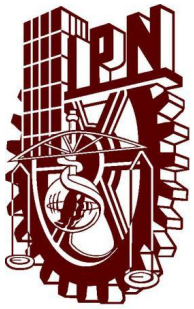

INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

Laboratorio de Procesamiento Inteligente de Información Geo-espacial

Estudio del tránsito vehicular en entornos urbanos basado en análisis espacio-temporal

TESIS

QUE PARA OBTENER EL GRADO DE:

MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A:

Ing. Diego Arturo Nava Chávez

Directores de tesis:
Dr. Miguel Jesús Torres Ruiz
Dr. José Giovanni Guzmán Lugo

Ciudad de México enero 2017

SIP-14 bis

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México siendo las 10:00 horas del día 30 del mes de noviembre de 2016 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"Estudio del tránsito vehicular en entornos urbanos basado en análisis espacio-temporal"

Presentada por el alumno:

NAVA	**CHÁVEZ**	**DIEGO ARTURO**
Apellido paterno	Apellido materno	Nombre(s)

Con registro:

B	1	4	0	6	2	7

aspirante de: **MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de Tesis

Dr. José Giovanni Guzmán Lugo

Dr. Miguel Jesús Torres Ruiz

Dr. Cornelio Yáñez Márquez

Dr. Marco Antonio Moreno Ibarra

Dr. Rolando Quintero Téllez

Dr. Amadeo José Argüelles Cruz

PRESIDENTE DEL COLEGIO DE PROFESORES

Dr. Marco Antonio Ramírez Salinas

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION
EN COMPUTACION
DIRECCION

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México el día 06 del mes diciembre del año 2016, el que suscribe Diego Arturo Nava Chávez alumno del Programa de Maestría en ciencias de la computación con número de registro B140627, adscrito al Centro de investigación en computación, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección de Dr. José Giovanni Guzmán Lugo y Dr. Miguel Jesús Torres Ruiz y cede los derechos del trabajo intitulado Estudio del tránsito vehicular en entornos urbanos basado en análisis espacio-temporal, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección diego.nava77@hotmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Diego Arturo Nava Chávez

Nombre y firma

Resumen

Las aplicaciones móviles de ruteo han tenido una gran demanda en los últimos años, basándose en la gran popularidad que han alcanzado los teléfonos inteligentes (smartphones) y los sensores con los que estos dispositivos cuentan, en especial el sensor GPS. El tratar de determinar la ruta óptima puede llegar a ser un verdadero problema, debido a una gran cantidad de variables que influyen en el comportamiento del tráfico.

Un problema que se puede presentar en las aplicaciones de ruteo consiste en que al momento de que el usuario solicita a la aplicación que genere una ruta entre dos puntos, la aplicación no cuenta con información actualizada que le permita determinar que una determinada vialidad ya está saturada, por lo que la emplea como una opción adecuada para el trayecto del usuario. Sin embargo, al momento de que la persona está transitando por dicha calle, ésta presenta una carga vehicular intensa; por otra parte, lo anterior, es más común cuando la persona se desplaza entre un origen y un destino que tiene una separación considerable, debido a que desde el momento que se calculó la ruta, hasta el momento en que se arriba a dicha zona, pudo haber ocurrido algún evento que complica la vialidad en la zona. Lo anterior, puede tener varias causas, como puede ser un accidente automovilístico, una falla en parte de la infraestructura de un servicio que ofrece la ciudad (drenaje, luz, teléfono) que requiere el cierre de una determinada calle, una manifestación, entre otros. Este tipo de situaciones no son previsibles y obviamente no es posible controlar su ocurrencia y tomar medidas preventivas al momento de estar realizando cálculos de rutas; pero otro tipo de situaciones si se pueden pronosticar y estimar a priori. Por ejemplo, en las grandes urbes del mundo, como la Ciudad de México, existen avenidas y/o calles que continuamente presentan un alto flujo vehicular en tiempos específicos, un caso de ello es Insurgentes Sur en un horario matutino en días laborales, mientras que

Insurgentes Norte presenta la carga vehicular intensa por las tardes.

En este trabajo, se presenta una metodología para poder hacer el estudio del tránsito vehicular en entornos urbanos, considerando la información que se recibe de forma voluntaria colaborativa. Con esta información se realiza un análisis histórico, el cual es utilizado al momento de determinar la ruta entre dos puntos. Con este análisis es posible identificar vías de comunicación donde se presenta carga vehicular intensa en horarios específicos de tiempo.

Abstract

Urban Routing apps have become widely used recently years, due to the acceptation of the smartphones among the population of the big cities and the sensors that come with them specially GPS; when we face the problem of trying to find the most optimized route can be quite a bit hard due to the great quantity of variables that are involved.

Sometimes when we try to get a route; mainly in rush hours or for big distances and when we take the way despite the applications shows there is no traffic when we get to certain avenue we noticed that actually there is, this can happen because many reasons but the time and day are one of the mainly factors indeed, we know based on the time and hour we are driving in we can face traffic in certain avenues, because such avenues usually have traffic at the same hour and day periodically.

In this present work we offer a methodology of how to get a route no only based on the amount of data in real time, we take in consideration de historical data as well, by trying to answer the question what happened? to know what could happen; there are some avenues that we can see a periodically behavior in such avenues we can try to estimate only in case there is not data in real time available, which the traffic conditions will be by a determined time window.

Agradecimientos

A Dios por todo lo que me ha dado y por dejarme creer.

A mis padres que siempre me han apoyado, a mi hermano y hermana por su cariño y compañía. A mis compañeros del laboratorio y a los amigos que conseguí en el CIC; a todos los amigos encontrados en la vida; a mis nuevos amigos de Rusia por cambiar mi percepción del mundo.

A mis asesores de tesis el Dr. José Giovanni Guzmán Lugo y Dr. Miguel Torres por su apoyo incondcional, al Doctor Grigori Sidorov que juntamente con mis asesores me apoyaron con mi estancia en Rusia.

Al Instituto Politécnico Nacional por todo lo que me ha permitido lograr, al Centro de Investigación en Computación y al Consejo Nacional de Ciencia y Tecnología por el apoyo en la realización de este trabajo, al programa PIFI.

Diego Arturo Nava Chavez

¡Gracias!

Índice de figuras

Índice de tablas

Índice general

Capítulo 1

Introducción

1.1. Antecedentes

1.1.1. Ruteo Urbano

Desde hace tiempo se ha buscado resolver el problema de cómo encontrar la ruta óptima entre dos puntos conocidos como punto origen y punto destino. Uno de los algoritmos más famosos es el algoritmo de Dijkstra, el cual se basa en la asignación de pesos a las uniones (aristas) entre los nodos representados en un grafo, con base en estos pesos se encuentra la ruta más corta, un ejemplo de lo anterior se muestra en la figura 1.1.

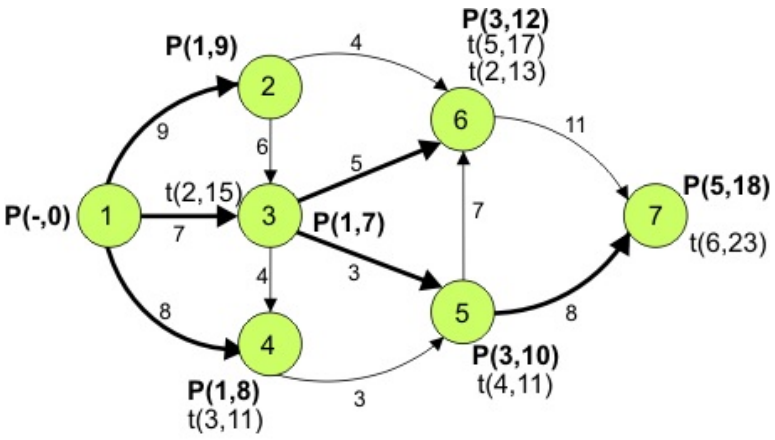

Figura 1.1: Algoritmo de Dijkstra

Dentro del ruteo urbano, la ruta más corta entre dos puntos no necesariamente es la más óptima, debido al tráfico que puede existir en un segmento de la ruta determinada. Puede darse el caso que se llegue más rápido al destino tomando una ruta más larga. Por lo anterior, se han desarrollado una serie de trabajos y aplicaciones que abordan este problema. En la actualidad, una de las aplicaciones más conocidas y populares entre los usuarios de dispositivos móviles es Waze. Esta aplicación se encarga de cuantificar el tráfico en las vialidades, por medio de la recolección de información voluntaria colaborativa proporcionada por los usuarios a través de sus dispositivos móviles. Con estos datos, Waze genera rutas evitando en la medida de lo posible, el tráfico excesivo. Otra aplicación muy conocida es la de Google Maps; esta aplicación (al igual que Waze) permite determinar el tráfico a través de la información proveniende de los dispositivos móviles (que cuenten con GPS) y recopila diversa información, como son la posición y la velocidad de los automóviles donde se desplazan los usuarios, como se muestra en la Figura 1.2.

Figura 1.2: Google Maps

1.2. Planteamiento del problema

Como se mencionó en el párrafo anterior, existen varias aplicaciones que permiten generar la ruta óptima acorde a un determinado criterio especificado por el usuario (usando vías de cuota y/o autopistas, ruta más corta o

más rápida). Gran parte de las aplicaciones, o por lo menos las más comerciales, basan sus algoritmos para encontrar la ruta en minimizar la cantidad de tiempo requerido para poder llegar al punto destino. Esto se realiza, principalmente, midiendo el tráfico en las ciudades, para lo cual utilizan los dispositivos móviles de los usuarios que tengan activa la aplicación, midiendo la posición y la velocidad en diferentes instantes de tiempo. Este mecanismo para medir el tráfico, ha demostrado ser una buena opción, pero tiene el inconveniente de que cuando no hay usuarios activos circulando por una avenida o calle, al momento de que se solicita a la aplicación la ruta para llegar a un destino, el algoritmo desconocerá el estado actual de esa vialidad, y podría seleccionarla como parte de la ruta que ofrecerá al usuario. Por lo anterior, se pretende seguir el mismo enfoque de recopilar información de los usuarios activos de la aplicación a desarrollar. Asimismo, esta información será almacenada en una base de datos espacial, con el objeto de poder realizar un análisis en diferentes ventanas de tiempo, para poder determinar el comportamiento de las vialidades sobre las que se recopilen datos. Con ello, la falta de información en un determinado instante, podrá ser reemplazada por datos históricos de aforo vehicular.

1.3. Objetivos

1.3.1. Objetivo general

Estudiar el tránsito vehicular en entornos urbanos por medio de un algoritmo que permita determinar la ruta entre un punto origen y un punto destino por medio del análisis de información histórica basado en ventas de tiempo de derivación parametrizada.

1.3.2. Objetivos específicos

- Buscar y analizar diferentes algoritmos de ruteo actualmente propuestos.
- Definir las capas de estudio, para recopilar información de áreas específicas, así como la base de datos espacial.
- Diseñar un algoritmo de ruteo parametrizado por ventanas de tiempo.

- Integrar el algoritmo de ruteo parametrizado como parte de una base de datos espacial.
- Diseñar una aplicación para dispositivos móviles que permita implementar el algoritmo propuesto.

1.4. Justificación

Si bien es cierto que existen aplicaciones para obtener un ruta basada en la información obtenida por análisis de datos colaborativo, tales como Waze y Google Maps, ninguna lleva un registro histórico para hacer predicciones o encontrar patrones de tráfico para ciertas avenidas en diversos instantes de tiempo. Por lo anterior, se puede presentar la situación de que al usar alguna de estas aplicaciones en la generación de una ruta entre dos puntos, para una sección en particular de la ruta calculada, no se cuente con datos recientes, por lo cual la aplicación considerará que la vialidad es adecuada y no presenta tráfico, pero cuando el usuario se encuentre en desplazamiento y llegue a esta vialidad, se encuentre con una situación completamente diferente y la vialidad tenga una carga de tráfico excesiva. Si se llevara un registro histórico de la información recopilada para las vías sobre las cuales los usuarios activos se han desplazado, se pueden identificar patrones en ciertas ventanas de tiempo y con ello hacer una predicción mas cercana al comportamiento real de la vialidad aunque no existan datos recientes por parte de los usuarios activos.

1.5. Beneficios esperados

Uno de los principales beneficios del presente trabajo de investigación será poder generar rutas entre dos puntos, considerando tanto la información disponible a corto plazo (de los usuarios activos) como la de mediano plazo (información histórica) obtenida por un análisis histórico basado en ventas de tiempo de derivación parametrizada, con lo cual se tendrá mayor certeza de que, a excepción de un evento inesperado y no predecible como una manifestación, cierre repentino por un choque, accidente, delito, etc.; la ruta generada evitará emplear vialidades con carga vehicular considerable.

1.6. Alcances y limitaciones

Llevar un registro histórico de cada calle o avenida es un gran impacto en el comportamiento del sistema, al momento de hacer la propagación del grafo según el tamaño del mismo y el registro puede llegar a tardar mucho tiempo, lo cual impacta en la frecuencia de actualización, para nuestro caso de estudio esto llega a tardar hasta 30 minutos en promedio, sin embargo, este tiempo podría ser mayor al considerar un grafo de mayores dimensiones.

Por otra parte como se describió un poco en el párrafo anterior, la presencia de un evento inesperado como un accidente vehicular, un delito, explosión, etc., puede requerir el cierre de la vialidad de forma inmediata. Por lo anterior, si no se cuenta con esta información por los datos obtenidos de parte de los usuarios activos, y asimismo, el análisis histórico describe la vialidad con poco aforo vehicular, el algoritmo a implementar podría suponer que es una vialidad adecuada, cuando en ese momento es lo contrario.

1.7. Organización de la tesis

El resto de la tesis se encuentra organizada de la siguiente manera:

- **Capítulo 2** Este capítulo se centra en el estado del arte, está compuesto por trabajos relacionados en un sentido científico y tecnológico, acerca de algoritmos relacionados con el presente trabajo de investigación.
- **Capítulo 3** En este capitulo se describen los conceptos y tecnologías relacionadas con el trabajo de tesis, útiles para la implementación del algoritmo propuesto.
- **Capítulo 4** En este capítulo se describe la metodología que permite hacer un estudio del tránsito vehicular así como también la forma en que se aplicaron las tecnologías descritas en el Capítulo 3.
- **Capitulo 5** Este capítulo presenta los experimentos realizados y los resultados obtenidos, los cuales servirán para un análisis de la metodología propuesta.
- **Capitulo 6** En este capítulo se presentan las principales contribuciones y conclusiones obtenidas en el proceso de investigación desarrollado en este trabajo, así como algunas propuestas relacionadas como trabajo a futuro.

Capítulo 2

Estado del Arte

En este capítulo se analizan diferentes trabajos que están relacionados con el presente trabajo de investigación. El trabajo [10] plantea que muchos de los algoritmos de planeación de rutas están mayormente basados en la longitud física de la ruta; sin embargo, existen otros factores tales como las reglas de tráfico, el número de cruces, o el tráfico en si mismo, etc., que deben ser tomados en consideración al momento de calcular rutas entre dos puntos. En este trabajo, se busca solucionar el problema de la planeación combinando estas reglas junto con el algoritmo de colonia de hormigas.

La forma en que se implementa el algoritmo de la colonia de hormigas es a través de un grafo, en el cual, la conectividad física de una calle puede ser representada por un vértice y un arco; la conectividad lógica es representada por la dirección y el peso de cada arco, como se muestra en la Figura 2.1.

En grafos para aplicaciones de ruteo urbano, los pesos se determinan por diversos factores, tales como la distancia física, calidad de la calle, número de carriles, tráfico actual y velocidad. En esta publicación solo se consideran condiciones restrictivas, como puede ser el número de semáforos, reglas de tráfico, calles cerradas, entre otros.

Existen distintos tipos de caminos tales como autopistas, carreteras, avenidas y calles, dependiendo del tipo de camino; lo cual es un parámetro a elegir[23]. Por ejemplo, una autopista es una buena opción para distancias muy largas, por lo que el peso que se debe asignar a este tipo de vialidad debe ser bajo. Por otro lado, una calle tiene mayor peso ya que la cantidad de automóviles que puede transportar es mucho menor. El trabajo [10] propone

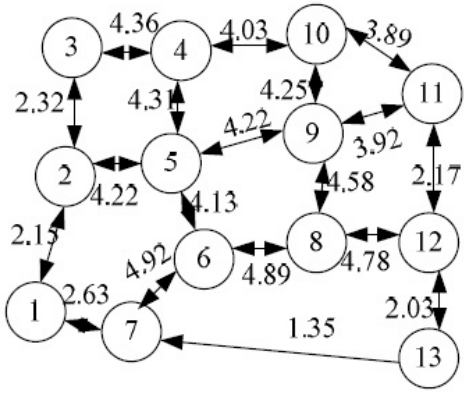

Figura 2.1: Ejemplo de una red de caminos

Tabla 2.1: Rango de pesos

Tipo de camino	Rango de pesos
Autopista	0.01 - 0.1
Carretera	0.1 - 1.0
Avenida	0.5 - 2.0
Calle	0.5 - 4.0

el siguiente rango de pesos mostrados en la Tabla 2.1.

Una vez obtenido el grafo, se ejecuta el algoritmo de la colonia de hormigas, el cual toma usa el sentido de las calles así como el tiempo de retraso ocasionado por los semáforos. Al ejecutar el algoritmo, no solo se busca la ruta más corta en distancia, sino también, el número de cruces; ya que entre más cruces se puede interpretar como un mayor tiempo de retraso, en la Figura 2.2 y en la Figura 2.3 se muestras los resultados sin y con estas reglas.

Esta publicación donde se hace el cálculo de la ruta más corta, es una buena aplicación para el algoritmo de la colonia de hormigas; sin embargo el trabajo no involucra en el algoritmo la velocidad que se puede alcanzar durante el trayecto y en consecuencia estimar el tiempo que requiere desplazarse a través de una vía.

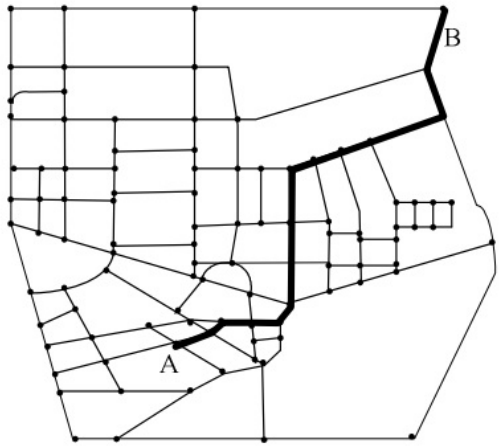

Figura 2.2: Test sin tomar en consideración reglas de tráfico

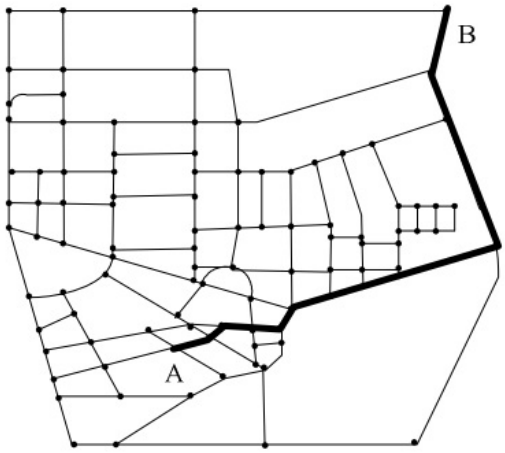

Figura 2.3: Test al tomar en consideración reglas de tráfico

En el trabajo de [19] se habla sobre una optimización del método SNN (Ren Neural de Impulsos o de sus siglas en inglés Spiking Neural Networks)[15] para tratar de resolver el problema de la ruta más corta. La eficiencia de una SNN es independiente del número de conexiones en el grafo. Uno de los principales algoritmos para resolver el problema de la ruta más corta es el algoritmo de Dijkstra[2]; este algoritmo es útil para grafos con un número limitado de nodos, pero para grafos con una alta cantidad de nodos requiere de un alto de tiempo de ejecución, lo cual limita sus aplicaciones[24].

Biológicamente, la comunicación entre dos neuronas se da a través de pulsos. Cuando una neurona dispara, el pulso es transmitido a través de la Sinapsis; después de un tiempo de espera, esta célula disparará. Ignorando el tiempo de transición a través del Axón y si solo consideramos el tiempo que las células les toma en disparar que es justamente el tiempo que al pulso le toma transmitirse a través de la sinapsis[19].

El modelo que propone el trabajo [19], se basa en el tiempo que tarda una célula en responder. Este tiempo de respuesta se puede calcular de acuerdo al peso de su arista: si una neurona n manda un pulso a una neurona m, la neurona m tardará un tiempo determinado en responder, y si se toma en cuenta que una neurona puede mandar un pulso a diferentes neuronas, la neurona con el tiempo de respuesta menor, definirá la ruta del grafo; ya que en este caso el peso será definido por el tiempo de respuesta.

Para poder aplicar el modelo SNN, el peso del grafo topológico tiene que ser cambiado por el tiempo de transmisión. El algoritmo se explica con el grafo de la Figura 2.4, donde cada neurona es representada en el grafo por medio de un nodo. Cuando todas las neuronas hayan disparado exitosamente, el algoritmo genera un árbol, el cual contendrá la ruta más corta.

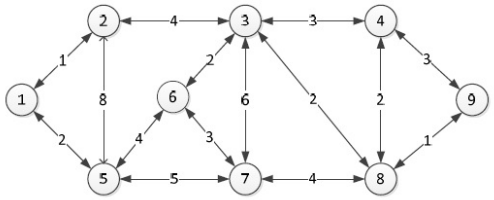

Figura 2.4: Grafo de demostración

Cuando la neurona 1 es activada esta envía un pulso a todas sus vecinas, cada neurona tendrá un tiempo de respuesta dependiendo del peso asignado a la arista, una vez que una neurona haya disparado quedará debilitada; en analogía con el algoritmo de Dijkstra, pasará a la lista de nodos visitados. Conforme las neuronas vayan disparando se irá formando un árbol, en el cual la rama que tenga como nodo final (hoja) será la ruta más corta.

Para ejemplificar esto se describirá una ejecución del algoritmo, ver Figura 2.4.

- En T=0, la neurona 1 dispara, entonces el tiempo de respuesta de la neurona 2 y 5 empieza a disminuir.
- En T=1, la neurona 2 responde, entonces la neurona 3 y la neurona 5 empiezan a disminuir, la segunda petición hacia la neurona 5 se procesa en paralelo con la petición anterior.
- En T=2, la neurona 5 alcanza su tiempo de respuesta proveniente de la neurona 2 así que se ignora la segunda petición, esta neurona manda un pulso a las neuronas 6 y 7.
- En T=5, la neurona 3 alcanza su tiempo de respuesta, entonces las neuronas 6, 7, 8 y 4 empiezan a disminuir.
- En T=6, la neurona 6 alcanza su tiempo de retraso, por lo tanto manda un pulso a la neuronas 7 y 3.
- En T=7, las neuronas 7 y 8 alcanzan su tiempo de espera, por lo tanto, la neurona 8 manda un pulso a la neurona 4 y 9, para la neurona 7 todos sus vecinos ya han disparado, así que la petición se ignora.

- En T=8, la neurona 9 alcanza su tiempo de espera, y como es el nodo destino, el algoritmo termina.

La ejecución del agoritmo generado como resultado, lo mostrado en la Figura 2.5

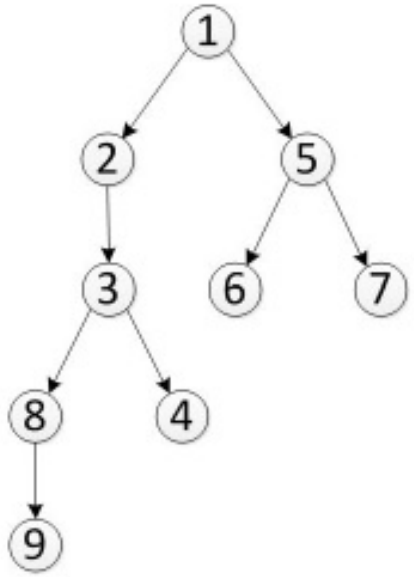

Figura 2.5: Árbol obtenido

Este algoritmo tiene la ventaja de que el tiempo de ejecución no está relacionado con el número de conexiones, sino más bien depende de la distancia en pesos, de la fuente hacia el destino. Es un algoritmo que no depende si un nodo ya fue visitado por todos su vecinos, ya que se basa en el tiempo de respuesta de cada nodo. Lo anterior quiere decir que si un nodo n es disparado por un nodo m antes que un nodo k, significa que la ruta más corta se encuentra en la arista entre el nodo n y m; lo cual da la posibilidad de anular cualquier otra petición ahorrando tiempo de ejecución. Tiene la desventaja de que los pesos tienen que ser enteros, esto es debido a que el peso representa tiempo o una iteración. Si se tienen pesos no enteros, la ejecución del algoritmo se puede incrementar en orden de 10.

En el trabajo [4] se propone utilizar un método difuso para grafos, donde los costos no son conocidos de forma precisa y se modelan con numero difusos, para posteriormente utilizar el método de Monte Carlo con el propósito de obtener la solución a este problema.

Se considera una red unida por arcos, donde los arcos pueden ser bidireccionales y se asigna un peso a todos los arcos de acuerdo a un algoritmo evolutivo [5, 12, 13]. Finalizado lo anterior, se aplica el método difuso de Monte Carlo para obtener la solución del problema, el cual escoge de forma aleatoria números, para así determinar la ruta más corta. Este algoritmo no es útil para situaciones en las cuales no conocemos con mucha precisión el peso de las aristas, esto podría aplicarse para ruteo urbano. En lo que respecta al ruteo urbano, existen muchos factores los cuales podrían dar una mejor aproximación para el peso de la arista.

En el trabajo [20] se abordan cuestiones de análisis de sensibilidad en relación con el problema del camino más corto y el problema de la capacidad máxima de la ruta en una red no dirigida. Para ambos problemas, se determinan los pesos máximos y mínimos que cada borde puede tener para que una trayectoria determinada siga siendo óptima. Para ambos problemas, muestra cómo determinar estos valores máximos y mínimos para todos los bordes en O (M + K log K), donde M es el número de aristas en la red, y K es el número de aristas en la propuesta de camino óptimo.

En el trabajo [25] se propone establecer la ruta óptima más corta en un grafo no dirigido G = (V, E), en la que algunos vehículos tienen que ir de un nodo de origen *s* a un nodo de destino *t*. Sin embargo, la mayoría de los ejes *k* en el grafo pueden ser bloqueadas durante el viaje. El objetivo es encontrar un mínimo conjunto de rutas para los vehículos antes de que comiencen para asegurar la llegada más rápida de al menos un vehículo, sin importar que l(0<l<k) bordes estén bloqueados. Se consideran dos escenarios para este problema: en el primer escenario, con *k = 1*, se propone el concepto de ruta de reemplazo frecuente y se diseña el algoritmo de menor superposición para encontrar el camino de reemplazo común. Con base en esto, se presenta un algoritmo para calcular el conjunto de rutas más cortas óptimas y también se prueba que su complejidad en tiempo es O (n$\hat{2}$). En el segundo escenario con *k > 1*, se considera el caso en el que los bordes bloqueados son consecutivos, en una ruta más corta desde *s* a *t* y los vértices que conectan los dos bordes

también están bloqueados (es decir, las rutas que pasan a través de estos vértices no están permitidas), y el algoritmo que se presenta para calcular la ruta óptima más corta para este escenario tiene complejidad O(mn+k$\hat{2}$n$\hat{2}$logn).

En el trabajo [17] las aplicaciones basadas en Sistema de Posicionamiento Global (GPS), han demostrado ser de gran utilidad en diversas áreas de investigación, en sistemas de ruteo, notifican a los usuarios las instrucciones que deben seguir en sus recorridos; no obstante, frecuentemente no incluyen elementos de apoyo, como puntos de referencia, que pudieran auxiliar a los usuarios durante los desplazamientos.

Planteemos un ejemplo, una persona desea ir de Plaza Torres al Centro de Investigación en Computación. Al solicitar la ruta a su navegador GPS, este último le dará las siguientes instrucciones: "camine a la izquierda cinco metros, camine a su derecha veinte metros en línea recta, gire a la derecha, camine 30 metros en línea recta, gire a la derecha, camine 100 metros en línea recta". Debido a la necesidad de calcular de manera aproximada la cantidad de metros a recorrer y la orientación que debe tomarse, es probable que para algunos usuarios las instrucciones obtenidas por el navegador no sean del todo satisfactorias, dado que es difícil para una persona tener una percepción real de la distancia, cuando se encuentra desplazándose. Para resolver esta problemática, se propone la Metodología para la Generación de Rutas basadas en Puntos de referencia (GRP) la cual, por medio de una ontología de aplicación encargada de hacer una descripción de lugares, como son: tiendas, escuelas, hospitales, entre otros; muestra a los usuarios en una aplicación móvil, los puntos de interés sobre la ruta diseñada con base en las coordenadas de la posición del dispositivo móvil donde se realiza la consulta y las coordenadas del destino al que se desea llegar . El trabajo combina diferentes herramientas de análisis geoespacial, tecnologías web y móviles.

La metodología del trabajo se divide principalmente en tres etapas: Adquisición, Ruteo y Sincronización. La Adquisición, consiste en la definición del origen y del destino, lo cual se usa en la etapa de Ruteo para generar la ruta, y por último, la etapa de Sincronización, en la que se hace uso del modelo ontológico, que mediante una consulta permite localizar los negocios ubicados en las calles que componen el recorrido estudiado.

Como caso de estudio, se analiza un fragmento de la Delegación Gustavo

A. Madero, ubicada en la Ciudad de México. El sistema propuesto, genera una ruta descriptiva que muestra los negocios cercanos a los diferentes tramos que la componen, obtenidos a través de la búsqueda de los nombres de las vialidades en la ontología de Sistema de Vialidades. La ontología proporciona el tipo de vialidad por la que se va a circular, su nombre, y los negocios que posee.

En el trabajo [3] se presenta una metodología para tecnologías móviles la cual tiene como propósito la detección de tránsito vehicular, que sea de fácil uso y que le ayude al usuario a tomar una decisión evitando congestionamientos viales; utilizando información histórica o en tiempo real según sea su necesidad. Este sistema tiene una forma de trabajar inspirada en el monitoreo colaborativo y de masas, ya que toda la información es proporcionada por cada usuario que utilice el sistema.

La metodología que se propone se divide en tres módulos: el primero tiene como nombre "Recolección de datos" el cual es a través de un monitoreo móvil de multitudes, esto significa que los usuarios comparten voluntariamente lecturas de ubicación y velocidad de forma automática en función del tránsito en las zonas en las que se trasladan, los datos que se obtienen son los siguientes: ubicación, la fecha, la hora, el IMEI de cada móvil y es almacenado directamente en una base de datos espacial, a través del protocolo de comunicación.

El segundo módulo tiene como nombre "Análisis de datos", se lleva a cabo en dos órdenes diferentes de tiempo: a corto y a largo plazo. El análisis a corto plazo se utiliza cuando tenemos lecturas de velocidad en tiempo real de algunas calles; el análisis a largo plazo se utiliza para determinar las tasas históricas que presentan las vialidades tomando aspectos de granularidad temporal, es decir se pueden hacer consultas por tiempo, horario y temporadas. En este mismo módulo se obtiene la velocidad de cada punto geográfico y después es clasificado en alguna de las siguientes clases: vialidad fluida, vialidad densa y congestionamiento vial.

El tercer módulo tiene como nombre "Resultados" en donde se realiza la visualización del comportamiento vial representada en un mapa utilizando líneas de colores diferentes para la representación del estado de las vialidades.

Finalmente con la unión de los tres módulos da como resultado un sistema colaborativo para la detección de tránsito vehicular.

Capítulo 3

Marco Teórico

En este capítulo se describirá, de manera general, la metodología propuesta, y de manera exhaustiva las etapas que la componen a la misma, describiendo cuales técnicas, estándares y protocolos fueron implementados en cada etapa.

3.1. Sistemas Operativos Móviles

Hoy en día existen diferentes Sistemas Operativos destinados para el uso móvil de los cuales los que a través del tiempo han destacado han sido: IOS, Android, Windows Phone Figura 3.1; sin embargo el ultimo recientemente ha tenido una caída de popularidad. El uso de dichos sistemas se ha vuelto parte de la vida diaria debido a la cantidad de funcionalidades que poseen y a esto se le agregan las cientos de miles de aplicaciones que se pueden instalar, esta cualidad le otorga un aumento de usabilidad el cual nos da la capacidad de realizar un gran numero de tareas que anteriormente eran muy costosas o debido a su naturaleza prácticamente imposibles de implementar.

La cantidad de problemas que podemos resolver ahora con las tecnologías móviles son muchas, pero particularmente nos han dado la capacidad de resolver problemas relacionados con el análisis de datos, ya que dichos dispositivos cuentan con una gran cantidad de sensores tales como: acelerómetro, bluetooth, GPS, etc. Y debido a la cantidad de dispositivos con los que ahora se cuentan, es posible hacer análisis de distintos tipos sin la necesidad de salir a buscar la materia prima (Datos) haciendo trabajo de campo, gracias a esto se puede recolectar una gran cantidad de datos sin afectar al usuario ya que

Figura 3.1: Sistemas Operativos Móviles

la cantidad de datos que se extraen de su teléfono es mínima, lo que vuelve grande a este tipo de proyectos es la cantidad de usuarios aportándolos.

3.1.1. Android

Android es un sistema operativo inicialmente pensado para teléfonos móviles, al igual que iOS, Symbian y Blackberry OS. Lo que lo hace diferente es que está basado en Linux, un núcleo de sistema operativo libre, gratuito y multiplataforma.

El sistema permite programar aplicaciones en una variación de Java llamada Dalvik. El sistema operativo proporciona todas las interfaces necesarias para desarrollar aplicaciones que accedan a las funciones del teléfono (como el GPS, las llamadas, la agenda, etc.) de una forma muy sencilla en un lenguaje de programación muy conocido como es Java.

Android era un sistema operativo para móviles prácticamente desconocido hasta que en 2005 Google lo compró. Hasta noviembre de 2007 sólo hubo rumores, pero en esa fecha se lanzó la Open Handset Alliance, que agrupaba a muchos fabricantes de teléfonos móviles, chipsets y Google y se proporcionó la primera versión de Android, junto con el SDK para que los programadores empezaran a crear sus aplicaciones para este sistema.

Aunque los inicios fueran un poco lentos, debido a que se lanzó antes el

Figura 3.2: Android

sistema operativo que el primer móvil, rápidamente se ha colocado como el sistema operativo de móviles más vendido del mundo, situación que se alcanzó en el último trimestre de 2010.

En febrero de 2011 se anunció la versión 3.0 de Android, llamada con nombre en clave Honeycomb, que está optimizado para tabletas en lugar de teléfonos móviles. Por tanto Android ha transcendido los teléfonos móviles para trascender a dispositivos más grandes[1].

3.1.2. IOS

Es un sistema operativo móvil desarrollado por Apple Inc. Inicialmente fue creado para el iPhone, pero con el tiempo fue adaptado para los demás dispositivos móviles de esta compañía (iPad y el iPod touch).

Este sistema operativo móvil está basado en el concepto de manipulación directa. Es decir, que el usuario puede interactuar directamente con la pantalla del dispositivo por medio de gestos multitáctiles como toques y deslices[14].

Figura 3.3: IOS

3.1.3. Windows Phone

Windows Phone es un sistema operativo móvil desarrollado por la empresa Microsoft para teléfonos inteligentes y otros dispositivos móviles. Fue lanzado al mercado el 21 de octubre de 2010 en Europa y el 8 de Noviembre en Estados Unidos, con la finalidad de suplantar el conocido Windows Mobile.

Figura 3.4: Windows Phone

Microsoft decidió realizar un cambio completo en este nuevo sistema ope-

rativo con respecto al otro, no solo se cambio el nombre, sino que se desarrollo desde cero, presentando una interfaz completamente nueva, mejor comportamiento y un mayor control sobre las plataformas de hardware que lo ejecutan, todo con el propósito de volver a ser competitivo en el mundo de los móviles.

La primera generación de Windows Phone es Windows Phone 7 Series conocido también como Windows Phone 7, dicho número fue tomado debido a que su antecesor en el mercado era Windows Mobile 6.5. Cabe señalar que el Windows Phone presenta incompatibilidad con los Windows Mobile anteriores, los usuarios no serán capaces de actualizar el Windows en su teléfono y por ende deberán comprar uno nuevo con el reciente sistema operativo[8].

3.2. Bases de Datos Espaciales

Una base de datos espacial es, en primer lugar, una base de datos. Dicho de otro modo, una base de datos espacial es capaz de modelar, almacenar y consultar tanto datos estándar no espaciales (o alfanuméricos) como datos espaciales. En la práctica, los primeros siempre están conectados con los segundos, por lo que una base de datos que manejara solamente información espacial específica sería insuficiente para hacer un modelaje correcto. Hoy en día existen muchos motores de bases de datos pero no todas ellas cuentan con el soporte para trabajar con componentes espaciales, los motores que se han vuelto mas populares son MySql y PostgreSQL, las dos son capaces de manejar la componente geoespacial a través de una extensión[7].

3.2.1. MySql

MySQL es un sistema de administración de bases de datos (Database Management System, DBMS) para bases de datos relacionales. Así, MySQL no es más que una aplicación que permite gestionar archivos llamados de bases de datos.

Existen muchos tipos de bases de datos, desde un simple archivo hasta sistemas relacionales orientados a objetos. MySQL, como base de datos relacional, utiliza multiples tablas para almacenar y organizar la información.

Figura 3.5: MySql

MySQL fue escrito en C y C++ y destaca por su gran adaptación a diferentes entornos de desarrollo, permitiendo su interactuación con los lenguajes de programación más utilizados como PHP, Perl y Java y su integración en distintos sistemas operativos.

También es muy destacable, la condición de open source de MySQL, que hace que su utilización sea gratuita e incluso se pueda modificar con total libertad, pudiendo descargar su código fuente. Esto ha favorecido muy positivamente en su desarrollo y continuas actualizaciones, para hacer de MySQL una de las herramientas más utilizadas por los programadores orientados a Internet.

MySQL implementa extensiones espaciales siguiendo la especificación del Consorcio Open GIS (OGC), un consorcio internacional de más de 250 compañías, agencias y universidades que participan en el desarrollo de soluciones conceptuales públicamente disponibles y que pueden ser útiles para todo tipo de aplicaciones que manejan datos espaciales[22].

3.2.2. PostgreSQL

PostgreSQL es una de las opciones más interesantes en bases de datos relacionales open-source. Michael Stonebraker inició el proyecto bajo el nom-

bre Post Ingres a mediados de los 80's con la idea de solucionar problemas existentes en las bases de datos en esa época.

Figura 3.6: PostgreSQL

MySQL fue por mucho tiempo el motor más popular; pero hoy es propiedad de Oracle y esto limita su evolución. Por otro lado, PostgreSQL es gratuito y libre, además de que hoy nos ofrece una gran cantidad de opciones avanzadas. De hecho, es considerado el motor de base de datos más avanzado en la actualidad.

Una característica interesante de PostgreSQL es el control de concurrencias multiversión; o MVCC por sus siglas en inglés. Este método agrega una imagen del estado de la base de datos a cada transacción. Esto nos permite hacer transacciones eventualmente consistentes, ofreciéndonos grandes ventajas en el rendimiento. Por ejemplo, no se requiere usar bloqueos de lectura al realizar una transacción lo que nos brinda una mayor escalabilidad.

Así mismo también cuenta con la capacidad de agregar componentes espaciales a través de su extensión espacial PostGIS. Debido a que está construido sobre PostgreSQL, PostGIS hereda automáticamente sus características, así

como los estándares abiertos. Actualmente es la base de datos espacial de código abierto más ampliamente utilizada. [18].

3.3. Grafos

Un grafo puede tener muchos significados dependiendo el campo pero para las ciencias de la computación y la matemática, un grafo es una representación gráfica de diversos puntos que se conocen como nodos o vértices, los cuales se encuentran unidos a través de líneas que reciben el nombre de aristas. Al analizar los grafos, los expertos logran conocer cómo se desarrollan las relaciones recíprocas entre aquellas unidades que mantienen algún tipo de interacción.

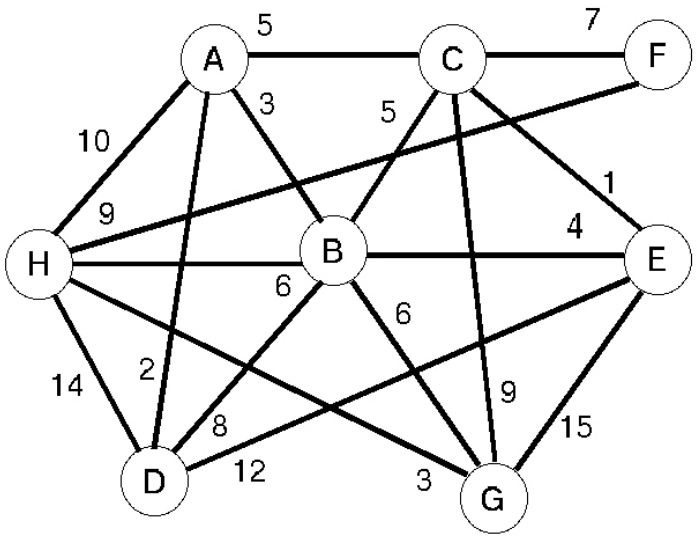

Figura 3.7: Representación de un grafo

En este sentido no podemos pasar por alto el hecho de que el primer documento escrito que tenemos acerca de lo que son los grafos fue realizado en el siglo XVIII, y más concretamente en el año 1736, por Leonhard Euler. Este fue un matemático y físico, de origen suizo, que destacó por ser una de

las figuras más importantes de su tiempo en la citada materia.

En concreto dicho autor realizó un artículo basándose en los puentes que existen en la ciudad de Kaliningrado. A partir de ellos, y mediante lo que es la teoría de los grafos, desarrolló una exposición acerca de los grafos y los vértices que se sustenta en el hecho de que es imposible regresar al vértice que ejerce como punto de partida sin antes no pasar por alguna de las aristas en dos ocasiones.

Los grafos pueden ser clasificarse de diversas maneras según sus características. Los grafos simples, en este sentido, son aquellos que surgen cuando una única arista logra unir dos vértices. Los grafos complejos, en cambio, presentan más de una arista en unión con los vértices[6].

3.3.1. Algoritmo de Dijkstra

Dado un grafo con etiquetas no negativas, se trata de calcular el coste del camino mínimo desde un vértice dado al resto (ing., single-source shortest paths). La utilidad de un procedimiento que solucione esta cuestión es clara: el caso más habitual es disponer de un grafo que represente una distribución geográfica, donde las aristas den el coste (en precio, en distancia o similares) de la conexión entre dos lugares y sea necesario averiguar el camino más corto para llegar a un punto partiendo de otro (es decir, determinar la secuencia de aristas para llegar a un nodo a partir del otro con un coste mínimo). La solución más eficiente a este problema es el denominado algoritmo de Dijkstra, en honor a su creador, E.W. Dijkstra. Formulado en 1959 en .A note on two problems in conexión with graphs", Numerical Mathematica, 1, pp. 269-271, sobre grafos dirigidos, el algoritmo de Dijkstra es un algoritmo voraz (algoritmo goloso) que genera uno a uno los caminos de un nodo a al resto por orden creciente de longitud; usa un conjunto S de vértices donde, a cada paso del algoritmo, se guardan los nodos para los que ya se sabe el camino mínimo y devuelve un vector indexado por vértices, de modo que para cada uno de estos vértices podemos determinar el coste de un camino más económico (de peso mínimo) de a a tales vértices. Cada vez que se incorpora un nodo a la solución, se comprueba si los caminos todavía no definitivos se pueden acortar pasando por el[21].

3.3.2. Algoritmo de búsqueda A estrella

El algoritmo de búsqueda A*(pronunciado "A asterisco" o "A estrella") se clasifica dentro de los algoritmos de búsqueda en grafos. Presentado por primera vez en 1968 por Peter E. Hart, Nils J. Nilsson y Bertram Raphael, el algoritmo A* encuentra, siempre y cuando se cumplan unas determinadas condiciones, el camino de menor coste entre un nodo origen y uno objetivo. Él método busca el camino en un grafo de un vértice inicial hasta un vértice final. Su aplicación va desde aplicativos para encontrar rutas de desplazamiento entre localidades la resolución de problemas, como la resolución de uno quiebra-cabezas, es muy usado en juegos[9].

3.4. GPS

El sistema de posicionamiento global (GPS) es un sistema que permite determinar en toda la Tierra la posición de un objeto (una persona, un vehículo) con una precisión de hasta centímetros (si se utiliza GPS diferencial), aunque lo habitual son unos pocos metros de precisión. El sistema fue desarrollado, instalado y empleado por el Departamento de Defensa de los Estados Unidos. Para determinar las posiciones en el globo, el sistema GPS se sirve de 24 satélites y utiliza la trilateración.

El GPS funciona mediante una red de 24 satélites en órbita sobre el planeta Tierra, a 20 200 km de altura, con trayectorias sincronizadas para cubrir toda la superficie de la Tierra. Cuando se desea determinar la posición, el receptor que se utiliza para ello localiza automáticamente como mínimo tres satélites de la red, de los que recibe unas señales indicando la identificación y la hora del reloj de cada uno de ellos. Con base en estas señales, el aparato sincroniza el reloj del GPS y calcula el tiempo que tardan en llegar las señales al equipo, y de tal modo mide la distancia al satélite mediante el método de trilateración inversa, el cual se basa en determinar la distancia de cada satélite al punto de medición. Conocidas las distancias, se determina fácilmente la propia posición relativa respecto a los satélites. Conociendo además las coordenadas o posición de cada uno de ellos por la señal que emiten, se obtiene la posición absoluta o coordenadas reales del punto de medición. También se consigue una exactitud extrema en el reloj del GPS, similar a la

de los relojes atómicos que lleva a bordo cada uno de los satélites[11].

3.5. Servidor

3.5.1. Linux

Linux es un sistema operativo, una gran pieza de software que controla un computador. Es parecido a Microsoft Windows, pero completamente libre. El nombre correcto es GNU/Linux pero "Linux"se usa más.

Linux no es el producto de una sola compañía, es el resultado de la contribución de un gran número de compañías y grupos de personas. De hecho, el sistema GNU/Linux es un componente central, el cual se transforma en muchos productos diferentes: las llamadas distribuciones.

Las distribuciones cambian la apariencia y funcionamiento de Linux completamente. Las hay desde grandes sistemas completos totalmente equipados (respaldadas por compañías) hasta las más ligeras que entran en un llavero USB o funcionan en computadores viejos (usualmente desarrolladas por voluntarios).

GNU/Linux no es más difícil de usar que Windows, y tiene muchas más funcionalidades. Uno se tarda sólo unos minutos en familiarizarse con una distribución como Ubuntu o Fedora, la cual viene con muchos programas instalados.

Si necesita software de calidad comercial para trabajar con documentos de negocios, Internet, conexión de redes, o trabajar con gráficos, está listo para que lo use. ¿Quiere aún más? Linux puede hacerlo: Existen muchos miles de programas que puede encontrar, instalar y desinstalar de una forma intuitiva y sencilla.

Sin embargo, no deberías asumir que Linux es un clon de Windows. Para saber qué le espera al adentrarse en Linux, le sugerimos leer nuestra página relacionada con migrar a Linux.

3.5.2. PHP

"El PHP es un lenguaje de script incrustado dentro del HTML. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características especificas de si mismo. La meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de paginas".

Con PHP se puede hacer cualquier cosa que podemos realizar con un script CGI, como el procesamiento de información en formularios, foros de discusión, manipulación de cookies y páginas dinámicas. Un sitio con páginas dinámicas es el que permite interactuar con el visitante, de modo que cada usuario que visita la pagina vea la información modificada para requisitos articulares. Las aplicaciones dinámicas para el Web son frecuentes en los sitios comerciales (e-commerce), donde el contenido visualizado se genera de la información alcanzada en una base de datos u otra fuente externa.

Una de sus características más potentes es su suporte para gran cantidad de bases de datos. Entre su soporte pueden mencionarse InterBase, mSQL, MySQL, Oracle, Informix, PosgreSQL, entre otras. PHP también ofrece la integración con las varias bibliotecas externas, que permiten que el desarrollador haga casi cualquier cosa desde generar documentos en pdf hasta analizar código XML.

Como producto de código abierto, PHP goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y se reparan rápidamente. El código se pone al día continuamente con mejoras y extensiones de lenguaje para ampliar las capacidades de PHP. Es utilizado en aplicaciones Web-relacionadas por algunas de las organizaciones más prominentes tales como Mitsubishi, Redhat, Der Spiegel, MP3-Lycos, Ericsson y NASA.

PHP es la opción natural para los programadores en máquinas con Linux que ejecutan servidores web con Apache, pero funciona igualmente bien en cualquier otra plataforma de UNIX o de Windows, con el software de Netscape o del web server de Microsoft. PHP también utiliza las sesiones de HTTP, conectividad de Java, expresiones regulares, LDAP, SNMP, IMAP, protocolos de COM (bajo Windows).

Para trabajar con capacidades PHP, se puede conseguir mayor información en PHP.net, sitio encargado de mantener al día a todos los desarrolladores con las últimas descargas relacionadas con el lenguaje y documentación.

3.5.3. AJAX

En esencia, AJAX permite que una página web que ya ha sido cargada solicite nueva información al servidor. Dicho así, no supondría en realidad ningún invento novedoso. Una página web que contiene un enlace permite que se solicite al servidor nueva información cada vez que se pincha dicho enlace. Una página web que contiene un formulario envía información al servidor y recibe de él nueva información, normalmente la respuesta ante los datos que se han enviado. En ambos casos hay una conexión entre el cliente y el servidor.

¿Cuál es la diferencia cuando usamos AJAX? La diferencia es que con AJAX no es necesario recargar toda la página web, como ocurre cuando pinchamos en un enlace o cuando pulsamos el botón submit de un formulario. Con AJAX es posible realizar una conexión a un servidor desde dentro de una página web usando un programa Javascript. Dicho servidor enviará una respuesta; esta respuesta se almacenará en una variable del programa Javascript y, una vez almacenada en la variable, podremos hacer con ella lo que deseemos.

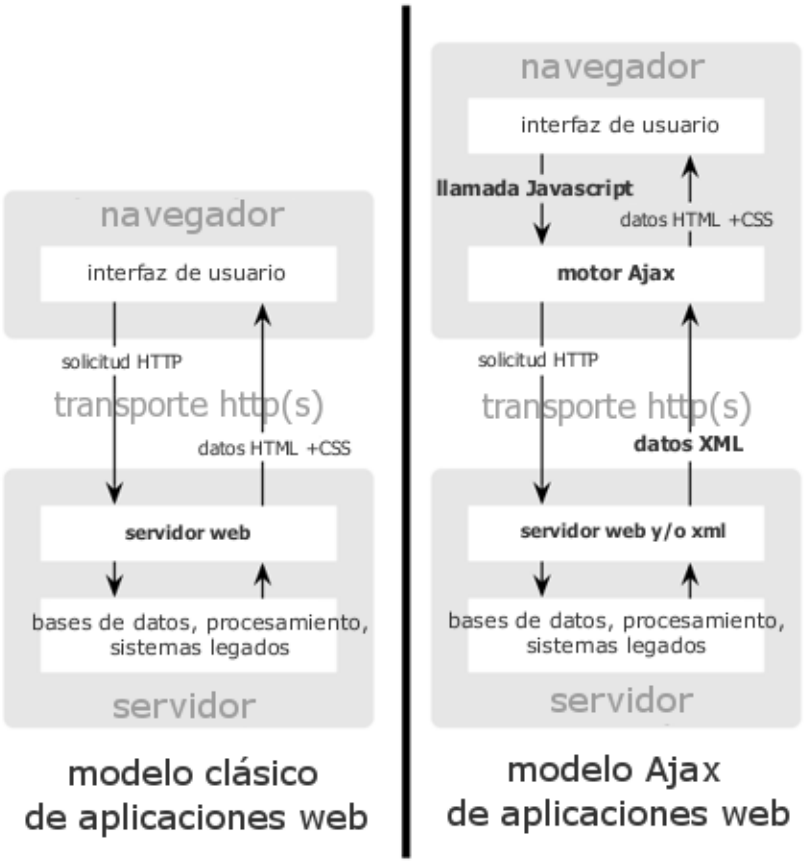

Figura 3.8: AJAX

Capítulo 4

Metodología

La metodología propuesta para la generación de rutas basadas en el registro de trafico según ventanas de tiempos, en el cual el usuario obtendrá una ruta tomando como referencia el trafico actual y/o el histórico ya sea completo o por ventanas de tiempo, la metodología se resume en: Adquisición, Calculo y Obtención. La adquisición de datos es el proceso por el cual se obtiene la materia prima, que son el registro de trafico (posición geoespacial y velocidad) juntamente con la fecha y la hora en la que se tomo el registro. Se cuenta con una base de datos espacial con un formato arco nodo en la cual se almacenan la calles y su posición así como los datos recolectados.. El calculo es el momento en el que dichos datos se procesan, se busca la pertenencia a un segmento de vía para posteriormente hacer el calculo de pesos usando los puntos asociados a dicha vía este proceso se repite periódicamente para mantener actualizados los pesos. La obtención es el resultado que recibe el usuario al hacer la consulta, dicho resultado es obtenido con base a el ultimo calculo realizado por el algoritmo.

4.1. Marco de trabajo

Se establecen las siguientes premisas:

- Los datos de vialidades empleados para comprobar el funcionamiento de la metodologia, pertenecen a la Delegacion Gustavo A. Madero de la CDMX

- Los datos usados que describen las calles, así como los nombres de las mismas, provienden del proyecto opensource OpenStreet Maps.

Como se menciona al inicio de este capitulo la metodología se basa en Adquisición, Calculo y Obtención. Para poder entender mejor dicho proceso se explicara en las siguientes imágenes.

A través de una aplicación desarrollada en Android se hace la recolección de datos figura 4.1, dicha recolección se realiza únicamente cuando el usuario se encuentra en movimiento cada dato censado incluye la posición geográfica, la velocidad y la fecha, los cuales son guardados en la memoria del teléfono para su posterior envió, un problema en la recolección de datos es la frecuencia de recolección; si es una frecuencia muy alta en situaciones de mucho trafico caeríamos en el problema de la redundancia de datos almacenando información innecesaria y repetitiva, pero por el otro lado si es muy baja cuando se viaja a velocidades muy altas se podría perder información incluso de segmentos enteros de vía, por eso se opto por frecuencia dinámica dependiendo de la velocidad del usuario.

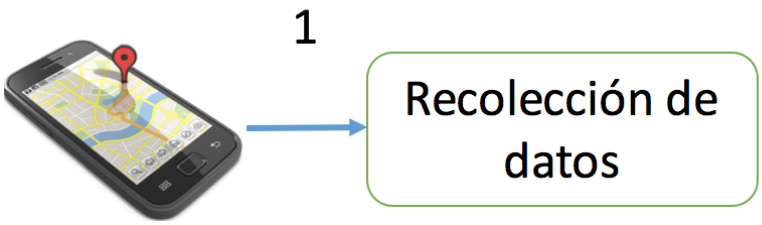

Figura 4.1: Metodología 1

En cuanto figura 4.2 el usuario se conecte a una red wifi la aplicación hará el envió de datos al servidor, esto se logra a través de un evento accionado por el sistema operativo Android.

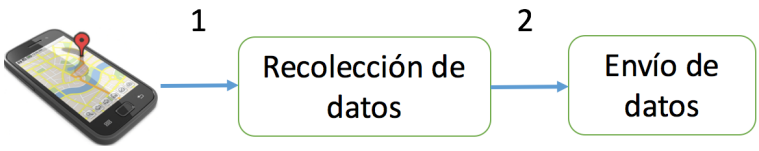

Figura 4.2: Metodología 2

El dispositivo espera la confirmación del servidor figura 4.3, esto para garantizar que los datos fueron recibidos en caso de que haya habido un error en la transmisión, en cuanto se recibe la confirmación se borran los datos de la memoria local.

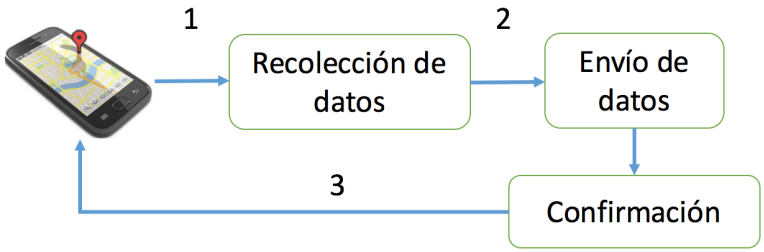

Figura 4.3: Metodología 3

En cuanto el servidor recibe los puntos censados y mediante una operación espacial conocida como buffer de la cual se hablara mas adelante, determina si dicho punto pertenece a una sección de vía o no, en otras palabras si la muestra fue censada mientras se conducía si fue así se almacena en la base de datos (figura 4.4), en caso contrario se ignora.

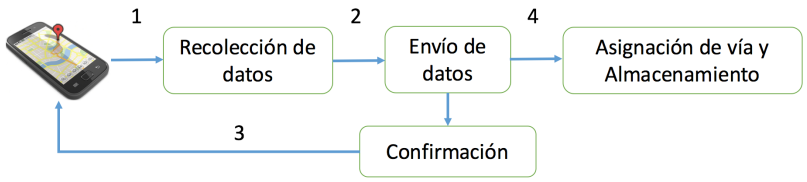

Figura 4.4: Metodología 4

En la base de datos, los puntos se almacenan como una relación uno a mucho con las vías, una vía tiene cero o muchos puntos, esta relación nos ayudara a determinar el peso de la vía mas adelante figura 4.5. La estructura en la base de datos de las calles y sus intersecciones están almacenadas como un grafo dirigido arco-nodo, esta metodología se basa en la actualización de pesos en dicho grafo.

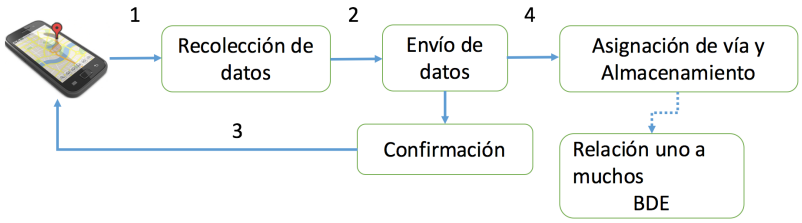

Figura 4.5: Metodología 5

El calculo del peso se realiza por cada arco del grafo, se toma una arista y todos las muestras asociadas a ella figura 4.6, posteriormente del universo de muestras tomadas pertenecientes a dicho arco se determina cuantas fueron tomadas dentro de un plazo de 10 minutos figura 4.7.

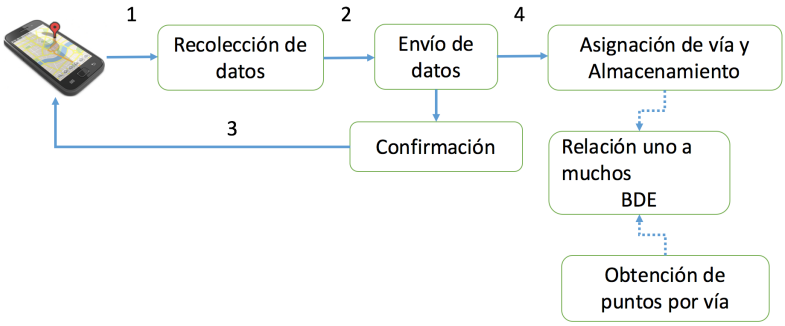

Figura 4.6: Metodología 6

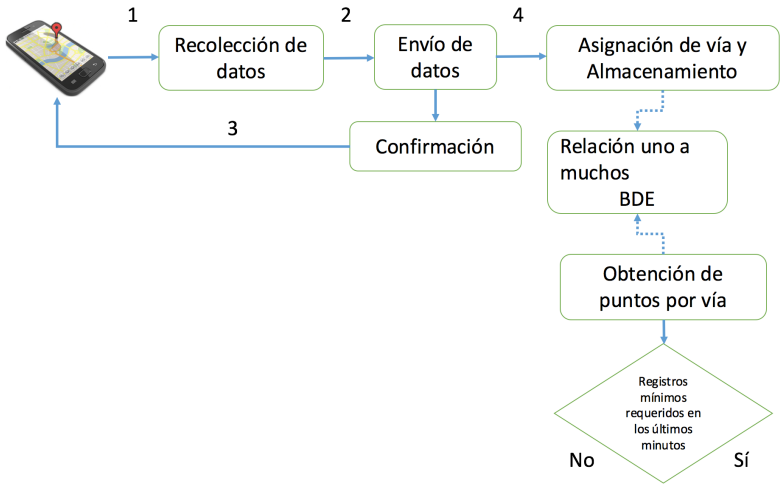

Figura 4.7: Metodología 7

Si existe un numero suficiente de puntos a evaluar dentro de dicho umbral el peso será el promedio de dichos puntos, ya que se tiene de suficiente información en tiempo real para calcular el peso de la vía; entonces se filtran (figura 4.8) los puntos que cumplen con el requisito de ser tomados en un tiempo menor a 10 minutos y se promedian, el resultado de esta operación será el peso de la vía figura 4.9.

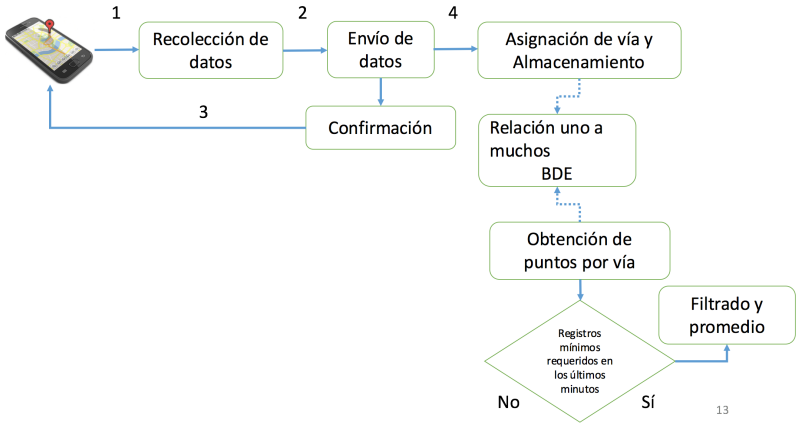

Figura 4.8: Metodología 8

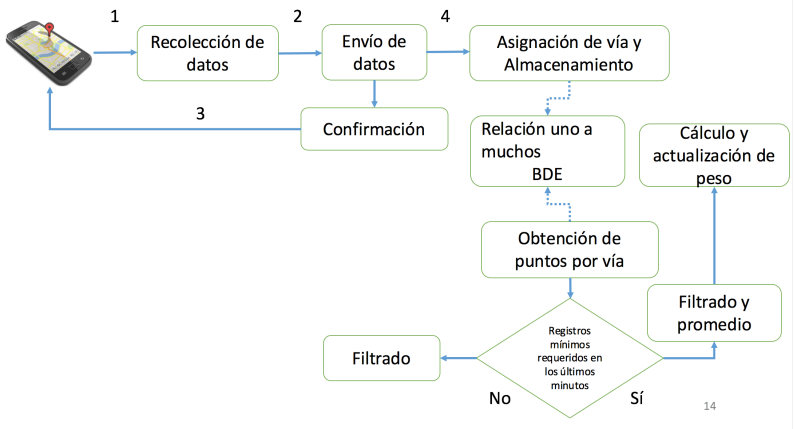

Figura 4.9: Metodología 9

En caso de que no sea así, de que no exista un numero suficiente de puntos tomados dentro del umbral de tiempo, se procede a calcular el peso en base al histórico para lograr dicho proceso se filtran los datos en base a la ventana de tiempo elegida por el usuario (figura 4.9) , y se ordenan los datos de forma descendente para su posterior procesamiento (figura 4.10) .

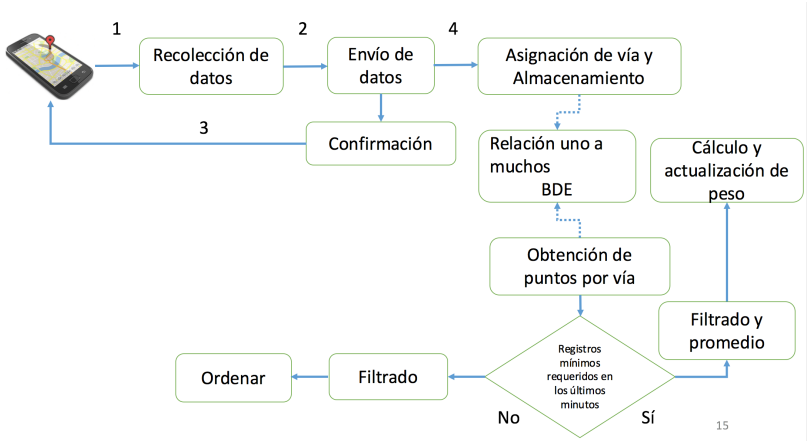

Figura 4.10: Metodología 10

Tenemos una lista ordenada de datos en forma descendente basándonos en la fecha, este conjunto de datos representa el historial según una ventana de tiempo definida por el usuario para un arco en especifico, usamos ese conjunto para calcular el peso este proceso se logra a través del suavizado linear (figura 4.11), con este proceso se logra obtener una predicción la cual equivaldría a la velocidad esperada en ese momento, una vez realizado el calculo se actualiza el valor del peso para el arco que estamos calculando (figura 4.12).

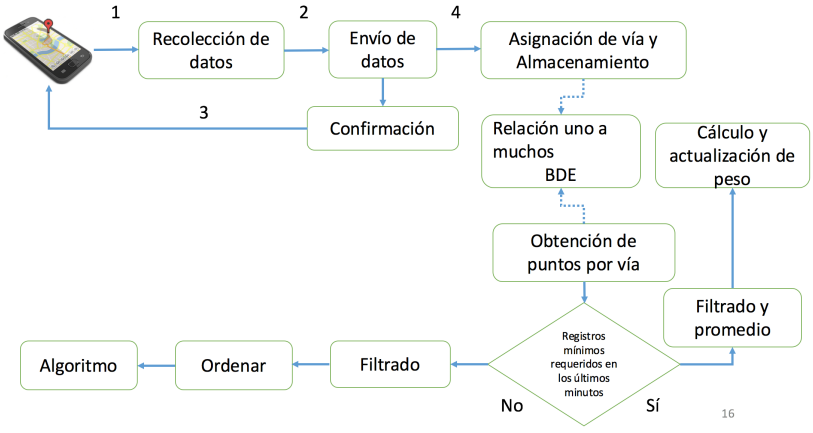

Figura 4.11: Metodología 11

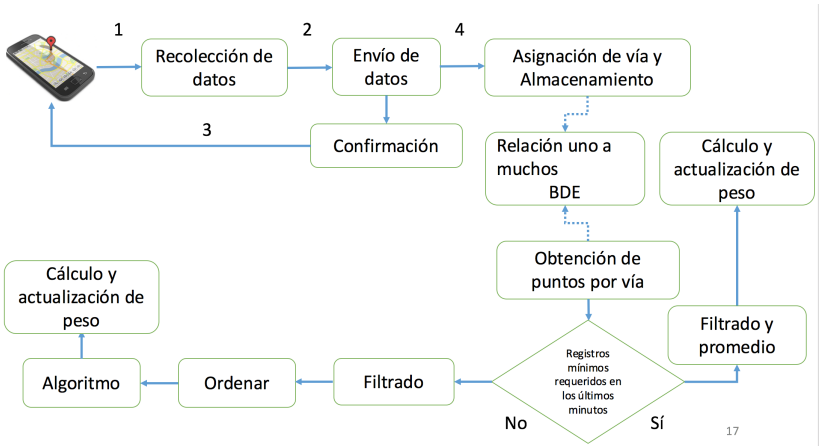

Figura 4.12: Metodología 12

Este proceso se repite con cada arco del grafo, si bien es efectivo es sumamente tardado dependiendo del tamaño del grafo, servidor, sistema operativo, este proceso puede variar entre 10 minutos a 40 minutos o incluso mas; esto representa un problema debido a que el umbral de tiempo es de 10 minutos y además que los pesos se actualizaran por completo cada vez que el proceso termine esto hace que el sistema pierda en cierto sentido la propiedad de "tiempo real", para solucionar dicho problema existe otro subproceso que toma las muestras que fueron tomadas en un tiempo menor a 10 minutos (figura 4.13), se agrupan según el arco al que estén asociados (figura 4.14) si el numero de muestras por agrupación es mayor al umbral establecido para actualizar el peso de dicha sección de vía se promedian y se procede a la actualización (figura 4.15).

Figura 4.13: Metodología 13

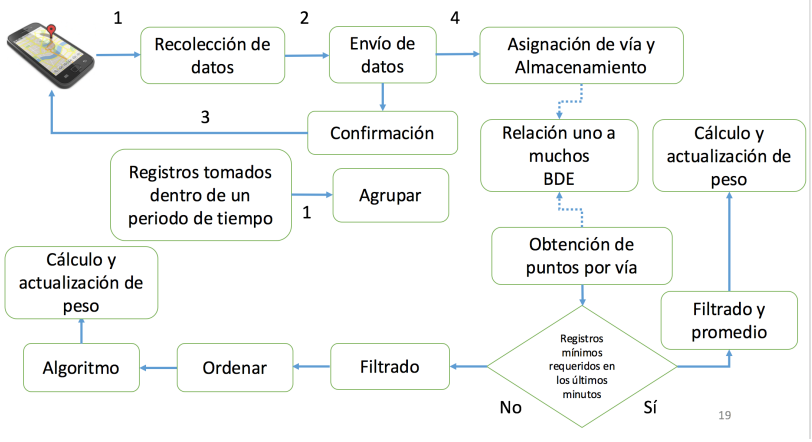

Figura 4.14: Metodología 14

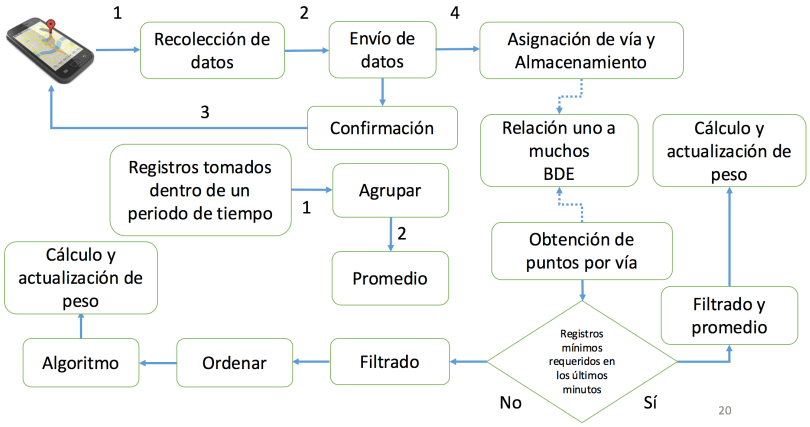

Figura 4.15: Metodología 15

Este proceso nos permite tener nuestro grafo actualizado según los datos de trafico obtenidos, de tal forma que cuando un usuario desde la aplicación hace una solicitud (figura 4.16) para buscar la ruta mas corta el servidor responderá basándose en los datos mas actuales (figura 4.24).

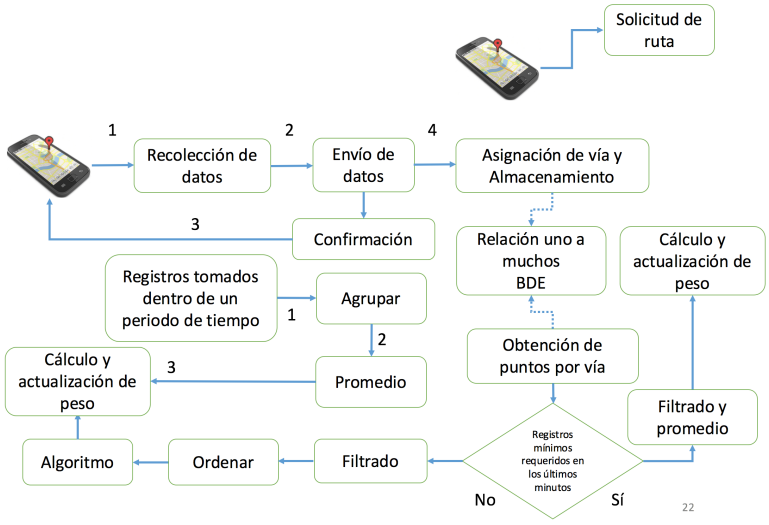

Figura 4.16: Metodología 16

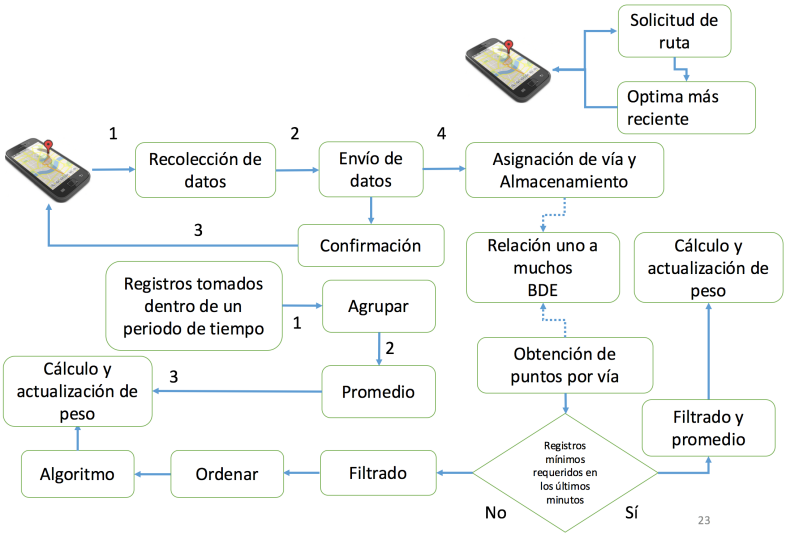

Figura 4.17: Metodología 17

En las siguientes secciones se describirá mas a detalle la metodología mostrada.
newline

4.2. Obtención de la base de datos

La base de datos es la fuente de información para el proyecto en ella se almacenan las muestras, las calles, las muestras que una están sin procesar, los arcos, los nodos, etc. Sin importar que tan pequeña sea tu zona de estudio ingresar de forma manual los registros de calles puede ser un trabajo realmente largo, para hacer el registro de calles nos basamos en un proyecto de OpenSource conocido como OpenStreetMaps (OSM) de dicho proyecto nos basamos para obtener nuestros datos, para lograrlo nos dirigimos a la pagina oficial, y después nos vamos a la sección de exportar una vez ahí seleccionamos el área que queremos exportar y después damos click en el botón de exportar (figura 4.18), esto nos descargara un ShapeFile el cual usaremos para hacer la carga de nuestra zona de estudio a la base de datos, hay que tener en cuenta que si el área es muy grande la api de OpenStreetMaps no podrá procesarla, por lo cual será necesario otra alternativa conocida como Overpass API esta opción si bien no se actualiza tan continuamente como

OSM es suficiente para propósitos de investigación y prueba.

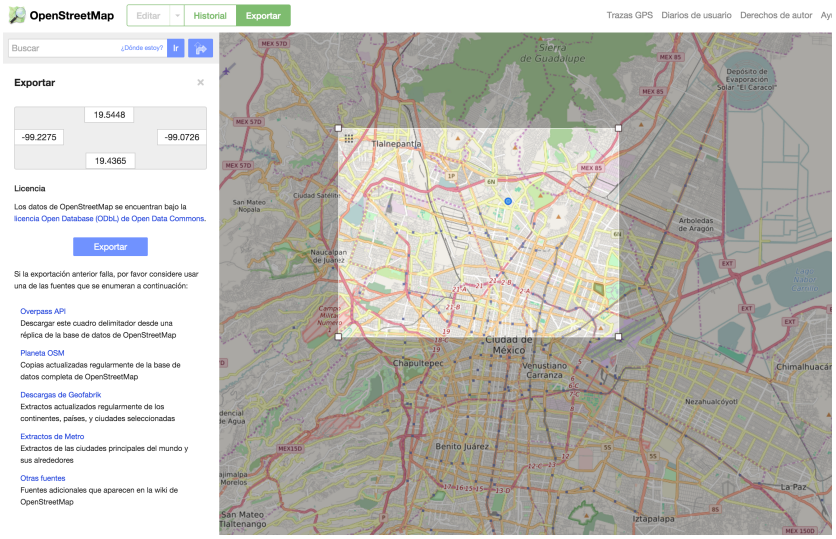

Figura 4.18: OpenStreetMaps

Para el siguiente proceso es necesario tener el motor de base de datos PostgreSQL con la extensión PgRouting, para poder exportar el ShapeFile a la base de datos usamos un paquete llamado osm2pgrouting, dicho paquete exporta el documento que obtuvimos del servidor de OSM a nuestra base de datos en un formato arco nodo, para utilizar este paquete se puede correr el siguiente comando:

```
osm2pgrouting --f your-OSM-XML-File.osm --conf
mapconfig.xml --dbname routing --username
postgres --clean
```

Este comando pide una copia de tu .osm que obtuviste del sitio de OSM que tipo de configuración estas configuraciones vienen en tu instalación de osm2pgrouting y las credenciales de tu base de datos.

Este comando se encargara de crear la topología dentro de nuestras bases de datos, al ejecutar el comando de arriba veras una salida como la que se muestra en la figura 4.19

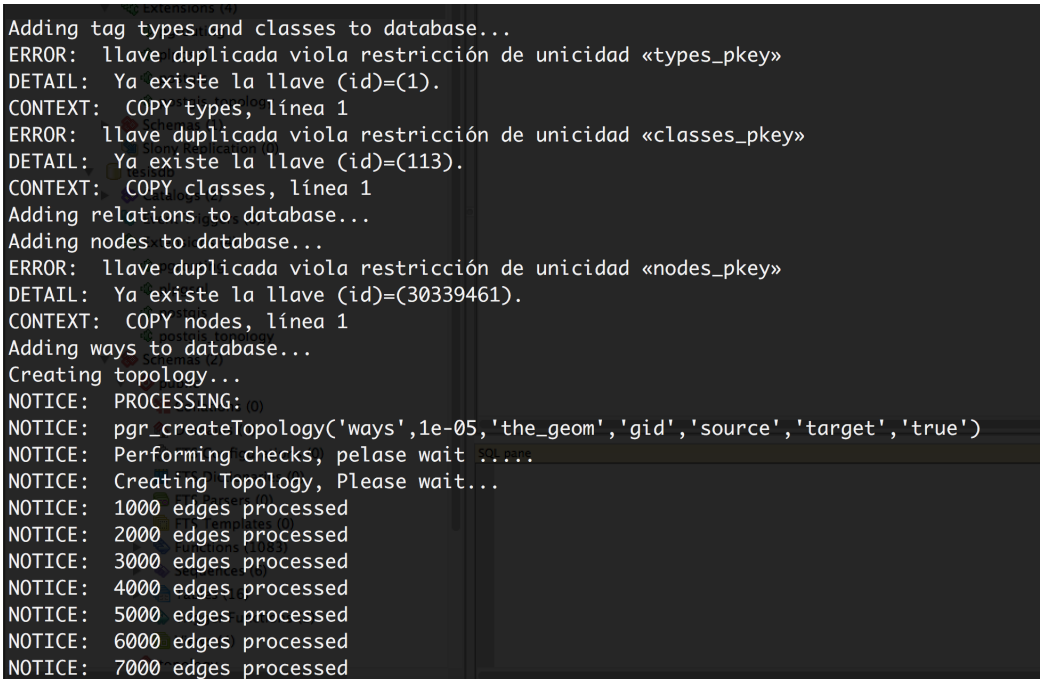

Figura 4.19: Salida Comando

Al finalizar la corrida del comando se puede visualizar la tabla creada en la figura 4.20

gid	class_id	length	name	x1	y1	x2	y2	reverse_cost	rule	to_cost	maxspeed_forwa	maxspeed_backwa	osm_id	priority
1757	112	0.171003862...	Eugenia	-99.1761307	19.3893828	-99.1745496	19.3890077	171003.8628...		NULL	50	50	8163899	2.5
6297	110	0.087900590...	Morena	-99.1558438	19.3983489	-99.1566803	19.3983975	87900.59085...		NULL	90	90	24734574	1.75
6394	112	0.097105482...		-99.1458209	19.404851	-99.1448952	19.4048335	0.097105482...		NULL	50	50	24734857	2.5
6573	112	0.037431243...		-99.1457349	19.4087216	-99.1457857	19.4083884	0.037431243...		NULL	50	50	24736816	2.5
14511	109	0.006332860...		-99.2252722	19.4665209	-99.2252748	19.466464	6332.860281...		NULL	90	90	24963307	1.5
17125	107	0.026931664...		-99.1591493	19.5265077	-99.1589368	19.5263715	26931.66433...		NULL	90	90	24984441	1.15
19835	109	0.123671931...		-99.1956249	19.4097293	-99.1946101	19.4102958	123671.9310...		NULL	90	90	25026911	1.5
20352	105	0.063832231...		-99.2171608	19.4436052	-99.2174155	19.4441266	63832.23145...		NULL	110	110	25042080	1.05
22688	110	0.068411599...	Cedros	-99.239931	19.2722696	-99.2392844	19.2723469	0.068411599...		NULL	90	90	25113937	1.75
28719	105	0.136316841...		-99.2108271	19.3066986	-99.2120767	19.3070334	136316.8412...		NULL	110	110	25225332	1.05
29185	112	0.072141706...	Xoco	-99.1504183	19.3854373	-99.1498175	19.3851215	0.072141706...		NULL	50	50	25275227	2.5
38301	117	0.074150030...		-99.1127054	19.391546	-99.1128435	19.390892	0.074150030...		NULL	90	90	26550497	3
41440	110	0.008019142...		-99.1127169	19.5278879	-99.1127919	19.5278736	0.008019142...		NULL	90	90	27457433	1.75
44192	112	0.083287921...		-99.0322676	19.3886393	-99.0326798	19.3879991	0.083287921...		NULL	50	50	27785822	2.5
46509	112	0.072869906...		-99.1533262	19.3211809	-99.1529948	19.3217568	0.072869906...		NULL	50	50	29445104	2.5
46744	108	18546.40407...		-99.2107379	19.5539668	-99.2105609	19.5539668	0.018546404...		NULL	90	90	29986658	1.5
47565	109	0.012729618...		-99.2145003	19.439134	-99.2145734	19.4390426	12729.61824...		NULL	90	90	30265395	1.5
50909	112	0.077104939...		-99.2419182	19.5075444	-99.2420048	19.508233	0.077104939...		NULL	50	50	31974334	2.5
52565	112	0.090478484...		-99.17862	19.4944562	-99.1783749	19.4952364	0.090478484...		NULL	50	50	32556653	2.5
56348	112	0.269333538...		-99.0803141	19.4735513	-99.0798797	19.4759386	0.269333538...		NULL	50	50	32871487	2.5
59613	112	0.082709671...		-99.0930668	19.470387	-99.0923547	19.4700668	0.082709671...		NULL	50	50	32999373	2.5
68044	112	0.104137978...	Durango	-99.190757	19.4766751	-99.1906745	19.4776084	0.104137978...		NULL	50	50	33726050	2.5
70619	112	0.057406554...		-99.2329555	19.3806041	-99.232849	19.3800977	0.057406554...		NULL	50	50	33762651	2.5
76922	112	0.112346044...		-99.0959176	19.5054785	-99.0949874	19.5049765	0.112346044...		NULL	50	50	33902865	2.5
84333	112	0.131624756...		-99.0409851	19.5373938	-99.0405211	19.5384938	0.131624756...		NULL	50	50	34379245	2.5
99836	100	0.012638882...		-99.0678287	19.3286286	-99.0679481	19.3286436	0.012638882...		NULL	50	50	41373304	5
101626	112	0.032337017...		-99.0147695	19.3712328	-99.0148719	19.3709585	0.032337017...		NULL	50	50	41499621	2.5

Figura 4.20: Tabla vías

Este proceso nos genera una tabla con el campo espacial de la vía, así como su longitud como se puede observar en la imagen existe un campo llamado lenght y otro llamado reverse_cost en especial el ultimo mencionado es el que nos permite que nuestro grafo sea dirigido ya que si es una vía de una sola dirección dicho campo será millones de veces mas grande que la longitud, para que cuando el algoritmo se corra ya sea Dijsktra o A estrella ignore esa vía, esta es la forma en la que creamos un grafo dirigido a nivel base de datos.

El campo de la longitud de la vía es útil pero la tesis se basa en encontrar la ruta mas rápida en tiempo, para esto es necesario agregar otras dos columnas que serian cost_speed y reverse_cost_speed, el objetivo de la segunda columna es el mismo que para el caso de la longitud para respetar la dirección de la vía ya sea única o bidireccional, es necesario agregar estos campos a la tabla ways de la base de tipo double. Por propósitos de comparación también incluirá ruteo por tipo de vía, esta opción dará preferencia a las vías primarias de igual forma para el caso anterior es necesario agregar los campos a la tabla vías cost_way y reverse_cost_way. Que valores tendrían dichas columnas para el tipo de vía es muy sencillo la tabla ways trae una columna llamada protity dicha columna se puede ponderar con los campos

de longitud, y para la velocidad como valor inicial se puede agregar el tiempo que tarda en recorrer la vía y como datos para calcular el dato se usaría la longitud y una fracción de la velocidad máxima permitida en esa via dato también proporcionado por la tabla ways, al final de este proceso se debería ver algo como se muestra en la figura 4.21.

	source integer	target integer	cost_speed double precision	reverse_cost_speed double precision	reverse_cost_way double precision	cost_way double precision
	121184	121185	2.9774697343401	2.9774697343401	0.9625	0.9625
0FA409	121185	121186	47.834080000086	47.834080000086	0.9625	0.9625
	121186	121187	10.820743147595	10.820743147595	0.9625	0.9625
03D105	121187	98899	23.172108842682	23.172108842682	0.9625	0.9625
	98899	98910	3.2085960476755	3.2085960476755	0.9625	0.9625
	120730	121188	22.929995623573	22.929995623573	1.875	1.875
	121188	121189	34.060915930548	34.060915930548	1.875	1.875
	121189	121190	4.707579965197	4.707579965197	1.875	1.875
	121190	121191	25.100081624001	25.100081624001	1.875	1.875
	121192	121193	25.987959861999	25.987959861999	1.875	1.875
	121193	121194	2.9379745111783	2.9379745111783	1.875	1.875
	121194	121195	28.485457061495	28.485457061495	1.875	1.875
	121195	121196	27.517328614282	27.517328614282	1.875	1.875
	121196	121197	4.7009140857139	4.7009140857139	1.875	1.875

Figura 4.21: Campos necesarios

También es necesario crear una tabla para almacenar las muestras es importante agregar la llave foránea de la tabla ways para poder relacionar las muestras con las vías.

4.3. Envió de datos muestreados

Esta es la etapa en la cual se recolectan los datos muestreados desde la aplicación móvil y son enviados al servidor para su evaluación y clasificación. Los datos muestreados contienen la posición geográfica, la fecha con ora en la que fue tomada la muestra y la velocidad a la que se desplazaba el usuario

en ese momento, estos datos son particularmente importantes por que nos ayudaran a hacer el registro histórico con el cual se hará el calculo de los pesos (proceso descrito mas adelante) de nuestro grafo. Se analizara esta etapa primero desde lado del usuario (aplicación) y después del lado del servidor.

4.3.1. Recolección y Envío

Esta parte es realizada del lado de la aplicación (usuario), la aplicación se encarga de censar la posición y velocidad del usuario.

Mientras el usuario se encuentra en movimiento la aplicación mide sus datos geoespaciales, para esto se utiliza un evento con el cual cuenta Android a lo cual es posible también pasarle la frecuencia con la que se desea que dispare el evento en caso de haberse presentado un cambio, para que la aplicación pueda continuar capturando datos incluso cuando se encuentra cerrada o suspendida es necesario levantar un servicio en Android, e iniciar el servicio cuando la aplicación inicie.

La instancia del LocationManager clase incluida en Android útil para nuestro propósito se inicia de la siguiente forma:

```
1  private static final int LOCATION_INTERVAL = 100;
2  private static final float LOCATION_DISTANCE = 20f;
3
4  LocationListener[] mLocationListeners
5      = new LocationListener[] {
6  new LocationListener(LocationManager.GPS_PROVIDER, this)
7      };
8
9  mLocationManager.requestLocationUpdates(
10                       LocationManager.GPS_PROVIDER,
11                       LOCATION_INTERVAL,
12                       LOCATION_DISTANCE,
13                       mLocationListeners[0]);
```

Esto iniciara nuestro servicio de localización, seguidamente esto nos permitirá en el método onLocationChanged sobrescrito de la interfaz LocationListener, cada vez que existe un cambio en la posición que también respete nuestra distancia mínima y nuestro tiempo de muestreo se ejecutara el có-

digo que se encuentra dentro del método, en el momento en que el método es activado lo primero que se evalúa es que la velocidad no sea 0 a pesar de que el dispositivo se encuentra en movimiento esto puede suceder por que los satélites nos se encontraban sincronizados, si la velocidad es mayor a cero y es una muestra valida procedemos a checar si el dispositivo esta conectado a una red Wifi en caso afirmativo se envía al servidor la información en caso opuesto se almacena internamente, el usuario tiene la posibilidad de enviar los datos en tiempo real usando su red de datos figura 4.22, si no lo desea o no cuenta con un servicio de datos la información se guardara de forma continua hasta que el dispositivo sea conectado a una red wifi en ese momento se sincronizara la información con el servidor el código de envió de datos es mostrado a continuación:

```
1  if (intent.getAction().equals("diego.wifi.receiver.CUSTOM_WIFI"))
       {
2              // Do whatever
3              long time= System.currentTimeMillis();
4              LastHttpSend mApp =
5                ((LastHttpSend)context.getApplicationContext());
6              long oldTime = mApp.getGlobalMilis();
7              Log.e(TAG, "Old Time: "+ oldTime);
8                  long difTime = time-oldTime;
9                  if(difTime < 20000L && difTime > 0L)
10                 {
11                     Log.e(TAG, "no ha pasado suficiente tiempo");
12                     return;
13                 }
14             mApp.setGlobalMilis(time);
15             jo = getResults();
16             if(jo.length() == 0)
17             {
18                 Log.e(TAG, "JSON Vasio");
19                 return;
20             }
21
22             Log.e(TAG, "JSON: " + jo);
23
24             new HttpAsyncTask().execute("http://dominio/ruta");
25         }
```

Primeramente se verifica que la acción provino del intent de WIFI, lo siguiente es ver el tiempo en el teléfono esto es para evitar que se haga una

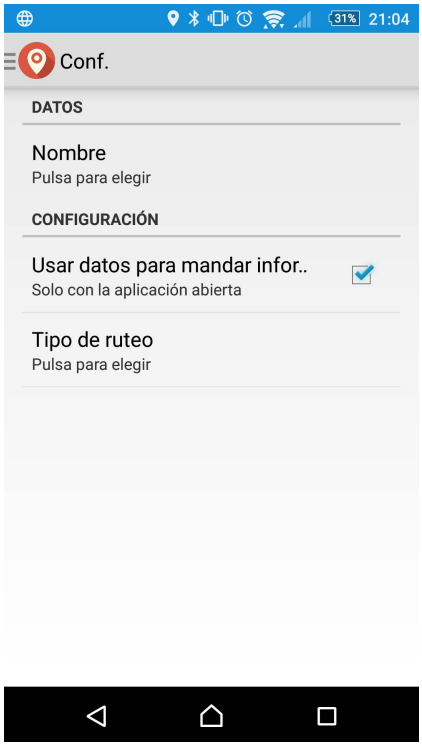

Figura 4.22: Menú Aplicación

misma petición seguida y así evitar que se envíen datos de forma duplicada, se leen los registro del archivo donde se almacenan y se les da un formato tipo JSON, en caso de que el JSON venga vacío no se realiza ninguna acción en caso contrario se envía la solicitud al servidor. La variable jo es una variable de clase por lo cual al ejecutarse el método execute() también carga los datos usando una método sobrecargado llamado doInBackground el al cual nos permite mandar el JSON obtenido en nuestra solicitud AJAX tipo POST.

En cuanto se hace la solicitud el dispositivo espera la respuesta del servidor en caso de recibir una respuesta positiva borra los datos del mismo, en caso contrario lo volverá a intentar dentro de un tiempo continuamente hasta recibir la respuesta esperada.

4.3.2. Recepción y clasificación

Estas tareas se ejecutan del lado del servidor, la recepción es la que captura los datos recibidos por el cliente y se encarga de darle la respuesta positiva que espera. La clasificación como su nombre lo dice se encarga de ver si un punto pertenece a una vía y en caso afirmativo a cual pertenece.

Un problema al que se enfrenta es la recepción de datos, mientras sean pocos el proceso será rápido y se dará respuesta de la misma forma, pero que pasa si hablamos de muchos datos que un dispositivo envía digamos de un viaje de un día, la cantidad de datos seria abismal y procesarlos todos en ese momento podría darse el caso de que el tiempo de respuesta máximo para una solicitud AJAX expire; esto traería muchos problemas al no dar respuesta al dispositivo el mismo intentara enviar los datos otra vez esto provocaría datos repetidos de forma masiva. La solución mas sencilla a este problema es encolar las solicitudes; esto nos permite recibir la solicitud responder al dispositivo casi al instante para que el mismo borre la información y después con un micro servicio del lado del servidor procesar las solicitudes encoladas.

La pregunta principal es cómo es que se determina si una muestra pertenece a una sección de vía y en caso afirmativo a cual, esto se logra a través de una operación espacial conocida como BUFFER básicamente dado un punto geoespacial y una distancia generalmente en metros te regresa todos los objetos espaciales de una capa en especifico que hacen intersección con el buffer generado. Si aplicamos esta operación a la capa de las calles y a su ves ordenamos y tomamos el objeto mas cercano mas cercano a nuestro punto de referencia esta será nuestra sección de vía que buscamos; si regresa vacío es que esa muestra no pertenece a una vía, a través de las componentes espaciales que nos brinda postGis la consulta necesaria para realizar esta operación se muestra a continuación; donde la variable point es un String conformado con la latitud y la longitud separadas por espacio, a su vez se ve un ejemplo de repuesta en la figura 4.23.

```
1 SELECT gid,name
2                 FROM ways
3                 WHERE ST_DWithin(the_geom,
4                   ST_GeographyFromText('POINT($point)'), 7)
5                 ORDER BY ST_Distance(the_geom,
6                   ST_GeographyFromText('POINT($point)'));
```

gid integer	name text
29572	Wisconsin
29547	Kansas
90595	Av. Pennsilvania
29571	Wisconsin
29548	Kansas
29546	Kansas
90594	Av. Pennsilvania

Figura 4.23: Resultado de la consulta

Esta operación regresa todos los puntos contendidos en un área de 7 grados con centro en el valor de nuestro punto de referencia y los ordena del mas cercano al mas lejano.

Esto nos regresa una lista ordenada la cual el primer resultado es la calle mas cercana, para nuestro propósito tomamos el gid que será la llave foránea para enlazar nuestra muestra con la sección de vía en la figura 4.24 se muestra la tabla donde se guardan las muestras, ahí se puede ver el campo osm_id que es la llave foránea utilizada para saber que dicha muestra pertenece a una determinada sección de vía, así mismo en la figura 4.25 se muestra un ejemplo de cómo se ve el conjunto de muestras pintados en un mapa.

id	osm_id	vel	fecha	created_at
87195	261154	1.06118	2016-03-28T...	2016-03-28T...
87194	150313	2.43132	2016-03-28T...	2016-03-28T...
87193	150212	13.57	2016-03-28T...	2016-03-28T...
87191	150212	13.2504	2016-03-28T...	2016-03-28T...
87192	150212	13.44	2016-03-28T...	2016-03-28T...
87190	136414	12.69	2016-03-28T...	2016-03-28T...
87189	150211	14.4414	2016-03-28T...	2016-03-28T...
87187	150211	15.39	2016-03-28T...	2016-03-28T...
87188	150211	14.5802	2016-03-28T...	2016-03-28T...
87185	150211	14.9801	2016-03-28T...	2016-03-28T...
87186	150211	15.2002	2016-03-28T...	2016-03-28T...
87183	150210	14.9303	2016-03-28T...	2016-03-28T...
87184	150210	14.4201	2016-03-28T...	2016-03-28T...
87180	150210	16.8101	2016-03-28T...	2016-03-28T...
87181	150210	16.99	2016-03-28T...	2016-03-28T...
87182	150210	16.4409	2016-03-28T...	2016-03-28T...
87178	150210	16.1004	2016-03-28T...	2016-03-28T...
87179	150210	16.48	2016-03-28T...	2016-03-28T...
87176	150210	14.07	2016-03-28T...	2016-03-28T...

Figura 4.24: Tabla de muestras

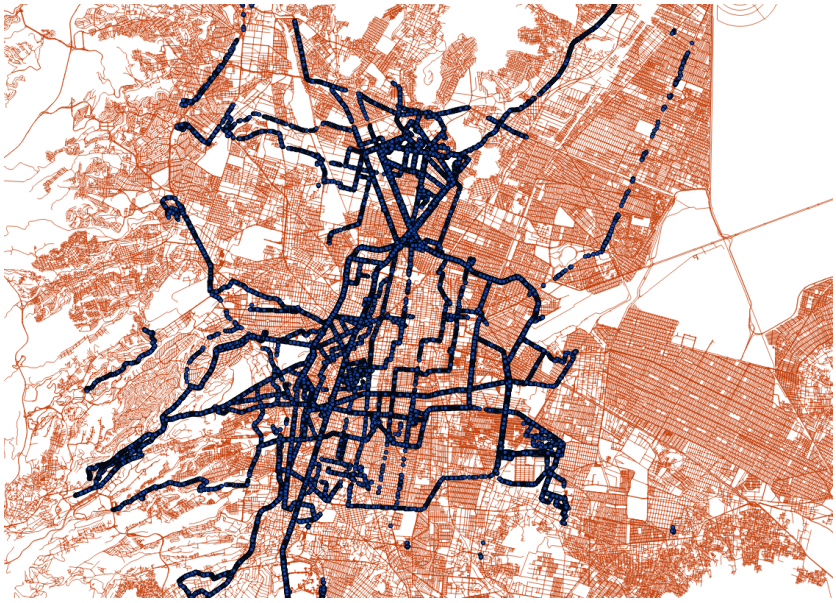

Figura 4.25: Muestras

4.4. Actualización de pesos

Aquí es donde se actualizan pesos, es donde el grafo se vuelve dinámico y nos dará la posibilidad de crear rutas dinámicas, se tiene un script en PHP el cual se encarga de ejecutar el proceso que vamos a describir, una y otra vez.

El peso se calcula por cada sección de vía o en términos de grafos por cada arista, partimos por tomar una arista; de dicha arista se buscan las muestras asociadas a ella (figura 4.26) lo cual es posible gracias a la llave foránea que se guarda en cada muestra.

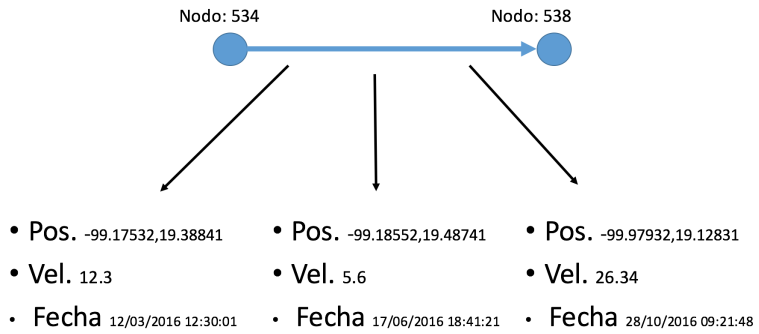

Figura 4.26: Relación vía muestras

Ya que tenemos nuestras muestras se buscan aquellas muestras que hallan sido tomadas dentro de un intervalo de 10 minutos, si el numero de muestras es el suficiente, no hay nada mas que calcular es lo que esta pasando en ese momento, lo que se tiene son datos reales entonces solo bastara con el promedio y hacer la actualización en la base de datos, el framewrok de php/Laravel tiene métodos que facilitan en gran manera hacer este tipo de consultas como se muestra a continuación:

```
$way = Way::find($row->gid);
            $ahora = Carbon::now();
            $antes = Carbon::now()->subMinutes(10);

            $locsRecientes = $way
              ->locations()
              ->whereBetween('fecha', [$antes, $ahora])
              ->avg('vel');
```

Si esto es valido la variable locsRecientes debe venir con algún valor en caso contrario se pasaría al caso por default.

Cuando es necesario tomar del histórico para calcular el valor del peso de dicha arista, para esto traemos todos los elementos relacionados con la sección de vía, después los filtramos por la sección de tiempo deseada ya sea la hora o días de la semana o le día y la hora de la consulta esto dependerá de la selección del usuario, a continuación se muestra como se hace el filtrado

por día de la semana el mismo principio se sigue para los demás tipos de filtro.

```
$filtrarPorDia = $locTodos->filter(function($item){
    $diaHoy = Carbon::now()->dayOfWeek;
    $diaItem =
    Carbon::createFromFormat('Y-m-d H:i:s', $item->fecha)
      ->dayOfWeek;
    if($diaHoy == $diaItem)
        return true;
});
```

Otro problema es que al ser muchos datos se puede presentar el detalle de haber para un intervalo de tiempo muy corto (0 a 30 Segundos) se tengan datos con valores similares esto puede bajar el rendimiento del algoritmo de forma innecesaria, para estos casos se da por hecho que el valor es el mismo por lo que se agrupan y se promedian. Al final de este proceso tenemos una lista ordenadas de muestras según una ventana de tiempo ya sea por ejemplo todos los lunes a las 3 de la tarde según sea el momento en que se haga el calculo, contamos con nuestro histórico ya filtrado y optimizado ahora es el momento de ingresarlo al algoritmo.

Fue implementado el algoritmo de suavizado lineal el cual es una mejora de la media móvil un algoritmo el cual busca hacer predicciones a corto plazo; se basa en la idea de tomar los últimos datos conocidos los cuales los pondera y los divide entre la suma del vector de ponderación dándole mas peso a los datos mas recientes, con el método de suavizado se evita el problema de dar pesos aleatorios asignándole pesos ya establecidos por el algoritmo que se calculan según el numero de muestras utilizadas (Figura 4.27); donde N es el numero de elementos que se tomaran para hacer el calculo de la predicción, el numero recomendado del numero de datos que se pueden utilizar para hacer el calculo es de 4 [16].

$$SMA = \frac{p_M + p_{M-1} + \cdots + p_{M-(n-1)}}{n}$$
$$= \frac{1}{n}\sum_{i=0}^{n-1} p_{M-i}$$

Figura 4.27: Suavizado Lineal

Para un registro de velocidades de 70 Km/h, 85 Km/h, 87 Km/h, 77 Km/h, primero se calcula la ponderación esta se calcula con la siguiente ecuación (n*n + n) / 2 donde n es 4, el resultado es 10, entonces para cada dato la ponderación seria 1/10, 2/10, 3/10, 4/10; lo cual nos deja la siguiente operación [(70 * 0.1) + (85 * 0.2) + (87 * 0.3) + (77 * 0.4)] / (0.1 + 0.2 + 0.3 + 0.4) dándole mas peso al valor mas reciente, para este ejemplo el valor esperado es 80.9 Km/h.

Este proceso se repite con cada arco registrado en la base de datos. Esto trae un problema que depende de que tan extenso sea el grafo, si el servidor se tarda en procesar los datos 30 o 40 minutos y si solo se toman en cuenta para datos reales todos aquellos que lleguen en un plazo de 10 minutos; si llega un dato no se tomaría en cuenta como algo que pasa en el momento si llega cuando el proceso solo lleva 15 minutos de haber empezado, también que las actualizaciones se harían al finalizar ese tiempo, cada 30 minutos la aplicación pierde hasta cierto punto la propiedad dinámica. Para esto se creo un micro servicio el cual se encarga solo de leer cada 30 segundos los nuevos puntos que llegan de los usuarios y que cumplan la regla de estar dentro de los 10 minutos, recordemos que estos puntos pudieron ser tomados horas atrás dependiendo si tenia red o si tenia la opción de sincronizar en tiempo real, una vez seleccionados esos puntos se agrupan según a la vía a la que pertenecen; si son suficientes muestras para esa vía se promedian, siguiendo el principio de que es lo que pasa en ese momento por lo tanto no hay necesidad de predecir, y por ultimo se actualiza.

4.5. Ruteo

Esta es la parte donde el usuario hace la petición y obtiene como respuesta la ruta mas rápida en ese momento y con una ventana de tiempo. Como primer paso el usuario hace la petición desde su dispositivo no es mas que una solicitud AJAX de tipo GET como parámetros se envían la coordenada de partida y la coordenada destino, desde la aplicación la coordenada de partida es tu ubicación actual y la coordenada de destino es el punto en el mapa donde el usuario presione en cuanto se haga un click largo sobre la pantalla en ese momento se hace la solicitud como se muestra en la figura 4.28.

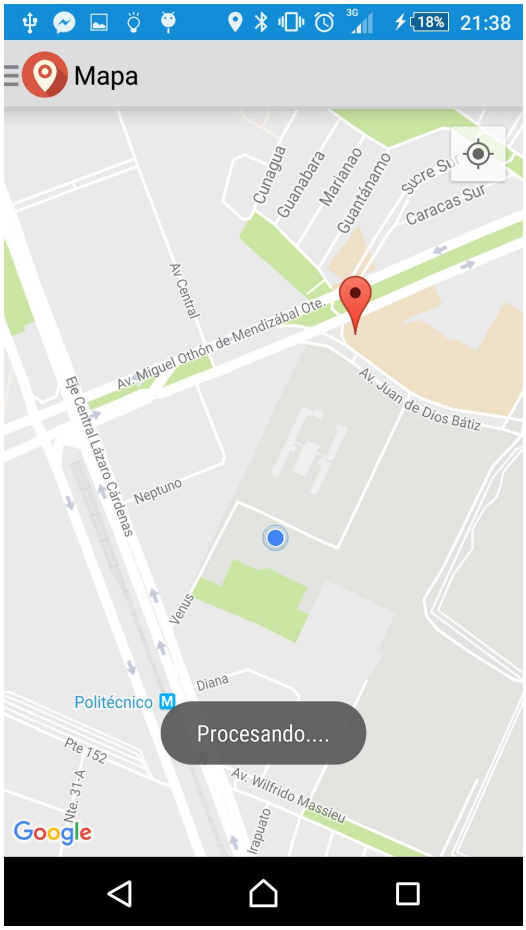

Figura 4.28: Solicitud

El servidor recibe un par de coordenadas como entrada como salida debe devolver una lista con las calles que ha de recorrer. Recordemos que la ciudad esta representada en forma de grafo dentro de la base de datos y para poder encontrar una ruta es necesario recorrerlo de un nodo a otro, lo primero que se hace es encontrar el nodo mas cercano a los puntos recibidos, para solucionar este problema podríamos hacer un BUFFER pero para la función buffer necesitamos conocer o ingresar un radio o una distancia, para la operación que se pretende realizar no se conoce la distancia del nodo mas cercano. Para solucionar esto PostGis nos provee con las funciones necesarias para realizar la solicitud necesaria, la cual se muestra a continuación:

```
SELECT name, source, target FROM ways
    ORDER BY the_geom <->
    ST_GeometryFromText('POINT($punto)',4326)
    LIMIT 1;
```

Esta consulta encuentra la sección de vía mas cercana a nuestro punto de referencia, pero recordemos que nuestra vía es una arista de un grafo por lo que esta consulta nos devuelve también un par de nodos el nodo de inicio y de salida de la vía, el nodo que se elije se basa en el sentido de la vía, si es dual es arbitrario pero si es de una sola dirección se toma el nodo de salida.

Ya con los nodos de partida y de llegada, ahora si podemos trazar nuestra ruta aplicando el algoritmo de A* en el grafo almacenado, para hacer dicha función la extensión PgRouting de Postgres cuenta con una extensión llamada pgr_astar la cual obtiene la ruta optima al destino, la consulta para realizar dicha operación se muestra a continuación y un ejemplo de su respuesta se muestra en la figura 4.29.

```
SELECT seq, id1 AS node, id2 AS edge,
 cost, ST_AsText(the_geom) as path
  FROM pgr_astar(
       'SELECT gid as id, source, target,
       cost_speed as cost,
       reverse_cost_speed as reverse_cost,
       x1,y1,x2,y2
      FROM ways',
      $startNode, $targetNode, true, true
      )as di
         JOIN ways pt
```

```
12          ON di.id2 = pt.gid;
```

seq integer	node integer	edge integer	cost double precision	path text
0	19111	31536	35.404195442993	LINESTRING(-99.135172 19.3919973,-99.1346166 19.3919128)
1	19184	31537	33.560875509715	LINESTRING(-99.1346166 19.3919128,-99.1340901 19.3918328)
2	21902	31605	33.732301846654	LINESTRING(-99.1340901 19.3918328,-99.1341668 19.3913324)
3	21951	37850	13.142044384492	LINESTRING(-99.1341668 19.3913324,-99.1347015 19.3913791)
4	19183	37851	12.208108838437	LINESTRING(-99.1347015 19.3913791,-99.1351982 19.3914225)
5	19109	37852	3.3087268983664	LINESTRING(-99.1351982 19.3914225,-99.1353311 19.3914459)
6	16947	37853	15.002782969472	LINESTRING(-99.1353311 19.3914459,-99.1359349 19.3915458)
7	19179	37854	13.964077818547	LINESTRING(-99.1359349 19.3915458,-99.1365013 19.3916108)

Figura 4.29: Respuesta pgr_astar

La función pgr_astar nos retorna los ids de las secciones de vía que el usuario deberá de seguir para obtener los demás campos necesarios se hace un join con la misma tabla ways para obtener los demás datos necesarios, al final obtenemos una lista ordenada con las calles que el usuario deberá de seguir.

El servidor retorna en formato JSON la respuesta obtenida de la consulta, y desde la aplicación dicha respuesta se toma y se pintan utilizando las coordenadas que vienen en cada elemento de la lista figura 4.30.

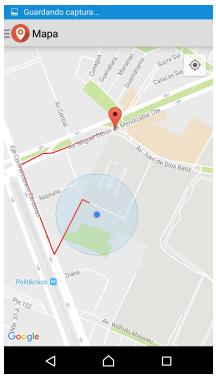

Figura 4.30: Respuesta del servidor

Capítulo 5

Resultados Experimentales

Como se ha podido observar a lo largo de la metodología se ha buscado trazar rutas en base a la ventana de tiempo elegida por el usuario, durante el desarrollo de este capitulo se mostraran los resultados obtenidos y como el usuario puede llegar a dichos resultados.

Al ingresar a la aplicación en la sección de opciones el usuario puede elegir la ventana de tiempo que desea, solo puede elegir el día del cual desea tomar los datos para trazar la ruta; la hora que se toma es la que se realiza en ese momento. En la figura 5.1 se puede ver la selección del menú, el usuario selecciona el día también tiene la opción de tomar los datos de todos los días.

Ya que el usuario eligió la ventana se dirige al mapa donde dejara presionado al punto donde desea ir la ruta se trazara del punto donde se encuentra al punto que eligió (Figura 5.2).

Se siguió la ruta mostrada por la aplicación desde el CIC hasta la unidad de PEMEX ubicada en Av. Miguel Bernard no. 473 con el propósito de comparar el tiempo que la aplicación propone con el tiempo real, la ruta obtenida por la aplicación se muestra en la Figura 5.3 y en la Figura 5.4 se puede ver las secciones de vía retornadas por la solicitud donde se puede ver el tiempo en segundos por cada sección la suma nos da un total de 7.1 minutos en la Tabla 5.1 se puede ver la comparación del tiempo medido con el tiempo calculado por la aplicación y al final el total del tiempo medido en segundos.

La prueba se realizo un viernes a las 6:30 de la tarde, hora en la que la

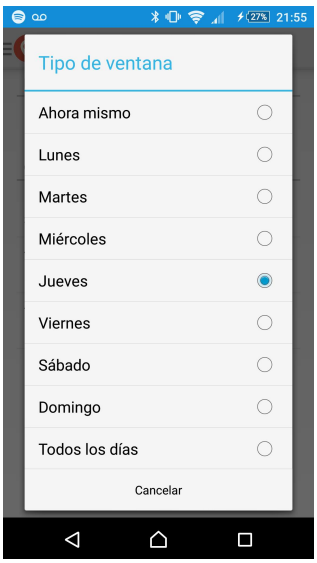

Figura 5.1: Menú ventanas de tiempo

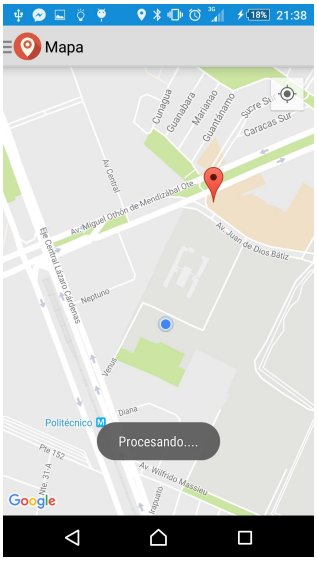

Figura 5.2: Solicitud servidor

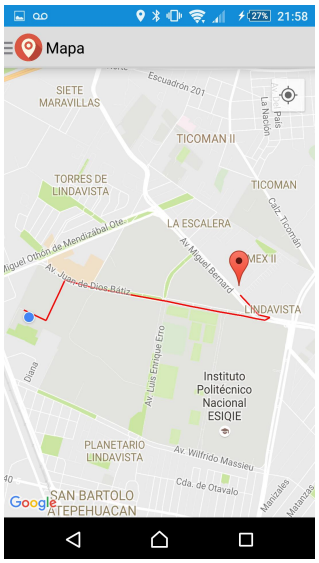

Figura 5.3: Ruta obtenida

seq integer	node integer	edge integer	tiempo double precision	name text	path text
3	87484	137141	47.2563965459806		LINESTRING(-99.1463934 19.5030897,-99.1458135 19.5042525)
4	87474	137142	15.4865389594456		LINESTRING(-99.1458135 19.5042525,-99.1455573 19.5047587)
5	87485	137143	25.2492157184525		LINESTRING(-99.1455573 19.5047587,-99.1452952 19.5053001)
6	87486	137144	23.7287704337961		LINESTRING(-99.1452952 19.5053001,-99.1452228 19.5054309)
7	87487	138628	31.6947397992139	Juan de Dios Batiz	LINESTRING(-99.1452228 19.5054309,-99.1445994 19.5052032,-99.1436541 19.5050238)
8	88229	138629	5.3551430807075	Juan de Dios Batiz	LINESTRING(-99.1436541 19.5050238,-99.1435246 19.5049971)
9	88231	211956	5.54775434781417	Juan de Dios Bátiz	LINESTRING(-99.1435246 19.5049971,-99.1432704 19.5049415)
10	85790	211957	18.3034716770764	Juan de Dios Bátiz	LINESTRING(-99.1432704 19.5049415,-99.1421084 19.504733)
11	85781	211958	29.1357508355667	Juan de Dios Bátiz	LINESTRING(-99.1421084 19.504733,-99.1403956 19.5044433)
12	85784	211959	15.5698071419481	Juan de Dios Bátiz	LINESTRING(-99.1403956 19.5044433,-99.1392509 19.5042515)
13	85800	211960	5.50681400613139	Juan de Dios Bátiz	LINESTRING(-99.1392509 19.5042515,-99.1389872 19.5042065)
14	88226	211961	20.9388700663906	Juan de Dios Bátiz	LINESTRING(-99.1389872 19.5042065,-99.1374764 19.5039065)
15	85779	211962	16.0836180018808	Juan de Dios Bátiz	LINESTRING(-99.1374764 19.5039065,-99.1371328 19.5038492)
16	85778	211963	15.4362829238039	Juan de Dios Bátiz	LINESTRING(-99.1371328 19.5038492,-99.1361995 19.5036771)
17	85796	211964	6.95472958241056	Juan de Dios Bátiz	LINESTRING(-99.1361995 19.5036771,-99.1357229 19.5035896)
18	87083	250751	12.6045032727997	Juan de Dios Bátiz	LINESTRING(-99.1357229 19.5035896,-99.1352361 19.5035135)
19	87086	250752	13.2126820879336	Juan de Dios Bátiz	LINESTRING(-99.1352361 19.5035135,-99.1347565 19.5034337)
20	88233	138634	0.991214084583444	Juan de Dios Bátiz	LINESTRING(-99.1347565 19.5034337,-99.1346942 19.5034234)
21	85793	138635	5.90829288490111	Juan de Dios Bátiz	LINESTRING(-99.1346942 19.5034234,-99.1343906 19.5033803)
22	85792	138636	10.6906128908194	Juan de Dios Bátiz	LINESTRING(-99.1343906 19.5033803,-99.1334571 19.5031949)
23	88234	138637	0.627890014445167	Juan de Dios Bátiz	LINESTRING(-99.1334571 19.5031949,-99.133388 19.5031834)
24	88235	138638	5.55365032743722	Juan de Dios Bátiz	LINESTRING(-99.133388 19.5031834,-99.1329393 19.5031103)
25	27336	39821	4.76492949525667	Juan de Dios Bátiz	LINESTRING(-99.1329393 19.5031103,-99.132635 19.5032671)
26	27280	39758	12.8553772001525	Juan de Dios Bátiz	LINESTRING(-99.132635 19.5032671,-99.132957 19.5033238,-99.1333187 19.5034142)
27	27281	133694	13.5951387786075	Miguel Bernard	LINESTRING(-99.1333187 19.5034142,-99.1339945 19.5040921)
28	85616	133695	13.1733872165606	Miguel Bernard	LINESTRING(-99.1339945 19.5040921,-99.1344684 19.5045675)
29	85617	133696	16.3535411644781	Miguel Bernard	LINESTRING(-99.1344684 19.5045675,-99.1348208 19.504921,-99.1348431 19.504944)

Figura 5.4: Resultado con tiempo

Tabla 5.1: Valores obtenidos en prueba

Predicción (S)	Tiempo de Ejecución (S)	Calle	Longitud (Km)
12.13162145	11.56342		0.158445
19.05748746	17.32433		0.142871
47.25639655	11.75464		0.062364
15.48653896	8.36923		0.066172
25.24921572	7.654345		0.016404
23.72877043	14.35433		0.037280
31.6947398	29.34232	Juan de Dios Batiz	0.170538
5.355143081	6.766522	Juan de Dios Batiz	0.013894
5.547754348	7.546443	Juan de Dios Bátiz	0.027351
18.30347168	21.23234	Juan de Dios Bátiz	0.123980
15.56980714	11.23532	Juan de Dios Bátiz	0.182392
5.506814006	4.34322	Juan de Dios Bátiz	0.121861
20.93887007	17.233223	Juan de Dios Bátiz	0.028088
16.083618	18.23323	Juan de Dios Bátiz	0.1618289
15.43628292	13.2332	Juan de Dios Bátiz	0.0365735
6.954729582	9.24223	Juan de Dios Bátiz	0.0996776
12.60450327	11.73343	Juan de Dios Bátiz	0.050893
13.21268209	10.7775	Juan de Dios Bátiz	0.051720
0.991214085	2.86554	Juan de Dios Batiz	0.051046
5.908292885	6.115423	Juan de Dios Batiz	0.006629
10.69061289	10.63433	Juan de Dios Batiz	0.032182
0.627890014	0.87896	Juan de Dios Batiz	0.0999928
5.553650327	12.643234	Juan de Dios Batiz	0.007354
4.764929495	3.56443	Juan de Dios Batiz	0.0477277
12.8553772	5.234533	Juan de Dios Batiz	0.0363496
13.59513878	16.65454	Miguel Bernard	0.0735048
13.17338722	11.876665	Miguel Bernard	0.1034379
16.35354116	28.34664	Miguel Bernard	0.0725372

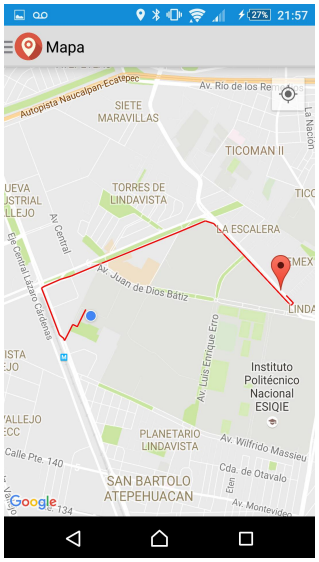

Figura 5.5: Ruta obtenida prueba 2

avenida “Miguel Othón de Mendizábal” viene muy llena y se puede ver como la evita para ir por "Juan de Dios Batiz"la cual esta mucho mas vacía. El tiempo medido en la prueba real fue de 6.8 minutos las principales diferencias se dieron en las secciones de vía del CIC a la salida que se encuentra hacia plaza torres, esto debido a que en estas secciones también se toma en cuenta el tiempo medido durante el proceso de estacionado o cuando se pasa caminando lo cual puede resultar en una medición mucho mas lenta, y en el ultimo tramo de “Juan de Dios Batiz” esto es por que es donde se encuentra el semáforo y este depende del tiempo que en ese momento toque esperar, pero al final el tiempo calculado al real fue bastante cercano.

Se realizo el mismo experimento para el Domingo a la misma hora los resultados se muestran en las Figura 5.5 y la Figura 5.6. El tiempo calculado en esta prueba es de 5.5 minutos, no se muestra la tabla de comparación debido a la extensión de la misma pero el valor real obtenido fue de 5.3 minutos. Para este caso la ruta que se tomo fue por avenidas mucho mas rápidas debido que para este día no hay trafico entonces la ruta predilecta es a través de las avenidas principales.

Se realizo la prueba de generación de rutas en diferentes ventanas desde

seq integer	node integer	edge integer	tiempo double precision	name text	path text
0	87480	137137	8.67195625619087		LINESTRING(-99.1484564 19.5030726,-99.1483181 19.503
1	87476	137136	19.3701024070063		LINESTRING(-99.1486257 19.5027629,-99.1484564 19.503
2	87482	137130	23.04592993391		LINESTRING(-99.1488319 19.5023709,-99.1486257 19.502
3	87481	171955	7.137084430607	Venus	LINESTRING(-99.1491706 19.5017536,-99.1488319 19.502
4	104903	211815	6.33545591597125	Av. de los 100 Metros	LINESTRING(-99.1491706 19.5017536,-99.1492327 19.501
5	12489	211816	13.7890303314988	Av. de los 100 Metros	LINESTRING(-99.1492327 19.5019055,-99.1492565 19.501
6	12488	211817	12.8258924110112	Av. de los 100 Metros	LINESTRING(-99.1499966 19.5038028,-99.1505054 19.505
7	85823	211818	0.903429957001513	Av. de los 100 Metros	LINESTRING(-99.1505054 19.5050307,-99.1505439 19.505
8	85834	211994	16.0623079168437	Miguel Othón de Mendizábal	LINESTRING(-99.1505439 19.5051233,-99.1496147 19.505
9	85822	211995	9.14309835230025	Miguel Othón de Mendizábal	LINESTRING(-99.1494339 19.5055793,-99.1492571 19.505
10	104901	211996	2.85072770561262	Miguel Othón de Mendizábal	LINESTRING(-99.1487173 19.5057853,-99.148373 19.5059
11	85785	211997	8.20632646963263	Miguel Othón de Mendizábal	LINESTRING(-99.148373 19.5059257,-99.1471942 19.5063
12	88230	211991	4.0454926249485	Miguel Othón de Mendizábal	LINESTRING(-99.1471942 19.5063838,-99.1466984 19.506
13	124353	211992	9.1000163331955	Miguel Othón de Mendizábal	LINESTRING(-99.1466984 19.5065809,-99.146259 19.5067
14	88245	211969	0.170683226888375	Miguel Othón de Mendizábal	LINESTRING(-99.146259 19.5067452,-99.1462242 19.5067
15	88244	211970	32.1883628798263	Miguel Othón de Mendizábal	LINESTRING(-99.1462242 19.5067581,-99.1417476 19.508
16	88241	211971	1.5048068431455	Miguel Othón de Mendizábal	LINESTRING(-99.1417476 19.5084703,-99.1415503 19.508
17	85787	211972	8.46027744245275	Miguel Othón de Mendizábal	LINESTRING(-99.1415503 19.5085452,-99.1414125 19.508
18	88247	138654	16.7134608958888		LINESTRING(-99.1414125 19.5085998,-99.1412125 19.508
19	88248	202058	0.4038907051576	Miguel Bernard	LINESTRING(-99.1399343 19.5097369,-99.1398258 19.509
20	119814	202059	29.6411923555188	Miguel Bernard	LINESTRING(-99.1398258 19.5096337,-99.1395576 19.509
21	119815	202060	23.0035052697475	Miguel Bernard	LINESTRING(-99.1395576 19.5093303,-99.1392775 19.509
22	88240	202061	17.3295669186037	Miguel Bernard	LINESTRING(-99.1392775 19.5090527,-99.1376419 19.507
23	119816	202062	5.03483664815063	Miguel Bernard	LINESTRING(-99.1376419 19.5074361,-99.1373482 19.507
24	85798	202063	2.35348088906925	Miguel Bernard	LINESTRING(-99.1373482 19.5071458,-99.137131 19.5069
25	88228	202064	2.4032968211995	Miguel Bernard	LINESTRING(-99.137131 19.5069251,-99.1369534 19.5067
26	119817	202065	27.9307463250675	Miguel Bernard	LINESTRING(-99.1368695 19.5066659,-99.1368539 19.506
27	119818	303197	1.5880971291052 5		LINESTRING(-99.1341767 19.5039472,-99.1339945 19.504
28	85616	133695	12.0080601136674	Miguel Bernard	LINESTRING(-99.1339945 19.5040921,-99.1344684 19.504
29	85617	133696	8.090415394733	Miguel Bernard	LINESTRING(-99.1344684 19.5045675,-99.1348208 19.504

Figura 5.6: Resultado con tiempo prueba 2

el CIC hasta la esquina opuesta al metro Lindavista.

En la figura 5.7 se realizo para el día domingo a las 10 AM, y se puede observar que la ruta trazada toma la ruta mas directa por que la cantidad de trafico ese día normalmente es bajo.

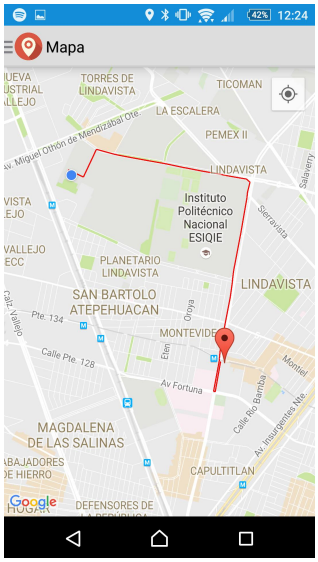

Figura 5.7: Ruta obtenida prueba sabado 10 AM

En la figura 5.8 la prueba se realizo para el día lunes a las 7 AM un día que a esa hora suele haber mucho trafico y s puede ver que la ruta trazada va por calles mucho menos transitadas.

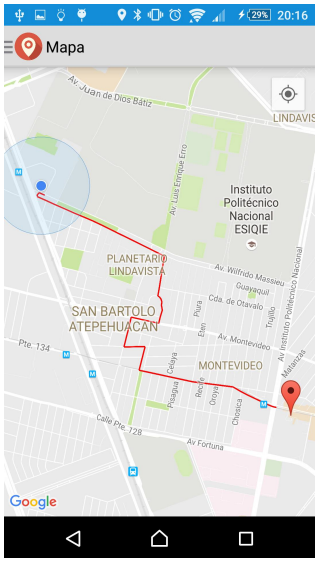

Figura 5.8: Ruta obtenida prueba lunes 7 AM

En la figura 5.9 se observa el resultado para el día martes a las 11 AM, una hora en la que generalmente no hay tanto trafico pero el algoritmo opta por mandar a través del politécnico.

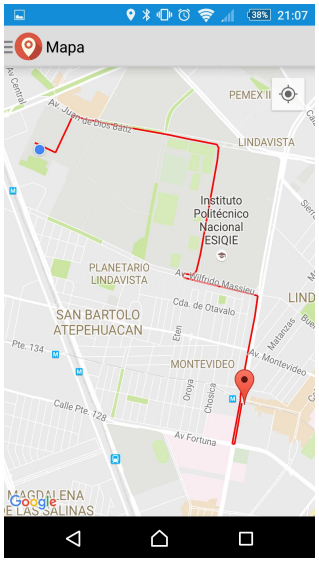

Figura 5.9: Ruta obtenida prueba martes 11 AM

El resultado obtenido para el día miércoles a las 3 PM se puede observar en la figura 5.10, es un horario en el que hay algo de trafico pero generalmente la avenida Vallejo se logra con fluidez y es la avenida por la cual el algoritmo recomienda viajar.

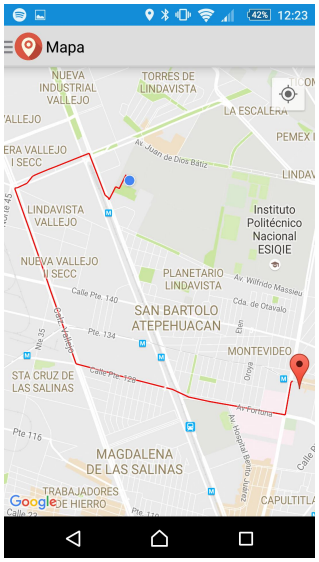

Figura 5.10: Ruta obtenida prueba miércoles 3 PM

Para el día jueves a las 7PM una hora en la que empieza a haber mas trafico el algoritmo recomienda la ruta mostrada en la figura 5.11.

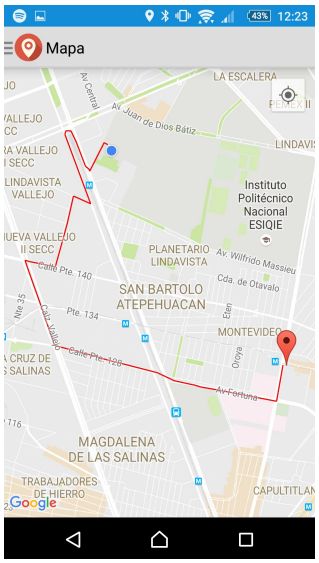

Figura 5.11: Ruta obtenida prueba jueves 7 PM

Para el día viernes a las 10 PM, el algoritmo recomienda la ruta mostrada en la figura 5.12 una hora en la que avenida 100 metros se encuentra con mayor fluidez y avenidas como politécnico no presentan tanta fluidez esto debido a que es la avenida por la cual se llegan a varios centros de entretenimiento y generalmente los viernes presenta mayor transito.

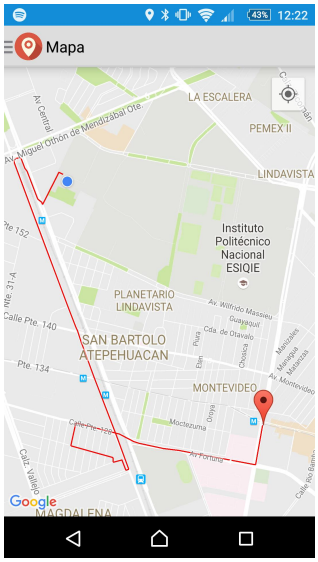

Figura 5.12: Ruta obtenida prueba viernes 10 PM

Capítulo 6

Conclusiones y Trabajos Futuros

6.1. Conclusiones

En el presente trabajo de tesis, se presento una metodología que genera rutas basados en un registro histórico y en tiempo real del trafico en la CDMX, también es posible asignar ventanas de tiempo según el día que se elija.

La selección de ventanas de tiempo nos permite hacer un estudio de trafico fuera del momento en el que estamos, y nos permite generar rutas basados en criterios históricos por días de la semana.

La metodología presentada es una combinación de distintas tecnologías tales como Andoird, PHP, Laravel, postrges, postgis, Linux. La interacción de cada tecnologías nos permitieron generar nuestro registro histórico y la generación de rutas basados en tal registro.

Si bien existen muchas aplicaciones con las que puedes obtener rutas basándonos en el trafico en tiempo real, ninguna te permite hacer dichas estimaciones tomando en cuenta ventanas de tiempo a las cuales el usuario tiene acceso.

6.2. Trabajos futuros

Agregar crowdsourcing en para la edición de calles y avenidas.

Agregar restricciones y reportar eventos.

Agregar Geo codificación para que el usuario sea capaz de ingresar direcciones.

Para aquellas calles de las que no se tenga registro, realizar una aproximado basándose en las calles cercanas.

Que el usuario pueda elegir no solo el día si no también la hora

6.3. Limitaciones

La cantidad de puntos medidos por vías, esto crearía un registro muy grande de datos.

Se requiere una cantidad grande de datos para que el trazo de las rutas sea mas optimo, ya que si una vía se queda con un registro en el cual no había trafico y no hay registros hasta dentro de algunas horas, el algoritmo la tomara como vía preferida ya que el dato no estará actualizado.

No hay total control sobre la información de avenidas y calles.

Bibliografía

[1] Xatak Android. ¿qué es android? http://www.xatakandroid.com/sistema-operativo/que-es-android, 2016. [Acccessed: 01.06.16].

[2] Bast, Mehlhorn, Schäfer, and Tamaki. A heuristic for dijkstra's algorithm with many targets and its use in weighted matching algorithms. Algorithmica, 36(1):75–88, 2003.

[3] XOCHITL OLVERA BERNARDINO. Sistema colaborativo para el monitoreo de trafico vehicular. Master's thesis, CIC IPN, 2014.

[4] JamesJ. Buckley and LeonardJ. Jowers. Fuzzy shortest path problem. In Monte Carlo Methods in Fuzzy Optimization, volume 222 of Studies in Fuzziness and Soft Computing, pages 191–193. Springer Berlin Heidelberg, 2008.

[5] Tzung-Nan Chuang and Jung-Yuan Kung. The fuzzy shortest path length and the corresponding shortest path in a network. Comput. Oper. Res., 32(6):1409–1428, 2005.

[6] Definicion De. Definición de grafos. http://definicion.de/grafos/, 2016. [Acccessed: 05.06.16].

[7] Base de Datos Uno. ¿qué son las bases de datos espaciales? https://basededatosunounivia.wordpress.com/tag/espacial/, 2016. [Accessed: 04.06.16].

[8] Concepto Definicion. Definición de windows phone. http://conceptodefinicion.de/windows-phone/, 2016. [Acccessed: 03.06.16].

[9] Datos en abierto. A* – definición. `https://datosenabierto.wordpress.com/2011/08/14/implementacion-algoritmo-a-estrella-definicion/`, 2016. [Accessed: 07.06.16].

[10] Yan Ge, Jian Wang, Guijia Wang, and Feng Jiang. Urban vehicle routing research based on ant colony algorithm and traffic rule restriction. In Yanwen Wu, editor, High Performance Networking, Computing, and Communication Systems, volume 163 of Communications in Computer and Information Science, pages 95–102. Springer Berlin Heidelberg, 2011.

[11] Mark Track GPS. ¿que es gps ? `http://www.marktrackgps.com/que-es-gps.html`, 2016. [Acccessed: 07.06.16].

[12] Fábio Hernandes, Maria Teresa Lamata, José Luis Verdegay, and Akebo Yamakami. The shortest path problem on networks with fuzzy parameters. Fuzzy Sets and Systems, 158(14):1561–1570, 2007.

[13] Takeshi Itoh and Hiroaki Ishii. A model of fuzzy shortest path by possibility measure. Japanese J. Fuzzy Theory Systems, 8(6):977–990 (1997), 1996.

[14] GCF Aprende Libre. Sistema operativo móvil ios. `https://www.gcfaprendelibre.org/tecnologia/curso/ipad/caracteristicas_generales_del_ipad/3.do`, 2016. [Acccessed: 03.06.16].

[15] Wolfgang Maass and Anthony M. Zador. Dynamic stochastic synapses as computational units. Neural Computation, 11(4):903–917, 1999.

[16] Oracle. Understanding forecast levels and methods, September 2016.

[17] Ana Maria Magdalena Saldana Perez. Metodologia para la generacion de rutas basadas en puntos de referencia. CIC IPN, 2014.

[18] Platzi. Qué es postgresql y cuáles son sus ventajas. `https://platzi.com/blog/que-es-postgresql/`, 2016. [Acccessed: 04.06.16].

[19] Hong Qu, Zhi Zeng, Changle Chen, Dongdong Wang, and Nan Yao. An optimization method of snns for shortest path problem. In Hisashi Handa, Hisao Ishibuchi, Yew-Soon Ong, and Kay Chen Tan, editors, Proceedings of the 18th Asia Pacific Symposium on Intelligent

and Evolutionary Systems, Volume 1, volume 1 of Proceedings in Adaptation, Learning and Optimization, pages 185–194. Springer International Publishing, 2015.

[20] Ramkumar Ramaswamy, James B. Orlin, and Nilopal Chakravarti. Sensitivity analysis for shortest path problems and maximum capacity path problems in undirected graphs. Mathematical Programming, 102(2):355–369, 2005.

[21] Alvaro H. Salas S.*. Acerca del algoritmo de dijkstra. Dijkstra, 2008.

[22] Esepe Studio. ¿qué es mysql? `http://www.esepestudio.com/noticias/que-es-mysql`, 2016. [Acccessed: 04.06.16].

[23] L Tang, Cheng. Representation of traffic network inclusive of node costs. Highway Traffic Technology, 2007.

[24] Fuhao Zhang and Jiping Liu. An algorithm of shortest path based on dijkstra for huge data. In Yixin Chen, Hepu Deng, Degan Zhang, and Yingyuan Xiao, editors, FSKD (4), pages 244–247. IEEE Computer Society, 2009.

[25] Huili Zhang, Yinfeng Xu, and Xingan Wen. Optimal shortest path set problem in undirected graphs. Journal of Combinatorial Optimization, 29(3):511–530, 2015.

Glosario

Android. Es un sistema operativo basado en el kernel de Linux diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o tableta.

Algoritmo. Secuencia explicita y finita de operaciones que conduce a la solución de un problema; aplicado a los SIG suele tratarse de un conjunto de operaciones algebraicas sobre atributos de mapas, o bien, de operaciones aplicadas sobre bases de datos que permiten obtener un resultado mediante combinación de información espacial y alfanumérica.

Validación de un algoritmo. Proceso de verificación mediante el cual, se asegura que el algoritmo esta libre de errores sintácticos y de escritura, y que genera resultados correctos para cualquier combinación coherente de valores de las variables de entrada.

Atributo. Propiedad o característica de un elemento perteneciente a una clase determinada, que forma parte de un sistema de datos.

Base de datos. Conjunto de datos estructurado para permitir su almacenamiento, consulta y Actualización en un sistema informático. Las bases de datos relacionales son un caso concreto en el que la información se organiza en relaciones llamadas tablas, que son conjuntos de tuplas (registros), cada una de las cuales integra información de un elemento en un conjunto de campos; si dos tablas comparten un campo con valores dentro del mismo dominio, puede aplicarse una operación de unión mediante la cual, las tuplas se enlazan en función de los valores del campo de enlace.

Cartografía. Ciencia encargada del estudio y construcción de mapas,

considera las herramientas empleadas en el diseño, construcción y manipulación de datos en mapas. Elemento de tipo mapa.

Clasificación. Proceso de agrupamiento de un conjunto de elementos en clases, dependiendo de sus características o las funciones que cumplen. Las clases son internamente homogéneas pero diferenciables entre ellas por los valores de una o varias variables.

Compresión. Técnica de reducción del número de bits necesario para almacenar o transmitir información concreta, existen técnicas de compresión sin pérdida de la información original, o con pérdida controlada de información.

Coordenada. Cantidad usada para definir una posición en un sistema de referencia las coordenadas pueden ser lineales (cartesianas) o angulares (esféricas), según el sistema de referencia aplicado.

Dato. Representación simbólica de un atributo o variable cuantitativa; describen hechos empíricos, sucesos y entidades. En computación, representan la información que el programador manipula en la construcción o en el desarrollo de un algoritmo.

Emulación. Imitación de un proceso real mediante un modelo virtual.

Entorno. Conjunto de valores de los factores influyentes bajo los cuales se realiza una simulación. Es un concepto equivalente a escenario y representa bajo que condiciones se ejecuta la simulación de un proceso; en este contexto, la experimentación es la realización de simulaciones bajo condiciones de entorno controladas.

Estándar. Propiedad que garantiza la uniformidad en los métodos de capturar, representar, almacenar y documentar la información.

GIS. Acrónimo de Geographic Information System.

GPS. Acrónimo de Global Positioning System, o sistema de localización global hace referencia a un sistema mediante el cual es posible estimar las coordenadas actuales de una estación en tierra mediante la recepción simultá-

nea de señales emitidas por varios satélites, llamados en conjunto constelación GPS.

Intersección. Operación de combinación de dos elementos pertenecientes a mapas, en la cual se conservan las zonas incluidas en el dominio espacial común a los dos elementos.

JSON. Es un lenguaje de programación de propósito general, concurrente, orientado a objetos y basado en clases Acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

Línea. Conjunto ordenado de vectores encadenados en el modelo de datos vectorial. La línea se usa para representar objetos geográficos como carreteras, tendidos eléctricos, etc. En una estructura topológica, las líneas tienen un sentido y están definidos los lados izquierdo y derecho.

Mapa. Modelo gráfico de la superficie terrestre donde se representan objetos espaciales y sus propiedades métricas, topológicas y atributivas. Un mapa puede ser analógico (impreso sobre papel, por ejemplo) o digital (codificado en cifras, almacenado en un ordenador y presentado en una pantalla). Existen mapas métricos, diseñados para representar distancias, superficies o ángulos, y mapa topológicos, diseñados para representar vecindad, inclusión, conectividad y orden. En el contexto de los GIS, un mapa es la presentación de cualquier estructura de datos usada para reflejar cartográficamente una variable espacial (nominal o cuantitativa) independientemente del modelo de datos utilizado (vectorial o raster).

Modelo. Representación simplificada de un objeto o proceso en la que se representan algunas de sus propiedades; un modelo reproduce solamente algunas propiedades del objeto o sistema original; los modelos se construyen para conocer o predecir propiedades del objeto real.

Nodo. Vértice inicial o final de una línea se aplica por extensión a las entidades puntuales que están interconectadas en una estructura en red. El orden de los nodos (inicial, final) permite asignar a la línea un sentido y dejar definidos los conceptos topológicos de izquierda y derecha.

Pgrouting. Provee funciones de ruteo espacial a Postgresql y Postgis.

Polígono. Figura geométrica plana formada por al menos un anillo externo.

Postgis. Extensión de bases de datos espaciales para Postgresql. Proporciona soporte para elementos geográficos permitiendo aplicarles consultas en SQL.

Postgresql. Sistema gestor de bases de datos relacionales, orientado a objetos, de distribución libre publicado bajo la licencia de BSD.

Red. Modelo de datos formado por nodos y conexiones entre ellos; tanto los nodos como las conexiones pueden tener atributos propios como, longitud, resistencia y sinuosidad. El análisis de redes agrupa un conjunto de técnicas implicadas en la resolución de cuestiones que pueden ser modeladas mediante una red, por ejemplo determinación del camino de mínimo coste entre dos puntos.

QGIS. Llamado también Quantum GIS, es un Sistema de Información Geográfica (GIS) de código libre para plataformas GNU/Linux, Unix, Mac OS y Microsoft Windows.