para generar rutas óptimas para el usuario. Adicionalmente de proveer una interfaz de control sobre distintos recursos, esta información permite generar una fiel simulación del problema de enrutamiento específico a la situación real. Al dejar de asumir variables como el número de vehículos, utilizamos información verdadera que nos permite obtener soluciones mas útiles.

El análisis se realiza a través de la información mencionada en el párrafo anterior. Se configuran distintos algoritmos heurísticos para obtener una respuesta que consista de la información interna y externa a la compañía para generar múltiples soluciones al iterar el problema varias veces. Una vez llegado a un punto en que la solución no parece mejorar, plateau [7], se escoge la ruta con mejor tiempo y distancia, local maximum [8].

Para poder realizar un análisis correcto de la administración de logística se modelan los elementos relacionados al flujo de trabajo. Luego, se prosigue a analizar la data modelada y generar múltiples soluciones al problema, las cuales son comparados para obtener la ruta óptima.

Mientras que la arquitectura base de la solución es simple, también es sumamente flexible al momento de expandir la funcionalidad del mismo. El producto es liberado con algunos ejemplos de extensiones para incrementar su alcance.

2.2 Análisis Bibliográfico

Para un mejor entendimiento se ha divido el proyecto en cuatro partes principales que son: Sistema de navegación, Sistema de enrutamiento, Importación de data externa y Administración de recursos. Cada unos de estos sistemas tienen un comportamiento y objetivo especifico para generar información propia y/o aumentar data que provenga de otros servicios.

A continuación se describen las diferentes herramientas que se consideraron para cada parte del proceso, las técnicas utilizadas y las razones por las que dichas herramientas fueron seleccionadas como la solución más adecuada para este proyecto.

2.2.1 Sistema de Navegación

Los sistemas de navegación son capaces de generar caminos de un punto geográfico a otro. El uso de almacenamiento de rutas en modo de grafo permite obtener soluciones rápidas, consumiendo los pesos de distancia y tiempo hallados en las aristas [9].

Originalmente se pensó en utilizar Google o Waze debido a su conocida eficiencia y facilidad de uso. Otra ventaja era la información de tránsito que ya poseen debido a sus vastos servidores. Esta decisión duró poco debido, una vez más, a las restricciones que imponen estas plataformas en su modo gratis. A partir de este inconveniente, se decidió en buscar alternativas de navegación.

Finalmente, se decidió integrar la librería de Graphhopper [10]. Esta librería, escrita en Java, permite consumir archivos “osm” [11] para simular rutas viales de múltiples mapas en el mundo. Las ventajas que acompañan es que al ser código abierto podía ser encapsulado en un servidor propio limitado únicamente por el hardware en el que se sitúa. La mayor desventaja en cambio fue la pérdida de data de tráfico.

2.2.2 Sistema de Enrutamiento

Un problema presente en empresas que necesitan realizar envíos, entregas o movimiento de materia prima, productos o paquetes es que al realizar estas rutas no se toman en cuenta muchas variables en el ambiente y se remite a calcular de forma empírica la mejor forma de navegar entre los puntos de tarea.

Para resolver esto se decidió utilizar motores de enrutamiento capaces de generar estas rutas optimas. Un servicio que se considero es el API de

enrutamiento que provee Google, el cual es comprobado de ser muy efectivo. El problema aparece con respeto a los limites que impone esta plataforma hacia el número de puntos en la ruta. Esta limitación fue crucial al momento de descartar la plataforma, ya que para generar una ruta con más de 8 puntos era necesaria una membresía bastante costosa.

Por estas razones se optó por utilizar un proyecto de código abierto llamado jsprit [12]. Jsprit es un set de herramientas en Java para resolver problemas de vendedor viajero con contenido rico (Rich-Content Traveling Salesman Problem) y problemas de enrutamiento de vehículos (Vehicle Routing Problem). Es una plataforma ligera, flexible y fácil de usar, la cual utiliza varios algoritmos con meta heurística para la resolución de problemas de navegación.

2.3 Algoritmos de Enrutamiento

La meta-heurística aplicada para solucionar los problemas de enrutamiento de vehículos fue propuesta y desarrollada por Schrimpf et que formuló el principio de ruin-and-recreate [13]. Es una gran zona de búsqueda que combina elementos de threshold-accepting algorithms [14] y simulated annealing [15].

Esencialmente, funciona de la siguiente manera:

Ruin: A partir de una solución inicial, se desintegra partes de la solución que conduce a un conjunto de trabajos que no son atendidos por ningún vehículo y una solución parcial que contiene todos los trabajos.

Recreate: Sobre la base de la solución parcial todos los trabajos del conjunto de trabajos no asignados son reintegrados una vez más, la cual provee a una nueva solución.

Si la nueva solución proveída por el algoritmo es aceptado como una mejor solución, se establece como la nueva solución parcial. Estos pasos se repiten iterativamente hasta que se cumple un determinado criterio de terminación (por ejemplo, el tiempo de cálculo, número de iteraciones, etc.).

Los autores de la librería ampliaron el algoritmo básico descrito por Schrimpf con las estrategias inspiradas en las publicaciones de Pisinger y Röpke [16].

El sistema de meta heurísticas es el más adecuado para los problemas complejos que tienen muchas limitaciones y un espacio de soluciones discontinuas [17], ya que permite distinguir claramente entre las etapas de ruin-

and-recreate lo cual hace mas sencillo la verificación y validación de restricciones en un problema de rico contenido.

2.4 Metodología de Planeación de Proyecto

Durante la planeación y desarrollo del proyecto, se utilizó la metodología planeación ágil de desarrollo (SCRUM)[18]. Su comportamiento iterativo e incremental permite liberar features nuevos en períodos de tiempo similares permitiendo entregas continuas.

Se utilizó la herramienta en línea Taiga para facilitar el proceso de desarrollo y colaboración de equipos de trabajo en el proyecto. Provee funcionalidad tal como administración de tareas, sprints, e hitos que permite rastrear el progreso del proyecto.

2.5 Comparación con Soluciones Actuales

Cuando se compara la herramienta contra simple percepción humana, el servicio supera, incluso en su forma más básica, en eficiencia y ejecución en todas las categorías evaluadas.

En escenarios relativamente simples, que evalúan la generación de rutas únicamente por la localización de los puntos de interés, APIs de servicios existentes como los de Google, y Waze tienen una distintiva ventaja debido a la robustez de sus herramientas en su plan gratis.

En escenarios más complejos, en los cuales es necesario evaluar características como tiempo, prioridad, dependería del capital disponible. Google Maps for Work es la herramienta más completa entre todas las opciones, teniendo obviamente ventaja sobre el servicio desarrollado; la desventaja es el costo de la subscripción, la cual se encuentra alrededor de $10,000 [19].

Tercerizar el flujo logístico ata al negocio ante un API competente pero a cambio de un pago continuo. Consecuentemente, el proceso queda a merced de cláusulas y regulaciones de la compañía que provee el servicio.

El servicio Carrozza, aunque que no tiene la robustez de servicios más maduros como los proveídos por Google, o Waze, permite flexibilidad y contenido por encima del que provee la mayoría de servicios en sus planes gratis. Al ser código abierto, puede ser adquirido e implementado una sola vez por los negocios y solo sería necesario tercerizar servicios técnicos en caso de mantenimiento.

CAPÍTULO 3

3 EXTENSIÓN DEL SISTEMA BASE

La adquisición de la información de los datos es de vital importancia para el motor de enrutamiento; de la precisión de ésta parte dependerán los resultados obtenidos. Al definir los parámetros y recursos que controla el usuario hay suficiente información para proveer de soluciones en un ambiente ideal. Lastimosamente, muchos factores externos tienen suficiente influencia para transformar soluciones “suficiente buenas” en inútiles.

Información como tráfico, bloqueos, clima, disponibilidad son algunos de los factores que juegan un rol muy importante en poder crear rutas eficientes.

Este capítulo describe el proceso seguido para la adquisición de información en tiempo real. El sistema de adquisición tiene como propósito obtener información que represente el estado de un área geográfica a través del tiempo. De esta forma, se conseguirá simular la carga vial en las zonas de enfoque.

3.1 Sistema de Adquisición

La adquisición de datos en tiempo real se realiza de forma externa al servicio de enrutamiento. A pesar de esto, el servicio expone una interfaz para integrar información de terceros a su base de datos interna. Esto permite que desarrolladores pueden integrar sistemas como data de tráfico de ciudades para alimentar el grado con datos más cercanos a la realidad.

La interfaz expuesta es bastante simple de integrar; cualquier fuente de datos debe integrar un único método en su implementación el cual exige un cierto formato de respuesta para poder ser acoplada con el servicio.

3.2 Proceso de Integración

Una vez establecida la fuente de información en el modelo del servidor se le asigna un periodo de tiempo para refrescar la data dependiente del servicio que se esté utilizando.

Todas las fuentes deben ser iniciadas antes de que el servidor empiece a escuchar peticiones para evitar proporcionar data errada o inconclusa. Cuando la última fuente haya sido cargada, el servidor empezará a aceptar peticiones.

3.3 Código Ejemplo de Integración

Para poder incluir una fuente de data externa es necesario que esta extienda la interfaz RoadDataSource la cual expone un único método llamado fetch, el cual requiere que retorne, no importa la forma, un arreglo de puntos de cambios de velocidad como se muestra en la Figura 3.1:

```
public interface RoadDataSource {
   RoadData fetch() throws Exception;
```

Figura 3.1. Interfaz que debe implementar toda fuente de datos externo para modificar pesos del grafo de calles.

El proyecto base incluye un ejemplo de inserción de una fuente de data de tráfico basada en medidores de velocidad proveídos por el New York Police Department (NYPD). La figura 3.2 muestra una llamada HTTP a un servidor que contiene información que representan los cambios de velocidades en calles de Nueva York, y una vez obtenidos, son transformados a una estructura que reconoce nuestro sistema.

```
public class NewYorkRoadDataSource implements RoadDataSource {
   public RoadData fetch() throws Exception {
      JSONArray arr = new JSONArray(fetchJSONString("***"));
      RoadData data = new RoadData();

      for (int i = 0; i < arr.length(); i++) {
         JSONObject obj = arr.getJSONObject(i);
         double speed = obj.getDouble("speed");
         JSONArray paths = obj.getJSONArray("points");
         List<Point> points = new ArrayList();
         for (int pointIndex = 0; pointIndex < paths.length(); pointIndex++) {
            JSONObject point = paths.getJSONObject(pointIndex);

points.add(Point.builder().latitude(point.getDouble("latitude")).longitude(point.getDouble("longitude")).
build());
         }
         if (!points.isEmpty()) {
            data.add(new RoadDataEntry(speed, points));
         }
      }
      return data;
   }
```

Figura 3.2. Ejemplo de implementación para modificar velocidades de calles en Nueva York utilizando los datos de tráfico del NYPD.

3.4 Evaluación del Sistema

Finalizado el proyecto se notó las fortalezas y debilidades que exhibía al ser evaluado por cuenta propia y en comparación a otras plataformas y planes que proveen servicios similares.

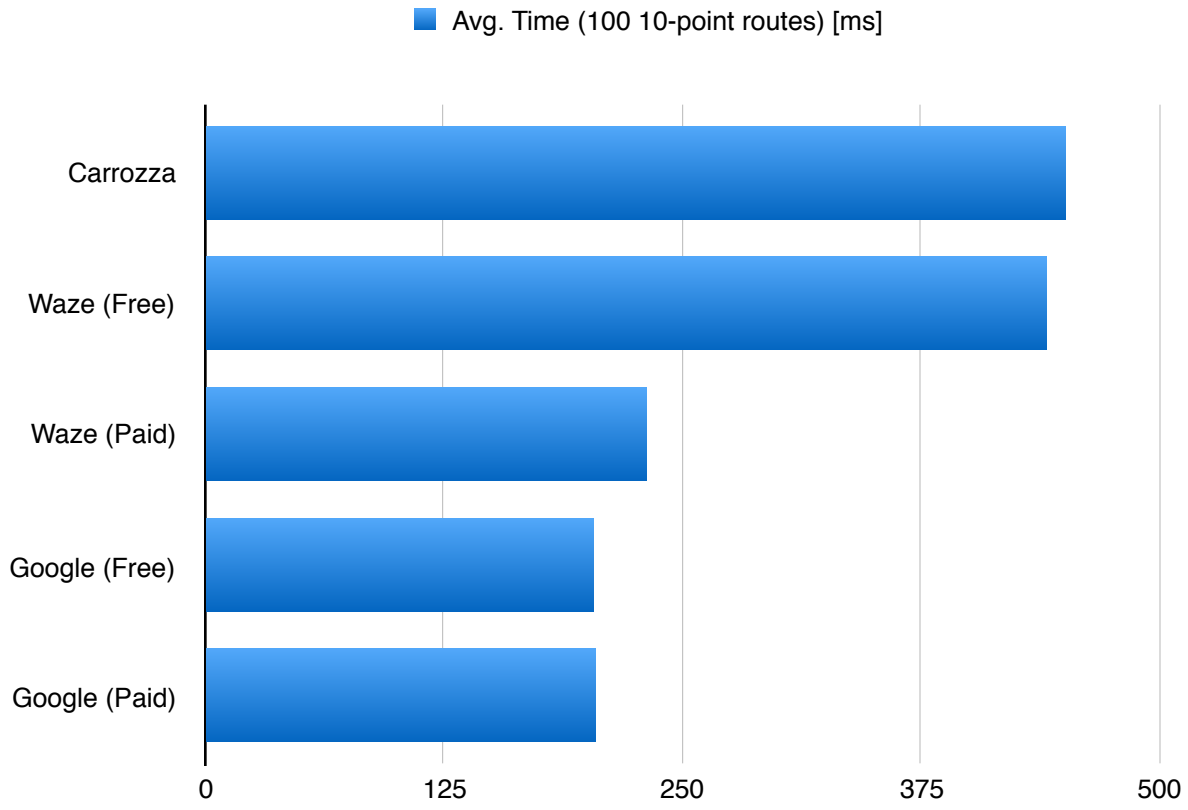

Figura 3.3. Resultados en tiempo de respuesta por parte de los motores de enrutamiento probados. Se basó en el cálculo de 100 escenarios de rutas con 10 puntos de acción cada una.

Algunos de los aspectos en los que fue evaluado fueron:

3.4.1 Velocidad

El servidor Carrozza proveyó rutas a un nivel similar que las de Waze en su plan gratuito, mas Google superó en velocidad de respuesta a ambos servicios como se puede notar en la Figura 3.3.

Es de considerar que servicios como Google y Waze proveen rutas que por defecto incluyen información de múltiples fuentes para llegar a una solución más confiable mientras que Carrozza utiliza una única fuente de datos.

Para realizar estas pruebas fue necesario ignorar la cache del lado del servidor y del cliente para que todos los servicios puedan realizar las pruebas bajo los mismos parámetros.

3.4.2 Solución

Se decidió medir y probar con rutas conocidas para averiguar que tan cercana era la solución que entregaba cada uno de los servicios de enrutamiento que fueron probados. Debido a que otros servicios miden data de tráfico se alteró las pruebas a rutas sin tráfico alguno para que no se vuelva un factor que desequilibre las pruebas.

Al no considerar tráfico, todos los servicios retornaban rutas similares entre ellos por lo cual bajo estas condiciones no había un claro ganador.

CONCLUSIONES Y RECOMENDACIONES

Las funcionalidades que provee Carrozza, las cuales no están presentes en los planes gratis de Google, ni Waze incluyen: la capacidad de realizar dependencias entre puntos de recolección y entrega. Esta ventaja permite la creación de tareas con múltiples puntos de acción lo cual es necesario para simular envíos. Esta característica prueba ser muy importante al momento de usar esta herramienta con propósitos de logística.

La desventaja más grande en términos de funcionalidades proveídas es la carencia de información de tráfico por defecto. Mientras que Google y Waze la proveen en todos sus planes de forma automática, Carrozza requiere que esta sea agregada manualmente.

Para aumentar el valor del proyecto es necesario mejorar la estabilidad y robustez del producto. Actualmente, el servidor espera un requerimiento adecuado sin validaciones ni métodos de autenticación los cuales son necesarios en todo servicio a nivel profesional.

Con respecto a futuras mejoras, se puede situar también la creación de nuevas restricciones que se pueden asignar a envíos para así, cubrir más aspectos que puedan influenciar al proceso de generación y planeación de rutas.

BIBLIOGRAFÍA

[1] Adam Robinson. (2014, April 30). E-Commerce Logistics: The Evolution of Logistics and Supply Chains from Direct to Store Models to E-Commerce [Online]. Available: http://cerasis.com/2014/04/30/e-commerce-logistics/

[2] Stefan Schröder. (2014). Jsprit [Online]. Available: http://jsprit.github.io/

[3] Tim Stenovec. (2015, November 20). The 2 simple reasons why Google Maps is better than everything else [Online] Available: http://www.techinsider.io/the-reason-google-maps-is-the-best-traffic-app-2015-11

[4] Barth, Dave (2009, August 25). "The Bright Side of Sitting in Traffic: Crowdsourcing Road Congestion Data". Google. [Online]. Available: https://googleblog.blogspot.com.br/2009/08/bright-side-of-sitting-in-traffic.html

[5] Google (2016). Google Maps API [Online]. Available: https://developers.google.com/maps/?hl=pt-br

[6] Waze (2016). Waze for Developers [Online]. Available: https://www.waze.com/pt-BR/about/dev

[7] Matt Ginsberg, Essentials of Artificial Intelligence. 1993.

[8] Larson, Ron; Edwards, Bruce H. (2009). Calculus (9th ed.). Brooks/Cole. ISBN 0-547-16702-4.

[9] Hazewinkel, Michiel, ed. (2001), "Graph theory", Encyclopedia of Mathematics, Springer, ISBN 978-1-55608-010-4

[10] Graphhopper (2015). Graphhopper Directions API [Online]. Available: https://graphhopper.com/api/1/docs/

[11] Lardinois, Frederic (2014, August 9). "For the Love of Mapping Data". TechCrunch. [Online]. Available: https://techcrunch.com/2014/08/09/for-the-love-of-open-mapping-data/

[12] Stefan Schröder. (2016). Jsprit [Online]. Available: https://github.com/graphhopper/jsprit

[13] Gerhard Schrimpf. “Record Breaking Optimization Results Using the Ruin and Recreate Principle” (Volume 159, Issue 2). 2000. pp. 139-171

[14] Dueck, Scheuer. “Journal of Computational Physics” (Volume 90, Issue 1). 1990. pp. 161-175

[15] Yuval Baror. (2016). Simulated Annealing visualization A visualization of a simulated annealing solution to the N-Queens puzzle by Yuval Baror [Online]. Available: http://yuval.bar-or.org/index.php?item=9

[16] Álvarez, Munari. (2016, June). Abordagens metaheurísticas para o problema de roteamento de veículos com janelas de tempo e múltiplos entregadores [Online]. Available: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-530X2016000200279

[17] Battiti, Roberto; Mauro Brunato; Franco Mascia (2008). Reactive Search and Intelligent Optimization. Springer Verlag. ISBN 978-0-387-09623-0.

[18] Münch, Jürgen; Armbrust, Ove; Soto, Martín; Kowalczyk, Martin (2012). "Software Process Definition and Management". Retrieved July 16, 2012.

[19] Google (2016). Google Maps for Work [Online]. Available: https://www.google.com/work/mapsearth/products/mapsapi.html#