UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

CENTRO UNIVERSITARIO UAEM TEXCOCO

Sistema de información para la distribución de suministros en casos de desastres con la tecnología de Google Maps

TESIS

QUE PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

Moisés Gómez Díaz

Tutor Académico:
M. en C. José Sergio Ruiz Castilla

Tutores Adjuntos:
Dra. Cristina Juárez Landín
Dra. Yulia Nikolaevna Ledeneva

TEXCOCO, ESTADO DE MEXICO MARZO 2013

Texcoco, Méx. , a 09 de Abril de 2013

DICTÁMEN DE AUTORIZACIÓN Y OBTENCIÓN DE GRADO DE MAESTRÍA

Título del proyecto:

Sistema de información para la distribución de suministros en casos de desastres con la tecnología de Google Maps.

Tesista:

Gómez Díaz Moisés

Dictamen: **No. de revisión:** 2

- ☐ **Rechazado**
- ☐ **Sujeto a modificaciones**
- ☐ **Aceptado, condicionado**
- ■ **Aceptado**

Observaciones generales:

Sin observaciones
Aceptado para la impresión
Aceptado para defensa de grado

Tutor Adjunto	Tutor Académico	Tutor Adjunto
Dra. Ledeneva Yulia Nikolaevna	M. en C. Ruiz Castilla José Sergio	Dra. Juárez Landin Cristina

Agradecimientos

A Maricela, con mucho cariño, por su ayuda y apoyo.

A mi madre y mis hermanos, hombres y mujeres fuertes en carácter, sentimientos y búsqueda de una vida satisfactoria, profesionistas de éxito, son inspiración en mi vida.

A la UAEM, mi alma Mater, a mis maestros, profesores que contribuyeron a mi crecimiento profesional.

A mi tutor académico, M.C. Jose Sergio Ruiz Castilla, por su tiempo y apoyo.

A mis tutores Adjuntos: Dra. Cristina Juárez Landín y Dra. Yulia Nikolaevna Ledeneva por su tiempo.

A todos ellos, muchas gracias.

La mayor necesidad del mundo es la de hombres que no se vendan ni se compren; hombres que sean sinceros y honrados en lo más íntimo de sus almas; hombres que no teman dar al pecado el nombre que le corresponde; hombres cuya conciencia sea tan leal al deber como la brújula al polo; hombres que se mantengan de parte de la justicia aunque se desplomen los cielos.

La Educación, E. G. White

Contenido

Lista de Figuras

Lista de Tablas

Resumen

Ante la problemática que se presenta en situaciones de emergencia para el reparto de suministros, se hace una propuesta de solución para la repartición terrestre de estos, partiendo de un centro de distribución hacia refugios temporales para repartir ayuda como medicamentos, materiales peligrosos, vacunas, y todos aquellos que se acoplen al modelo de TSP (Problema del agente viajero). Para lo anterior se ha desarrollado un sistema de información con la tecnología de Google Maps utilizando el algoritmo del vecino más cercado para la solución de TSP que puede contribuir a que los damnificados reciban ayuda oportuna. Se busca generar una ruta con la distancia total mínima que recorra todos los refugios temporales utilizando la tecnología de Google Maps y proporcionar los suministros de manera oportuna. La propuesta se presenta como un sistema de información Web para acceder desde cualquier dispositivo con conexión a internet.

Introducción

Objeto de Estudio

El objeto de estudio de esta investigación consiste en proponer una ruta en el orden en que deben visitarse los refugios temporales partiendo de un centro de distribución cuando sucede una situación de emergencia en México.

En esta tesis también se hablará del problema de TSP aplicado al reparto de suministros, sin olvidar que se busca repartir suministros en los refugios temporales.

Por lo tanto propone una ruta con distancia mínima que recorra todos los refugios temporales a elegir partiendo de un centro de distribución**.**

Motivación

Las emergencias y los desastres suponen pruebas muy severas para la capacidad logística y de la organización de los países afectados. El reto se siente en el sector salud donde la deficiencia en la administración de suministros puede tener graves consecuencias.

El problema no sólo radica en la adquisición de bienes y equipos de emergencias. También deben prestarse rigurosa atención al manejo de aquellos suministros que ya tenemos a mano o se encuentran en camino. Puede ser que los suministros sean muy abundantes en los niveles centrales de distribución mientras que en el terreno en el lugar de emergencias se producen carencias de consecuencias muy graves. Por otro lado las

donaciones no solicitadas también compiten por el uso de medios de trasporte e instalaciones de almacenaje que pueden estar saturadas (Salud, 2001).

En (Salud, 2001), se presentan los conceptos básicos de la gestión logística de los suministros humanitarios. Aunque el manejo de suministros médicos y farmacéuticos reciben especial atención, los principios por los que se rige la cadena logística tienen aplicación multisectorial, no solo en situaciones de emergencia sino también en operaciones cotidianas que deben ser parte de la prevención y la preparación de los desastres.

La distribución de los suministros humanitarios que comprende, alimentos, materiales peligrosos, medicinas, voluntariado, médicos, etc. Tienen el objetivo de entregar la asistencia a las personas afectadas por el desastre o las organizaciones encargadas de su manejo, procurando que esta sea proporcional equitativa y controlada, para evitar los abusos y el desperdicio (Suministros, 1999), (Salud, 2001).

Aunque el término “logística” se refería a su origen a la técnica militar de transporte, provisión, y movimientos de tropas, hoy en día tiene aplicaciones prácticas en la vida civil. En general se concibe como un sistema en el que la interrelación de sus partes facilita la obtención de un objetivo de manera rápida y ordenada mediante la optimización ordenada de recursos. Esto implica que el éxito o la falla de uno de los segmentos repercuten en el resultado final.

La gran mayoría de compañías comerciales tienen, bajo uno u otro nombre, un departamento de logística que coordina bajo un sistema lógico y secuencial los aspectos relacionados con las compras, los trasportes, el mantenimiento, los inventarios, flujo de

materia prima y en general todas aquellas actividades auxiliares del proceso de producción y comercialización (Salud, 2001).

En operaciones de emergencia la logística es requerida para apoyar la organización e implementación de las acciones de respuesta para que estas sean no solo rápidas, si no también efectivas. La movilización del personal del equipo y del material necesario para el trabajo de las organizaciones que brindan asistencia y hasta las actividades relacionadas con la evaluación de heridos o la reubicación de poblaciones afectadas por el desastre, requieren de un sistema logístico para ser llevadas a cabo eficientemente (Salud, 2001).

Formulación Matemática de TSP

EL TSP puede ser definido como un grafo completo no dirigido $G= (V, E)$ si es simétrico o es grafo dirigido $G= (V, A)$ si es asimétrico. El conjunto $V = [1, \ldots, n]$ es un conjunto de vértices, $E=\{(i, j): i, j \in V, i<j\}$ es un conjunto de aristas y $A=\{(i, j): i, j \in V, i \neq j\}$ es un conjunto de arcos. La matriz $C= (C_{ij})$ es definida en E o A. La matriz satisface la desigualdad triangular cuando $C_{ij} \leq C_{ik} + C_{kj}$, para todos $_{i,j,k}$. Particularmente en estos problemas los vértices son puntos $p_i=(X_i,Y_i)$ en el plano, y $C_{ij} = \sqrt{(X_i + X_j)^2 + (Y_i + Y_j)^2}$ es la distancia euclidiana. La desigualdad del triángulo también se cumple si c_{ij} es la longitud de una ruta más corta de *i* a *j* en *G*.

Existen varias formulaciones de TSP disponibles en la literatura. En estudios recientes por (Orman & Williams, 2004), (Öncan, et al., 2009) puede ser referido para un análisis detallado. Una de las formulaciones más citadas en la literatura es la de (Dantzig, et al., 1954) para TSP. Esta formulación asocia una variable binaria x_{ij} con cada arista (i, j),

igual a 1 si y solo si la arista aparece en el viaje óptimo. La formulación para TSP es la siguiente:

Minimizar

$$\sum_{i<j} c_{ij} x_{ij} \tag{1}$$

Sujeto a

$$\sum_{i<k} x_{ij} + \sum_{j>k} x_{kj} = 2(k \in V) \tag{2}$$

$$\sum_{i,j \in S} X_{ij} \leq |S| - 1 \qquad (S \subset V, 3 \leq |S| \leq n-3 \tag{3}$$

$$X_{ij} = 0 \; or \; 1 \; (i,j) \in E \tag{4}$$

En la formulación de las restricciones (2), (3) y (4) se conocen como restricciones de grado, eliminación de restricciones e integridad de restricciones de un sub viaje respectivamente. En la presencia de la restricción (2), restricción (3) son algebraicamente equivalentes a las restricciones de conectividad.

$$\sum_{i \in S, j \in V \setminus S, j \in S} X_{ij} \geq 2 \quad (S \subset V, 3 \leq |S| \leq n-3) \tag{5}$$

La formulación se extiende fácilmente a un caso asimétrico (Dantzig, et al., 1954). Aquí x_{ij} es una variable binaria, asociada con el arco (i,j) igual para 1 si y solo si el arco aparece en el viaje óptimo. La formulación es la siguiente:

Minimizar

$$\sum_{i<\neq j} c_{ij} x_{ij} \tag{6}$$

Sujeto a

$$\sum_{j=1}^{n} X_{ij} = 1 \qquad (i \in V, i \neq j) \tag{7}$$

$$\sum_{i=1}^{n} X_{ij} = 1 \qquad (i \in V, j \neq j) \tag{8}$$

$$\sum_{i,j \in S} X_{ij} \leq |S| - 1 \qquad (S \subset V, 2 \leq |S| \leq n-2) \tag{9}$$

$$X_{ij} = 0 \; o \; 1 \quad (i,j) \in A \tag{10}$$

En nuestro problema el agente de ventas representaría el vehículo en que se llevan los suministros (medicinas, ropa, alimentos, etc.) que sale del centro de distribución, y las ciudades a visitar representa el número de los refugios temporales en las que hay que proporcionar ayuda. En este trabajo no se consideran los casos de carreteras bloqueadas, se asume que todos los refugios temporales están conectados por vías terrestres disponibles.

En la Figura 1 se muestra un problema del agente viajero con siete ciudades. La ciudad 1 es la ciudad de residencia del agente. Por lo tanto si se comienza desde su ciudad, el agente debe elegir una ruta para visitar cada una de las otras ciudades exactamente una

vez antes de regresar a su punto de partida. El número colocado junto a la ligadura entre cada par de ciudades representa la distancia entre estas ciudades.

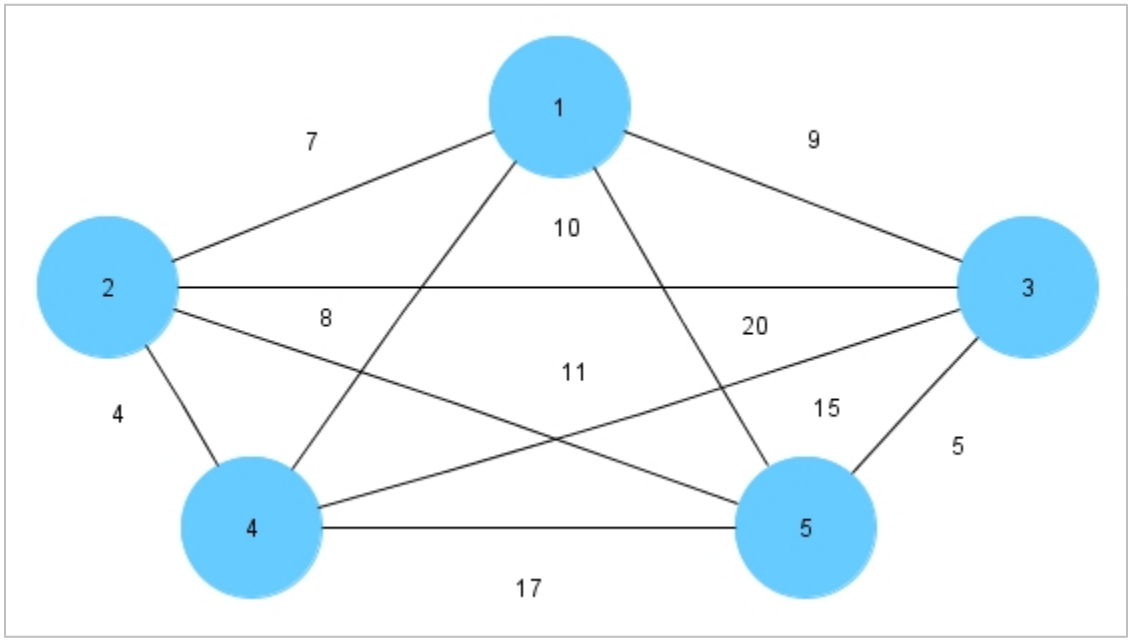


Figura 1. Ejemplo del problema del agente viajero (Hiller& Lieberman, 2010)

Ante tal situación, se propone un sistema de información que permita determinar la distancia mínima del centro de distribución a los refugios temporales en los que se proporcionará la ayuda utilizando la tecnología de Google Maps.

¿Cómo obtener la ruta terrestre que minimizará la distancia del centro de distribución a los refugios temporales para enviar suministros en situaciones de emergencia usando la tecnología de Google Maps?

Planteamiento del Problema

En esta investigación el problema se centra en entregar suministros del centro de distribución que es el lugar donde llegan todos los suministros para repartirlos equitativamente en el lugar donde ocurrió la emergencia (damnificados concentrados en refugios temporales en distintos puntos del lugar donde ocurrió la emergencia). Se concentra en suministros que tienen la misma prioridad; por ejemplo, medicinas, vacunas, personal médico.

Se busca distribuir ayuda desde el centro de distribución de donde saldrán los suministros hasta el lugar donde se necesita la ayuda tratando de llegar a los afectados con una herramienta que le permita tomar una decisión en planificación de una ruta que proponga un recorrido lo más corta posible para visitar todos los refugios temporales donde están concentrados los damnificados partiendo desde el centro distribución y regresando al punto de origen.

Las medidas y las decisiones que se toman las primeras 72 horas de una situación de emergencia sientan las bases de una respuesta de emergencia eficaz durante las seis a ocho semanas.

El problema descrito anteriormente corresponde a un problema de optimización combinatorial NP-Completo. Se dice que los problemas NP-Completo son los problemas más difíciles de NP (tiempo polinomial no determinista). No tienen un algoritmo en tiempo polinómico que los resuelva. No se ha podido demostrar formalmente que no exista, pero los matemáticos creen que realmente no existe. Este tipo de problemas es un subconjunto

de los problemas NP, es decir, existe un algoritmo polinómico que puede determinar si un valor es solución al problema (Duarte Muñoz, 2007).

En especial, se identifican con el clásico problema del viajante de comercio; en donde se trata de optimizar la distancia recorrida al pasar por una secuencia de nodos (stepways), y donde dicha secuencia forma un ciclo (camino) hamiltoniano.

El problema del agente viajero, en inglés *Traveling Salesman Problem* (TSP). Recibe este nombre porque puede describirse en términos de un agente de ventas que debe visitar cierta cantidad de ciudades en un solo viaje. Si comienza desde una ciudad de residencia, el agente determinará que ruta debe seguir para visitar cada ciudad exactamente una vez antes de regresar a su casa de manera que se minimice la longitud del viaje (Hillier & Lieberman, 2010). Este problema puede ser representado por un grafo, cuyos nodos representan cada ciudad y los arcos la distancia entre cada par de nodos; formando, de esta manera, un ciclo hamiltoniano.

Justificación

La implementación de un sistema de información que sea capaz de obtener las rutas, evaluarlas y definir la distancia más corta para la entrega de suministros ayudará a la mejor toma de decisiones para las instituciones civiles auxiliares en situaciones de emergencia.

Se logrará que la llegada de los suministros sea en un tiempo razonable, considerando que los grupos vulnerables no cuenten con medios para obtenerlos.

Las organizaciones que atienden emergencias, obtendrán una mejor planeación sobre el trazado de rutas de emergencia y hacer un uso eficiente de recursos.

Los afectados tendrán ayuda oportuna, en lo que se refiere a suministros (ropa, medicinas, recursos humanos, voluntarios, etc.).

La versión 3 del API de Google Maps está diseñada para cargarse de manera rápida y funcionar correctamente en dispositivos móviles avanzados, por ejemplo, los dispositivos que utilizan iOs y Android. Los dispositivos móviles tienen comportamientos diferentes a los de una PC de escritorio, como la ampliación y reducción de la imagen pellizcando la pantalla. La pantalla de navegador de un dispositivo móvil es más pequeña que la de los típicos navegadores de escritorio, por lo que, diseñar un sistema compatible con estos dispositivos de última generación es de suma importancia para garantizar la portabilidad del sistema.

Hipótesis

Es posible obtener rutas utilizando herramientas de programación en Google Maps, poder obtener la distancia total mínima recorriendo todos los refugios temporales con un algoritmo y mostrar este resultado gráficamente en un sistema de información modelado con Google Maps apoyando a la mejor toma de decisiones de las organizaciones que atienden emergencias.

Objetivo

Desarrollar un sistema de información Web con tecnología de Google Maps que determine la distancia mínima total para el reparto de suministros desde el centro de distribución hacia los refugios temporales donde está la emergencia utilizando un algoritmo del vecino más cercano.

Objetivos Específicos

1. Sistematizar la localización de la latitud y longitud (geo localización) de los puntos de distribución.
2. Obtener las distancias terrestres mediante Google Maps y guardarlas en una base de datos.
3. Definir la función de aptitud como problema del agente viajero.
4. Resolver la función de aptitud través del algoritmo del vecino más cercano.
5. Mostrar gráficamente la solución para recorrer todos los nodos en Google Maps.

Capítulo 1

Marco Teórico

1.1.El modelo Scrum

Scrum es una metodología de desarrollo muy simple, que requiere trabajo duro, porque la gestión no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto.

Scrum se considera como metodología de desarrollo ágil, definiendo la forma de abordar un proceso de desarrollo de software de forma ágil y liviana, a través de la descripción de un conjunto de roles, componentes y organización de la actividad diaria.

La base fundamental de esta metodología consiste en la división del trabajo completo (Product Backlog) en distintos apartados o bloques que pueden ser abordados en periodos cortos de tiempo (1-4 semanas) que se denominan Sprints.

Esta organización del proceso de creación de software permite potenciar los siguientes aspectos:

- Ágil: La división del trabajo en pequeñas unidades funcionales (*sprints*) permite mantener una política de entregas frecuentes de software que ofrecen una visión clara del estado del proceso y permite la introducción de modificaciones.

- Simple: Se centra especialmente en facilitar el desarrollo rápido, por lo que su complejidad (por ejemplo desde el punto de vista de la documentación a generar o de la organización de equipos) se ha tratado de reducir al máximo.

- Flexible: Todo el desarrollo se contempla como un ciclo de iteraciones continuas de desarrollo, lo que facilita la introducción de modificaciones "sobre la marcha", mejorando continuamente el proceso.

- Colaborativa: El planteamiento, desde el punto de vista de la organización del equipo, resulta bastante horizontal (en contraposición a una organización jerárquica férrea), otorgando a los miembros del equipo de desarrollo una elevado grado de autonomía y auto-organización de su trabajo.

1.1.1.Roles en la metodología Scrum

- *Product owner*: Es la persona responsable de transmitir al equipo de desarrollo la visión del producto que se desea crear, aportando la perspectiva de negocio. Representa al resto de interesados (*Stakeholders*, *clientes*, *directivos* etc.) en el desarrollo del producto. Sobre el *Product Owner* recae la responsabilidad de definir el conjunto de requerimientos (*Product Backlog*), de priorizarlos, y de finalmente validarlos.

- *Stakeholders*: Conjunto de personas que no forman parte directa del proceso de desarrollo pero que si deben ser tenidos en cuenta, por ser personas interesadas en el mismo, tales como directores, gerentes, comerciales etc. El *Product Owner* será el encargado de recoger sus opiniones y sugerencias y decidir si las aplica a la definición del proyecto (*Product Backlog*), así como

decidir si invita a alguna de estas personas al proceso de revisión de entregables.

- *Usuarios*: Al igual que los *Stakeholders* no forman parte del proceso de creación directamente (podrían estar en la fase de revisión de entregables si se considera necesario), son los destinatarios finales de la aplicación a desarrollar, el público objetivo del mismo. Una vez que la aplicación esté completada serán los que accedan a ella con mayor frecuencia.

- *Scrum Master*: En un contexto SCRUM el equipo de trabajo se auto-organiza y gestiona, por lo que no hay que confundir este rol con el de un jefe de proyecto. El papel principal del Scrum Master consiste en garantizar que el equipo de trabajo no tenga impedimentos u obstáculos para abordar sus tareas dentro del proyecto.

- *Equipo de Desarrollo*: Equipo responsable de desarrollar y entregar el producto. Mantiene una organización horizontal en la que cada miembro del equipo se auto-gestiona y organiza libremente en la definición y ejecución de los distintos sprints.

1.1.2. Componentes de la metodología Scrum

Definición del proyecto (*Product Backlog*): Consiste en un documento que recoge el conjunto de requerimientos que se asocian al proyecto. Es responsabilidad del *Product Owner* realizar esta definición y establecer las prioridades de cada requerimiento. Es un documento de alto nivel, que contiene descripciones genéricas (no detalladas), y que está sujeto a modificaciones a lo largo del desarrollo.

Definición del *Sprint* (*Sprint Backlog*): Un *sprint* debe entenderse como un subconjunto de requerimientos, extraídas del *product backlog*, para ser ejecutadas durante un periodo entre 1 y 4 semanas de trabajo. El *sprint backlog* sería el documento que describa las tareas que son necesario realizar para abordar los requerimientos.

Ejecución del Sprint: Sería el periodo de entre 1 y 4 semanas (periodo definido previamente en base a las tareas recogidas en el *sprint backlog*) durante el cual el equipo de trabajo abordaría las tareas de desarrollo correspondientes. Una vez iniciada la ejecución de un *sprint* definido, este no podrá ser modificado, y en caso de ser necesario introducir cambios estos se harán una vez concluido el periodo a través de la definición de otro *sprint backlog*.

Entrega: Una vez concluida la ejecución del *sprint*, se dispondrá de una porción de la aplicación potencialmente definitiva.

Evolución del proyecto (*Burn down*): Es un documento que refleja el estado del proyecto, indicando el volumen de requerimientos que en ese momento se encuentran pendientes de ser abordados (en el *product backlog*), los requerimientos que en ese momento se están desarrollando (*sprint backlog*) y los requerimientos cuyo desarrollo ya se ha completado en su totalidad.

1.1.3.Reuniones de trabajo en un contexto SCRUM

- Planificación de *sprint*: Se realiza al principio de cada ciclo de sprint, y está encaminada a seleccionar el conjunto de requerimientos del *product backlog*, que serán abordado, el equipo de trabajo que será necesario y el tiempo que se estima (entre 1 y 4 semanas) para su desarrollo.

- Reunión diaria: Conocida como *daily scrum*, se realiza al comienzo de cada día en que ese esté ejecutando un *sprint*. Es una reunión corta (no más de 30 minutos) en la que los integrantes del equipo responden las siguientes preguntas:

    1. ¿Qué has hecho desde la última reunión?
    2. ¿Qué problemas has encontrado para realizar el trabajo previsto?
    3. ¿Qué planeas hacer antes de la próxima reunión?

Revisión de *sprint*: Una vez concluido el ciclo de *sprint* se mantiene una reunión en la que se define qué parte del trabajo previsto se ha completado y qué parte permanece pendiente. En cuanto al trabajo completado se realiza una revisión (demo) del mismo al *product owner* y otros usuarios que pudiesen estar involucrados.

Retrospectiva de *sprint*: Es una reunión en la que todos los miembros del equipo realizan una valoración del trabajo realizado en el último *sprint*, identificando puntos de mejora de cara a los siguientes a realizar. El objetivo principal es introducir un componente de mejora continua en el proceso (Palacio, 2008) .

1.2. Google Maps

Google maps es un servicio de mapas al que puedes acceder desde cualquier navegador web. Se pueden consultar mapas básicos o personalizados, así como información sobre empresas locales como, por ejemplo, su ubicación, su información de contacto o indicaciones para llegar en coche (Google, 2013).

1.2.1. API (Interfaz de Programación de Aplicaciones) de JavaScript de Google Maps Versión 3

El API de JavaScript de Google Maps permite insertar Google Maps a las páginas Web. Esta versión está especialmente diseñada para proporcionar una mayor velocidad y que se pueda aplicar más fácilmente tanto a móviles como a las aplicaciones de navegador de escritorio tradicionales. El API proporciona diversas utilidades para manipular mapas y para añadir contenido al mapa mediante diversos servicios, permitiendo crear sólidas aplicaciones de mapas en un sitio Web, la documentación sobre el uso de esta herramienta se puede ver en (Google, 2013).

1.3. Estado del Arte

1.3.1. Métodos de Solución para TSP

TSP puede resolverse de diferentes maneras:

- Enumeración de todas las soluciones factibles: es decir, enlistar todas las posibles soluciones al problema, calcular sus costos asociados e identificar por comparación, cual es la solución con un costo más conveniente.

- Métodos exactos. También llamados algoritmos óptimos intentan descartar familias enteras de posibles soluciones, tratando así de acelerar la búsqueda y llegar una óptima. Uno de los que más se usan para resolver TSP es Ramificación y Acotación

- Heurísticas. Son métodos que obtienen una solución en tiempos muy cortos aunque no garantizan la optimización de la solución. Buscan una solución en un tiempo de computo razonable, por esto cuando una instancia de grandes dimensiones se resuelve con algún método exacto toma un extenso periodo de tiempo. Con el uso de heurísticas se obtienen soluciones de buena calidad en tiempos de cómputo más pequeños.

1.3.2.Métodos Convencionales para TSP

Las propuestas para resolver el problema del viajante de comercio son muchas. Algunas de ellas están basadas en programación dinámica, métodos branch and bound. Existen otras opciones más rápidas son los métodos aproximados como; recocido simulado, búsqueda del tabú, redes neuronales, algoritmos evolutivos, las cuales no aseguran el hallazgo de la solución óptima.

A continuación se describen algunos de estos métodos.

1.3.2.1.Branch and Bound

Branch and Bound es un método de búsqueda general, a grandes rasgos funciona de la siguiente manera. En cada paso, el conjunto de todas las soluciones posibles se divide en dos o más subconjuntos las cuales son representadas por ramas en un árbol de decisión. Para el TSP, se dividen todos los viajes en subconjuntos si contiene un borde dado o no. El enfoque se vuelve más útil cuando se añade una idea adicional: en cada paso, se calculan las cota inferiores para el peso de todas las soluciones con respecto a subárboles (usando una relajación adecuada) y comparar un límite superior calculado previamente (por ejemplo la longitud de un buen viaje encontrado por una heurística seguido de post-optimización).

Entonces ninguna rama del árbol para que el límite inferior excede el límite superior posiblemente puede conducir a un viaje óptimo, de modo que la rama puede ser descartada; poco frecuente, un gran número de recorridos serán excluidos (Jungnickel, 2008).

La calidad de este método dependerá de los criterios de ramificación y relajación, solo puede ser juzgado heurísticamente, no se puede esperar ninguna garantía de rendimiento.

1.3.2.2.Recocido Simulado (Simulated Annealing)

Es una opción más compleja para la optimización combinatoria. Está basado en el proceso de enfriado de metales. El termino *annealing* se relaciona con la manera en que los metales en estado líquido son enfriados lentamente para asegurar un estado de energía mínima. Se puede decir que el algoritmo *simulated annealing* enfría la solución lentamente hasta alcanzar el objetivo posible (Ilog, 2010). Esto es, en cada una de las evaluaciones de solución perteneciente al vecindario siempre toma el movimiento que mejora la solución.

Sin embargo permite hacer movimientos que incrementen el costo de la solución basado en una función de probabilidad. La facultad de realizar movimientos hacia arriba permite escapar de un mínimo local; explorando así en forma extensiva el espacio de búsqueda. En los pasos descendientes no existe un mecanismo real para salir del mínimo local. Algunas técnicas para evitar quedar atrapados en estos puntos, incluyen el incremento del tamaño del vecindario y el uso del muestreo aleatorio pero esto no siempre es completamente satisfactorio.

La función de probabilidad depende del cambio en el costo de un movimiento candidato y de la temperatura actual del sistema. La temperatura es análoga a la

temperatura en el proceso de enfriado físico. Es decir la temperatura se reduce a través de proceso de enfriado. A altas temperaturas cualquier movimiento es permitido, lo que posibilita una extensa exploración del espacio de soluciones; mientras que con las bajas se aceptan pocos movimientos hacia arriba. La idea detrás de la reducción de la temperatura, es más tarde sobre el proceso de enfriado simplemente se explote el vecindario de una solución óptima local. La temperatura se reduce a través de una planificación adecuada de enfriamiento.

La Figura 2 presenta el algoritmo correspondiente al procedimiento *simulated annealing*.

```
Procedure Simulated Annealing
begin
 n ← 0;
 inicializar temperatura t;
 seleccionar aleatoriamente un string actual T;
 evaluar T;
 repeat
  repeat
  Seleccionar un nuevo string T en el vecindario de T al mutar un único bit de T;
  if f(T) < f(T')
   then T ← T';
   else if random [0,1]< exp {( f(T')-f(T)/t}
         then T ← T';
 until (condición de terminación)
 t ← g(t,n);
 n ← n+1;
 until (criterio de detención);
end
```

Figura 2. Procedimiento Simulated Anneling

1.3.2.3.Búsqueda del Tabú

El concepto básico de Búsqueda del Tabú (*Tabu Search*) es descrito por Grover en 1986 (Glover, 1986) como una meta-heurística impuesta sobre otra heurística. El objetivo es evitar ciclos al prohibir o penalizar movimientos de la solución, a puntos en el espacio de solución previamente visitados "tabú". El método de búsqueda del Tabú ha sido motivado por la observación del comportamiento humano con un elemento aleatorio que conduce a un comportamiento inconsciente en circunstancias similares. Como Grover señala, la tendencia resultante para desviarse desde un curso programado, debe ser visto como una

fuente de error pero también puede ser probada como fuente de beneficios. La Búsqueda del Tabú procede de acuerdo a la suposición de que no existe una restricción para aceptar una nueva (pobre) solución, a menos que la restricción evite un camino ya investigado. Esto asegura que nuevas regiones del espacio de solución, del problema serán investigadas con el objetivo de evitar mínimos locales y encontrar la solución deseada.

Este procedimiento guía a una heurística de escalamiento descendiente con el objetivo de continuar la exploración evitando un retroceso a un óptimo local del cual ya se ha salido. En cada iteración se aplica un movimiento admisible a la solución actual, transformándola de esta manera en una solución vecina con menor costo, mientras que el movimiento inverso está prohibido para algunas iteraciones con el objetivo de evitar ocurrencia de ciclos.

Durante el proceso de búsqueda de movimientos son almacenados en una lista Tabú, representando la memoria de los pasos previos del algoritmo. Se cuenta con un proceso para determinar cuando las restricciones Tabú pueden ser sobrescritas.

En muchos casos, las diferencias entre las distintas implementaciones del método Tabú están relacionadas con el tamaño, variabilidad y adaptabilidad de la memoria Tabú y un dominio particular. La Figura 3 presenta el procedimiento del algoritmo de Búsqueda del Tabu.

Dada la flexibilidad inherente al método de búsqueda del Tabú existe una gran variedad de trabajos que propusieron para TSP como (Glover, 1986), (Glover, 1991), (Knox & Glover, 1989), (Knox, 1994). No está claro cuál de estos mecanismos es

preferible todos ellos requieren un tiempo de ejecución de n*n*n (n^3) y es poco probable que sean prácticos para instancias de gran tamaño.

```
Procedure Tabu-Search
begin
 x← solución inicial factible;
 inicializar nmax con número máximo de iteraciones;
 mejor solución ← x;
 inicializar el número de iteraciones n=0;
 inicializar la lista Tabu;
 repeat
  elegir un movimiento no tabu Δx(n+1);
  x(n+1) ← x(n) + Δx(n+1);
  if (fobj(x(n+1)) > fobj(major solución))
    then major solución ← x(n+1);
  actualizar lista Tabu;
  n ← n+1;
until (n=nmax o cualquier movimiento posible de la solución actual es Tabu);
end
```

Figura 3. Procedimiento Búsqueda del Tabú.

1.3.2.4.Redes Neuronales

Las redes neuronales se definen como colecciones de procesadores paralelos conectados entre sí en forma de un grafo dirigido, organizado de tal modo que la estructura de la red sea la adecuada para el problema que se esté considerando. La mayoría de modelos de redes neuronales necesitan de una colección de ejemplos representativos de la traducción

deseada. La red neuronal se adapta, al entrenarse para reproducir las salidas deseadas cuando se le presentan las entradas dadas como ejemplo (Russell & Norvig, 2011).

La red neuronal es robusta en el sentido de que responderá con alguna salida; incluso en el caso que se presente una entrada que no haya visto nunca.

El proceso de entrenamiento es una codificación de la información acerca del problema que hay que resolver, y que la red invierte la mayor parte de su existencia productiva siendo ejercitada una vez que concluye el entrenamiento; entonces se descubre un método para permitir que los sistemas automatizados evolucionen sin ser reprogramados explícitamente.

Cuando se describen las bases matemáticas de los modelos de redes, suele resultar útil que la red es un sistema dinámico; esto es, un sistema que evoluciona a lo largo del tiempo.

El proceso de aprendizaje consiste en hallar los pesos que codifican el conocimiento que se desea que el sistema aprenda. Para la mayor parte de los sistemas reales, no es fácil determinar una solución en forma cerrada para este sistema de ecuaciones.

Otra representación de una red neuronal para TSP es geométrica, donde las neuronas pueden ser vistas como un conjunto de $M \geq N$ en el plano, inicialmente ubicados como los vértices de un polígono regular, de M lados, en el método de la instancia. El objetivo es mover iterativamente estos vértices hacia las ciudades, esto significa deformar el polígono. Este proceso continua hasta que el polígono luzca como un viaje, y cada ciudad está representada por un vértice, los vértices que no representen a ninguna cuidad están situados en una línea recta entre otros dos, que si representan ciudades.

1.3.2.5.Algoritmo Lin-Kernigham

Es una variante de la búsqueda local en profundidad (Martin & Otto, 1994). Comienza con estructurar el conjunto de todas las soluciones factibles (rutas). Se define una ruta, T obteniendo las rutas correspondientes al intercambiar k arcos de T. Se inicia con una ruta aleatoria T_1 y se construye una secuencia de rutas T_1, T_2, T_3,..Tn. En otras palabras, cada tour es obtenido (Lin, 1965) desde uno previo al realizar k-cambios, por ejemplo, al eliminar K enlaces y reconectar las partes finales desprendidas para obtener una ruta. Los k- cambios son necesarios para decrementar la longitud del tour.

Cuando el proceso se detiene en una ruta para el cual no hay posibles mejoras después de k-cambios, la ruta es k-opt. Lin ha introducido y estudiado el caso de k=2 y k=3 y han mostrado que se pueden obtener rutas bastante buenas rápidamente. Para encontrar una ruta óptimo globalmente sugiere repetir la búsqueda desde varios puntos de inicio aleatorios. Posteriormente Lin y Kernighan (Lin & Kernighan, 1973) (Martin & Otto, 1994) han realizado una mejora a este algoritmo, la cual consiste en una búsqueda local en ancho 3-Opt (Análisis que consiste en eliminar tres conexiones o bordes en una red, reconectarlas de todas las maneras posibles y evaluar cada reconexión para encontrar el óptimo. Este proceso se repite para un conjunto diferente de tres conexiones) seguida de una búsqueda local en profundidad.

1.3.2.6.Heurísticas

Las heurísticas son métodos inteligentes que buscan una buena solución en un tiempo de cómputo razonable, pero sin garantizar que esta sea óptima (Johnson , et al., 2002).

Existen diferentes tipos de heurísticas (Glover, et al., 2001):

- Heurísticas constructivas: Procedimiento que se encarga de obtener una solución a partir de un criterio inicial, esto es, construyen una solución factible.

- Heurística de búsqueda local: Procedimientos para mejorar soluciones ya encontradas. Tratan de optimizar localmente alrededor de una solución, ubicando mínimos locales.

- Heurísticas combinadas: Procedimientos que constan de una heurística constructiva y una heurística de búsqueda local.

1.3.2.6.1. El vecino más cercano (Closest neighbor heuristic)

Esta heurística constructiva se basa en la idea de moverse de una ciudad a la siguiente, de tal forma que todas las opciones sean visitadas, la ciudad elegida será la más cercana a donde se encuentra ubicado el agente viajero (Figura 4). Es una heurística miope ya que en alguna iteración se podría escoger la mejor opción que tiene disponible, sin ver que esta pueda obligar a tomar malas decisiones posteriormente (Johnson , et al., 2002).

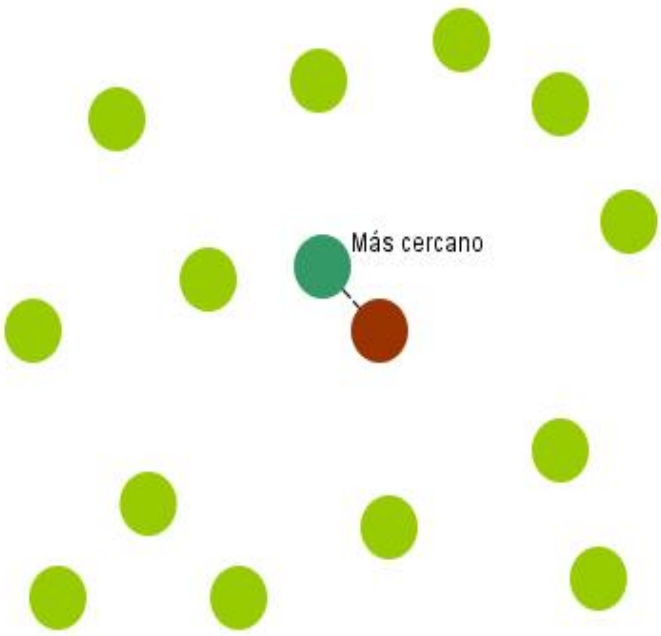


Figura 4. En la heurística del vecino más cercano, el agente viajero se mueve a la ciudad más próxima a la donde se encuentra.

Esta heurística TSP es sencilla y directa. La clave de este enfoque es visitar siempre la ciudad más cercana. La complejidad polinomial asociada con esta heurística es n*n (n^2). La aproximación más cercana es muy similar al algoritmo de árbol de expansión mínimo. Los pasos del vecino más cercano se presentan como (Johnson & McGeoch, 1995):

1. Selecciona una ciudad aleatoriamente.
2. Encuentre las más cercana ciudad sin visitar e ir allí.
3. ¿hay ciudades sin visitar? Si es así repita el paso 2

Dentro de la literatura nos encontramos trabajos relacionados con el problema de TSP, las variaciones de TSP y sus algoritmos de solución. Por otra parte se tienen las aplicaciones de estos algoritmos que dan solución a problemas reales.

Métodos que solucionan el problema de TSP y sus variaciones

La forma general de TSP fue formulada en las universidades de Viena Y Harvard. Más adelante, el problema se retomó de la mano de Hassler Whitney y Merril M. Flood en la universidad de Princeton.

Una forma más directa de encontrar la solución exacta sería formular todas las posibles permutaciones, siendo el número de soluciones posibles *n!*. Esta posibilidad se convierte en impracticable a medida que aumenta el número de variables. En (Applegate, et al., 2006), propone *ramificación y acotamiento* para la resolución de TSP basado en programación lineal con un buen funcionamiento hasta con 200 ciudades o bien combinar varias opciones. En (Cartel & Ragsdale, 2006) tratan el m-TSP (Multiple Traveling Salesman Problem), el cual costa de $m>1$ vendedores para visitar $n>m$ localidades, las cuales deben ser visitadas solo una vez, minimizando el recorrido o es coste total, resuelven el problema mediante la técnica de *Algoritmos Genéticos* probando un nuevo cromosoma no estándar ("*Two-part- chromosome*"). Comparando el desempeño computacional de su propuesta con enfoques convencionales de estudios previos con cromosomas estándar, encuentran en algunos casos mejores soluciones para el m-TSP, con instancias hasta de 150 ciudades. Por otra parte (Snyder & Daskin, 2006) tratan el problema del vendedor viajero generalizado (GTSP), en el que existen clusters o grupos definidos y el viajero debe visitar al menos un nodo de cada cluster, minimizando el costo total del viaje. Presentan una

heurística que combinan *algoritmos genéticos de llaves aleatorias* con una *búsqueda local*. De 41 problemas de prueba con distancias simétricas y hasta 442 nodos, encuentran el valor óptimo en la mayoría de los casos, y en los demás, las soluciones estuvieron dentro del 1% del óptimo, con una eficiencia computacional dentro de los 10 segundos. En (Campbell, 2006) se orienta hacia PTSP (problema del vendedor viajero probabilístico); en este caso solo un subconjunto de clientes debe ser visitado, siendo el número de clientes a visitar una variable aleatoria. Ante tal dificultad de resolver esta variante para instancias de tamaños propios de casos reales, una de las alternativas es asignar clientes a regiones y resolver para una instancia reducida. Es así que la forma de dividir a los clientes y la escala necesaria para representar razonablemente la función objetivo, es la interrogante que abordaron los autores. Presenta resultados computacionales en problemas hasta de 1000 ciudades, donde las probabilidades de visitar a los clientes son bajas y el tiempo computacional disponible es reducido; es decir juega un papel preponderante la eficiencia computacional. Manifiestan que la agregación de clientes en regiones puede proporcionar estimaciones cercanas al objetivo de manera rápida y, resolver el primer problema (reducido), puede conducir a soluciones satisfactorias a menor costo.

En (Nguyen, et al., 2007) hacen una implementación de un *algoritmo genético con participación de un método de búsqueda local* basado en la *heurística LK* (Lin & Kernighan, 1973) para encontrar soluciones de alta calidad en instancias TSP de gran escala. En (Liu, 2007) se enfoca en PTSP desarrolla un *algoritmo de búsqueda dispersa hibrido*, que incorpora, entre otros, la regla del *vecino más cercano*, aceptación por *umbrale*s y el *operados de recombinación de bordes* (ER). Destacan la potencialidad de la propuesta para resolver problemas de gran tamaño, mostrando que la incorporación de

umbral de aceptación dentro de la búsqueda dispersa favorece la eficiencia del procedimiento manteniendo la calidad de la solución. En (Carrabs, et al., 2007) se enfocan en el TSPPDL, problema del vendedor viajero con recogida y entrega de mercancía y restricciones de carga y descarga LIFO (últimas en entrar, primeras en salir). Para referir a este problema es necesario el "Traveling Salesman Problem with Pickup and Delivery" (TSPPD), que consiste en determinar el recorrido de longitud mínima que se requiere para que un solo vehículo, con capacidad de carga limitada, satisfaga los requerimientos de los clientes, a los cuales debe recogerles la mercancía en un determinado sitio de origen, para su posterior entrega en el sitio destino. En el TSPPDL se agregan restricciones de precedencia que incorporan la política LIFO al cargar y extraer la mercancía. Proponen tres nuevos operadores de búsqueda local, los cuales se incorporan con la heurística de búsqueda de entorno variable, obteniendo resultados satisfactorios en eficiencia y eficacia en comparación con el método de prueba: tipo búsqueda entorno variable descendente (VND) de (Cassani & Righini, 2004).

En (Ohlmann & Thomas, 2007) orientan sus esfuerzos hacia TSPTW (Travelling Salesman Problem with Windows Time), en el cual deben considerar restricciones de tiempo para las visitas a los clientes, llevando a que cada ciudad debe ser visitada en un tiempo determinado, no permitiendo llegar después de dicho instante; igualmente, si el viajero llega antes, debe esperar que se cumpla el tiempo preestablecido. Usan una variante de recocido simulado, denominado, *Compressed-Annealing,* la cual relaja las restricciones de ventanas de tiempo e incorporan el método de penalización de variable asociado al concepto de presión. Bajo instancias hasta de 200 ciudades, el método propuesto satisfactorio en la mayoría de los casos. En (Duan & Yu, 2007) resaltan la robustez de la

optimización bajo *colonia de hormigas* y su facilidad para integrarse con otros métodos. Los autores combinan *colonia de hormigas* con *algoritmos meméticos*. Al final proponen un nuevo enfoque y lo prueban con un TSP de 51 ciudades, mostrando eficacia y viabilidad práctica.

En (Savla, et al., s.f.) hacen una aproximación del TSP a la robótica, enfocado en el enrutamiento de vehículos tipo Dubins; tal caso es denominado por los autores como DTSP. Destacan que el TSP y sus variaciones continúan atrayendo esfuerzos de diferentes disciplinas, como son, matemáticas, ciencias de la computación e investigación de operaciones. El DTSP es similar al TSP, puesto que el vehículo debe visitar una serie de puntos, una sola vez y regresar al origen, pero con las características que deben considerarse restricciones asociadas a las trayectorias de curvatura que sigue el vehículo, delimitadas en el plano. Proponen, entre otros, un algoritmo llamado "*alternating algorithm*", al cual incorporan dentro de la categoría "*algoritmo de aproximación de factor constante*".

Aplicaciones de los algoritmos que solucionan TSP

Hung-Chang Lee aplica el algoritmo Búsqueda del Tabú para planificar el orden de visitas que los trabajadores tienen que planear, definen el problema como TSP con Google Maps para ayudar a los trabajadores a planificar sus visitas (Hung-Chang Lee , 2011). Por otra parte, (Fajardo & Oppus, 2009) proponen un aplicación móvil usando un algoritmo genético para el rescate y socorro de zonas afectadas por un desastre en Filipinas usando teléfonos con sistema operativo Android, la aplicación se llama MyDisasterDroid que determina la ruta óptima a lo largo de diferentes localizaciones geográficas que el

voluntarios y socorristas deben tomar con el fin de servir a la mayoría de las personas y proporcionar una cobertura máxima de la zona menor tiempo posible. Se implementó el algoritmo genético para la obtención de la ruta óptima, sin embargo no menciona el número de ciudades que soporta, cabe mencionar que no todas las instituciones que se encargan de emergencias no cuentan con teléfonos inteligentes.

Capítulo 2

Propuesta Metodológica

Para describir la propuesta metodológica que se uso es necesario describir los requerimientos del sistema, el diseño y la implementación de la heurística del vecino más cercano**.**

2.1. Descripción del sistema:

Desarrollar un sistema de información Web con tecnología de Google Maps que determine la distancia mínima total para el reparto de suministros desde el centro de distribución hacia los refugios temporales donde está la emergencia utilizando un algoritmo del vecino más cercano.

Por el momento, la herramienta del software solo debe tener acceso a un administrador general. Sin embargo el administrador de software puede crear perfiles de usuario para los diferentes módulos que contendrá el sistema. La Tabla 1, muestra la tabla de actividades para el administrador del sistema.

Tabla 1. Actividades para Administrador del Sistema

Administrador General
1. CRUD de Centros de Distribución
2. CRUD de Centros de Refugios Temporales
3. Sistematizar la localización de la latitud y longitud (geo localización) de los puntos de distribución.
4. Obtener las distancias terrestres mediante Google Maps y guardarlas en una base de datos.
5. Definir la función de aptitud como problema del agente viajero.
6. Resolver la función de aptitud través del algoritmo del vecino más cercano.
7. Mostrar gráficamente la solución para recorrer todos los nodos en Google Maps.

2.2. Arquitectura del sistema

El sistema está basado en la arquitectura cliente-servidor, en la que los usuarios del sistema Web se conectan mediante el uso de un navegador Web.

La arquitectura cliente-servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores y los demandantes llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, que le da respuesta. La Figura 5 muestra la arquitectura del sistema.

Se pueden diferenciar 7 módulos distintos:

- Módulo de Google Maps: Se encuentra todo lo relacionado con la API de Google Maps, así como el servidor de Google Maps al cuál se le harán las peticiones para la geo codificación de los puntos de latitud y longitud, cálculo de rutas, visualización de mapas, etc. El motor de la API de Google Maps es consultado con AJAX.

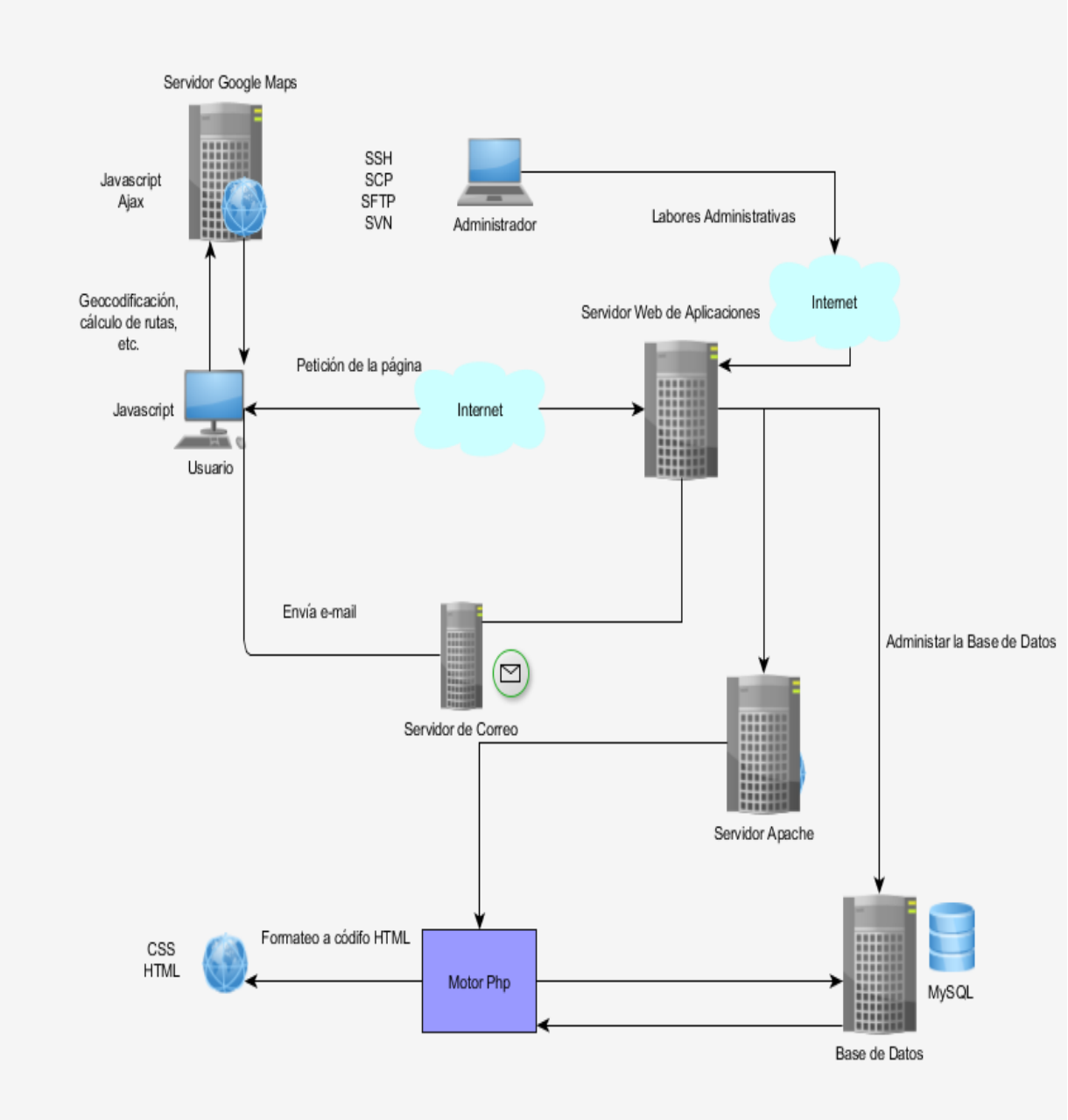


Figura 5. Arquitectura del Sistema

- Módulo de usuarios: Es el entorno donde el cliente accede a nuestra Web. El navegador que use, tiene que tener habilitado JavaScript para su ejecución.
- Módulo del Administrador: Son las tecnologías que emplean los administradores para conectar al servidor y hacer tareas oportunas. En concreto se usa SSH para

conectar al servidor, así como SCP y SFTP para trasferir archivos y colocarlos en el servidor.

- Módulo Apache: Se utiliza el servidor Web apache para el desarrollo del proyecto.
- Módulo de Base de Datos: La base de datos se encuentra en el gestor MySQL.
- Modulo Motor de PHP: El lenguaje de programación que se usó es PHP en el lado del servidor web.

2.3. Patrones de Diseño

Esta fase se modela el diseño del sistema en base a su arquitectura y sus requerimientos. En concreto los patrones de diseño que hemos empleado son MVC (Modelo Vista Controlador).

2.3.1. Modelo Vista Controlador

Este patrón de diseño es uno de los más comunes y usados en aplicaciones Web, debido a su idoneidad en este tipo de aplicaciones. Este patrón sigue un modelo que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. Estos tres componentes pueden asociar naturalmente: la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo en el sistema gestor de base de datos y la lógica de negocio, y el controlador es responsable de recibir los eventos de entrada desde la vista y comunicar con el modelo. En la Figura 6 se muestra la arquitectura MVC.

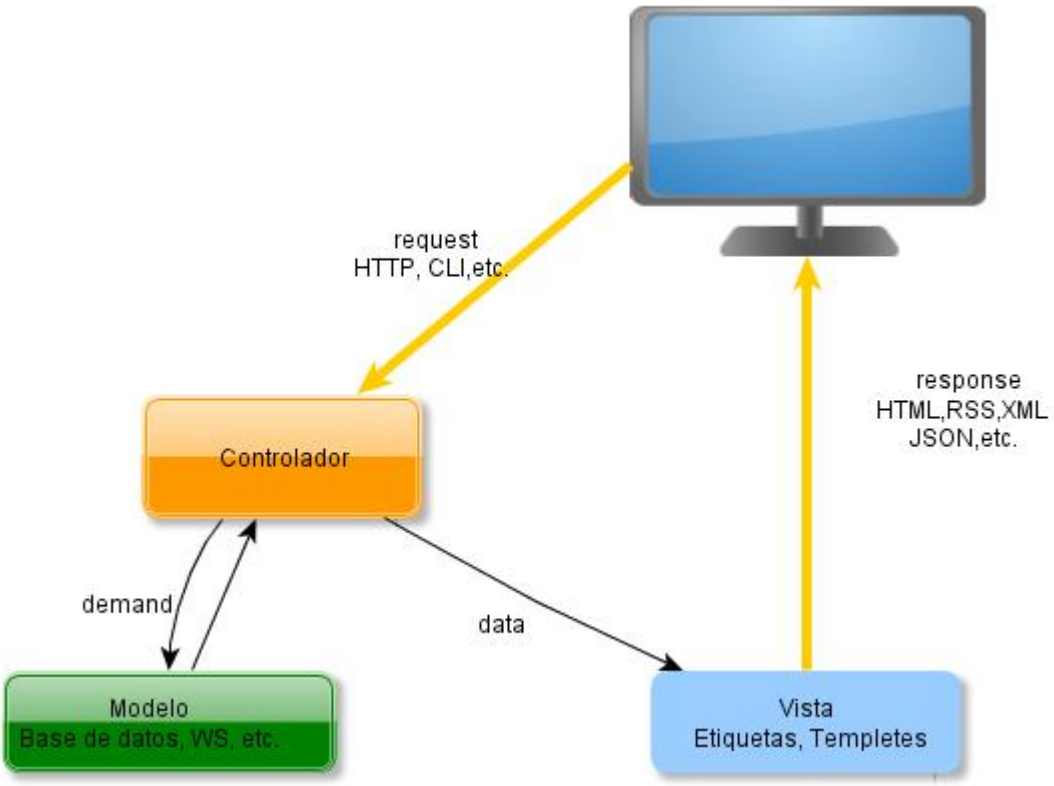


Figura 6. Arquitectura MVC

2.4. Proceso de desarrollo

2.4.1. Metodología

Dado que el tamaño del proyecto es pequeño utilizamos la metodología ágil *Scrum* (Palacio, 2008), agregando a los roles un *tester*. Y en la parte de desarrollo se utilizaron diagramas *UML* como apoyo al *product backlog*.

2.4.1.1.Roles asignados

Tabla 2. Asignación de roles

Rol	Nombre	Identificación
Gerente del Proyecto	Moisés Gómez Díaz	MGD
ProductOwner	Moisés Gómez Díaz	MGD
ScrumMaster	Moisés Gómez Díaz	MGD
Equipo:	Moisés Gómez Díaz	MGD MGD MGD MGD
Tester	Moisés Gómez Díaz	MGD

2.4.1.2.Ciclos del Proyecto

Se trabajaron 6 sprint cada sprint en un tiempo de desarrollo con tres días de holgura distribuyendo las cargas de trabajo como se muestra en la de descrito en la Tabla 3

Tabla 3. Cargas de trabajo

Equipo	Sprint	Fecha de desarrollo
MGD	Administración	4, 5, 6,7,8,11
MGD	Sistematizar la localización de la latitud y longitud (geo localización) de los Refugios temporales	18, 19,20,21,22
MGD	Obtener las distancias terrestres mediante Google Maps y guardarlas en una base de datos.	25, 26,28, 29
MGD	Definir el problema del agente viajero	2,3,4, 5,6
MGD	Resolver través del algoritmo del vecino más cercano.	9, 10,11,12,13,16,17
MGD	Mostrar gráficamente la solución obtenida recorriendo todos los nodos en Google Maps.	18,19,20,23,24,25,26

1. ProductBacklog - una junta general de trabajo para diseño de base de datos, clases, generación de artefactos de software (casos de uso, componentes, clases).
2. Una vez obtenido el modelado y el entendimiento del sistema, el desarrollador genera el modulo asignado.
3. Se programaron juntas diarias de 15 minutos para observar el avance de cada sprint y resolución de dudas.
4. Se integró el sistema en máximo 2 días tomando la holgura.

5. Se tomaron dos días de pruebas.

La Tabla 4 representa el calendario general del ciclo del proyecto

Tabla 4. Calendario General

2012	L	M	M	J	V
Septiembre	4	5 △	6	7	8
Septiembre	11 ☆	12 △	13 ☆	14	15 ☆
Septiembre	18	19 △	20	21	22 ☆
Septiembre	25	26 △	27	28	29 ☆
Octubre	2	3 △	4	5	6 ☆
Octubre	9	10 △	11	12	13
Octubre	16	17 △ ☆	18	19	20
Octubre	23	24 △	25	26	27
Noviembre	30	31 △	1	2 ☆	3
Análisis Diseño e implantación					
Integración del sistema					
△ **Junta semanal** ☆ **Prueba por modulo**					

La Tabla 5, Tabla 6, Tabla 7, Tabla 8, Tabla 9 y Tabla 10 representan el detalle de subconjuntos de requerimientos (*sprint*) extraidos del documento que recoge el conjunto de requerimientos asociados al proyecto (*product backlog*). Los *sprint* al igual que los gráficos *Burn-Up* representados por la Figura 7, Figura 8, Figura 9, Figura 10, Figura 11 y Figura 12 de acuerdo a la metodología *Scrum* para el desarrollo del Sistema.

Tabla 5. Detalle del *sprint* 1

N° de sprint	**1**
Identificación	CRUD de Administración
Descripción de la funcionalidad	Pantalla de acceso al sistema CRUD de usuarios para el rol de administrador general del sistema
Campo	Pantalla de acceso: Usuario y clave CRUD de usuarios Nombre de usuario Clave de usuario Rol: Administración General
Modulo del sistema al que pertenece	Administración
Personas asignadas	MGD
Estimación	4 días con 2 días de holgura

Para el sprint de CRUD de administración se estimaron 6 días

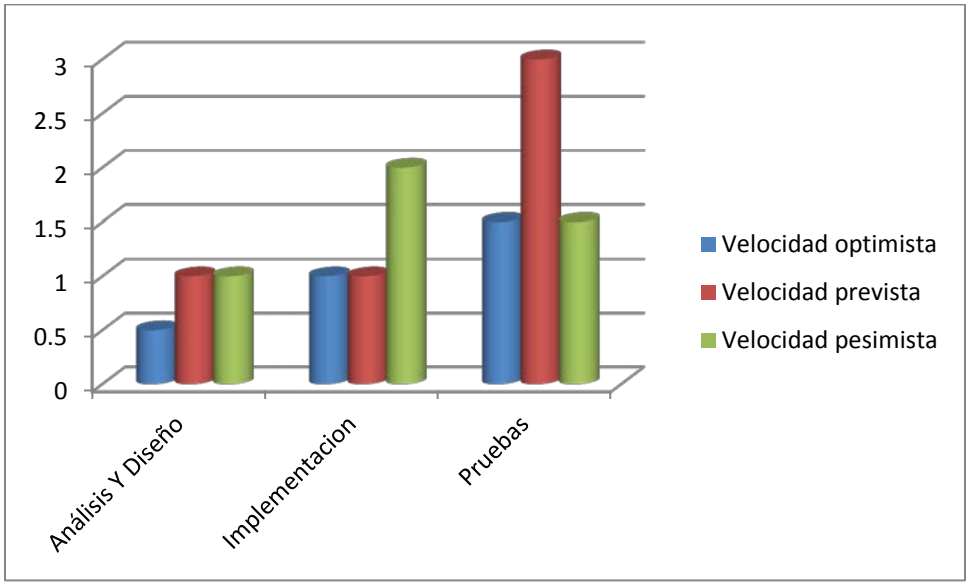


Figura 7. Grafico *Burn-Up* sprint CRUD de administración

Tabla 6. Detalle del *sprint* 2

N° de sprint	**2**
Identificación	Sistematizar la localización de la latitud y longitud (geo localización) de los puntos de los Refugios Temporales
Descripción de la funcionalidad	Geo localización de latitud y longitud a través de Google Maps. El usuario podrá marcar los refugios temporales a través de un Mapa.
Campo	Longitud y latitud
Modulo del sistema al que pertenece	Sistematizar la localización de la latitud y longitud (geo localización) de los puntos de los Refugios Temporales.
Personas asignadas	MGD
Estimación	3 días con dos días de holgura

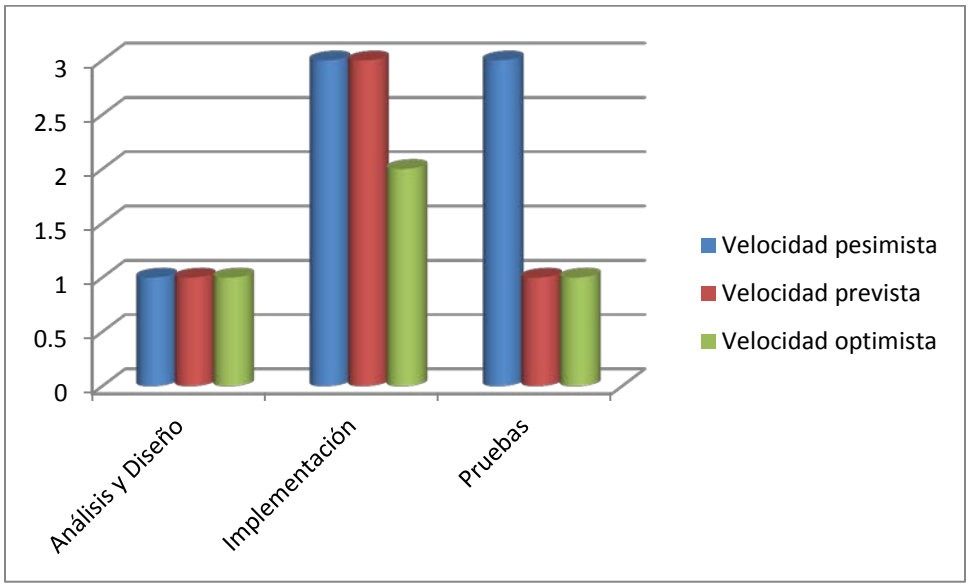


Figura 8. Grafico *Burn-Up* sprint Geo localización

Tabla 7. Detalle del *sprint* 3

N° de sprint	**3**
Identificación	Obtener las distancias terrestres mediante Google Maps y guardarlas en una base de datos.
Descripción de la funcionalidad	Obtener las distancias terrestres de cada uno de los refugios temporales. Sacar las combinaciones de distancias n*m. Obtener de todos refugios temporales sus distancias del nodo n al nodo m.
Campo	Id distancia Latitud y longitud del refugio origen Latitud y longitud del refugio destino Distancia Dirección origen Dirección de Destino
Modulo del sistema al que pertenece	Obtener las distancias terrestres mediante Google Maps y guardarlas en una base de datos.
Personas asignadas	MGD
Estimación	2 días con dos días de holgura

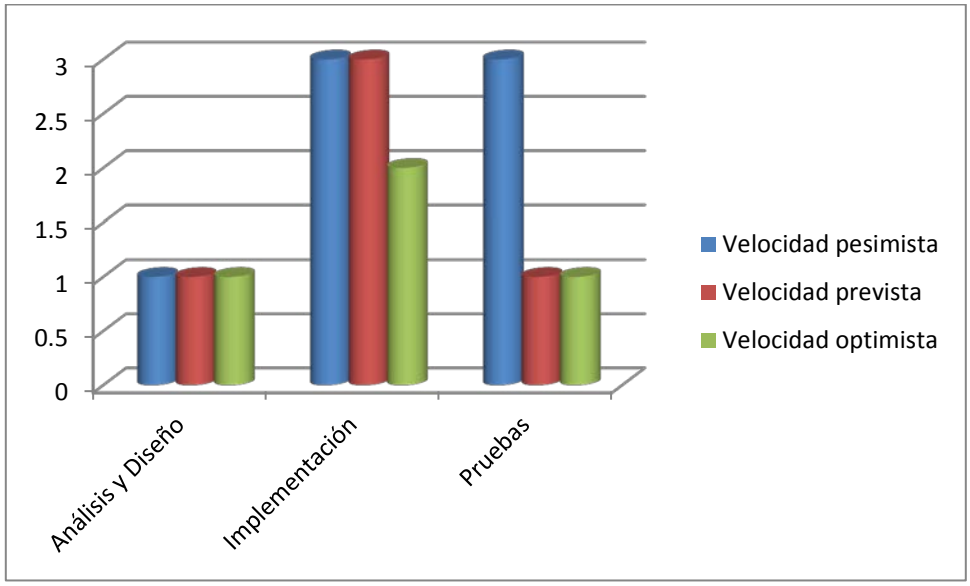


Figura 9. Grafico *Burn-Up* sprint Obtención de Distancias Terrestres

Tabla 8. Detalle del *sprint* 4

N° de sprint	**4**
Identificación	Definir el problema del agente viajero
Descripción de la funcionalidad	Una vez obtenida la latitud y longitud de los refugios temporales y sus respectivas distancias. Se construye una consulta SQL para obtener las distancias que hay de un refugio temporal a otro y se construye un arreglo de Nodo – Nodo - Distancia
Campo	Origen Destino Distancia
Modulo del sistema al que pertenece	Definir el problema del agente viajero
Personas asignadas	MGD
Estimación	3 días con dos días de holgura

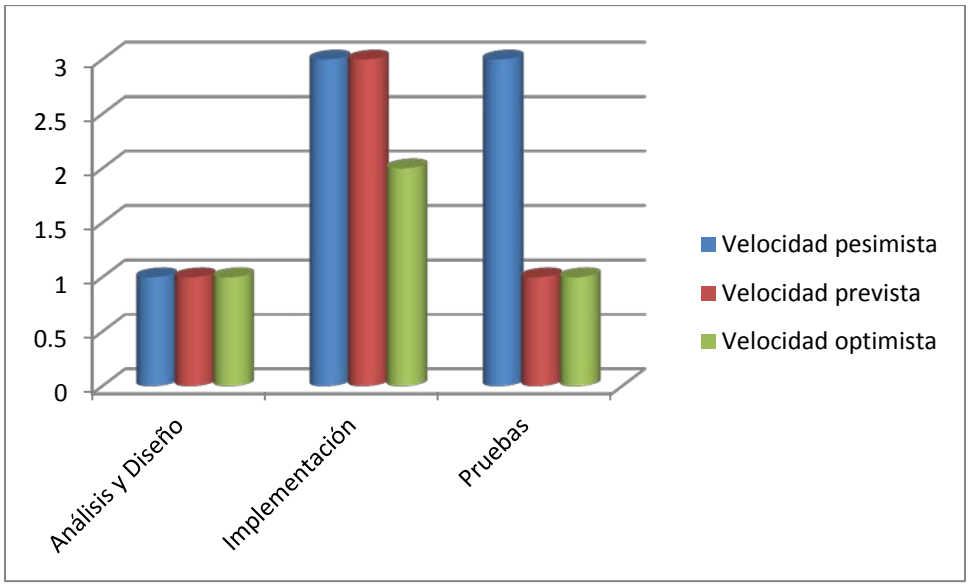


Figura 10. Grafico *Burn-Up* sprint definición del problema

Tabla 9. Detalle del *sprint* 5

N° de sprint	**5**
Identificación	Se aplica el método algoritmo del vecino más cercano.
Descripción de la funcionalidad	Los datos de entrada son los nodos distancia que se obtuvieren en el sprint 4. Se aplica el método del algoritmo del vecino más cercano. Al final se obtiene una ruta.
Campo	Origen Destino Distancia
Modulo del sistema al que pertenece	Se aplica el método algoritmo del vecino más cercano.
Personas asignadas	MGD
Estimación	5 días con dos días de holgura

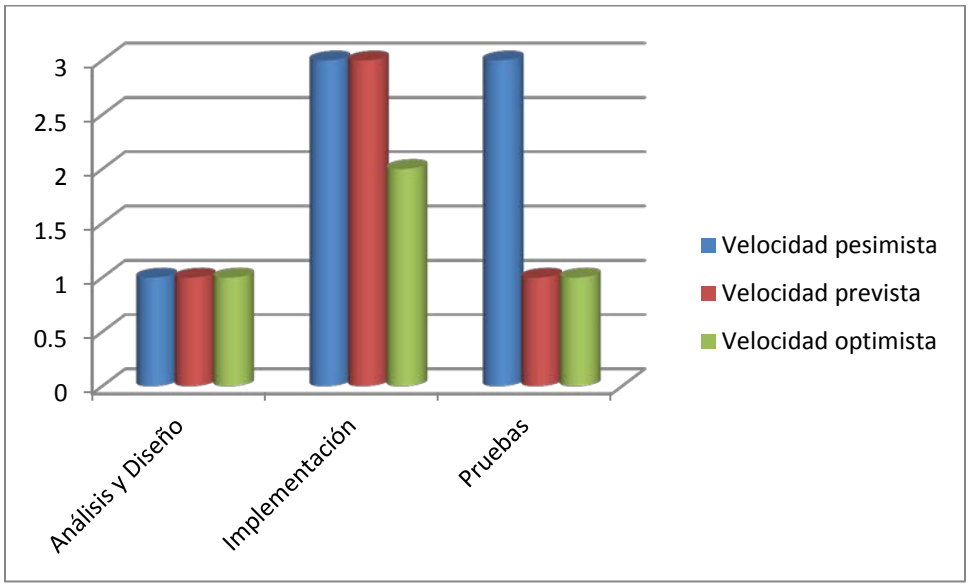


Figura 11. Grafico *Burn-Up* sprint implementa algoritmo

Tabla 10. Detalle del *sprint* 6

N° de sprint	**6**
Identificación	Mostrar gráficamente la solución obtenida recorriendo todos los nodos en Google Maps.
Descripción de la funcionalidad	Una vez que se obtiene la distancia mínima para recorrer todos los refugios temporales se muestra al usuario mediante un mapa mostrando la ruta obtenida por el método del algoritmo del vecino más cercano utilizando la API de Google Maps.
Campo	Origen Distancia Ruta Suma de distancias
Modulo del sistema al que pertenece	Mostrar gráficamente la solución obtenida recorriendo todos los nodos en Google Maps.
Personas asignadas	MGD
Estimación	7 días con dos días de holgura

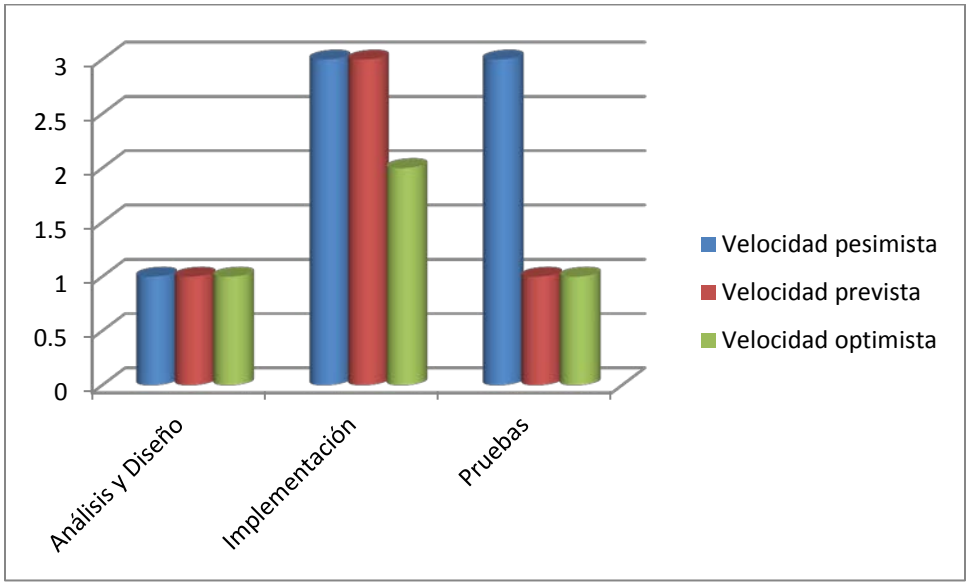


Figura 12. Grafico *Burn-Up* sprint solución

2.4.2. Artefactos Generados

A continuación se presentan los artefactos generados en el *product backlog* que comprende el modelo de la base de datos y los diagramas de análisis para el desarrollo del sistema.

Para realizar el modelo entidad-relación se utilizó la técnica de identificación de sustantivos.

Lista de sustantivos:

1. Nodos
2. Distancias

Donde cada nodo representa la ubicación de los refugios temporales y las distancias son las respectivas combinaciones que hay entre cada uno de los refugios temporales. Cada nodo tiene una distancia calculada. Para obtener el modelo relacional (Figura 15) fue indispensable realizar el diagrama entidad relación (Figura 13) y el diagrama conceptual (Figura 14).

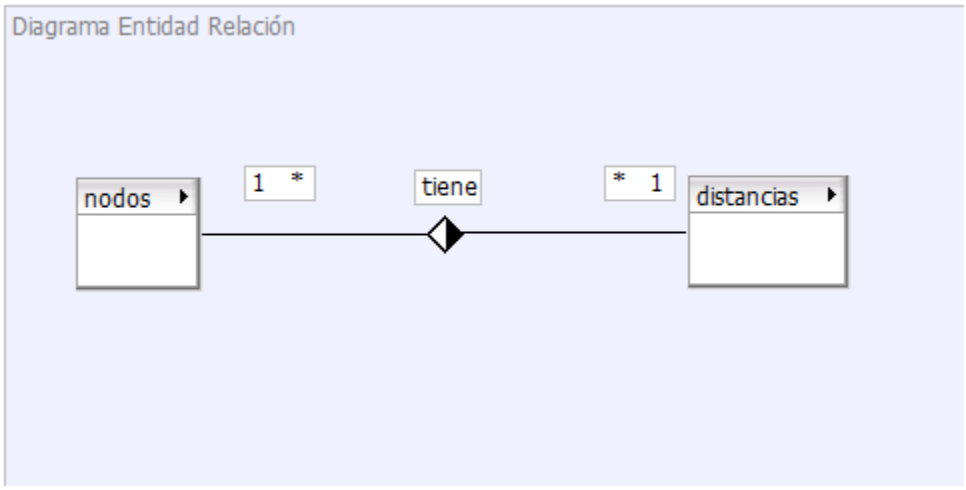


Figura 13. Diagrama entidad-relación

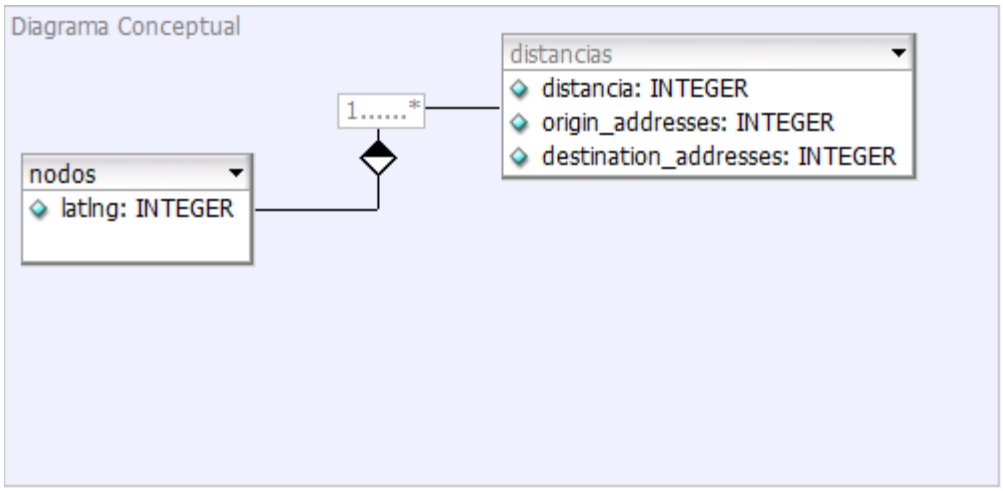


Figura 14. Diagrama Conceptual

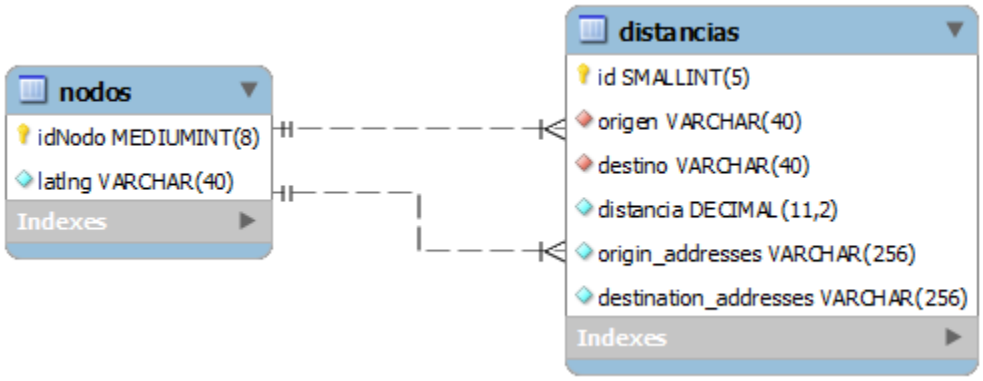


Figura 15. Modelo relacional

2.4.3. Requerimientos del Sistema

A continuación se presentan los diagramas de análisis que tienen como objetivo representar los requisitos funcionales del sistema. Cada uno de los diagramas muestra de manera general los módulos que tiene el sistema y los requisitos funcionales que deben tener cada uno de estos módulos (Figura 16, Figura 17, Figura 18, Figura 19, Figura 20, Figura 21).

Diagramas de análisis

		Modelo de Casos de Uso Administración	
ID Caso de Uso:	SITRA_01		
Proyecto:	SITRA-system	Subsistema:	
Fecha:	29/10/11	Requerimiento:	
Elaboró:	MGD		
Revisó			
Narrativa de Caso de Uso			
El usuario con privilegios de crea perfiles y da acceso a nuevos usuarios al sistema Se crea un usuario Se le asigna un perfil Se le asigna una cuenta y contreseña de acceso			
Modelo de Caso de Uso			

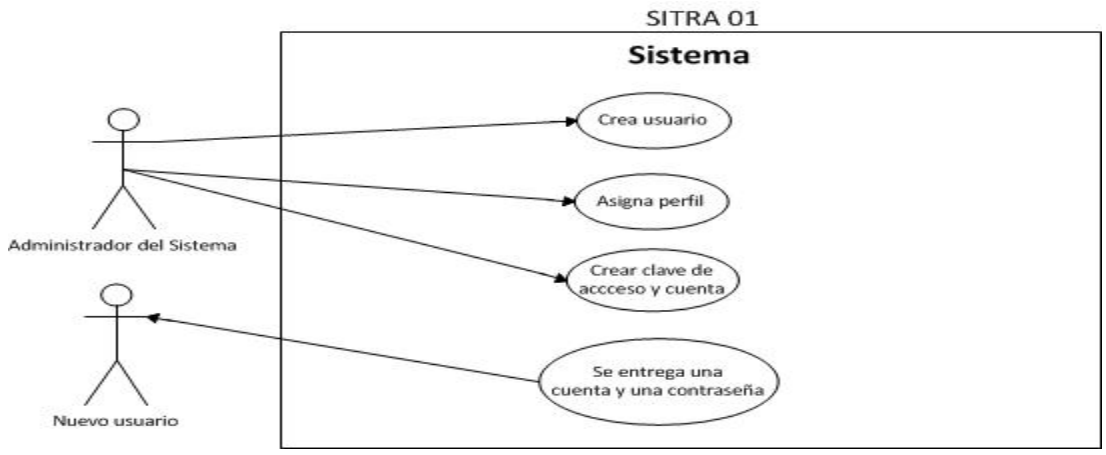


Figura 16. Diagrama de caso de uso Administración

		Sistematizar la localización de la latitud y longitud (geo localización) de los puntos de los Refugios Temporales	
ID Caso de Uso:	SITRA_05		
Proyecto:	SITRA-system	Subsistema:	
Fecha:	18-09-2012	Requerimiento:	
Elaboró:	MGD		
Revisó	MGD		

Narrativa de Caso de Uso

El usuario elige los refugios temporales a traves de Google Maps, obtiene la latitud y longitud de cada refugio, envia los datos y estos se guardan en la base de datos.

Modelo de Caso de Uso

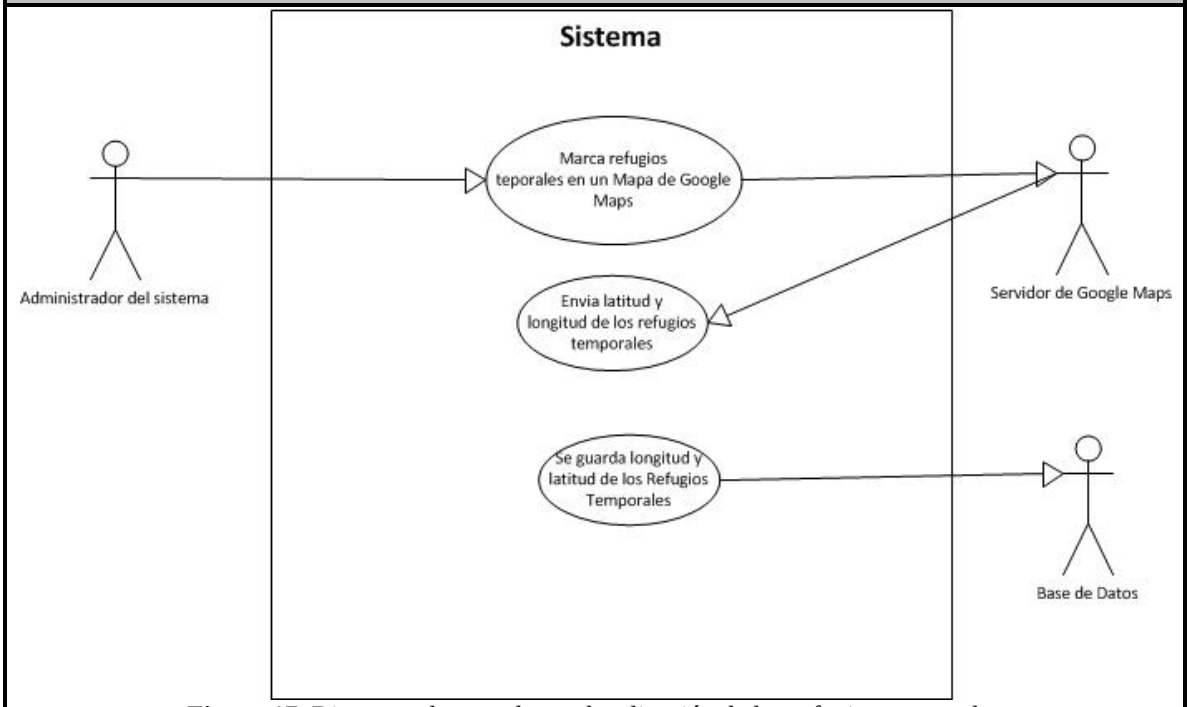


Figura 17. Diagrama de caso de geo localización de los refugios temporales

		Obtener las distancias terrestres mediante Google Maps y guardarlas en una base de datos	
ID Caso de Uso:	SITRA_05		
Proyecto:	SITRA-system	Subsistema:	
Fecha:	18-09-2012	Requerimiento:	
Elaboró:	MGD		
Revisó	MGD		

Narrativa de Caso de Uso

Una vez obtenida la latitud y longitud de los Refugios Temporales, se hizo uso de una API de Distancias de Google Maps para poder obtener las combinaciones de las diferentes distancias entre todos los Refugios temporales.
La cantidad de los refugios temporales para n destinos esta definido por el producto del mismo. Se guardan en la base de datos todas las combinaciones posibles (origen, destino, distancia, dirección de origen y dirección destino).

Modelo de Caso de Uso

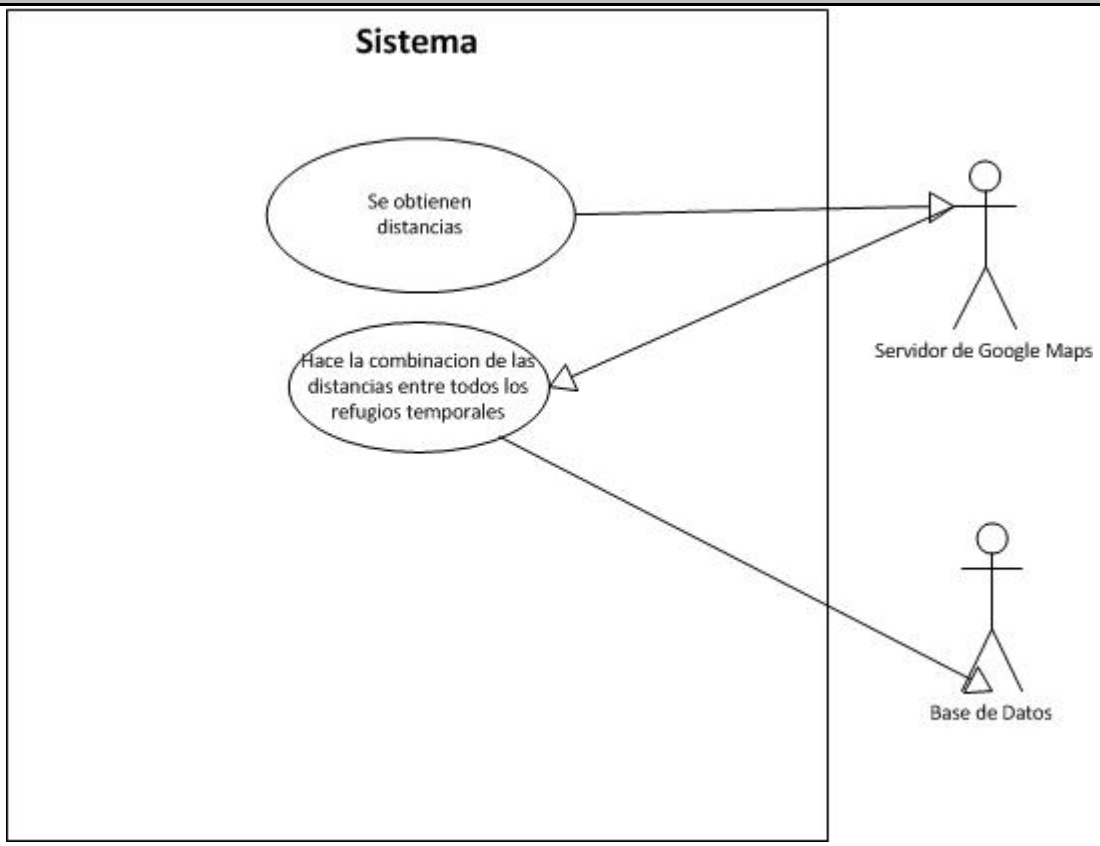


Figura 18. Diagrama de caso de obtener distancias mediante Google Maps

		Definir el problema del agente viajero	
ID Caso de Uso:	SITRA_05		
Proyecto:	SITRA-system	Subsistema:	
Fecha:	18-09-2012	Requerimiento:	
Elaboró:	MGD		
Revisó	MGD		

Narrativa de Caso de Uso

A partir de una Consuta SQL se obtienen origen, destino, distacia, direccion de origen y direccion destino

Modelo de Caso de Uso

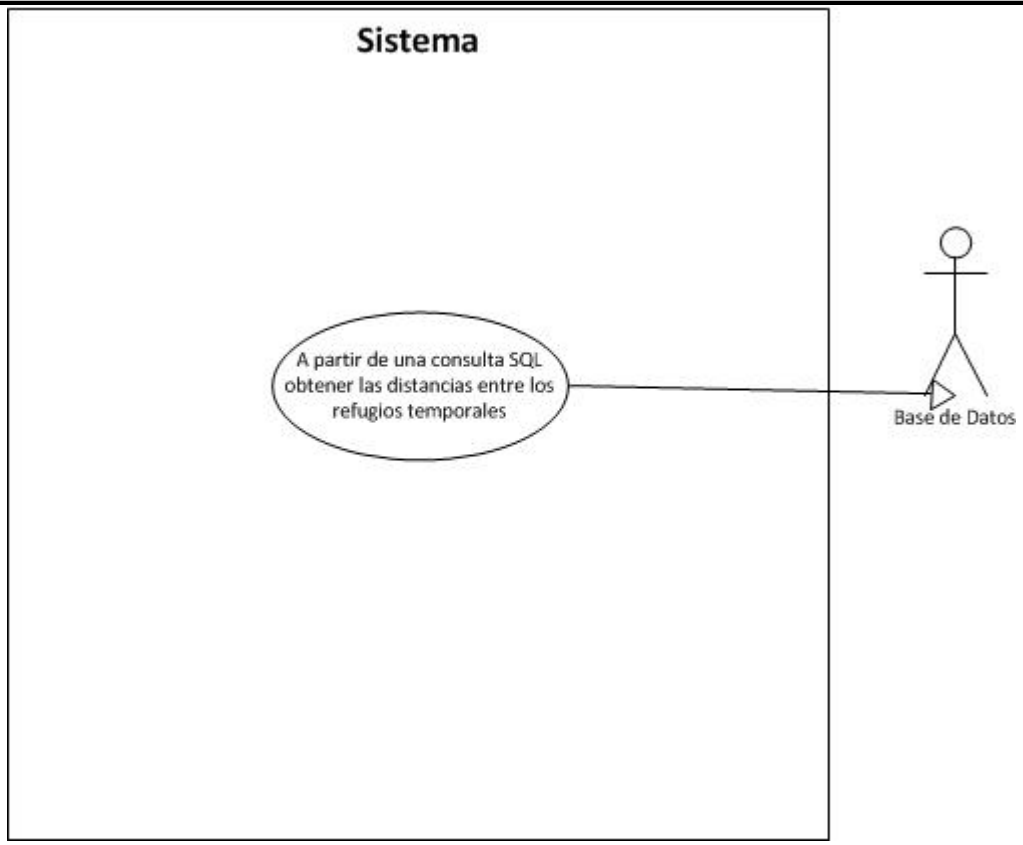


Figura 19. Diagrama de caso de definición del problema de Agente Viajero

		Resolver a través del algoritmo del vecino más cercano.	
ID Caso de Uso:	SITRA_05		
Proyecto:	SITRA-system	Subsistema:	
Fecha:	18-09-2012	Requerimiento:	
Elaboró:	MGD		
Revisó	MGD		

Narrativa de Caso de Uso

Con los datos obtenidos en el caso de uso anterior se resuelve el problema mediante el método del algoritmo del vecino más cercano. Este punto es muy importante pues a futuro se podrán incorporar otros metodos de solución.

Modelo de Caso de Uso

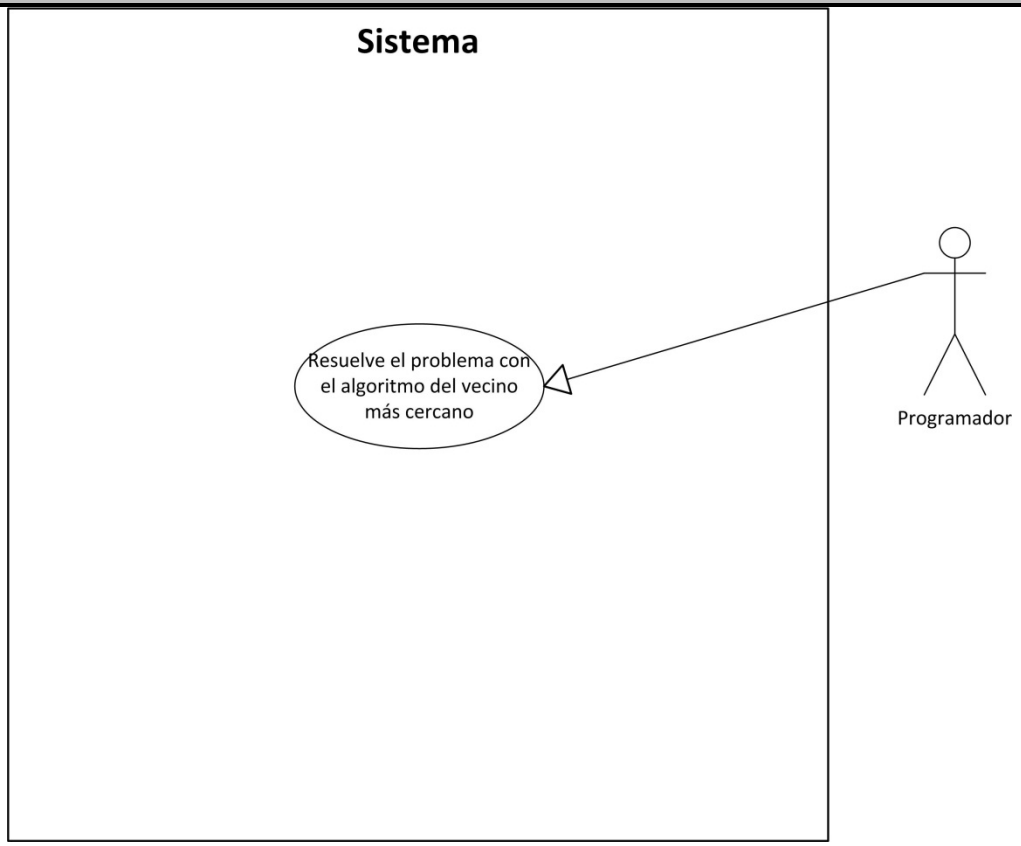


Figura 20. Diagrama de caso de uso Implementar algoritmo del vecino más cercano

		Mostrar gráficamente la solución obtenida recorriendo todos los nodos en Google Maps.	
ID Caso de Uso:	SITRA_05		
Proyecto:	SITRA-system	Subsistema:	
Fecha:	18-09-2012	Requerimiento:	
Elaboró:	MGD		
Revisó	MGD		

Narrativa de Caso de Uso

Mostrarle al usuario a traves de Google Maps la ruta con la distancia minima.

Modelo de Caso de Uso

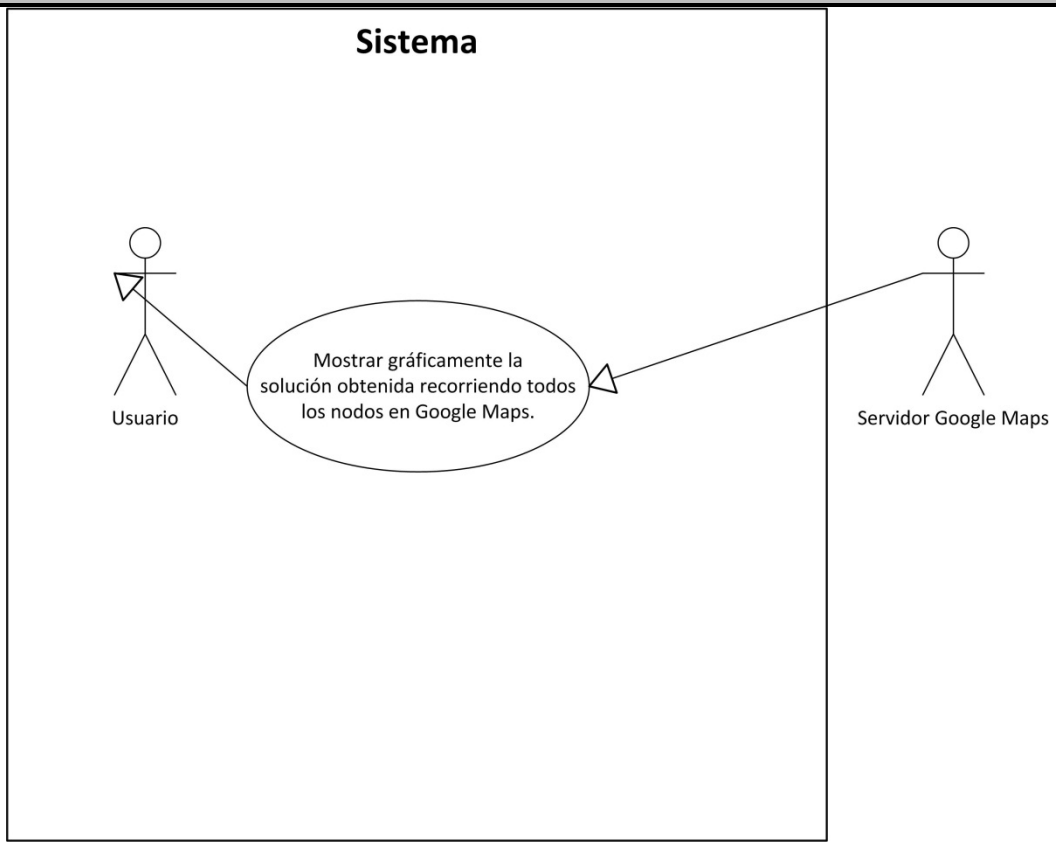


Figura 21. Diagrama de caso de mostrar solución al usuario

2.4.4. Pasos metodológicos para obtener la ruta.

Para la obtención de la ruta se siguieron los siguientes pasos metodológicos;

1. **Obtención de la geo localización de los refugios temporales:** Para poder utilizar el entorno de Google Maps para posicionar los marcadores que indican la posición de los refugios temporales se siguieron los siguientes pasos:
   1.1. Se consultó la documentación en línea del manual de servicios para la API versión 3 de JavaScript de Google Maps.
   1.2. Basado en las clases, objetos y métodos de esta API, se programaron 7 funciones:
      1.2.1. Función *initialize*: Inicializa un objeto de tipo *Map* llamado mapa. A este objeto se le incorpora un evento click, que presionar el botón izquierdo con el puntero del ratón, llamará la función *addMarker*. Además después de crear el objeto de tipo *Map*, se invoca a la función *setMarkers*.
      1.2.2. Función *addMarker*: Usa un método del objeto de tipo *Map* llamado *Marker*, el cual crear un marcador en la posición donde se selecciona *on* el puntero del ratón, este marcador tiene una propiedad llamada *position* (geo localización), la cual es enviada por una llamada *AJAX* con la función *ajax_request* hacia el servidor de bases de datos para grabar la geo localización (latitud y longitud) del marcador.
      1.2.3. Función *removeMarker*: Es llamada cuando se presiona sobre un marcador existente, elimina el marcador que indica la posición del albergue en el mapa y a su vez hace una llamada *AJAX* con la función *ajax_request* hacia el servidor de bases de datos para eliminar la geo localización del marcador.

1.2.4. Función *setMarkers*: En caso de que existan marcadores de latitud y longitud ya grabados en la base de datos, coloca marcadores de geo localización en el mapa para estos datos existentes, esta función requiere de la función *createMarker*.

1.2.5. Funcion *createMarker*: Invoca la imagen con el número de marcador que se está, retorna la imagen que será utilizada en la función *setMarkers*.

1.2.6. Función *deleteMarkers*: Esta función es invocada cuando se presiona el botón "Reiniciar Marcadores", borra todo los marcadores de posición localizados en el mapa, invoca la función *ajax_request* para eliminar cada uno de los registros de geo localización de los marcadores grabados en la base de datos.

1.2.7. Función *ajax_request*: Crea un objeto de tipo *http* que se comunica asíncronamente con un script *PHP* y la base de datos para grabar o eliminar las geo localizaciones de los marcadores.

Todas estas funciones se incorporaron en una página *HTML* para mostrar el mapa y permitir al usuario interactuar con el mapa creado.

2. **Calcular las distancias entre nodos m*n**. Se utilizó otro servicio de la API versión 3 de Google Maps llamado Servicio de Matriz de distancia.

La matriz de distancias de Google calcula la distancia y la duración del recorrido entre múltiples puntos de origen y destino con un medio de transporte específico, en este caso un vehículo.

El servicio de matriz de distancias tiene las siguientes limitantes.

1. Solo acepta 25 orígenes o 25 destinos máximos.

2. 100 elementos por solicitud (orígenes, tiempos y destinos) cada 15 segundos.

Basado en estas limitantes, si se desea saber la distancia entre todos los nodos, para un número N de nodos, los valores devueltos de las combinaciones de distancias entre todos los nodos será de N*N.

Para obtener las distancias entre todos los nodos (albergues) se siguieron los siguientes pasos:

1. Se crea una conexión a la base de datos para comprobar que no hay distancias calculadas de una matriz de distancias anterior, si hay datos de distancias se infiere que ya se llamó una vez el servicio de matriz de distancias y no es necesario llamarla de nuevo.
2. Se crea una conexión hacia la base de datos MySQL y se obtienen las geo localizaciones de todos los refugios temporales.
3. Con esta información se construyen 2 variables de tipo arreglo que contienen las geo localizaciones de estos nodos.
4. Se programaron las funciones que a continuación se describen:

    4.1.Se crea una Función *initialize*: crea un objeto de tipo *Map*, para cumplir con la tercera restricción del servicio de matriz de distancias de la API de Google Maps.

    4.2.Función *calculateDistances*: Accede al servicio de matriz de distancias de Google enviándole las 2 variables de orígenes y destinos, de tipo arreglo construidas en el punto 2, permitiendo llamar un método que procesará la matriz de distancias retornada por el servicio.

4.3.Función *callback*: Es llamada luego de ejecutar *calculateDistances* que recorre la matriz de distancias y por cada registro encontrado se hace una llama *AJAX* con la función ajax_*request* que inserta de manera asíncrona el nodo de origen, el nodo destino, la distancia entre estos dos nodos, la dirección del nodo origen y la dirección del nodo destino.

4.4.Función *ajax_request*: Crea un objeto de tipo http que se comunica asíncronamente con un script *PHP* y la base de datos para grabar los datos obtenidos de la matriz de distancia.

3. **Obtener ruta**. Para obtener la ruta sugerida por el método del algoritmo del vecino más cercano se siguieron los siguientes pasos:
   1. Se crean una conexión a la base de datos para obtener las distancias de nodo a nodo.
   2. Se ordenan las distancias en una matriz tridimensional con estructura origen – destino - distancia.
   3. Se obtiene la ruta más corta por el método del algoritmo del vecino más cercano. El pseudocódigo del algoritmo de vecino más cercano se explica con los siguientes pasos:
      a) Se elige como punto de partida el primer nodo V elegido por el usuario.
      b) Se elige un vértice arbitrario conectado al vértice V
      c) Se busca el nodo W de menor distancia que esté conectado al vértice V y que además sea un nodo no visitado.
      d) Se incorpora el nodo W a la ruta de solución.
      e) Se marca el nodo W como visitado
      f) El nodo W será el nuevo vértice V.

g) Si todos los vértices del dominio ya están visitados se termina el algoritmo incorporando el primer nodo al final de la ruta.

h) Si aún hay nodos sin visitar, vaya al paso c.

i) La secuencia en que fueron visitados los nodos es la salida del algoritmo como ruta propuesta.

4. **Mostrar la ruta sugerida.** Para modelar el resultado gráficamente se utiliza el servicio de rutas de la API versión 3 de *Google Maps*, los puntos intermedios en una ruta son llamados hitos o *waypoints*, este servicio en su versión gratis solo permite trazar una ruta con máximo 8 hitos o *waypoints*, sin contar el punto de inicio y de finalización, se logró superar esta limitante mediante la descripción de los pasos descritos a continuación:

   a. Se obtienen las geo localizaciones de los puntos de la ruta de los datos guardados en la base de datos.

   b. Debido a las restricciones propias del servicio de trazado de rutas de Google Maps, se hace un análisis de cuantos refugios temporales deben ser visitados, dependiendo del número de refugios temporales, se crean N instancias de rutas las cuales son múltiplos de 8, cada instancia trazará 8 refugios hasta trazar todos los refugios en el mapa. Se crean y definen las variables del color de las poli líneas para el trazado de la ruta mediante una función que genera colores al azar.

   c. Se programaron las funciones siguientes:

      i. Función *initialize*: Crea dos objetos de servicio de tipo *DirectionsRenderer* y un mapa de tipo *Map* donde se trazará la ruta. La razón de tener 2 objetos, es que si son más de 8 *waypoints* o nodos

intermedios no se podría trazar la ruta, por lo que se utilizan 2 instancias de objeto *DirectionsRenderer* para trazar dos rutas unidas entre sí que permitirá recorrer hasta 16 nodos intermedios. Se cuenta cuantos refugios temporales son y en base a eso se llama un método del objeto de tipo *DirectionsRenderer* llamado *route* que trazará la ruta.

ii. Función *createMarker*: Invoca la imagen con el número de marcador que se está, retorna la imagen que será utilizada en la función *setMarkers*.

iii. Función *setMarkers*: Coloca los marcadores numéricos en la ubicación de los refugios temporales en base a la latitud y longitud ya grabados en la base de datos, esta función requiere de la función *createMarker*.

iv. Función *computeTotalDistance*: Cuenta la distancia total obtenida de la respuesta obtenida mediante el método *route*.

2.4.5. Diseño y desarrollo de la Interfaz

Pantalla de acceso: Se solicita al usuario del sistema, su usuario y contraseña como se muestra en la Figura 22. En este punto se hace una consulta en una tabla de usuarios, si el usuario y la contraseña, encriptada en formato SHA-1 coinciden, se le permite el acceso al sistema.

Figura 22. Pantalla de acceso

Pantalla de Bienvenida: Si el usuario se autentifica de manera satisfactoria, se le presenta una pantalla de bienvenida con la fecha actual. En la parte superior izquierda se encuentra el menú para el sistema como se muestra en Figura 23.

Figura 23. Pantalla de Bienvenida

A continuación se presenta la descripción de los módulos del sistema.

1. **Módulo de administración de Personal**: Permite las altas, bajas, cambios y listado de personal (CRUD), todos estos usuarios pueden ser usuarios del sistema, aunque no necesariamente por estar en este listado son usuarios del sistema.

Figura 24. CRUD administración de personal

2. **Módulo de administración de usuarios:** Este módulo permite las altas, bajas, cambios y listado (CRUD) del personal con acceso al sistema, liga la tabla de personal con la tabla de usuarios mediante un id de identificación, el usuario y su clave, permite además ponerle una vigencia en base a una fecha para que el sistema automáticamente bloquee el acceso a este usuario después de la fecha especificada.

Figura 25. CRUD de usuarios

Estos dos módulos fueron programados con una librería llamada MySQL Ajax Table Editor (MATE), el cual permite gestionar y relacionar tablas de manera rápida.

Para gestionar estos módulos, se incluyen íconos o botones que a continuación se describen:

1. Para ver la información del registro, se presiona el ícono con el signo de admiración de color azul:
2. Para editar el registro, se presiona el ícono con el signo de pluma:
3. Para eliminar el registro, se presiona el ícono gris de la papelera:
4. Para añadir un registro se incluye un botón de “Añadir”, el cual permitirá añadir un registro.

5. Se puede exportar la información almacenada en la tabla con el botón “Exportar”, el cual genera un archivo con extensión csv (delimitado por comas) para que sea fácilmente importado en una hoja de cálculo de Excel.
6. Cuando hay demasiada información se puede utilizar el botón de “Buscar” para filtrar la información requerida.

A continuación se muestra una pantalla de edición de usuario, la pantalla de edición de personal tiene un comportamiento similar.

Figura 26. Pantalla de edición de usuario

3. **Módulo de Refugios:** Al acceder al módulo de refugios, desplegado en la sección de rutas, se abre un mapa gestionado con la tecnología de Google Maps, el cual se posiciona por defecto en el área de México D.F. y Texcoco, como se muestra en la Figura 27, sin embargo el usuario puede mover el mapa al área de interés con clásico método de arrastrar, para ubicar los refugios temporales mediante puntos o marcadores de los cuales desea obtener una sugerencia de ruta para ser recorridos.

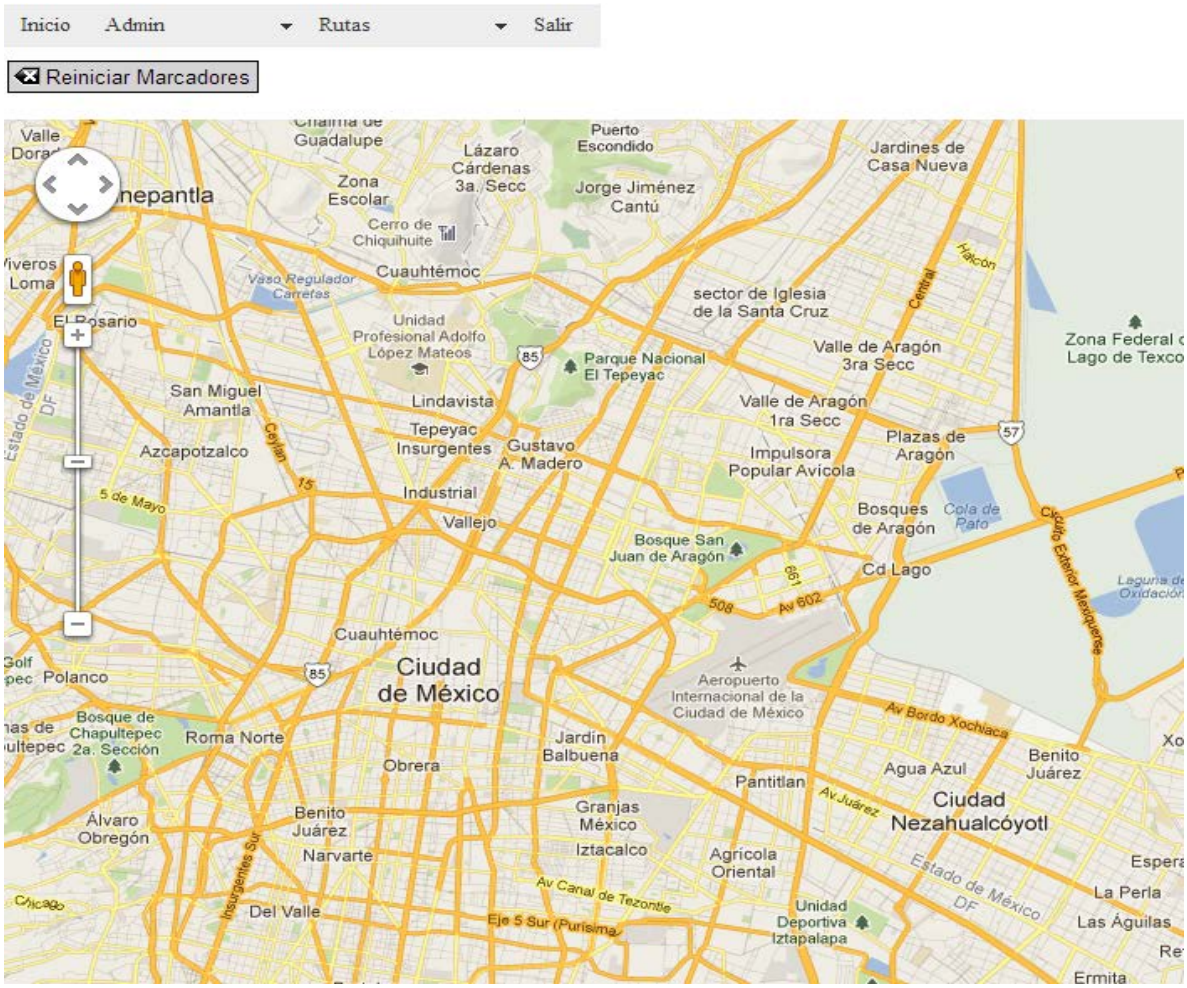


Figura 27. Pantalla de inicio del módulo de refugios.

A continuación el usuario debe marcar los refugios temporales dando un clic con el puntero del ratón en las ubicaciones localizadas en el mapa, puede utilizar las opciones de

zoom en el mapa. Si el usuario se equivoca seleccionando la ubicación del refugio temporal, debe eliminar el marcador dando nuevamente clic sobre el marcador que desea eliminar. La Figura 28 muestra la pantalla del sistema con refugios temporales ya marcados. El primer punto posicionado en el mapa será el punto de partida y retorno.

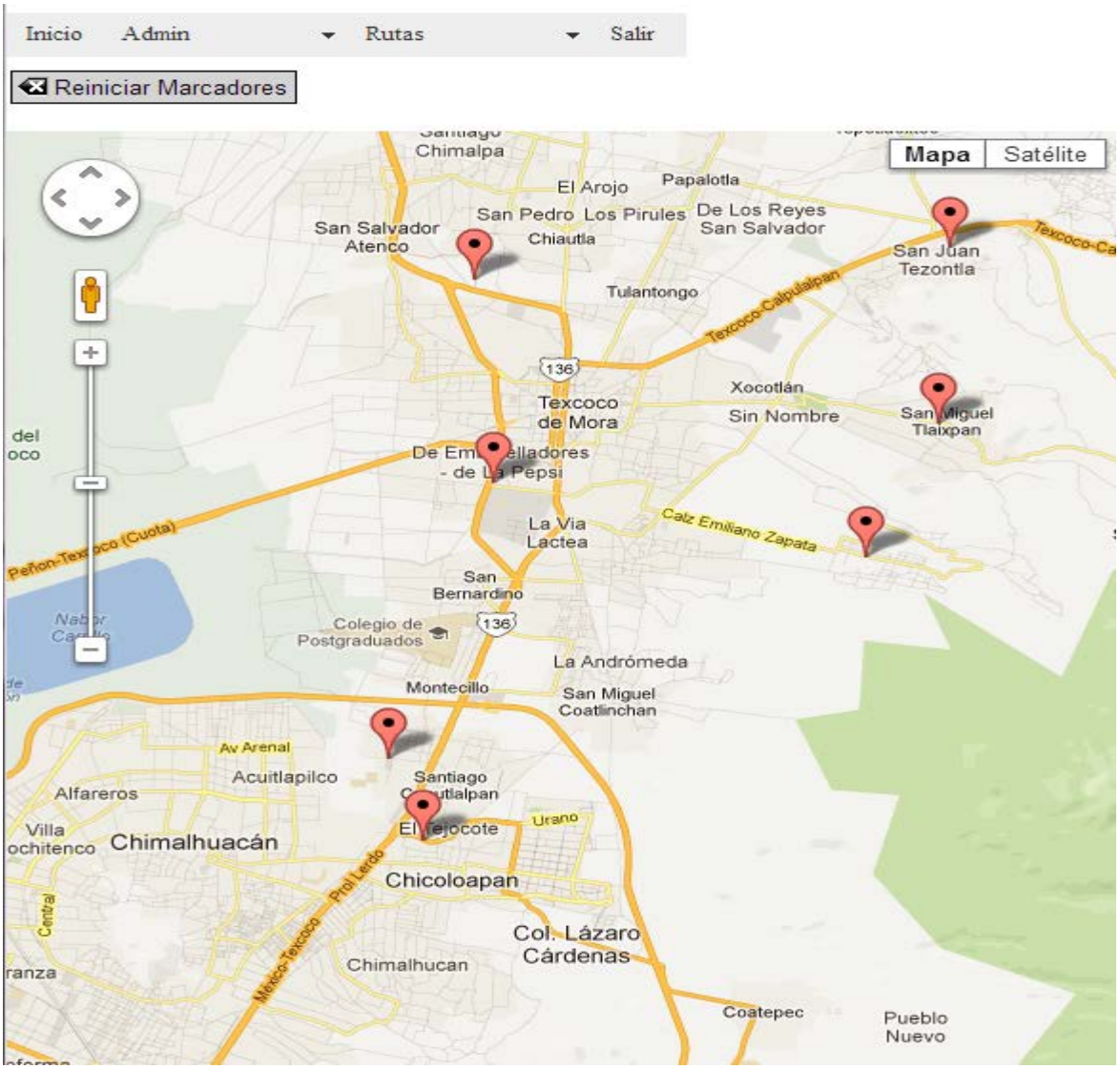


Figura 28. Pantalla donde se marcan o eliminan refugios temporales

En caso de que se desee saber el orden en que fueron elegidos las ubicaciones de los refugios temporales, basta con recargar la página (*reload*) para que el sistema muestre la secuencia en que fueron elegidas las ubicaciones tal y como se muestra en la Figura 29.

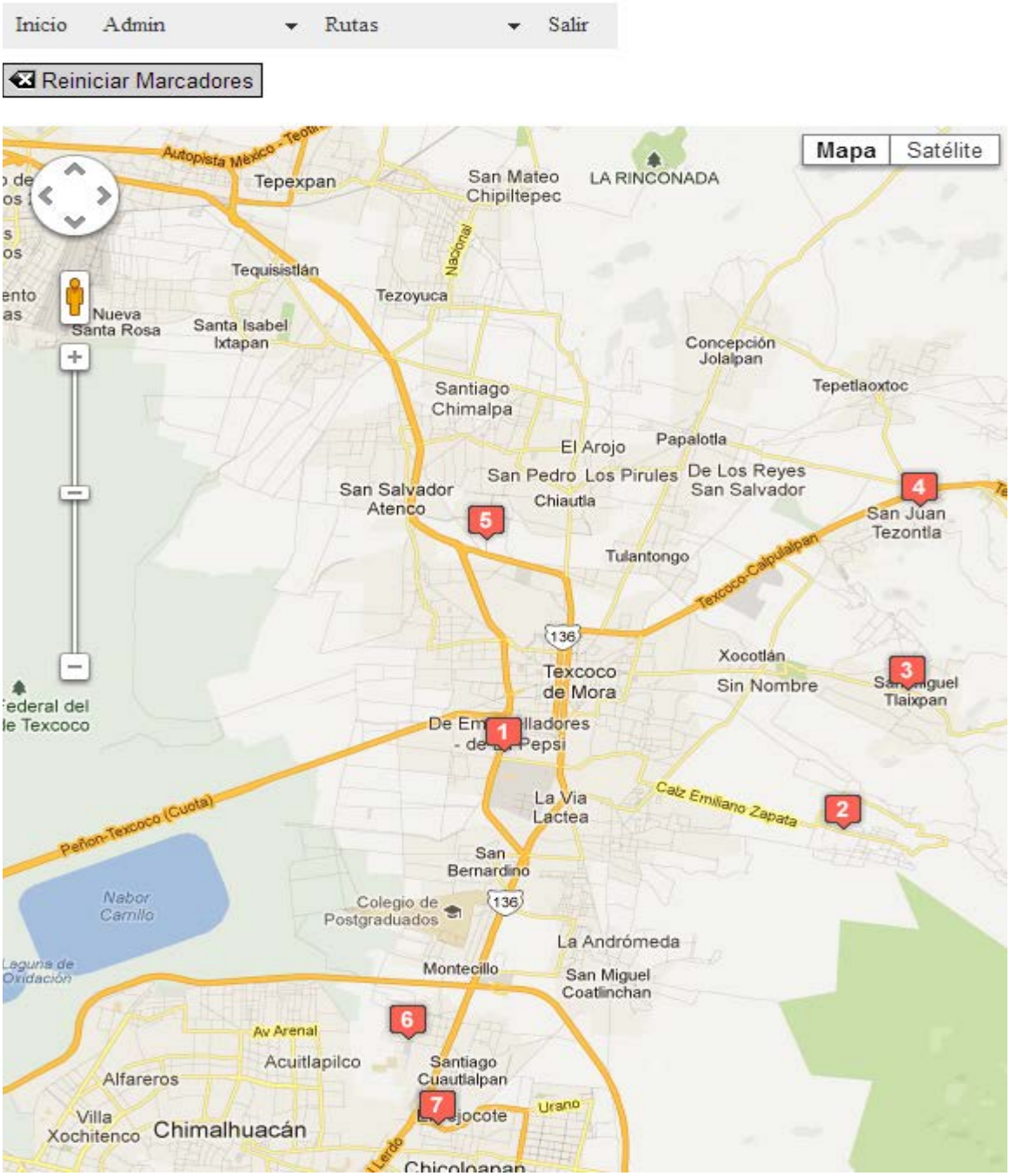


Figura 29. Enumeración de refugios temporales seleccionados

Mientras no se hagan los cálculos de distancias, se pueden seguir añadiendo más marcadores hasta el límite máximo del sistema. En caso de que el usuario desee reiniciar la colocación de los marcadores, puede borrar uno por uno cada marcador o presionar el botón de “Reiniciar Marcadores”, el cual eliminará todos los marcadores del mapa y eliminará las geo localizaciones de estos marcadores almacenados en la base de datos.

4. **Módulo Distancias:** A este módulo se accede a través del menú superior llamado Rutas, y tiene como requisito haber registrado más de 2 refugios para poder obtener las distancias reales entre los puntos mediante el servicio de Matriz de Distancias de Google Maps API.

Para obtener las distancias entre nodos, se presiona el botón “Calcular distancias”, a continuación se desplegarán las distancias calculadas de cada nodo N a los diferentes nodos M, además se muestra la dirección de origen para cada nodo y la dirección de destino, como se muestra en la Figura 30.

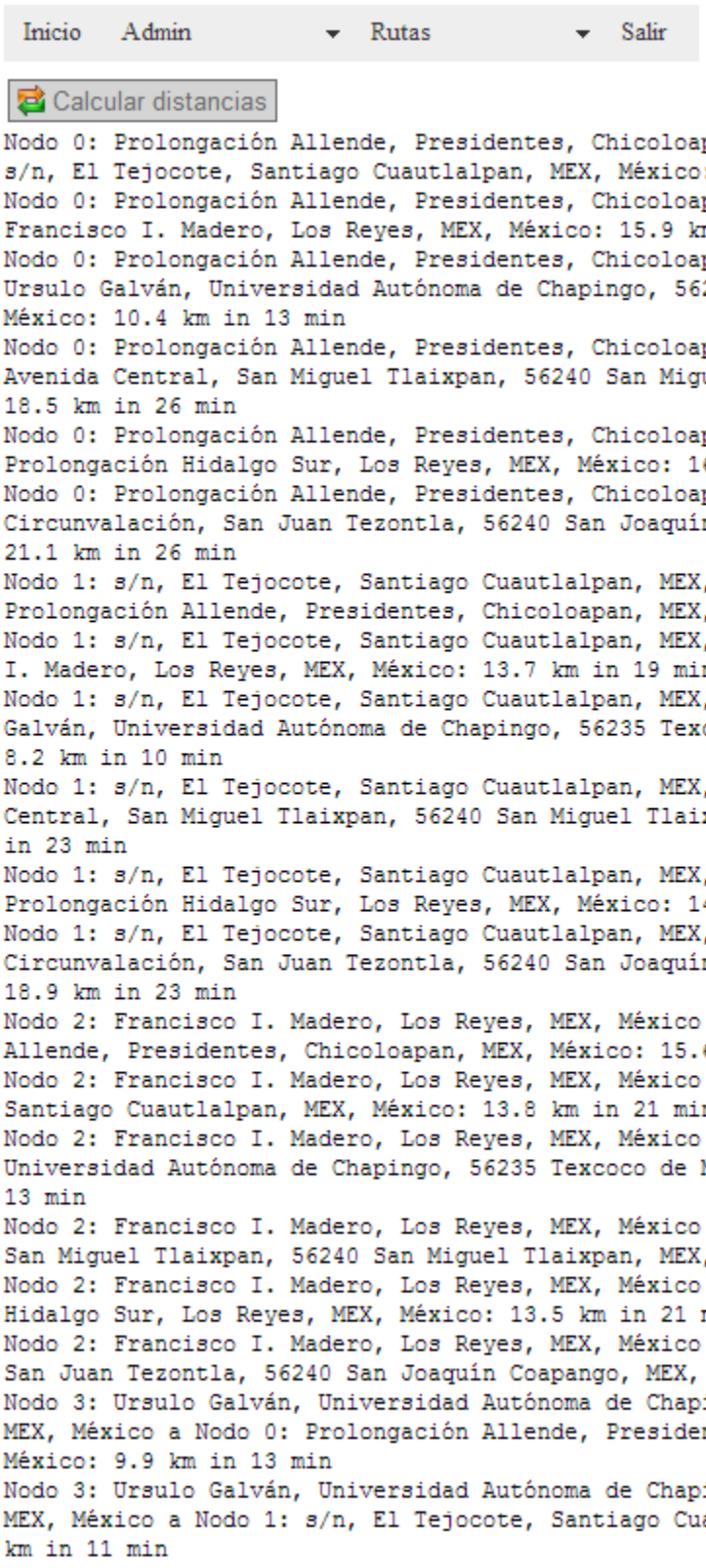


Figura 30. Distancias calculadas de cada nodo N a los diferentes nodos M

5. **Módulo de Ruta:** Este módulo está localizado en la sección Rutas, del menú superior, muestra la sugerencia de ruta calculada para recorrer todos los refugios regresando al punto de origen, los puntos elegidos en el módulo de Refugio son enumerados de acuerdo al orden sugerido por el algoritmo del vecino más próximo. Del lado derecho de la ruta se presentan las direcciones de los segmentos de ruta para recorrer así como las distancias de estos segmentos. Estas distancias se suman para presentar una suma total llamada Distancia Total en GMaps. Además se presenta una distancia total calculada por el algoritmo, esta distancia puede diferir de la distancia real de Google Maps debido a que en la distancia calculada por el algoritmo se usan los promedios de las distancias de ida y de regreso de un nodo a otro, en cambio en Google Maps ya se utilizan las distancias reales obtenidas para recorrer la secuencia de nodos calculados. La Figura 31 muestra la solución de la ruta sugerida de los refugios temporales que se eligieron en Figura 29. Este módulo utiliza las geo localizaciones de los refugios marcados en el módulo de Refugios y además requiere los datos de distancias calculados en el módulo de Distancias, de lo contrario no será posible sugerir una ruta.

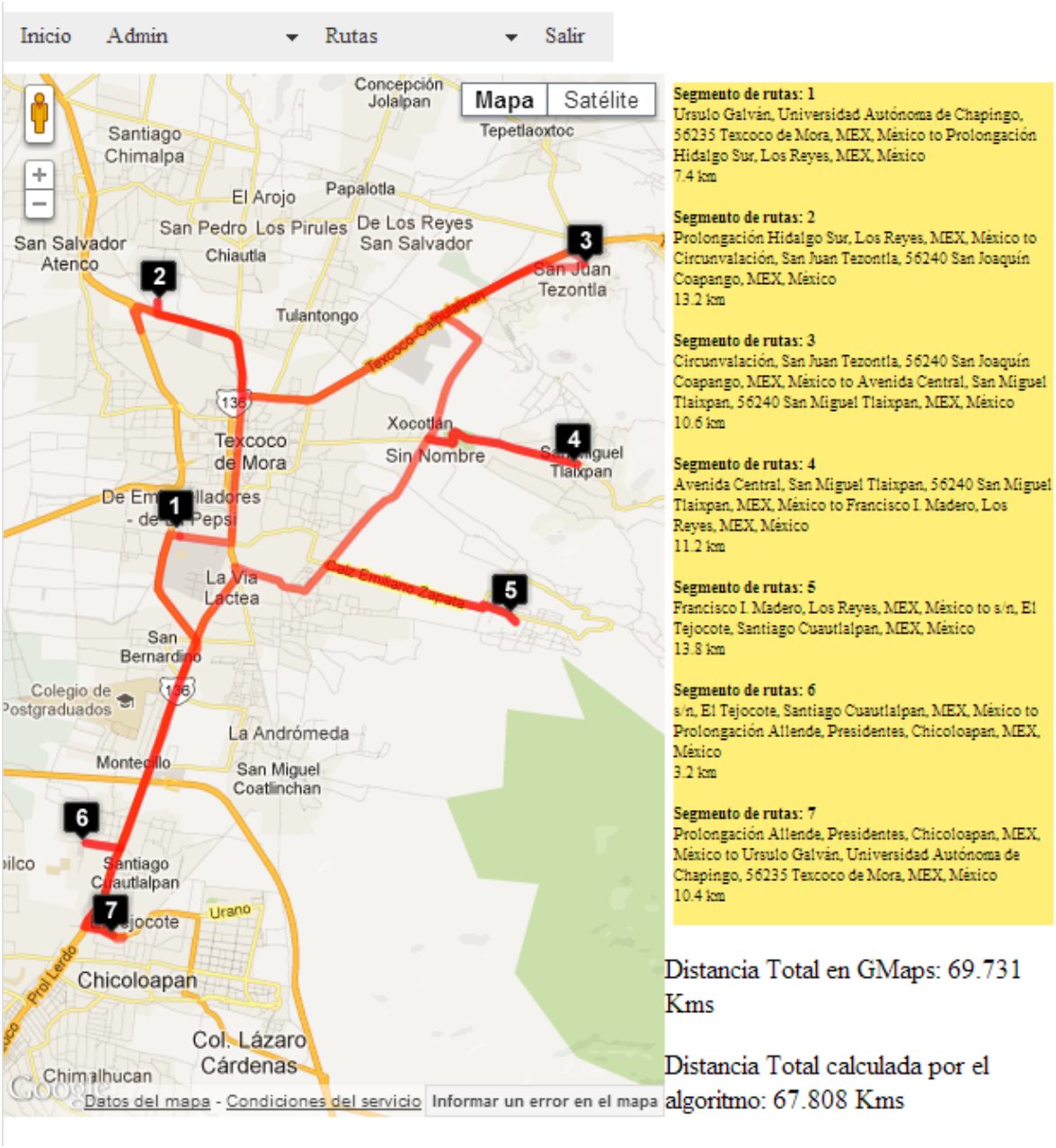


Figura 31. Ruta obtenida por el método del algoritmo del vecino más cercano

Resultados

Para demostrar los resultados, a continuación se presenta un caso práctico localizado en la zona de Villahermosa, Tabasco, donde el punto de origen será el palacio municipal de esta ciudad. Se eligieron 9 puntos como refugios temporales, colegio Inglés de Villahermosa, Miguel Hidalgo #401, Sindicato de Agricultura #302, 6 #69, Avenida de Los Ríos s/n, Aquiles Serdán #415, Paseo de las Ilusiones #1, Domingo Borrego s/n, Alfonso Taracena #207, con el supuesto que dichos puntos son ubicaciones de refugios temporales. La Figura 32 muestra los puntos marcados en el mapa.

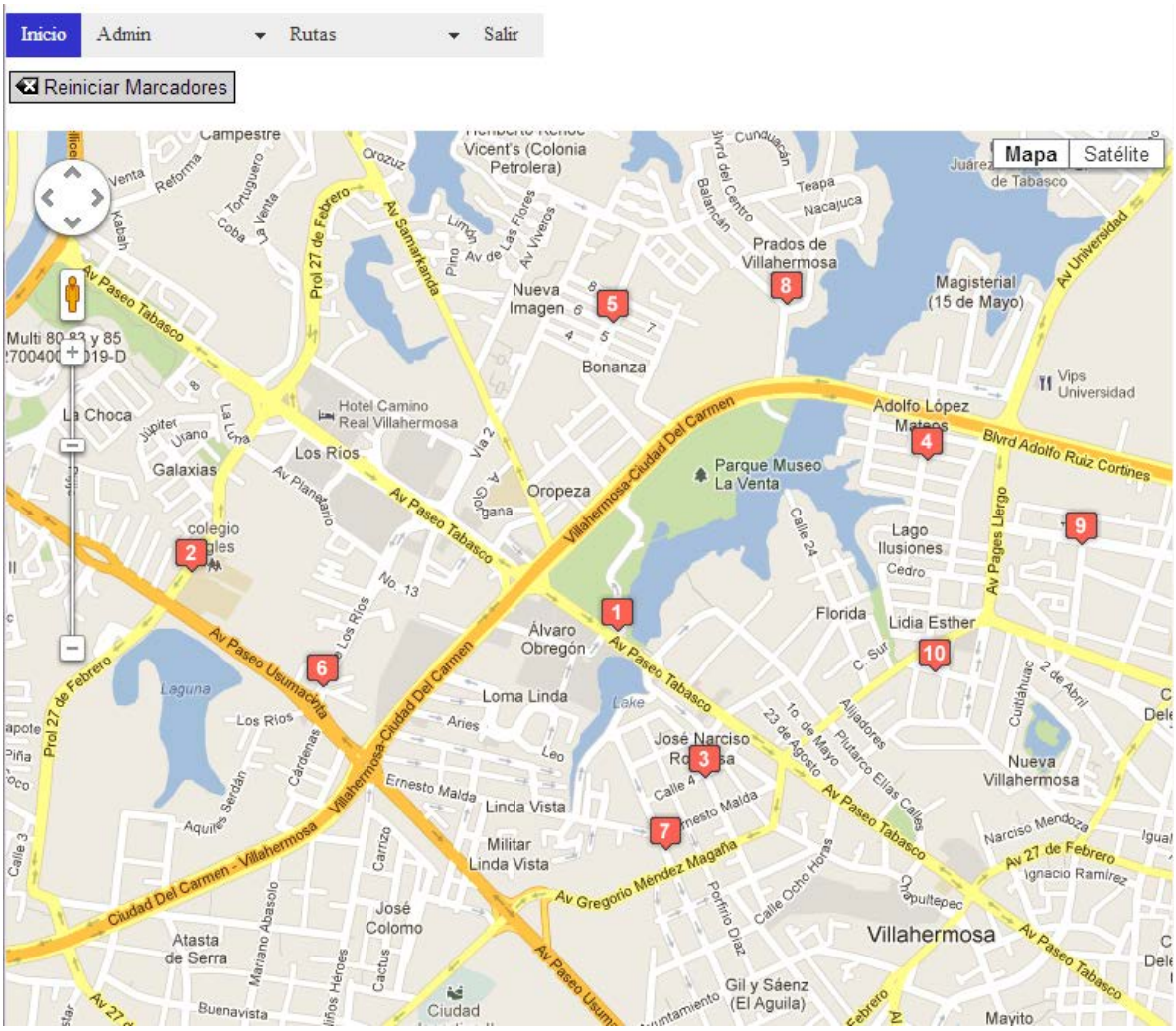


Figura 32. Elección de 9 refugios temporales, partiendo del centro de distribución de Villa Hermosa, Tabasco

El módulo de distancias arroja los siguientes datos resumidos en la Tabla 11.

Tabla 11. Distancias que existen entre los 9 refugios temporales

Origen	Destino	Distancia en KM	Origen	Destino	Distancia en KM	Origen	Destino	Distancia en KM
7	3	0.606	10	1	1.176	9	4	0.846
7	6	1.443	10	2	3.104	9	5	2.111
7	10	1.572	10	9	0.889	9	8	1.577
7	2	2.066	10	4	0.991	4	7	1.781
7	1	1.015	10	5	2.403	4	3	1.561
7	9	2.209	10	8	1.87	4	6	3.032
7	4	2.852	1	3	1.343	4	10	0.818
7	5	2.57	1	10	2.758	4	1	1.809
3	7	0.359	1	6	1.279	4	2	3.654
3	6	1.804	1	2	1.901	4	9	0.919
3	10	1.304	1	9	2.487	4	5	1.917
7	8	3.966	1	4	1.837	5	7	2.24
3	1	0.664	1	8	2.951	4	8	1.383
3	2	2.426	1	5	1.555	5	3	2.096
3	9	1.941	2	3	2.485	5	6	2.032
3	4	2.5	2	6	2.359	5	10	3.531
3	5	2.219	2	7	2.63	5	1	1.895
6	7	2.111	2	10	3.413	5	2	2.654
3	8	3.614	2	1	2.358	5	9	3.26
6	3	1.967	2	9	3.723	5	4	2.61
6	10	2.895	2	4	3.073	5	8	0.898
6	1	1.839	2	5	2.817	8	7	2.395
6	2	1.555	2	8	3.042	8	3	2.251
6	9	3.204	9	7	1.882	8	6	2.186
6	4	2.554	9	3	1.663	8	10	3.686
6	5	1.681	1	7	1.488	8	1	2.049
6	8	2.496	9	6	3.217	8	2	2.809
10	7	1.147	9	10	0.858	8	9	3.415
10	3	0.928	9	1	1.91	8	4	2.764
10	6	2.482	9	2	3.839	8	5	1.057

Para obtener la sugerencia de una ruta por medio del vecino más cercano, se construye un arreglo tridimensional cuyos índices son los refugios de inicio, los valores de la segunda dimensión son los refugios hacia los cuales se calculó la distancia, y los valores de la tercera dimensión son las distancias, la Figura 33 muestra el arreglo solo para el primer nodo, debe de construirse un arreglo más extenso que contenga todos los orígenes hacia todos los destinos.

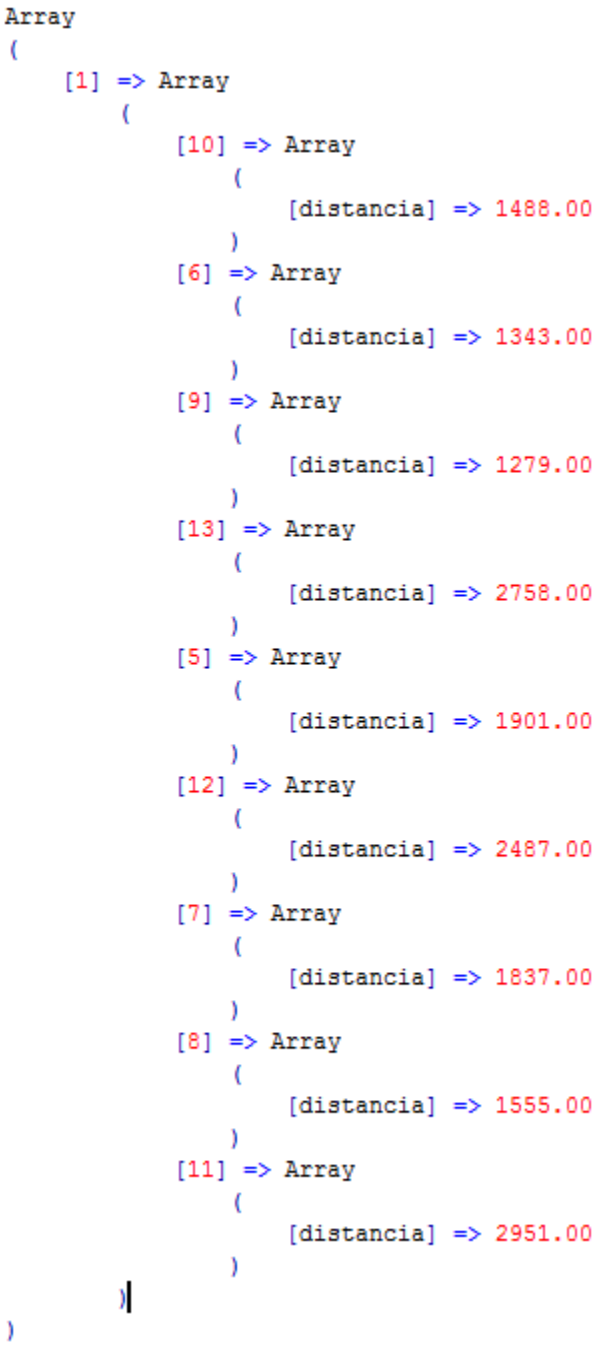


Figura 33. Arreglo de ejemplo para el nodo 1 y nodos destino

Después se itera n veces, donde n es el número de refugios temporales que deben de ser visitados, sobre una función que irá devolviendo el nodo más cercano, este nodo devuelto se incorporará a una variable ruta, dicha variable se le volverá a enviar como dato

de entrada a la función en cada iteración. La función recibe como datos de entrada el arreglo de distancias, el último nodo visitado, y la ruta que se está formando. La Figura 34 muestra el pseudocódigo de la función que realiza la tarea de recorrer el arreglo de distancias y retornar el nodo más cercano al último nodo visitado.

```
Funcion nodoMasCercano(distanciasNew,ultimoNodo,ruta)
  // distanciaNew arreglo bidimensional que contiene las distancia entre los puntos
  // ultimoNodo=Ultimo nodo visitado
  // ruta= Secuencia de nodos que forman la ruta separada por comas
  distancia=0;
  POR CADA nodo X con valores Y EN distanciasNew del registro ultimoNodo HACER
     SI X no se encuentra en la ruta ENTONCES
        SI distancia=0 ó Y[distancia]<=distancia ENTONCES
          distancia=Y[distancia]
          ultimoNodo=X;
        FIN DE SI
     FIN DE SI
     RETORNA ultimoNodo
  FIN PARA CADA X
```

Figura 34. Función que retorna el nodo más cercano del último nodo visitado

Al final de la iteración, una variable contiene la ruta sugerida, para este caso la ruta sugerida por el algoritmo del vecino más cercano es 1,6,10,13,12,7,11,8,9,5,1, la cual indica la secuencia en que se deben de visitar los refugios. Por lo que se procede a trazar la ruta siguiendo el recorrido sugerido.

En la Figura 35 se muestra el resultado gráficamente tranzando la ruta sugerida en Google Maps. La nueva numeración de nodos indica a secuencia en que deben de ser visitados los refugios, la ruta en verde indica el retorno al punto de inicio. La ruta tiene una distancia máxima de 13.393 kilómetros.

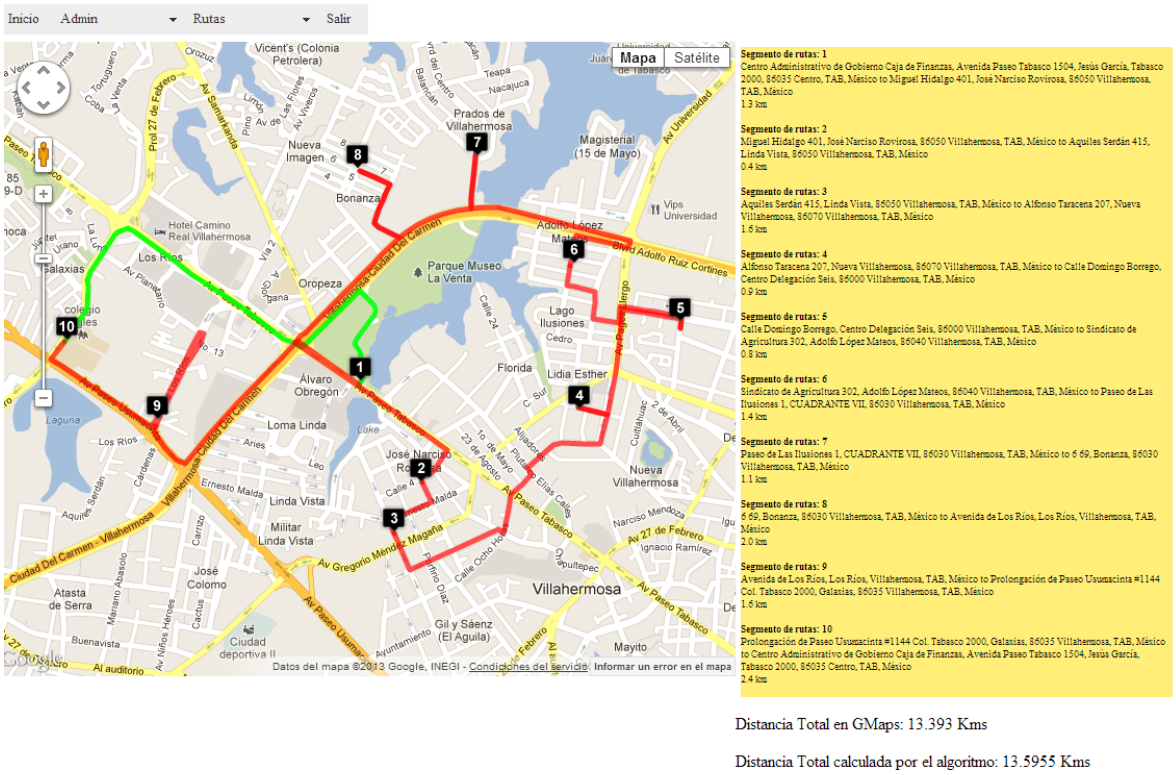
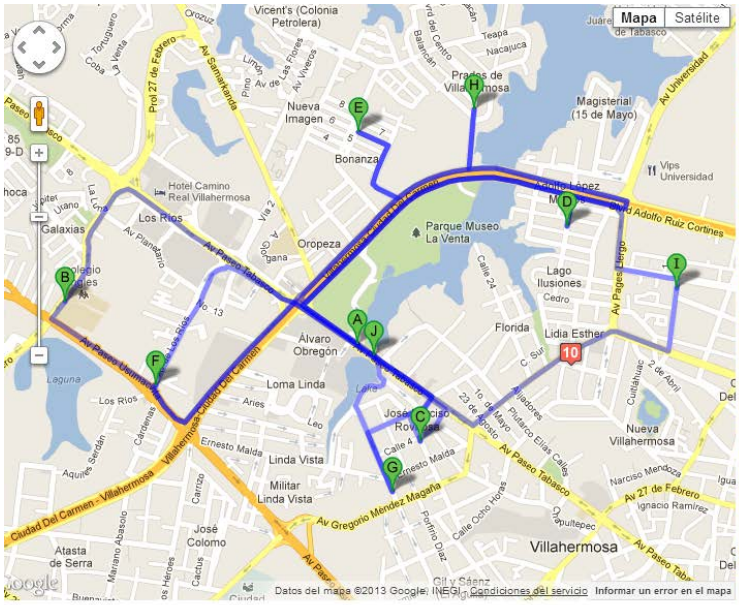


Figura 35. Ruta sugerida obtenida con el método del vecino más cercano en Google Maps

Se programó una ruta trazada con Google Maps, la ruta debería de recorrer todos los segmentos de ruta, iniciando en el punto de origen propuesto, sin optimización, sin embargo el servicio de rutas de Google Maps API 3 tiene la limitante de que solo puede trazar una ruta con máximo 8 puntos intermedios, y nuestro problema de ejemplo tiene 9 puntos intermedios (albergues temporales), por lo que a intentar trazar una ruta el servicio devuelve un error de MAX_WAYPOINTS_EXCEEDED.

Para lograr tranzar una ruta y tener un punto de referencia con respecto a la distancia sugerida por Google sin optimización de rutas, se eliminó el último refugio temporal, mostrado en la Figura 36. El nodo A y J, es el mismo punto de inicio y de retorno. La ruta trazada sin optimización tiene una distancia máxima de 22.15 kilómetros, aún con un

refugio faltante, aunque puede ser fácilmente despreciada esta diferencia ya que el refugio no.10 queda en la ruta entre I y J y éste puede ser visitado en este tramo.



Segmento de rutas: 1
Centro Administrativo de Gobierno Caja de Finanzas, Avenida Paseo Tabasco 1504, Jesús García, Tabasco 2000, 86035 Centro, TAB, México to Prolongación de Paseo Usumacinta #1144 Col. Tabasco 2000, Galaxias, 86035 Villahermosa, TAB, México
1.9 km

Segmento de rutas: 2
Prolongación de Paseo Usumacinta #1144 Col. Tabasco 2000, Galaxias, 86035 Villahermosa, TAB, México to Miguel Hidalgo 401, José Narciso Rovirosa, 86050 Villahermosa, TAB, México
2.5 km

Segmento de rutas: 3
Miguel Hidalgo 401, José Narciso Rovirosa, 86050 Villahermosa, TAB, México to Sindicato de Agricultura 302, Adolfo López Mateos, 86040 Villahermosa, TAB, México
2.5 km

Segmento de rutas: 4
Sindicato de Agricultura 302, Adolfo López Mateos, 86040 Villahermosa, TAB, México to 6 69, Bonanza, 86030 Villahermosa, TAB, México
1.9 km

Segmento de rutas: 5
6 69, Bonanza, 86030 Villahermosa, TAB, México to Avenida de Los Ríos, Los Ríos, Villahermosa, TAB, México
2.0 km

Segmento de rutas: 6
Avenida de Los Ríos, Los Ríos, Villahermosa, TAB, México to Aquiles Serdán 415, Linda Vista, 86050 Villahermosa, TAB, México
2.1 km

Segmento de rutas: 7
Aquiles Serdán 415, Linda Vista, 86050 Villahermosa, TAB, México to Paseo de Las Ilusiones 1, CUADRANTE VII, 86030 Villahermosa, TAB, México
4.0 km

Segmento de rutas: 8
Paseo de Las Ilusiones 1, CUADRANTE VII, 86030 Villahermosa, TAB, México to Calle Domingo Borrego, Centro Delegación Seis, 86000 Villahermosa, TAB, México
3.4 km

Segmento de rutas: 9
Calle Domingo Borrego, Centro Delegación Seis, 86000 Villahermosa, TAB, México to Centro Administrativo de Gobierno Caja de Finanzas, Avenida Paseo Tabasco 1504, Jesús García, Tabasco 2000, 86035 Centro, TAB, México
1.9 km

Distancia Total: 22.15 km

Figura 36. Ruta trazada sin optimización con Google Maps

Discusión

El método de solución propuesto funciona siempre y cuando se obtenga la geo localización del centro de distribución y de los refugios temporales. Esta geo localización se debe hacer marcando los puntos de ubicación con marcadores en el mapa mostrado en el sistema.

Se hace una propuesta de solución en la suposición de que todos los refugios están comunicados entre ellos, si algún refugio está incomunicado este deberá ser eliminado de mapa para evitar que el resultado generado por el algoritmo del vecino más cercano devuelva resultados inesperados.

El sistema funciona para una planificación de rutas exclusivamente terrestres, si un nodo no está comunicado por vía terrestre a los demás nodos, podría estar comunicado por vía aérea mediante un sistema de transporte aéreo (por ejemplo helicópteros), pero la distancia de los demás nodos al nodo incomunicado por vía terrestre debería ser calculada como una distancia euclidiana tomando en cuenta la curvatura del globo terráqueo, el sistema de información no está preparado para cálculos de distancia euclidiana de nodos incomunicados por caminos o carreteras terrestres.

El recorrido del transporte podría ser rastreado por medio de un GPS en tiempo real unido a la API de Google Maps, pero el presente trabajo no contempla la utilización del GPS integrado en los dispositivos móviles de última generación

Se superó la limitante original de la API de Google Maps que solo permite trazar una ruta con 8 nodos, en el sistema de información programado es posible trazar una ruta con hasta 25 nodos. Es posible superar la limitante de 25 nodos, aunque para esto se requiere de

una programación más exhaustiva para seccionar los orígenes y los destinos en más variables, pues la el servicio de la matriz de distancias que se utiliza en google maps para calcular las distancias entre nodos solo acepta máximo 25 nodos de origen o 25 nodos de destino.

Aportaciones

Los servicios gratuitos de la API versión 3 de Google Maps permiten trazar rutas con puntos intermedios, al ser un servicio gratuito está limitado para solo 8 puntos intermedios, la aplicación desarrollada puede trazar eficientemente una ruta con más de 20 puntos intermedios, como se muestra en la Figura 37. Si no se toma en cuenta el punto de partida (centro de distribución), las combinaciones de rutas para los 19 nodos restantes en el problema del viajante de comercio serian 19!, que nos daría un total de 121,645,100,408,832,000 de combinaciones posibles.

Al establecer una comparación con el servicio independiente de trazado de rutas de Google Maps, este servicio sólo permite trazar rutas con 8 puntos intermedios. La versión comercial (Google Maps API for Business) permite trazar rutas con hasta 23 puntos intermedios, sin embargo su precio mínimo es de $10,000 dólares por año, pudiendo aumentar su costo en base al número de mapas desplegados, páginas cargadas o número de rutas trazadas. Por lo que al utilizar la aplicación significa un ahorro económico significativo, además de que el sistema permite trazar rutas con hasta 23 puntos intermedios sin costo alguno.

Otra aportación de este sistema, es que es posible utilizarlo en diversas situaciones que tienen una aplicación práctica no solo para el reparto de suministros en casos de emergencia, también se puede aplicar a los carteros, la visita de clientes, ruta de enfermeras para vacunar, repartidores de diferentes oficios que deben de seguir rutas, etc.

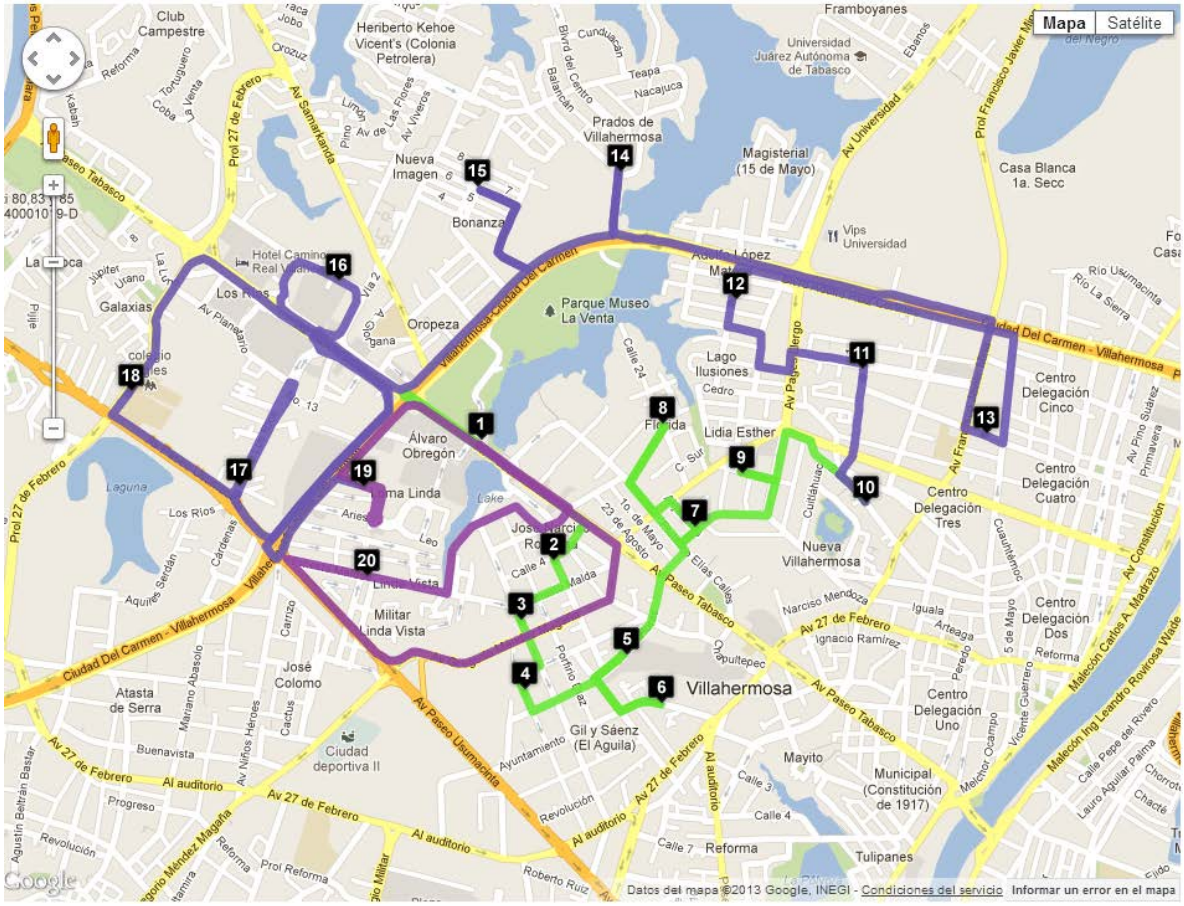


Figura 37. Ruta trazada con 1 punto de inicio y 19 puntos intermedios

Conclusiones

El problema se puede resolver con los datos obtenidos con la geo localización de los refugios temporales (latitud y longitud), y las distancias que hay entre ellos mediante el uso de los servicios que ofrece la API de Google Maps. Se puede obtener una sugerencia de ruta mejor que la que obtiene en Google Maps sin optimización de nodos.

La hipótesis ha resultado verdadera ya que fue posible recomendar una ruta con distancia mínima, utilizando la tecnología de Google Maps.

El objetivo general se cumplió ya que fue posible desarrollar un sistema Web con el uso de tecnología de Google Maps con el algoritmo del vecino más cercano, que sugiere una distancia mínima total para el reparto de suministros a refugios temporales partiendo de un centro de distribución.

Por la naturaleza del lenguaje en que fue programado, el sistema de información es compatible 100% con dispositivos móviles lo que garantiza su portabilidad, siendo posible usar el sistema en teléfonos celulares de última generación o tabletas con sistema operativo iOS o Android.

Trabajos futuros

Queda pendiente implementar un sistema de algoritmo más eficiente para resolver el problema del viajante de comercio en el módulo de rutas.

Aplicar un método de resolución del problema del viajante de comercio no solo para transporte terrestre, sino para transporte aéreo, para lo cual habrá que modificar el cálculo de distancia sin utilizar el servicio de matriz de distancias de la API de Google Maps, calculando la distancia euclidiana entre los nodos tomando en cuenta la curvatura del globo terráqueo solo con la geo localización de los nodos, esto permitiría trazar una sugerencia de ruta aérea con nodos ilimitados pues el sistema no estaría limitado por los 25 nodos de origen o 25 nodos de destino de la matriz de distancias de la API de Google Maps, ya que este servicio no sería requerido.

Si el vehículo de transporte de suministros lleva integrado un GPS, en un trabajo futuro se podría programar un módulo de rastreo del vehículo para darle seguimiento a la distribución de suministros.

Será necesario implementar el sistema en un caso práctico y evaluar los resultados en base a su eficacia y eficiencia del sistema de información.

Bibliografía

Applegate, D. L., Bixby, R. E., Vasek, C. & Cook, W. J., 2006. *The Traveling Salesman Problem: A Computational Study.* Estados Unidos de America: Princeton University Press.

Biggs, N. L., Lloyd, E. K. & Wilson, R. J., 1976. *Graph Theory, 1936-1936.* U.S.A: Oxford University Press.

Campbell, A., 2006. Aggregation for the probabilistic traveling salesman problem. *Computers and Operations Research,* 33(9), pp. 2703-2724.

Carrabs, F., Cordeau, J. & Laporte, G., 2007. Variable neighborhood search for the pickup and delivery traveling salesman problem with LIFO loading. *NFORMS Journal on,* 19(4), pp. 618-632.

Cartel, A. & Ragsdale, C., 2006. A new approach to solving the multiple traveling salesperson problem using genetic algorithms. *European Journal of Operational Research,* 175(1), pp. 246-257.

Cassani, L. & Righini, G., 2004. *A new approach to solving the multiple traveling salesperson problem using genetic algorithms.* Lecce, Italia, s.n., pp. Heuristic algorithms for the TSP with rear-loading. 35th.

Cirasella, J., Lyle, D., McGeoch, L. & Zhang, W., 2000. The Asymmetric Traveling Salesman Problem: Algorithms, Instance Generators, and Tests. *Springer Lecture Notes in Computer Science 2153,* pp. 32-59.

Dantzig, G., Fulkerson, D. & Johnson, S., 1954. Solution of a large-scale traveling. *Operations Research,* Volumen 2, p. 393–410.

Duan, H. & Yu, X., 2007. Hybrid ant colony optimization using memetic algorithm for traveling salesman problem. *Proceedings of the 2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning, ADPRL 2007,* pp. 92-95.

Duarte Muñoz, A., 2007. *Metaheurísticas.* Madrid: DYKINSON.

Fajardo, J. T. B. & Oppus, C. M., 2009. A mobile disaster management system using the android technology. *International Journal of Communications,* 3(1), pp. 77-86.

Fiechter, C., 1994. A Parallel Tabu Search Algorithm for Large Traveling Salesman Problems. pp. 243-267.

Glover, F., 1986. *Future Paths for Integer Programming and Links to Artificial Intelligence.* s.l.:Computers & Ops. Res.

Glover, F., 1991. Multilevel Tabu Search and Embedded Search Neighbourhoods for The Traveling. *School of Business.*

Glover, F., Gutin, G., Yeo, A. & Zverovich, A., 2001. Construction Heuristics for the asymmetric TS. *European Journal of Operational Research,* 3(129), pp. 555-568.

Google, 2013. *Google Developers.* [En línea]
Available at: https://developers.google.com/maps/documentation/javascript/tutorial?hl=es
[Último acceso: 2013 02 5].

Google, 2013. *Google Maps.* [En línea]
Available at: http://support.google.com/maps/bin/answer.py?hl=es&answer=144352
[Último acceso: 2011 03 22].

Heap, M., Kapur, R. & Mourand, A., 1989. A Fault Tolerant Implementation of The Traveling Salesman Problem. *Technical Report, Dept. of EECS, University of Texas.*

Hillier, F. & Lieberman, G., 2010. *Introducción a la Investigación de Operaciones.* México: McGrawHill.

Hopfield, J. & Tank, D., 1985. Neural' Computation of Decisions in Optimization Problems. *Biol. Cybern 52,* pp. 141-152.

Hung-Chang Lee , 2011. *An application of Google Map and tabu-search Algorithm for Traveling Salesman Problem on Tel-Home Care.* Taiwan , IEEE, pp. 764-4767.

Ilog, D., 2010. *Optimization Technology White Paper, A Comparative Study of Optimization.* [En línea]
Available at: www.ilog.com
[Último acceso: 10 11 2010].

Johnson , D. y otros, 2002. A. Experimental Analysis of heuristics for the ATSP. The Traveling Salesman Problem and its Variations.. En: s.l.:s.n., pp. 445-487.

Johnson, D. & Aragon , C., 1991. *Optimization by Simulated Annealing: an Experimental Evaluation: Part II, Graph Colouring and Number Partitioning.* s.l.:s.n.

Johnson, D. & Aragon, C., 1989. *Optimization by Simulated Annealing: an Experimental Evaluation: Part I, Graph Partitioning.* s.l.:s.n.

Johnson, D. & McGeoch, L., 1995. The Traveling Salesman Problem: A Case Study in Local Optimization.

Jungnickel, D., 2008. *Graphs, Networks and Algorithms.* Berlin Heidelberg New York: Springer-Verla.

Knox, J., 1994. Tabu Search Performance on the Symmetric Traveling Salesman Problem. *Computers & Ops. Res,* pp. 867-876.

Knox, J. & Glover , F., 1989. Comparative Testing of Traveling Salesman Heuristics Derived from Tabu Search, Genetic Algorithms and Simulated Annealing. *Technical Report, Center for Applied Artificial Intelligence, University of Colorado.*

Laarhoven P, V. & Aarts, J. M., 1987. *Simulated Annealing: Theory and Applications, Mathematics and its applications.* s.l.:D. Reidel Publishing Company.

Lin, S., 1965. *Computer solutions of The Traveling Salesman Problem.* s.l.:Bell Syst. Tech. J..

Lin, S. & Kernighan, B., 1973. An Effective Heuristic Algorithm for the Traveling Salesman Problem. *Operations Research 1973,* pp. 498-516.

Liu, Y., 2007. A hybrid scatter search for the probabilistic traveling salesman problem. *Computers and Operations Research,* 34(10), pp. 2949-2963.

Martin, C. & Otto, S., 1994. *Combining Simulated Annealing with Local Search Heuristics,.* s.l.:Laport G. y Osman I.

Mellin, O., 1992. *A Study of BP-5 and its use in the context of international emergency food aid,* Noruega: Red Barna.

Nguyen, H., Yoshihara, I., Yamamori, K. & Yasunaga, M., 2007. Implementation of an effective hybrid GA for large-scale traveling salesman problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics,* 37(1), pp. 92-99.

Ohlmann, J. & Thomas, B., 2007. A Compressed-Annealing Heuristic for the Traveling Salesman Problem with Time Windows. *INFORMS Journal on Computing,* 19(1), pp. 80-90.

Öncan, T., Kuban Altinel, I. & Lapote, G., 2009. A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers & Operations Research,* Volumen 36, p. 637–654.

Orman, A. J. & Williams, H., 2004. *A survey of different integer programming formulations of the travelling salesman problem,* London, UK.: Department of Operational Research, London School of Economics and Political Science.

Palacio, J., 2008. *Flexibilidad con Scrum.Principios de diseño e implantación de campos de Scrum.* s.l.:Safe Creative.

Parberry, I. & Gasarch, W., 2002. *Problems on Algorithms.* s.l.:Prentice-Hall, Inc..

Reeves, C., 1993. *Diversity and Diversification in Algorithms: some Connections with Tabu Search,Artificial Neural Nets and Genetic Algorithms.* New York: Springer-Verlag.

Russell, S. & Norvig, P., 2011. *Inteligencia artificial un enfoque moderno.* 2 ed. España: Pearson Educacion.

Salud, O. P. d. l., 2001. *Logística y gestión de suministros en el sector salud.* Washington: Organización Panamericana de la Salud.

Savla, K., Frazzoli, E. & Bullo, F., s.f. Traveling salesperson problems for the Dubins vehicle. *IEEE Transactions on Automatic Control,* 53(6), pp. 1378-1391.

Snyder, L. & Daskin, M., 2006. A random-key genetic algorithm for the generalized traveling salesman problem. *European Journal of Operational Research,* 174(1), pp. 38-53.

Suministros, F. p. e. M. d. l., 1999. *Sistema de Manejo de Suministros.* Costa Rica: Fundación para el Manejo de los Suministros.

Anexos

1. 13° Seminario de Investigación en la Universidad Autónoma de Aguascalientes con la ponencia titulada ”Calculo de posición vertical y horizontal de nodos para modelado de grafos usando gráficos vectoriales escalables” en la mesa de ingenierías y tecnologías(Mayo 2012)

2. III Congreso Internacional y IV Congreso Nacional de Computación e Informática en Universidad Autónoma del Carmen con la ponencia titulada “Optimización de Rutas de transporte terrestre de alimentos a albergues en caso de desastres a través de un sistema de información con tecnologías Web 2.0.”(Septiembre 2012).

3. Resumen extendido en Memoria de resúmenes 13° Seminario de Investigación en la Universidad Autónoma de Aguascalientes con el título ”Calculo de posición vertical y horizontal de nodos para modelado de grafos usando gráficos vectoriales escalables” en la mesa de ingenierías y tecnologías, ISSN 1870-4921 (Mayo 2012)

4. Libro electrónico de artículos III Congreso Internacional y IV Congreso Nacional de Computación e Informática en Universidad Autónoma del Carmen con el título “Optimización de Rutas de transporte terrestre de alimentos a albergues en caso de desastres a través de un sistema de información con tecnologías Web 2.0, ISBN 978-607-7826-24-8.”(Septiembre 2012).