Universidad de las Ciencias Informáticas

Facultad 6

Título: Módulo de cálculo de rutas del SIGUCI

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor:

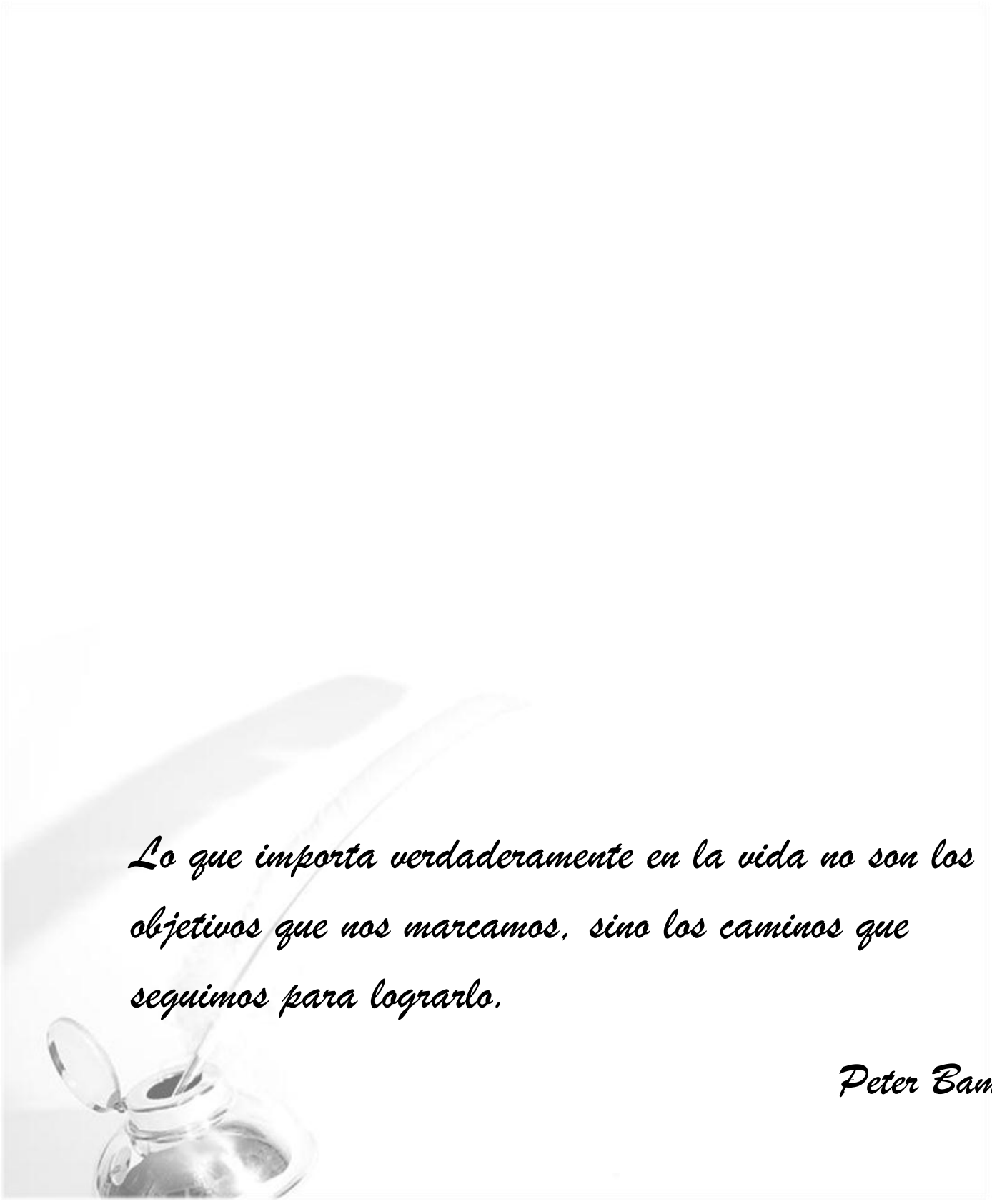
José Carlos Quiroga López

Tutor:

Ing. Leiber Fornaris Ramírez

La Habana, Cuba

Curso 2011-2012



Lo que importa verdaderamente en la vida no son los objetivos que nos marcamos, sino los caminos que seguimos para lograrlo.

Peter Bamm

Declaro que soy el único autor de este trabajo y autorizo al Centro GEYSED de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año_____.

José Carlos Quiroga López

_____

Firma del Autor

Leiber Fornaris Ramírez

_____

Firma del Tutor

Tutor: Ing. Leiber Fornaris Ramírez

Formación Académica: Ingeniero en Ciencias Informáticas (Julio/2011)

Centro Laboral: Universidad de las Ciencias Informáticas (UCI)

Correo Electrónico: lfornaris@uci.cu

Agradecimientos

A mis padres por hacerme la persona que soy, por apoyarme siempre y aconsejarme, ellos son la principal razón por la cual eh logrado graduarme de Ingeniero en Ciencias Informáticas.

A mi familia que siempre tuvo confianza en mí.

En especial a mi tío Aurelio por preocuparse siempre por mí y apoyarme en todo momento.

A mi novia por ayudarme en todo lo que podía y hasta en lo que no podía, fue un gran apoyo en los momentos difíciles, gracias por todos los sacrificios que hiciste por mí.

A mi tutor que me ayudo mucho y se preocupó por mí para que pudiera terminar bien mi carrera de ingeniero, gracias.

A todos mis amigos y amigas que estuvieron a mi lado en un momento u otro durante mi trascurso en la universidad.

En especial a piquete que me aguantaron estos 5 años.

Dedicatoria

Dedico el presente trabajo de diploma:

A mi madre que es lo más grande que tengo, sin ella no estaría donde estoy, gracias por darme tu apoyo y cariño en todo momento.

A mi padre que ha sabido guiarme por el buen camino, aconsejándome en los momentos difíciles, gracias por todo.

A mi abuela Julia que siempre me brindo mucho amor y cariño y a pesar de no estar físicamente siempre la llevo en mi corazón.

A mi tío Aurelio por preocuparse siempre por mí y ayudarme en todo lo que pudo.

A todos lo que aportaron un granito de arena para lograr que llegara a cumplir mi meta de ser Ingeniero, los que me ayudaron y apoyaron cuando más lo necesitaba.

Resumen

En la actualidad las Tecnologías de la Información y las Comunicaciones (TIC) han tenido un gran auge a nivel mundial debido a los innumerables avances que se han logrado en múltiples esferas de la sociedad. Un ámbito en el cual estos avances han desempeñado un papel fundamental es en el desarrollo de los Sistemas de Información Geográfica (SIG), que permiten representar información geográficamente referenciada. Como consecuencia se crea en el proyecto Aplicativos SIG de la facultad 6 de la Universidad de las Ciencias Informáticas (UCI) el sistema SIGUCI el cual brinda funcionalidades que agilizan la búsqueda de personas, lugares de interés así como la localización de los mismos sobre un mapa de la universidad. Sin embargo, aun cuando un usuario conozca la ubicación geográfica de algunos de estos elementos no podrá saber cuál es la forma más rápida para llegar a un destino determinado, recorriendo la menor distancia a través de la ruta más corta.

En el presente trabajo se propone como solución de dicho problema el desarrollo del módulo de cálculo de rutas del SIGUCI. Para ello se hace necesario realizar una valoración del estado del arte de los algoritmos que brindan soluciones para encontrar el camino mínimo, plantear sus características, importancia y facilidades de uso. Esta solución propuesta aportará a los usuarios del SIGUCI ahorros considerables de tiempo y esfuerzo, beneficiando de esta forma su desempeño en innumerables actividades diarias asociadas de manera directa e indirecta a la universidad.

Palabras claves: algoritmos, cálculo de rutas, camino mínimo, ruta, SIG.

Índice

Índice de Figuras

Índice de Tablas

Introducción

En la actualidad las Tecnologías de la Información y las Comunicaciones (TIC) han tenido un gran auge mundialmente debido a los innumerables avances que se han logrado en múltiples esferas a nivel mundial. En este sentido, las TIC han provocado: cambios en la forma de hacer los negocios a través de la configuración de nuevos productos y servicios, incremento de la eficiencia de la organización al mejorar la capacidad de respuesta a los problemas y aumento de la eficacia al incidir estas tecnologías sobre elementos claves de la estrategia empresarial. A partir de este conjunto de transformaciones se han logrado mejoras en la calidad de vida de la sociedad actual, brindando un mayor acceso a la información y propiciando un nivel elevado de comunicación entre las personas. Un ámbito en el cual estos avances han realizado un papel fundamental es en el desarrollo de los Sistemas de Información Geográfica (SIG).

Los SIG permiten representar información geográficamente referenciada, así como el manejo de mapas y la localización de puntos o lugares de interés sobre los mismos. Posibilitan además conocer la distancia existente de un punto determinado a otro, facilitando de esta manera que se encuentre la distancia más corta para arribar a un determinado lugar, así como un conjunto de funcionalidades que propician un desarrollo vertiginoso en el uso de las nuevas tecnologías. Ejemplos en la actualidad lo constituyen los teléfonos celulares que vienen equipados con estas facilidades, así como la red de redes Internet, donde se brindan servicios que se basan en los mencionados Sistemas de Información Geográfica.

En Cuba se hacen colosales esfuerzos para avanzar a la par de las tecnologías y los avances tecnológicos existentes a nivel global, con este propósito ha invertido gran cantidad de capital en la actualización de los sistemas en conjunto con el proceso de desarrollo de *software* y aplicaciones. Siguiendo esta política el Comandante en Jefe Fidel Castro Ruz decide crear una entidad para garantizar el desarrollo de aplicaciones informáticas en el país.

Así surge la Universidad de las Ciencias Informáticas (UCI), una de las instituciones educacionales cuya misión es la formación de profesionales en la rama de la informática, comprometidos con su Patria. Mediante la aplicación de un modelo pedagógico que integra la docencia, la investigación y la producción; con el objetivo de satisfacer las necesidades tecnológicas del país en todas las esferas. La producción en este centro se enfoca tanto al ámbito nacional como internacional, con la finalidad de fomentar el progreso económico de la nación. La UCI ha contribuido en gran medida al desarrollo de

SIG en el territorio nacional, ya que en la misma se han desarrollado múltiples sistemas de este tipo para una gran variedad de instituciones nacionales.

La UCI cuenta con un elevado número de instalaciones y lugares de interés que proporcionan una gran variedad de servicios a la comunidad universitaria. Debido a esto y a la extensión que posee la misma surge SIGUCI, un sistema que permite la representación geográfica de la universidad, brindándoles a sus habitantes un mayor conocimiento de su geografía, su estructura y composición.

SIGUCI cumple con un conjunto de especificidades de relevancia para sus usuarios como son: buscar persona, localizar estructura, buscar edificio, entre otras. Sin embargo, aun cuando un usuario conozca la ubicación geográfica de algunos elementos de interés (personas, estructuras, edificios) el mismo no podrá saber cuál es la forma más rápida para llegar a su destino. Además, no es posible conocer, en caso de que el destino no sea específico y existan varios del mismo tipo, ejemplo: cafeterías, comedores, entre otros, a cuál de ellos puede llegar primero recorriendo la menor distancia a través del camino más corto. Provocando insatisfacción en aquellos que hacen uso de la aplicación debido a que para lograr su objetivo deberá consumir tiempo y esfuerzo adicional.

Por la situación anteriormente mencionada se plantea el siguiente problema de la investigación: ¿Cómo conocer el recorrido mínimo a realizar para llegar de un lugar a otro en la Universidad de las Ciencias Informáticas?

De esta manera, para llevar a cabo la presente investigación el **objeto de estudio** lo constituyen los algoritmos de cálculo de rutas. Se define así como **campo de acción** la representación de la ruta más corta en el SIGUCI.

Teniéndose como **objetivo general** de la investigación desarrollar un módulo que permita el cálculo y representación de rutas en el SIGUCI.

El éxito en el progreso de la investigación contribuirá con el cumplimiento de la **idea a defender**: El desarrollo de un módulo para el cálculo y representación de rutas en el SIGUCI permitirá conocer el recorrido mínimo a realizar para llegar de un lugar a otro en la Universidad de las Ciencias Informáticas.

Para darle solución al problema planteado y cumplir con el objetivo propuesto es necesario trazarse un conjunto de tareas que contribuyen en el progreso de la investigación:

1. Caracterizar y analizar algoritmos de cálculo de rutas desarrollados a nivel mundial.

2. Fundamentar el uso de las tecnologías a utilizar en el proceso de desarrollo de un módulo para el cálculo de rutas en el SIGUCI.
3. Identificar las funcionalidades que debe brindar el módulo para el cálculo de rutas del SIGUCI.
4. Diseñar el módulo para el cálculo de rutas en el SIGUCI.
5. Implementar el módulo para el cálculo de rutas del SIGUCI.
6. Realizar las pruebas que permitan la validación de la solución desarrollada.

En el desarrollo de la investigación se utilizan diferentes métodos científicos para obtener información sobre el desarrollo del sistema, los cuales permiten enfocar los elementos más importantes de la misma para una mayor eficiencia y efectividad a la hora de desarrollar el *software*, los métodos que se emplean son los siguientes:

Métodos teóricos:

1. Inductivo-deductivo, este método permite llegar a conclusiones particulares de un tema en general, en este caso, conocer sobre el cálculo de la ruta más corta en un SIG.
2. Analítico-sintético, este método permite la extracción de los elementos más importantes del estudio y análisis de diversas fuentes bibliográficas que se relacionan con el desarrollo del módulo de cálculo de rutas del SIGUCI, de forma tal que permita aplicarlos en el posterior desarrollo del mismo.
3. Histórico-lógico, este método permite valorar en el transcurso de la historia, los sistemas y tecnologías que han sido empleados en la solución del objeto de estudio de la investigación y así definir sus ventajas y desventajas para apoyar a la selección de los elementos necesarios que pueden servir de base para el desarrollo del módulo cálculo de rutas en el SIGUCI.
4. Modelación, este método permite la elaboración de los diagramas y modelos asociados a la metodología de desarrollo de *software* seleccionada para llevar a cabo la solución del problema.

El presente trabajo consta con 4 capítulos cada uno con objetivos específicos.

El capítulo 1 abarca la caracterización general del objeto de estudio de la investigación, es decir, del proceso de desarrollo de Sistemas de Información Geográfica. Donde se exponen conceptos que

ayudan a un mayor entendimiento del problema planteado. Se identifican y analizan los algoritmos existentes a nivel mundial utilizados para el cálculo de rutas.

En el capítulo 2 se definen las herramientas, tecnologías y metodología que se emplean en el desarrollo del módulo para el cálculo de rutas en el SIGUCI, mediante un estudio detallado de estas definidas anteriormente en la arquitectura del sistema.

En el capítulo 3 se identifican los requisitos funcionales y no funcionales que debe brindar el sistema, este capítulo también contiene los modelos de la fase de inicio, el modelo de dominio y el diagrama de casos de uso. Así como la descripción de los casos de uso, lo que ayuda a una mejor comprensión por parte de los desarrolladores de la problemática en cuestión y la posible solución del problema planteado.

En el capítulo 4 se desarrolla el diseño del módulo, su implementación, además se define el modelo de clase del diseño, el modelo de datos, diagrama de despliegue y el diagrama de componentes. Se realiza el diseño y ejecución de pruebas mediante los casos de pruebas.

Capítulo I: Fundamentación Teórica

1.1 Introducción

Los SIG son ejemplos fehacientes del desarrollo de las nuevas tecnologías, ofrecen la posibilidad de realizar el cálculo de la distancia más corta que existe entre un lugar y otro. A través de la cual se puede determinar dado un punto de inicio y un punto final que sería el destino de un usuario, la ruta más corta a recorrer, permitiendo un ahorro considerable de tiempo y esfuerzo.

En este capítulo se abordan algunos conceptos relacionados directamente con el cálculo de rutas y en general con el objeto de estudio de la investigación científica, que ayudan a comprender mejor el desarrollo de la solución a la problemática planteada. Se realiza también un análisis de los diferentes sistemas existentes en el mundo que incluyen la funcionalidad cálculo de la ruta más corta, sus antecedentes y avances tecnológicos.

1.2 Conceptos asociados al dominio del problema

Sistema de Información Geográfica: un Sistema de Información Geográfica (SIG o GIS, en su acrónimo inglés) es una integración organizada de *hardware*, *software*, datos geográficos y personal, diseñado para capturar, almacenar, manipular, analizar y desplegar en todas sus formas la información geográficamente referenciada con el fin de resolver problemas complejos de planificación y gestión. También puede definirse como un modelo de una parte de la realidad referido a un sistema de coordenadas terrestre y construido para satisfacer unas necesidades concretas de información. (1)

La información que maneja un SIG puede ser almacenada de dos formas:

Modelo Ráster o de Retícula: divide el espacio en celdas, donde cada una de ellas representa una información específica con un único valor. Esto sucede puesto que este sistema se centra más que en la precisión de la localización en las propiedades que posee el espacio de trabajo. Si las resoluciones de las celdas son mayores, menores serán las precisiones de representación del espacio geográfico. (1)

Modelo Vectorial: este modelo se centra en la precisión de la localización o en los detalles de la información geográfica que se desee representar. Utilizando para la modelación de los objetos de la vida real tres representaciones diferentes: los puntos, las líneas y los polígonos. (1)

Información Geográfica: Se le denomina Información Geográfica al conjunto organizado de datos espaciales georreferenciados, que mediante símbolos y códigos genera el conocimiento acerca de las condiciones físico - ambientales, de los recursos naturales y de las obras de naturaleza entrópica del territorio nacional

Datos Geográficos: se puede definir un Dato Geográfico como: entidades, espacio-temporales que describen o cuantifican la distribución, el estado y los vínculos de los distintos fenómenos naturales y sociales. Por regla general, los datos geográficos se expresan gráficamente en mapas y se representan por signos convencionales especiales denominados signos cartográficos. (2)

Datos Espaciales: es el que diferencia a los SIG de otras bases de datos espaciales, representando el centro en torno al cual giran todas las posibles aplicaciones de los SIG, así se tiene que el dato espacial contiene en su acepción más elemental características de localización (X, Y) y tipo de característica temática (Z). (3)

Ruta: la ruta es un camino, vía o carretera que une diferentes lugares geográficos y que le permite a la personas desplazarse de un lugar a otro, especialmente mediante automóviles, aunque también es recurrente la presencia en estas de ómnibus. Se trata de un camino, carretera o vía que permite transitar desde un lugar hacia otro. En el mismo sentido, una ruta es la dirección que se toma para un propósito. (4)

Cartografía: La cartografía es la rama del grafismo que se ocupa de los métodos e instrumentos utilizados para exponer y expresar ideas, formas y relaciones en un espacio bidimensional o tridimensional. La cartografía parte del principio de que los seres vivos, los fenómenos físicos y sus interrelaciones ocurren en un contexto temporal y espacial y que por lo tanto es posible mapearlos. (5)

Georreferencia: proceso mediante el cual se logra una definición geográfica precisa de la ubicación de puntos, líneas y polígonos presentes en un mapa o foto, gracias a la correlación de estos y sus respectivos representados en un sistema de coordenadas reales. (6)

Se pueden distinguir dos formas distintas de georrefencia:

- La georrefencia directa o explícita: cada elemento temático tiene coordenadas asociadas.
- La georrefencia indirecta o implícita: una de las características de los datos sobre los elementos se refiere a otro tipo de elementos que tienen una georrefencia directa por ejemplo, datos censales pueden referirse mediante un código a “distritos” con una definida ubicación espacial. (6)

Algoritmo: es una secuencia de instrucciones que representan un modelo de solución para determinado tipo de problema. O bien como un conjunto de instrucciones que realizadas en orden conducen a obtener la solución de un problema.

En la vida cotidiana muchos algoritmos son empleados para resolver diversos problemas. Por tanto, un algoritmo no es más que una serie de pasos organizados de tal forma que se descaraba el proceso que se debe seguir para darle solución al problema en cuestión.
Los algoritmos de búsquedas: son aquellos que se diseñan para encontrar un elemento dentro de una estructura de datos. Estos algoritmos centran su solución en que dado un determinado elemento en un conjunto finito de elementos, puedan decidir si se encuentra o no y por supuesto mostrar la localización dentro de este. (7)
Tipos de algoritmos de búsqueda existentes:

- Algoritmo de Dijkstra (Soluciona el problema de los caminos más cortos de un vértice (origen) a los restantes).
- Algoritmo de Floyd-Warshall (Soluciona el problema de los caminos más cortos entre todos los pares de vértices).
- Algoritmo de Prim (Soluciona el problema de encontrar un árbol de expansión mínima).
- Algoritmo de Kruskal (Soluciona el problema de encontrar un árbol de expansión mínima).
- Algoritmo de Bellman-Ford.
- Algoritmo de búsqueda en anchura (BFS).
- Algoritmo de búsqueda en profundidad (DFS).
- Algoritmo de búsqueda A*.
- Algoritmo de Ford-Fulkerson.

1.3 Objeto de Estudio

El objeto de estudio de la investigación refiere a los algoritmos de cálculo de rutas, por lo que es de vital importancia realizar una caracterización de estos algoritmos, su importancia y facilidades de uso. Así como realizar una comparación entre ellos que facilite la futura selección del más adecuado para su empleo en el desarrollo de la investigación.

Descripción de algoritmos

Existen diferentes algoritmos que permiten encontrar la ruta o camino mínimo de un punto determinado a otro. En todos ellos se parte de una serie de nodos en un grafo que representan lugares de donde y

hacia donde se quiere ir y una serie de caminos que los interconectan cuyas propiedades indican distancia a recorrer, tiempo que tardaremos, entre otras.

A continuación se realiza una descripción de los algoritmos que comúnmente se emplean para resolver los problemas de distancias más cortas entre nodos de un grafo. Su uso posibilita el desarrollo de aplicaciones en las que el cálculo de rutas mínimas constituye su principal objetivo.

Algoritmo de Floyd-Warshall

El problema que intenta resolver este algoritmo es el de encontrar el camino más corto entre todos los pares de nodos o vértices de un grafo. Esto es semejante a construir una tabla con todas las distancias mínimas entre pares de ciudades de un mapa, indicando además la ruta a seguir para ir de la primera ciudad a la segunda. Este es uno de los problemas más interesantes que se pueden resolver con algoritmos de grafos.

Es decir, su procedimiento es bastante similar a construir una tabla con todas las distancias mínimas entre pares de ciudades en un mapa. Es un algoritmo que cuenta con un bucle que examina cada vértice y lo toma como pivote. Este tipo de algoritmo puede emplearse a diferentes problemas, incluyendo el diseño de circuitos, el diseño de rutas de transporte, aproximaciones al problema del viajante de comercio, o como base de otros algoritmos más complejos.

Teorema: la complejidad temporal es O (n3), pues la función Floyd presenta un triple bucle "*for*" anidado, dentro del cual se realizan operaciones sencillas de asignaciones y sumas. (8)

Algoritmo de Prim

El algoritmo de Prim es el algoritmo de Árbol de Expansión Mínima (AEM) más sencillo de implementar. Este algoritmo puede encontrar el AEM de cualquier grafo conexo ponderado y consiste en dividir los nodos de un grafo en dos conjuntos: procesados y no procesados. Al principio, hay un nodo en el conjunto procesado que corresponde al equipo central; en cada interacción se incrementa el grafo de procesados en un nodo (cuyo arco de conexión es mínimo) hasta llegar a establecer la conexión de todos los nodos del grafo a procesar. En otras palabras el algoritmo encuentra un subconjunto de aristas que forman un árbol con todos los vértices, donde el peso total de todas las aristas en el árbol es el mínimo posible. Si el grafo no es conexo, entonces el algoritmo encontrará el árbol de recubrimiento mínimo para cada uno de los componentes conexos que forman dicho grafo no conexo. (9)

Teorema: el algoritmo de Prim tiene un tiempo de ejecución O (n log n) en el peor de los casos.

Algoritmo de Ford-Fulkerson

El objetivo de este algoritmo es encontrar caminos en los que se vaya incrementando el flujo, hasta lograr el flujo máximo. Su concepción principal es hallar la ruta de penetración con flujo positivo neto que una el nodo origen y el nodo destino.

Sea (V, A, w) con V vértices, A aristas y w peso de las aristas, una red con una única fuente s y un único sumidero t; w (α) es la capacidad de α perteneciente a la arista A. Un flujo f es viable si f (α) <= w (α) para todo α perteneciente a la arista A. Se trata de hallar un flujo viable con el valor máximo posible. En una red con fuente j y sumidero k único el valor máximo que puede tomar un flujo variable es igual a la capacidad mínima que puede tomar un corte.

En otras palabras el algoritmo Ford-Fulkerson, es un algoritmo iterativo, que comienza con f (u,v)=0 para cada par de nodos y en cada iteración se incrementa el valor del flujo buscando un camino de aumento. Este proceso se va a repetir hasta no encontrar un camino de aumento.

Teorema de Ford-Fulkerson (1962): en cualquier red, el flujo máximo que fluye de la fuente al destino es igual a la capacidad del corte mínimo que separa a la fuente del destino. (10)

Algoritmo de Kruskal

Este algoritmo forma un árbol que incluye todos los vértices donde las aristas poseen menor peso. En el caso de que el grafo no sea conexo, entonces busca un bosque expandido mínimo.

Este algoritmo es de tipo *greedy*, ya que a cada paso, este selecciona el arco más barato y lo añade al sub-grafo. Este tipo de algoritmos pueden no funcionar para resolver otro tipo de problemas, por ejemplo para encontrar la ruta más corta entre los nodos a y b.

Entonces, se puede decir, que este algoritmo trabaja, creando dos conjuntos, uno con los árboles, formando un bosque, en el que los vértices del grafo constituyen un árbol separado, y otro que tenga a todas las aristas del grafo. Mientras el conjunto que contiene las aristas no es vacío, se van eliminando las aristas de peso mínimo de dicho conjunto y en el caso que esa arista conecta dos árboles diferentes se añade al bosque uniendo los dos árboles en un solo árbol y en caso contrario se desecha la arista. Como resultado final de este algoritmo se tiene un bosque con un solo componente, formando así un árbol de expansión mínimo en el grafo. Por lo expuesto hasta aquí se puede apreciar que el principal objetivo del algoritmo de Kruskal es generar disimiles bosques, en los que se encuentran las distintas aristas que va seleccionando en su ejecución hasta formar el árbol de expansión mínima. (11)

Teorema: debido a que las aristas se ordenan por su peso y después se eliminan las de menor peso en un tiempo de ejecución constante, este algoritmo posee complejidad O(n log n).

Algoritmo de Bellman-Ford

Dentro de los algoritmos del camino más corto también se encuentra el Bellaman-Ford. La eficiencia del mismo, radica en que se ejecute en un grafo dirigido ponderado, en el que algunas de sus aristas pueden llegar a tener peso negativo, a diferencia del algoritmo Dijkstra que será analizado más adelante que no puede tener aristas con peso negativo. A pesar de que Bellaman-Ford es bastante parecido al algoritmo Dijkstra, utiliza más cantidad de tiempo para resolver el mismo problema y se usa generalmente cuando existen arcos con peso negativo.

Soluciona el problema de la ruta más corta o camino mínimo desde un nodo origen, de un modo más general que el algoritmo de Dijkstra, ya que permite valores negativos en los arcos. El algoritmo devuelve un valor booleano si encuentra un circuito o lazo de peso negativo. En caso contrario calcula y devuelve el camino mínimo con su coste.

Este algoritmo posee dos variantes, las cuales le posibilita que sea usado para cuando existen arcos con pesos negativos. Una de estas variantes es la versión no optimizada utilizada en grafos con ciclos negativos. La otra versión es la optimizada, utilizada también en grafos, a diferencia de que en este grafo no existan ciclos de coste negativo. Existen variantes del algoritmo que son utilizadas en el Protocolo de encaminamiento de información (RIP), el cual es de gran importancia en las redes, dispositivos *router* IP y en otros que permiten la comunicación a diario con los servicios de internet. (12)

Algoritmo heurístico A*

Los algoritmos heurísticos son aquellos que cuando la solución no se determina de forma directa, se realizan una serie de ensayos, pruebas y reensayos. El método que ellos usan consiste en generar candidatos de soluciones posibles de acuerdo a un patrón dado; luego los candidatos son sometidos a pruebas de acuerdo a un criterio que caracteriza a la solución. Si un candidato no es aceptado, se genera otro; y los pasos dados con el candidato anterior no se consideran.

El algoritmo de búsqueda A* se encuentra en el grupo de los algoritmos de búsqueda en grafos y comienza, a partir de la creación de una lista con el nodo raíz. Mientras que dicha lista no esté vacía y no se alcance la meta y el primer elemento de la lista es la meta, se expanden todos los nodos que tengan menor costo que el nodo meta; de lo contrario se elimina el nodo que se encuentra en la

primera posición de la lista, luego se agregan sus hijos a la lista y se ordenan los elementos de acuerdo al costo y se eliminan los que estén repetidos de mayor costo. Por último, se expanden todos los nodos que posean menor costo que el nodo meta.

El algoritmo A* facilita los problemas en los que encontrar la mejor solución es el objetivo fundamental. Su coste en espacio y tiempo en el caso medio es mejor que los algoritmos de búsquedas a ciegas en el caso de que la heurística sea la adecuada. A pesar de ello, el algoritmo A* en ocasiones no puede dar respuesta a algunos problemas de búsquedas. Este algoritmo como muchos algoritmos de búsquedas posee un tiempo de ejecución exponencial. (13)

Algoritmo de Dijkstra

El algoritmo Dijkstra se utiliza para solucionar el problema de los caminos más cortos de un vértice, al que denominamos origen, a los restantes vértices del grafo. Es conocido como un algoritmo de tipo *greedy*, que a su vez se le conoce como algoritmos ávidos, glotones o voraces. Estos algoritmos trabajan por etapas, tomando en cada una de ellas la solución mejor (óptimo local) sin considerar las consecuencias futuras. El óptimo encontrado en una etapa puede posteriormente ser modificado si surge una mejor solución.

Con el uso de este algoritmo se pueden garantizan excelentes resultados. Dentro de estos resultados se encuentran la disminución sensible del riesgo de que existan errores, la racionalización del esfuerzo intelectual y las habilidades y conocimientos al propiciar que se consideren caminos y rutas que de otra manera pudieran pasar desapercibidos; entre otros resultados. No obstante, su uso también puede provocar la lentitud en la repercusión del proceso recursivo y la dificultad de aplicar el método a situaciones en las que la solución al problema general no se deriva de la suma directa y simple de una parte del problema.

Teorema: para una entrada consistente en un grafo simple, conexo, con pesos y n vértices, el algoritmo de Dijkstra tiene un tiempo de ejecución O (n2) en el peor de los casos. (14)

Comparación entre algoritmos

El conjunto de algoritmos analizados anteriormente permiten sin duda alguna la resolución de problemas complejos de búsqueda de caminos mínimos. Algunos de ellos, por sus características específicas permiten la resolución de problemas de cálculo de rutas, otros no. Debido a esto para seleccionar el más adecuado para la resolución del problema de la investigación en curso se deben tener presente algunos de sus elementos fundamentales.

En este aspecto es necesario resaltar que estos algoritmos se diferencian por dar solución a diferentes problemas clásicos, es decir, los problemas referentes a: un origen y muchos destinos, un origen y un solo destino, entre otros ejemplos. Así como en la complejidad de cada uno de ellos, característica esta que también manifiesta la diferencia entre cualquier algoritmo matemático.

A continuación se muestra una tabla comparativa que apoya los elementos mencionados:

Algoritmo	Grado de Complejidad	Problemas clásicos que resuelven
Dijkstra	O (n2)	Va de uno a los restantes vértices del grafo.
Floyd	O (n3)	Problema de los caminos más cortos entre los pares de vértices.
Prim	O (n log n)	Problema del Árbol de Expansión Mínima
A*	O (n log n)	Encuentra un elemento dado una estructura de datos y localiza el menor camino entre dos nodos.
Kruskal	O (n log n)	Problema del Árbol de Expansión Mínima.

Tabla 1 Comparación de algoritmos de búsqueda.

Luego de realizar este estudio, se identifican como posible solución al problema de la investigación los algoritmos A* y Dijkstra. A pesar de que el algoritmo A* cuenta con un tiempo de ejecución O (n log n), menor que el del Dijkstra O (n2), tiene la desventaja de que no siempre recorre todos los vértices del grafo corriendo el riesgo de que no encuentre la solución más adecuada. Además es un algoritmo usado para grafos en el que el número de nodos es muy grande. Teniendo en cuenta que el SIGUCI no cuenta con un grafo de mucha extensión, se seleccionó el algoritmo Dijkstra como el encargado de realizar el cálculo de rutas en el módulo a desarrollar, ya que permite encontrar el camino más corto entre un nodo origen y el resto de los nodos de manera más efectiva, debido a que el mismo recorre todos los vértices del grafo, volviéndose lento en grafos muy extensos, no siendo este el caso, pero garantizando así que siempre encuentre la solución requerida. Además, garantiza excelentes resultados entre los que se encuentran la disminución sensible del riesgo de que existan errores.

1.4 Soluciones Existentes

En el actual epígrafe se realiza un estudio de algunos de los sistemas existentes a nivel mundial que cuentan con funcionalidades que permiten el cálculo de la ruta más corta, así como las herramientas y algoritmos de búsqueda que utilizan para el desarrollo de estas funcionalidades.

ViaMichelin

Es un API (Application Programming Interface) perfecta para todas las empresas que deseen desarrollar aplicaciones relacionadas con la geolocalización. Sin necesidad de conocimientos técnicos elevados, esta solución permite una integración rápida, fácil y a la medida de las siguientes funcionalidades: (15)

- Generación de mapas.
- Cálculo de rutas para vehículos.
- Cálculo de rutas para peatones.
- Búsqueda por proximidad de alto rendimiento: cálculo de distancias por carretera o en línea recta.

ViaMichelin es útil a la hora de hacer viajes largos. Te permite ingresar un origen y un destino, y se obtiene detalladas instrucciones de viaje (ej. salida de la autopista, etc.), utilizando algoritmos como Dijkstra y A* para realizar el cálculo de las rutas. (15)

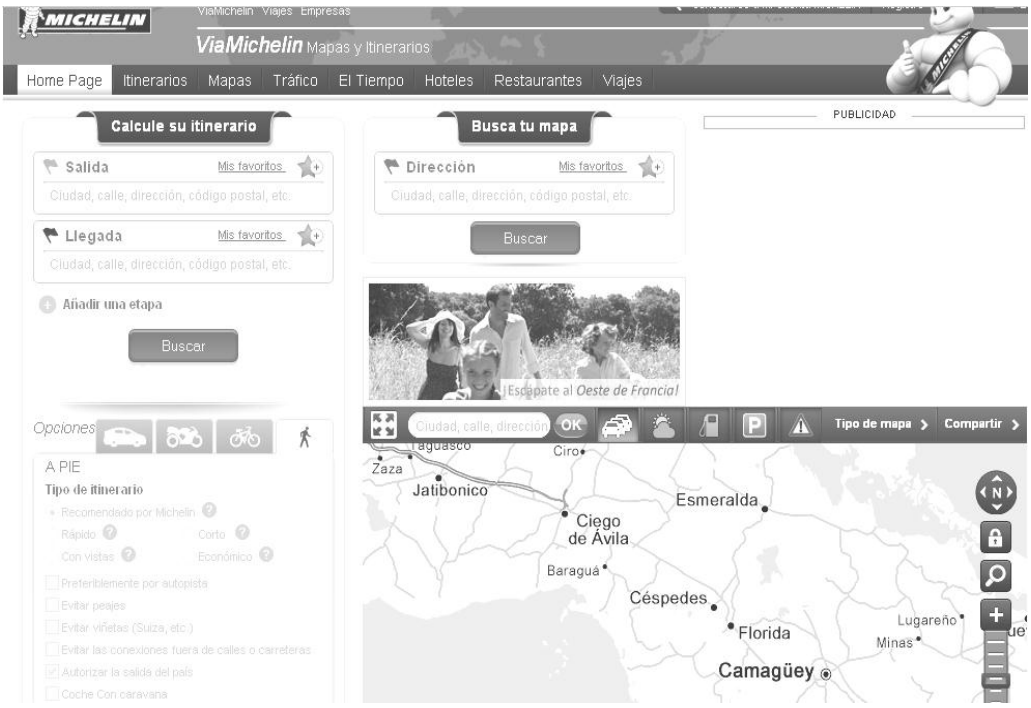


Figura 1 Interfaz web ViaMichelin. (15)

Bizitel

Es un sistema web de información geográfica adaptado para el cálculo dinámico de rutas óptimas en entornos urbanos, donde los usuarios pueden planificar sus desplazamientos.

Entre las utilidades que Bizitel, se encuentran:

- El trazado de la ruta sobre un mapa de Google Maps.
- Listado de instrucciones de circulación y su posicionamiento sobre el mapa.
- Ofrece información sobre ubicación de tiendas y talleres de bicicletas.
- Para el cálculo de la ruta se pueden seleccionar dos tipos de modalidades: ruta tranquila (realizada bajo criterios de circulación ciclista; es decir, se prioriza el uso de carriles-bici); o ruta rápida (donde todos los viales tienen la misma consideración).

El funcionamiento del calculador consiste en un algoritmo que resuelve el camino entre dos nodos de un grafo que representa la ciudad. La cual se modela mediante un grafo dirigido y con peso. Sobre el cual un algoritmo Dijkstra es el encargado de resolver el camino entre dos nodos dados utilizando la información almacenada en la base de datos. (16)

La ruta calculada se representa sobre un mapa de Google Maps. Toda la información y las utilidades que se muestran sobre el mapa se han desarrollado a través de la API de Google Maps.

Entre las herramientas utilizadas para el desarrollo de la aplicación se encuentran:

- Lenguajes de programación: JavaScript, PHP.
- Administrador de bases de datos: PHPmyAdmin.
- Sistema gestor de bases de datos: MySQL.
- Servidor de base de datos: Apache.

Figura 2 Interfaz web Bizitel. (16)

GeneSIG

GeneSIG es una plataforma soberana desarrollada en la UCI para la creación de SIG, la misma tiene como objetivos fundamentales:

- Permitir la representación geoespacial de la información asociada a cualquier negocio que lo requiera.
- Proporcionar servicios de acceso a la información geográfica, para su consulta, análisis y visualización, mediante una interfaz de usuario sencilla y de fácil manejo que pueda ser utilizada por usuarios no especializados en tecnología SIG.
- Integrar la información socioeconómica existente (recursos humanos, activos fijos, entidades de servicios, lugares de interés, etc.) con la información geográfica asociada.

Otro aspecto a destacar es que está implementada sobre herramientas y tecnologías libres, lo que facilita su uso y manejo de acuerdo a las necesidades específicas de sus usuarios. Además de contar con una interfaz bastante sencilla y de fácil manejo, permitiendo la personalización y uso de todos los componentes que la integran. Entre las tecnologías y herramientas fundamentales que le dan soporte se destacan:

- CartoWeb: Framework de Desarrollo.
- Mapserver: Servidor de mapas.
- PHP: Lenguaje de programación del lado del servidor.
- Ext JS: Lenguaje de programación del lado del cliente.
- PostgreSQL: Sistema Gestor de Base de Datos.
- PostGIS: Extensión espacial para PostgreSQL.
- pgRouting: Herramienta de enrutamiento para PostGIS/PostgreSQL.

Esta plataforma cuenta, entre la amplia gama de funcionalidades que implementa, con un módulo de Análisis de Rutas, para ello utiliza la herramienta pgRouting que brinda funciones que tienen implementados algunos de los algoritmos analizados en este capítulo para realizar el cálculo de rutas. Entre las operaciones que se pueden realizar en este módulo se encuentran la de capturar puntos de origen y destino, así como la de calcular, visualizar y eliminar rutas.

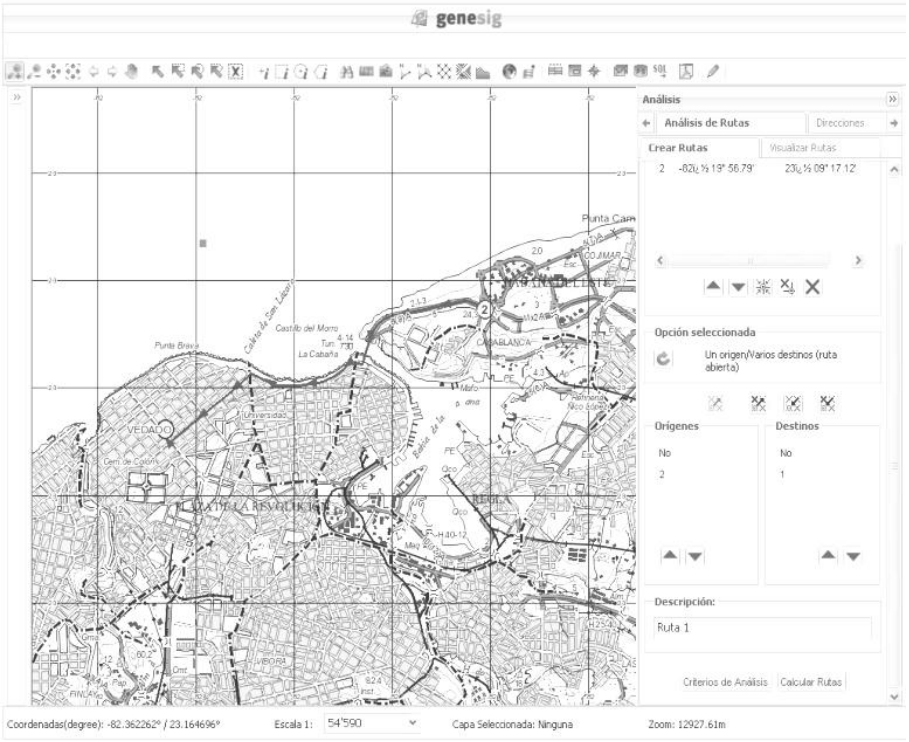


Figura 3 Interfaz web GeneSIG.

1.5 Conclusiones Parciales

En el capítulo actual se han explicado términos y conceptos asociados al tema de la investigación los cuales permiten un mayor entendimiento del mismo. Se realizó un estudio de los diferentes algoritmos de búsqueda existentes lo que permitió seleccionar el que se empleará como solución al problema en cuestión. También se realizó un estudio de varios sistemas similares, tanto a nivel internacional como nacional mediante el cual se puede afirmar que los mismos aportan elementos que deberían tenerse en cuenta en el desarrollo del sistema como es el uso de algoritmos para el cálculo de la ruta más corta.

El estudio de las soluciones existentes contribuye además a la solución del problema de la investigación ofreciendo características a tener en cuenta en el desarrollo del software como son: algoritmos que utilizan para el cálculo de rutas así como las herramientas que permiten el uso de los mismos, la manera de capturar de los parámetros para crear las rutas y herramientas para la representación de las mismas en la web. La solución que más se asemeja y aporta elementos al problema planteado es el módulo de Análisis de Rutas de la plataforma GeneSIG, el cual hace uso de la herramienta pgRouting y el algoritmo Dijkstra que esta implementa, este último seleccionado para darle solución a la investigación en curso.

A pesar de lo anteriormente planteado esta plataforma no constituye una solución directa al problema de la investigación debido fundamentalmente a las herramientas y tecnologías que sirvieron de base para su implementación. Las utilizadas por el SIGUCI y por consecuencia las que emplea el módulo, brindan mayores funcionalidades y facilidades de desarrollo. Algunas se emplean en versiones superiores a las usadas por GeneSIG posibilitando esto el aprovechamiento de las nuevas facilidades que estas brindan, permitiendo mayor usabilidad y rapidez de la aplicación. No obstante, se toma esta investigación como base para determinar el desarrollo de un módulo de cálculo de rutas para el SIGUCI.

Capítulo II: Tecnologías y herramientas

2.1 Introducción

En la actualidad existen innumerables herramientas que se emplean en el proceso de desarrollo de SIG. Cada vez son más sofisticadas y fáciles de usar, así como las tecnologías, lenguajes de modelado, metodologías y lenguajes de programación con el fin de obtener un producto final de mejor calidad.

En este capítulo se identifican y argumentan las diferentes tendencias y tecnologías actuales a utilizar en la construcción de la solución, a través de un estudio detallado de las mismas. Mediante este análisis se logra determinar la necesidad e importancia de su manejo y utilización.

Dado que la solución propuesta es un módulo para el sistema SIGUCI se utilizará la arquitectura base propuesta para el mismo. Utilizando herramientas, tecnologías y metodologías definidas con anterioridad por parte de los desarrolladores de este sistema.

2.2 Metodología de desarrollo de software

Una metodología de desarrollo de *software*, es aquella que hace posible la planificación, organización y construcción de un sistema o proyecto, con independencia de su temática o complejidad. Actualmente estas metodologías son una guía en el proceso de desarrollo de las aplicaciones informáticas, permitiendo que se obtengan resultados con la mayor calidad, rapidez y eficiencia posible, para evitar cometer errores futuros. (17)

2.2.1 RUP

Proceso Unificado de Desarrollo (Rational Unified Process, RUP) es una de las metodologías pesadas o tradicionales más conocidas y utilizadas. Ésta es robusta y precisa, permite controlar y documentar el desarrollo del *software*, eliminando los riesgos que puedan existir en este, propiciando en todo momento un enfoque de trabajo bien estructurado, así como la asignación correcta y óptima de tareas y responsabilidades. Para guiar el proceso de desarrollo del *software* emplea nueve flujos de trabajo organizados en cuatro fases de desarrollo. Las cuatro fases de desarrollo que definen el ciclo de vida son: (18)

Inicio: tiene como objetivo establecer la visión del proyecto o aplicación y lo que se quiere realizar.

Elaboración: es donde se define la arquitectura y los elementos base para la implementación.

Construcción: es donde se implementa y se va obteniendo una solución inicial parcial.

Transición: es donde se adquiere el producto acabado y definido, incluyendo su mantenimiento.

Flujos de trabajo:

- Modelado del Negocio.
- Captura de Requisitos.
- Análisis y Diseño.
- Implementación.
- Prueba.
- Despliegue.
- Gestión de Cambios y Configuración.
- Gestión de proyectos.
- Entorno.

RUP presenta tres características fundamentales, la primera es que está guiado por casos de usos que describen las funcionalidades del sistema que el usuario desea obtener, permitiendo la trazabilidad entre los artefactos generados en el proceso de desarrollo. La segunda característica es que esta metodología está centrada en la arquitectura, por lo cual permite una mejor organización y estructura de las partes del sistema, teniendo en cuenta elementos de calidad, rendimiento, reutilización y flexibilidad. Finalmente, su tercera característica es que es iterativo e incremental, puesto que permite obtener un incremento que produce un crecimiento en el producto, para ir mejorando los resultados. (19)

2.3 Sistemas Gestores de Base de Datos (PostgreSQL v9.1)

El PostgreSQL es un Sistema Gestor de Base de Datos (SGBD) que posee características significativas, entre las que se pueden incluir subconsultas, se puede usar, modificar y distribuir de forma gratuita debido a la liberación de su licencia. Es un sistema de gestión de bases de datos objeto-relacional tiene su código fuente disponible. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. (20)

Ventajas:

- ✓ PostgreSQL ha sido diseñado y creado para tener un mantenimiento y ajuste mucho menor que otros productos, conservando todas las características, estabilidad y rendimiento.

- ✓ Puede estar en funcionamiento varios años bajo una alta actividad y no presentar fallos en el servicio.
- ✓ PostgreSQL está disponible en la mayoría de los sistemas Unix (34 plataformas en la última versión estable), y ahora en versión nativa para Windows.

2.3.1 PostGIS v1.6

PostGIS es un módulo que añade soporte de objetos geográficos a la base de datos objeto-relacional PostgreSQL, convirtiéndola en una base de datos espacial para su utilización en Sistemas de Información Geográfica. Se publica bajo la Licencia pública general de GNU. (21)

Entre sus principales características se destacan:

- Alto rendimiento, robusta base de datos espaciales basado en PostgreSQL.
- Simples características para SQL.
- Ofrece representaciones espaciales de los tipos de geometría (puntos, líneas, polígonos)
- Soporte para operaciones espaciales comunes y avanzadas como la creación de la geometría y la conversión, reproyección, *buffer*, convexa, la generalización, la unión, entre otras.
- Geodésica de apoyo para las mediciones en todo el mundo.
- Línea de comandos y herramientas gráficas para la gestión flexible.

PostGIS está ampliamente reconocida como una base de datos espacial de *back-end* de *software* cliente y servidor, incluyendo:

- Servidor de código abierto: GeoServer, Mapserver, Mapnik, Deegree, SharpMap
- Escritorio de código abierto: GRASS, QGIS, uDig, gvSIG
- Propiedad del servidor: ArcServer, Enterprise iónicos, MapDotNet servidor
- Propiedad de escritorio: ArcGIS, Manifold, Caja fuerte FME, Cadcorp SIS, MapInfo Professional

2.3.2 pgRouting v1.05

pgRouting es una extensión de PostGIS la cual añade funcionalidades de ruteo a PostGIS/PostgreSQL. Además, ofrece una gran variedad de algoritmos para la búsqueda del camino más corto. Es desarrollado y mantenido por Georepublic y está disponible bajo la licencia GPLv2 y es apoyado por una creciente comunidad de individuos, empresas y organizaciones. (22)

pgRouting proporciona funciones para:

- La ruta más corta Dijkstra: algoritmo de enrutamiento sin heurística.

- La ruta más corta A*: el enrutamiento de grandes conjuntos de datos (con heurística).
- La ruta más corta estrella fugaz: de enrutamiento con restricciones de giro (con heurística).
- Problema del agente viajero (TSP).
- Distancia de conducción de cálculo (isolíneas).

Las ventajas del enfoque de enrutamiento de bases de datos:

- Los cambios de datos se pueden reflejar de forma instantánea a través del motor de enrutamiento. No hay necesidad de pre cálculo.
- El "costo" de parámetros se puede calcular de forma dinámica a través de SQL y su valor puede venir de varios campos o tablas.

Dado a sus características, pgRouting es el utilizado por la mayoría de los *frameworks* de desarrollo de SIG como herramienta fundamental para el cálculo de la ruta más corta entre dos nodos de un grafo.

2.4 Lenguaje de programación del lado del servidor (PHP5)

El lenguaje de programación PHP ofrece soporte para varios servidores web, es un lenguaje multiplataforma, contiene una sintaxis clara y bien definida, es sencillo de aprender y utilizar, permite las técnicas de Programación Orientada a Objetos (POO). (23)
Ofrece una solución simple y universal para las páginas dinámicas de la web de fácil programación, perceptiblemente más fácil de mantener. (24)

Ventajas:

- Es un lenguaje multiplataforma.
- Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una Base de Datos.
- El código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.

2.5 Framework de desarrollo (Symfony v1.4)

Symfony es un *framework* diseñado para optimizar el desarrollo de las aplicaciones web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web

compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de la aplicación.

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas como Unix, Linux, etc., además de Windows. A continuación se muestran algunas de sus características. (25)

Symfony se diseñó para que se ajustara a los siguientes requisitos:

- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y Linux).
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios de *phpDocumentor* y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

2.6 Lenguaje de programación del lado del cliente (ExtJS v4.0)

Ext JS es una librería JavaScript ligera y de alto rendimiento, compatible con la mayoría de navegadores que permite crear páginas e interfaces web dinámicas. Permite realizar aplicaciones web enriquecidas basándose en tecnología AJAX (Asynchronous JavaScript And XML), JSON (JavaScript Object Notation), DHTML (Dynamic HTML) y DOM (Document Object Model).

Con Ext JS, se pueden desarrollar aplicaciones web multiplataforma con facilidad. El modelo de componente mantiene su código bien estructurado por lo que incluso las aplicaciones más grandes pueden ser de fácil mantenimiento. Brinda la posibilidad de utilizar un gran número de componentes visuales que mejoran considerablemente la calidad de las aplicaciones y validaciones de formularios de todo tipo, basándose en expresiones regulares y tipos de datos. Trae implícitos componentes como

vista en árboles, arrastrado y soltado, cambio de tamaño de imágenes, rejillas, paginado, agrupado de objetos, tabs, asistentes, entre otros muchos. (26)

2.7 Lenguaje de modelado (UML v2.0)

El Unified Modeling Language (UML) es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener y controlar la información sobre tales sistemas. (27)

Está pensado para usarse con todos los procesos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. (28)

2.8 Herramienta CASE (Visual Paradigm v8.0)

Visual Paradigm soporta el ciclo de vida completo del desarrollo del *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El *software* de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste (29). Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Es muy fácil de usar y soporta la última notación UML 2.1, ingeniería inversa, generación de código, importación desde Rational Rose, exportación/importación XMI, integración con MS Visio, *plug-in*, integración IDE con Visual Studio, IntelliJ IDEA, Eclipse, NetBeans y otros. (30)

Características fundamentales:

- Generación de documentación: Brinda la posibilidad de documentar todo el trabajo sin necesidad de utilizar herramientas externas.
- Disponibilidad en múltiples plataformas: Microsoft Windows (98, 2000, XT, o Vista), Linux, Mac OS X, Solaris o Java.

2.9 Entorno de Desarrollo (NetBeans v7.1)

NetBeans es un entorno de desarrollo visual de código abierto. Su aprendizaje se ha convertido en fundamental para quienes están interesados en el desarrollo de aplicaciones multiplataforma. Está disponible para diversos sistemas operativos como Solaris, Windows, MacOS y GNU Linux. Su instalación y actualización es muy simple y una vez instalado se le pueden adicionar módulos que permiten extender sus funcionalidades. Debido a que los módulos pueden ser desarrollados

independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de *software*. (31)

Dado a las características planteadas anteriormente y que permite el trabajo con el *framework* de seleccionado (Symfony) y los lenguajes de programación (PHP, Ext JS) es ideal para el desarrollo del módulo a desarrollar.

2.10 Servidor de mapas (MapServer v5.2.2)

Los servidores de mapas tienen como objetivo acceder a información geoespacial existente, normalmente en servidores, en diferentes formatos y servir dicha información a clientes de mapas a través de protocolos estándares.

Algunas de las funciones que permiten realizar los servidores de mapas son:

- Zoom para alejar o acercar los elementos cartográficos.
- Identificación de atributos alfanuméricos en cada elemento cartográfico.
- Conexión de bases de datos locales a la base de datos remota del servidor de mapas.
- Selección de elementos por combinación de capas o análisis con operadores espaciales de superposición, contención, intersección, entre dos capas.
- Cálculo de rutas óptimas para la navegación de vehículos.

MapServer es una plataforma de Código Abierto para la publicación de datos espaciales y aplicaciones cartográficas interactivas para la web. Funciona en los principales sistemas operativos (Windows, Linux, Mac OS X). (32)

Básicamente ofrece las siguientes características:

- Servicios de mapas (WMS).
- Servicios de geometrías (WFS).
- Servicios de coberturas (WCS).
- Excelente rendimiento.
- Buen nivel de renderizado (generación de imágenes gráficas) de mapas.

2.11 Servidor web (Apache2)

Un servidor web cuenta con un esquema de funcionamiento muy simple, basado en ejecutar infinitamente el siguiente bucle:

- Espera peticiones en el puerto TCP indicado (el estándar por defecto para HTTP es el 80).
- Recibe una petición.
- Busca el recurso.
- Envía el recurso utilizando la misma conexión por la que recibió petición.
- Vuelve al segundo punto.

Apache es un servidor web flexible, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos (HTTP 1.1). Entre sus características destacan (33)

- Multiplataforma.
- Es un servidor de web conforme al protocolo HTTP/1.1.
- Modular: Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con la API de programación de módulos, para el desarrollo de módulos específicos.

2.12 Conclusiones Parciales

Luego de realizar un estudio de las herramientas y metodologías, teniendo en cuenta las definidas previamente para el sistema SIGUCI, se han seleccionado las que más se adecuan a las características del *software* que se desea desarrollar, garantizando un producto de una gran calidad, en el tiempo requerido y ajustándose a las necesidades del cliente.

El sistema emplea un conjunto de tecnologías que permiten una correcta comunicación entre el cliente y el servidor, de ahí que emplea PHP5 como lenguaje de programación del lado del servidor junto con el *framework* Symfony v1.4, así como Ext JS v4.0 como lenguaje de programación del lado del cliente ya que brinda la posibilidad de utilizar un gran número de componentes visuales que mejoran considerablemente la calidad de la aplicación y validaciones de formularios de todo tipo. El sistema en su conjunto se ejecuta a través del servidor de mapas MapServer v5.2.2, brindándole múltiples opciones para el trabajo con mapas y empleando como sistema gestor de base de datos PostgresSQL v9.1 ya que posee el módulo PostGIS v1.6 que cuenta con muchas funcionalidades para la gestión de datos espaciales, el cual al añadirle la extensión pgRouting v1.05, le brinda funciones para el cálculo de rutas, entre ellos el Dijkstra, algoritmo seleccionado para utilizar en la solución del problema planteado. Se utiliza UML v2.0 como lenguaje de modelado posibilitando el completo proceso de modelado del *software*.

Capítulo III: Descripción de la propuesta de solución

3.1 Introducción

En este capítulo se formula la propuesta de solución para el problema existente utilizando la metodología RUP para la planificación, investigación y diseño de la herramienta a desarrollar. Se presenta el modelo de dominio y se realiza la especificación de requisitos, así como el modelo de casos de uso.

3.2 Modelo de Dominio

El modelo de dominio facilita la identificación de los elementos principales del sistema a desarrollar, lo cual posibilita un mejor entendimiento al usuario mostrando los principales conceptos que se manejan. Este modelo permite la representación de conceptos mediante diagramas UML donde figuran las principales clases y roles del sistema. (Ver Figura 4)

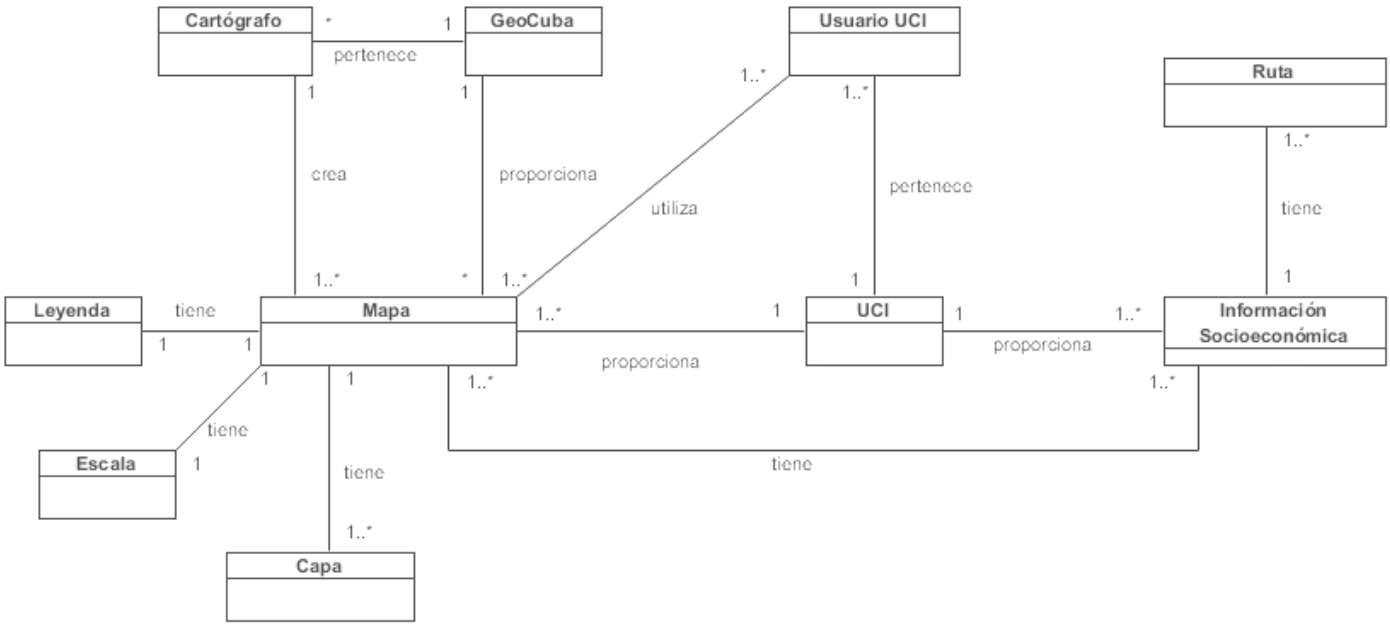


Figura 4 Modelo de clases del dominio.

Principales conceptos tratados en modelo de dominio

Cartógrafo: Persona que se dedica profesionalmente a la realización de cartas geográficas, al estudio y elaboración de mapas.

GeoCUBA: Es un grupo empresarial que se dedica a la elaboración, producción y venta de planos,

mapas y cartas náuticas con diversos fines, así como a la realización de Estudios Geográficos, de Impacto Ambiental, e investigaciones científicas, entregando a sus clientes, productos informativos con una alta calidad y fiabilidad.

Mapa: Es una representación gráfica y métrica de una porción de territorio sobre una superficie bidimensional, generalmente plana, pero que puede ser también esférica como ocurre en los globos terráqueos. El mapa que tenga propiedades métricas significa que ha de ser posible tomar medidas de distancia, ángulos o superficies sobre él y obtener un resultado aproximadamente exacto. Sobre este se van a tematizar los reportes sobre violaciones a la seguridad informática en la universidad.

Escala: Relación entre la distancia que separa dos puntos en un mapa y la distancia real de esos dos puntos en la superficie terrestre. En los mapas, la escala puede expresarse de tres modos distintos: en forma de proporción o fracción, con una escala gráfica o con una expresión en palabras y cifras. Cuanto mayor es la escala, más se aproxima al tamaño real de los elementos de la superficie terrestre. Los mapas a pequeña escala generalmente representan grandes porciones de la Tierra y por tanto, son menos detallados que los mapas realizados con escalas más grandes. La relación matemática entre las dimensiones del mapa, carta o plano y la superficie terrestre que representa.

Leyenda: Explicación de los símbolos, los colores, las tramas y los sombreados empleados en un mapa; suele encontrarse a pie de página o en un recuadro, situado en sus márgenes o bien en su dorso. Los símbolos empleados en los mapas pueden llegar a contener un gran volumen de información, que por su facilidad de lectura permiten una rápida interpretación.

Capa: Las capas son una forma de organizar la información temática para la elaboración de los SIG. Son transparencias colocadas a criterio del autor para facilitar la manipulación de la información. El sistema permite activar o no las capas disponibles e incluso variar su orden de acuerdo al orden de prioridad o posible combinación que se le puede dar a la información disponible.

Información socioeconómica: Es un conjunto organizado de datos procesados referentes al aspecto social y económico de cualquier lugar de interés del país.

Usuario UCI: Persona que habita las instalaciones de la Universidad de las Ciencias Informáticas (UCI) y que necesite trabajar o consultar algún tipo de información incluida en un mapa.

UCI: La universidad es la encargada de solicitar un servicio determinado utilizando un mapa y que proporcione la información socioeconómica referente a la misma.

Ruta: Camino seleccionado por el usuario para ir de un lugar a otro.

3.3 Especificación de requisitos

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir, sin tomar en consideración ningún tipo de restricción física, de manera que se mantienen invariables sin importar con qué propiedades o cualidades se relacionen. (34)

3.3.1 Requisitos funcionales

- RF1: Crear rutas por selección de puntos para vehículos.

Con este requisito se persigue que el usuario pueda una vez seleccionado un origen y un destino en el mapa, ver la ruta más corta entre los dos puntos determinado.

- RF2: Crear rutas por selección de estructuras para vehículos.

Con este requisito se persigue que el usuario pueda una vez seleccionada una estructura origen y otra destino, ver la ruta más corta entre ellas.

- RF3: Crear rutas por selección de lugar de interés para vehículos.

Con este requisito se permite que el usuario pueda una vez seleccionado un punto origen y el lugar de interés, ejemplo: edificio docente, cajero, nodo, etc., ver la ruta más corta entre el origen y el lugar de interés más cercano a este.

3.3.2 Requisitos no funcionales

A continuación se describirán los requisitos no funcionales que son aquellas propiedades que el sistema debe de tener y que aseguren que el producto sea usable, confiable, rápido.

Usabilidad

* El sistema debe permitir al usuario llegar de manera rápida y efectiva a la información buscada, además, ser una interfaz de manejo cómodo que posibilite a los usuarios sin experiencia una rápida adaptación.

Eficiencia

* El tiempo de respuesta estará dado por la cantidad de información a procesar, entre mayor cantidad de información mayor será el tiempo de procesamiento.

Restricciones de diseño

* Diseño sencillo, con no más de 4 entradas de datos por parte del usuario en cada funcionalidad, donde no sea necesario mucho entrenamiento para utilizar el sistema.

- Se deben emplear los estándares establecidos (diseño de interfaces, base de datos y codificación).

Interfaz

Interfaz de usuario

- La interfaz debe ser sencilla con colores suaves a la vista y sin cúmulo de imágenes u objetos que distraigan al cliente del objetivo.

Interfaces de *software*

- La construcción de la aplicación funcionará bajo los conceptos de arquitectura cliente/servidor. Por tanto, la PC cliente debe tener como requisitos mínimos de *software*:

Para las PCs cliente

- Un Navegador como Mozilla Firefox, Zafarí u otro navegador que cumpla con los estándares W3C.

Para los servidores

- Sistemas operativos GNU/Linux.
- Servidor web Apache 2.0 o superior, con módulo PHP5.
- PostgreSQL 9.1 como Sistema Gestor de Base de Datos.
- PostGIS como extensión de PostgreSQL 9.1 como soporte de datos espaciales.
- MapServer 5.2.2 o superior.

Hardware

Para los servidores

- El Servidor de Mapas tenga como mínimo 1GB de RAM y 2GB de disco duro.
- El Servidor de BD tenga como mínimo 2GB de RAM y 10GB de disco duro.
- Procesador 3 GHz como mínimo.

Requisitos de Licencia

- Se debe garantizar que el sistema será *software* libre y por tanto, los componentes de *software* que se utilicen también deberán serlo.

3.4 Modelo de casos de uso del sistema

Luego de haber definido los requisitos funcionales así como los no funcionales con los cuales el sistema debe cumplir se debe mostrar la relación que existe entre los actores y las funcionalidades del

módulo cálculo de rutas, mediante el diagrama de casos de uso del sistema.

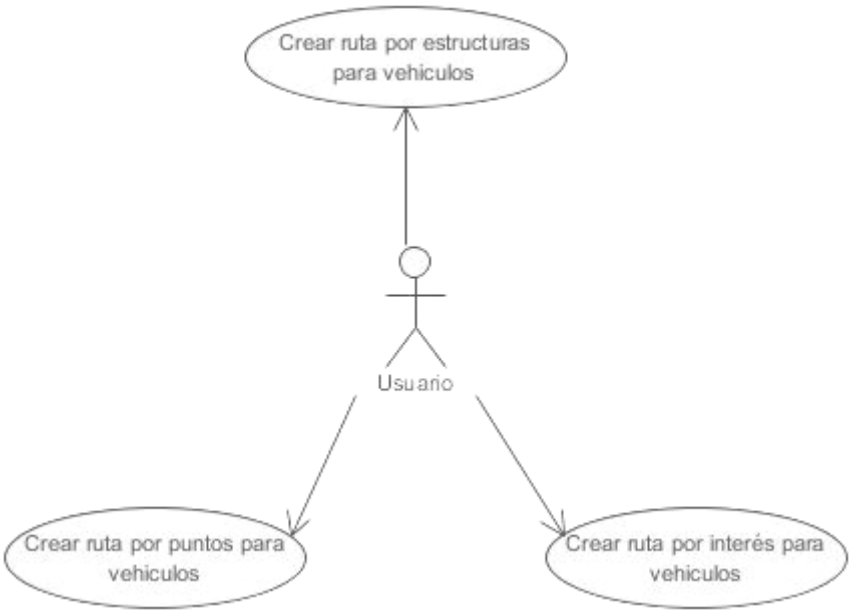
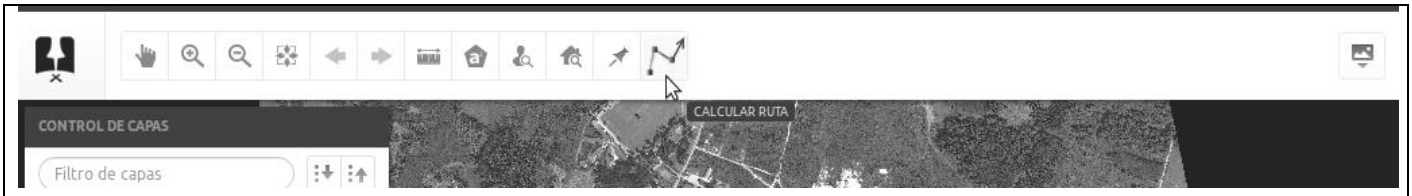


Figura 5 Diagrama de casos de uso del sistema.

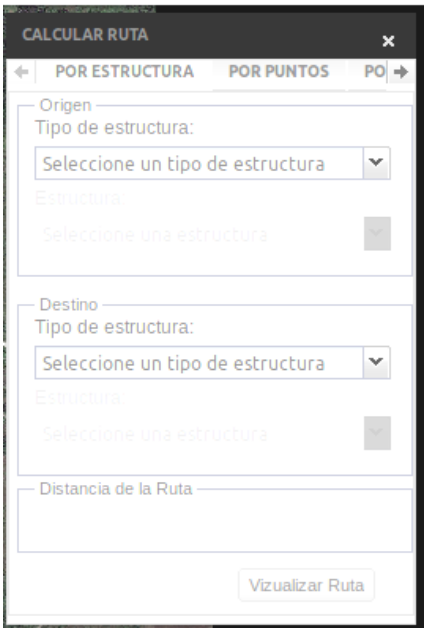
Caso de uso:	Crear ruta por estructuras para vehículos
Actores:	Usuario
Propósito:	Con este caso de uso se persigue que el usuario pueda una vez seleccionada una estructura origen y otra destino, calcular la ruta más corta entre ellas.
Resumen:	El caso de uso se inicia cuando el usuario necesita conocer la ruta más corta de un lugar a otro de la UCI y para ello selecciona dos estructuras, una de origen y otra de destino.
Precondiciones:	
Referencias:	RF2
Prioridad:	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del sistema**
1. El usuario selecciona la opción “CALCULAR RUTA” (ver interfaz 1).	2. El sistema muestra la ventana “CALCULAR RUTA”.
Prototipo de Interfaz	
Interfaz 1	



3 .El usuario selecciona la pestaña "POR ESTRUCTURA".	4. El sistema muestra el contenido de la pestaña (ver interfaz 2).

Prototipo de Interfaz

Interfaz 2



5. El usuario selecciona la estructura de origen y la de destino, mediante comboBox.	6. El sistema activa el botón "Visualizar Ruta" posibilitándole al usuario mostrar en el mapa, la ruta entre el origen y el destino seleccionado previamente.
7. El usuario oprime el botón "Visualizar Ruta".	8. El sistema visualiza la ruta anteriormente determinada por el usuario y muestra la distancia de la misma en metros (Ver Interfaz 3).

Prototipo de Interfaz

Interfaz 3

Flujos Alternos	
Acción del Actor	**Respuesta del sistema**
5.1 El usuario selecciona la misma estructura de origen y destino.	5.2 El sistema muestra un mensaje de error "Las estructuras de Origen y Destino no pueden ser las mismas" (Ver Interfaz 4).
Prototipo de Interfaz	
Interfaz 4 ERROR Las estructuras de origen y destino no pueden ser las mismas.	
Poscondiciones	Se crea y visualiza la ruta seleccionada por el usuario.

Tabla 2 Descripción de caso de uso "Crear ruta por estructuras para vehículos".

3.5 Conclusiones Parciales

En el presente capítulo se logra un mayor entendimiento del flujo actual de los procesos identificados en el desarrollo del módulo, mediante el modelado del negocio estableciendo un lenguaje común entre los desarrolladores y el cliente. Mediante la captura de requisitos se identificaron las características y

cualidades que la solución propuesta debe cumplir, definiéndose los requisitos funcionales y no funcionales. Se logró un mayor entendimiento a través de la descripción de los casos de uso del módulo. Una vez cumplido este objetivo, es posible pasar a la implementación del módulo de cálculo de rutas del SIGUCI, logrando así un producto de mayor calidad.

Capítulo IV: Construcción y Prueba de la solución propuesta

4.1 Introducción

En el presente capítulo se evidencia la construcción del módulo cálculo de rutas del SIGUCI para lo cual se realiza el modelo del diseño que describe las clases principales que se implementan con sus respectivas relaciones y atributos. Se definen los patrones de arquitectura a utilizar para el desarrollo de la solución, así como los patrones de diseño. Se diseña el modelo de la base de datos, además de la creación del diagrama de despliegue, mediante el cual se puede ver la aplicación desde un punto de vista general. Se crea el diagrama de componentes por el que se rige la implementación del *software.* También se realizan las pruebas al sistema permitiendo la validación del correcto funcionamiento de este mediante pruebas de caja negra garantizando la calidad del producto final.

4.2 Diseño del módulo de cálculo de rutas del SIGUCI

4.2.1 Patrón de arquitectura Modelo Vista Controlador (MVC)

Es un patrón de desarrollo que separa la parte lógica de una aplicación de su presentación. Básicamente sirve para separar el lenguaje de programación del HTML lo máximo posible y para poder reutilizar componentes fácilmente (35)

El modelo es el responsable de: (36)

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento,
- Definir las reglas de negocio (la funcionalidad del sistema).
- Llevar un registro de las vistas y controladores del sistema.
- Si se está ante un modelo activo, notificará a las vistas los cambios en los datos.

El controlador es responsable de: (36)

- Recibe los eventos de entrada.
- Contiene reglas de gestión de eventos, del tipo" SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas.

Las vistas son responsables: (36)

- Recibir datos del modelo y mostrarlos al usuario.
- Tienen un registro de su controlador asociado (normalmente porque además lo instancia).

- Pueden dar el servicio de "Actualización" para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

Con el uso del Modelo-Vista-Controlador se logra la independencia de la base de datos. Se reutiliza la lógica de negocio para diferentes clientes o sistemas. Mejora considerablemente la escalabilidad y la flexibilidad. (Ver Figura 6)

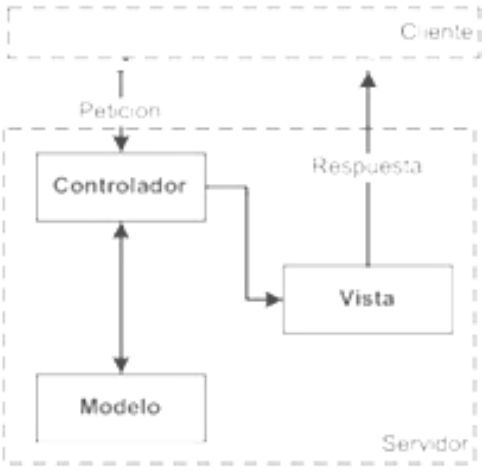


Figura 6 Diagrama de Modelo Vista Controlador.

4.2.2 Patrones de Diseño

Los patrones ayudan a manejar la complejidad del *software*. Cada patrón describe una manera para mejorar el problema que este resuelve. Cuando se encuentra un patrón que soluciona un problema de diseño en particular, no vale la pena invertir tiempo inventando una nueva solución. Si no, que aplicando correctamente el patrón se consigue obtener la solución al problema. (37)

Con el uso de estos se gana tiempo en el mantenimiento del sistema. Son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de *software* y otros ámbitos referentes al diseño de interacción o interfaces.

Patrones GRASP (General Responsibility Asignment Software Patterns)

Con la utilización de este patrón se describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. GRASP es un acrónimo que significa General Responsibility Asignment Software Patterns (Patrones Generales de Software para Asignar Responsabilidades). (38)

Bajo Acoplamiento

El acoplamiento es una medida de fuerza con que un elemento tiene conocimiento o confía en otros elementos. Este patrón es un principio que asigna la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Esto facilita la centralización de actividades. El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. Un error muy común es asignarle demasiada responsabilidad y alto nivel de acoplamiento con el resto de los componentes del sistema.

Este patrón se utilizó con la idea de tener las clases lo menos vinculadas posible entre sí. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. (39)

Ejemplo: permite el uso de los componentes de forma individual, evidenciando el bajo acoplamiento.

Alta Cohesión

La cohesión es una medida de la fuerza con la que se relacionan las clases y el grado de focalización de las responsabilidades de un elemento, cada elemento de nuestro diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto identificable. Una clase con baja cohesión hace muchas cosas no relacionadas o hace demasiado trabajo. En la solución propuesta es necesario controlar la complejidad de cada clase utilizada para mantener un buen comportamiento de las mismas, por esto, las clases que fueron identificadas con una gran cantidad de funcionalidades se dividieron en otras clases, de manera que se repartiera equitativamente el peso de la complejidad, manteniendo además la coherencia de las clases. (39)

Ejemplo: permite la dependencia entre los componentes, evidenciando la alta cohesión.

Patrones GOF (Gang of Four)

Los patrones estructurales son parte de los patrones GOF, describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases, la combinación clases y la formación de estructuras de mayor complejidad. Nos permiten crear grupos de objetos para ayudarnos a realizar tareas complejas.

Abstract Factory (Fábrica abstracta)

Permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando. Trabaja en la creación de familias de objetos permitiendo hacer los sistemas independientes de cómo sus productos se crean,

componen y representan, resultando fácil cambiar de familia de productos. Crear un objeto que sea la composición de muchos otros, que los hacen parte de este objeto padre, es decir el objeto compuesto representa la familia de los demás objetos que lo componen (40).

Singleton (Instancia única)

Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Su diferencia radica en que permite crear objetos que tengan una sola instancia global a la misma, por ejemplo cuando se quiere acceder por un solo punto a una base de datos o se quiere acceder a una impresora desde la red o se quiere que un objeto sea accedido desde múltiples formas (41).

Ejemplo: todas las clases controladoras, son instancias únicas.

4.2.3 Diagrama de clases del diseño

Un diagrama es la representación gráfica de una colección de elementos de modelado. Específicamente el diagrama de clases del diseño representa las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas. A continuación se muestra el diagrama antes mencionado del caso de uso Crear ruta por estructuras para vehículos, diseñado bajo la arquitectura MVC. Posteriormente se realiza una descripción de las clases que se representan para un mejor entendimiento de sus funciones y su interrelación. Se pueden consultar el resto de los diagramas en los anexos de la investigación.

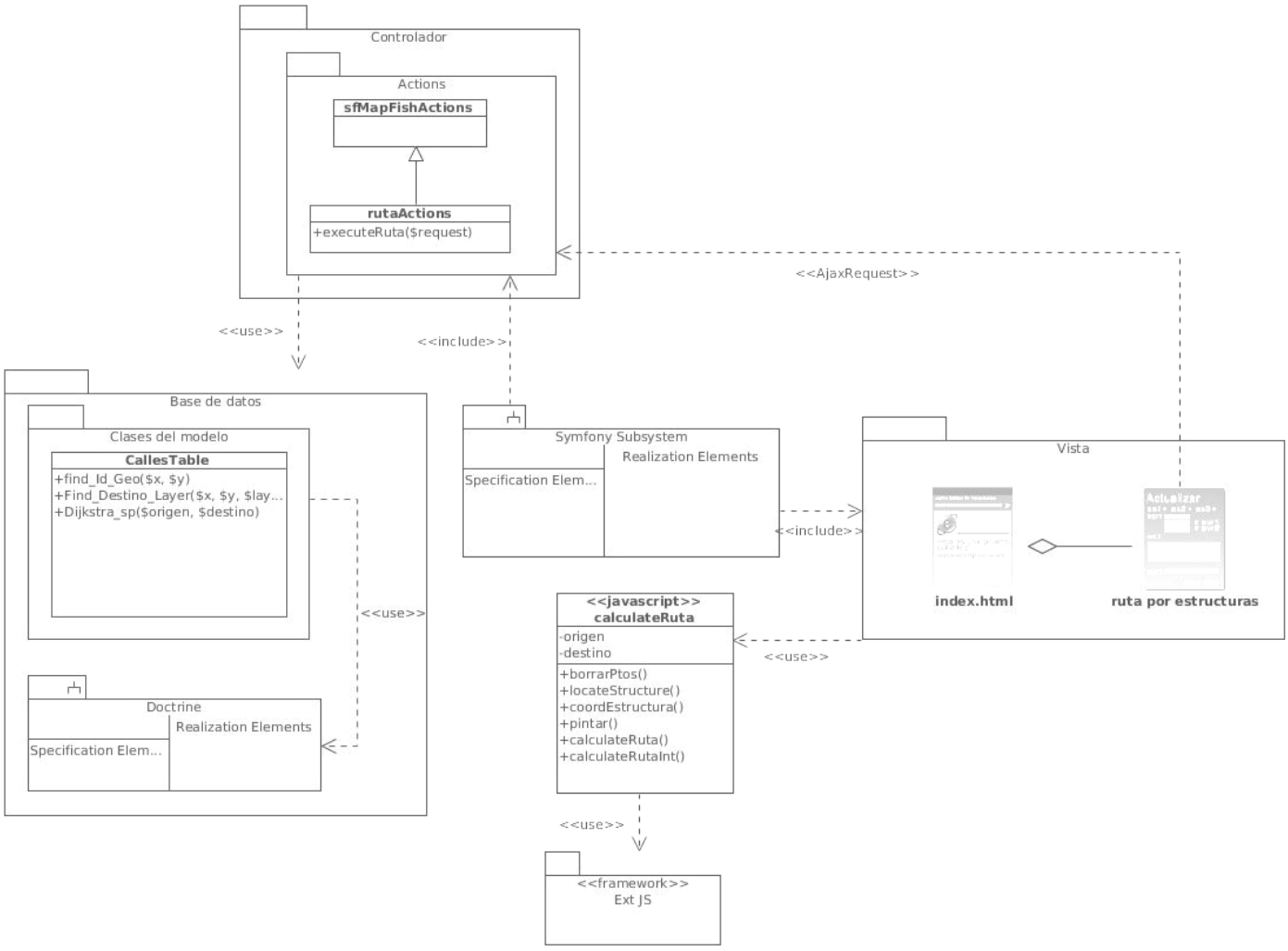


Figura 7 Diagrama de clases del diseño Crear ruta por estructuras para vehículos.

Descripción de clases

Cada uno de los diagramas de clases del diseño que emplean el *framework* Symfony poseen clases que son comunes para la mayoría de ellos, tanto las que son JavaScript como las que son PHP. A continuación serán descritas las clases fundamentales del diagrama y las relaciones de comunicación que se realizan entre ellas.

Clase calculateRuta: clase encargada de construir todos los componentes visuales del formulario, además recibe y procesa las peticiones del usuario.

Clase CallesTable: es la clase encargada de establecer la conexión con el servidor de base de datos para obtener los datos a procesar.

Clase rutaActions: tiene como propósito controlar la realización del CU en sí, recibe las peticiones realizadas por el cliente, gestiona las mismas y envía la respuesta a la vista.

4.3 Diseño de la base de datos

El diseño de la base de datos es un aspecto importante, teniendo en cuenta que brinda persistencia al modelo descrito con anterioridad. Este es capaz de reflejar la estructura del problema en el mundo real, siendo capaz de representar todos los datos esperados, incluso con el paso del tiempo. Evita el almacenamiento de información redundante y proporciona un acceso eficaz a la información.

Modelo de datos

Los modelos de datos aportan la base conceptual para diseñar aplicaciones que hacen un uso intensivo de datos, así como la base formal para las herramientas y técnicas empleadas en el desarrollo y uso de sistemas de información. También puede ser descrito como los requisitos de información y proceso de una aplicación de uso intensivo de datos, construir una representación de la misma que capture las propiedades estáticas y dinámicas requeridas para darle soporte. (Ver Figura 8)

Figura 8 Modelo de Datos del módulo cálculo de rutas.

4.4 Distribución física del sistema

El diagrama de despliegue modela la topología del *hardware* sobre el cual correrá nuestra aplicación y nos indica en donde se ejecutará cada uno de nuestros componentes; muestra las relaciones físicas entre los componentes de *software* y el *hardware* de nuestro sistema. Los diagramas de despliegue muestran la forma en que físicamente lucirá nuestro sistema, sólo deben mostrarse los nodos y componentes que utilizarán en su versión ejecutable. El término original para estos diagramas es *deployment diagram* que en nuestro idioma ha sido traducido como diagramas de distribución, emplazamiento, implantación o despliegue. (42)

El diagrama de despliegue correspondiente al SIGUCI al cual se le desarrolla el módulo de cálculo de rutas, como se muestra a continuación consta con una computadora cliente, en la cual los usuarios

pueden visualizar la ruta interactuando con la aplicación que se encuentra en el servidor. Este servidor es el encargado de responder las peticiones de las computadoras clientes y a su vez está conectado al servidor de mapas, el cual es el encargado de administrar los mapas necesarios para el funcionamiento correcto de la aplicación. Además está conectado a un servidor que provee los datos geográficos para visualizar en el mapa la información sobre la ruta escogida por el usuario. (Ver Figura 9)

Figura 9 Diagrama de despliegue del módulo cálculo de rutas.

4.5 Modelo de implementación

El modelo de implementación es una visión general del módulo cálculo de rutas donde se describen los elementos físicos del mismo y sus relaciones. Está constituido por una colección de componentes que representan todos los tipos de elementos de *software* que forman parte de la aplicación.

Los diagramas de componentes ilustran las piezas del *software* que conformarán un sistema. Un diagrama de componentes tiene un nivel más alto de abstracción que un diagrama de clases, usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. Estos son bloques de construcción, como eventualmente un componente puede comprender una gran porción de un sistema. (43)(Ver Figura 10)

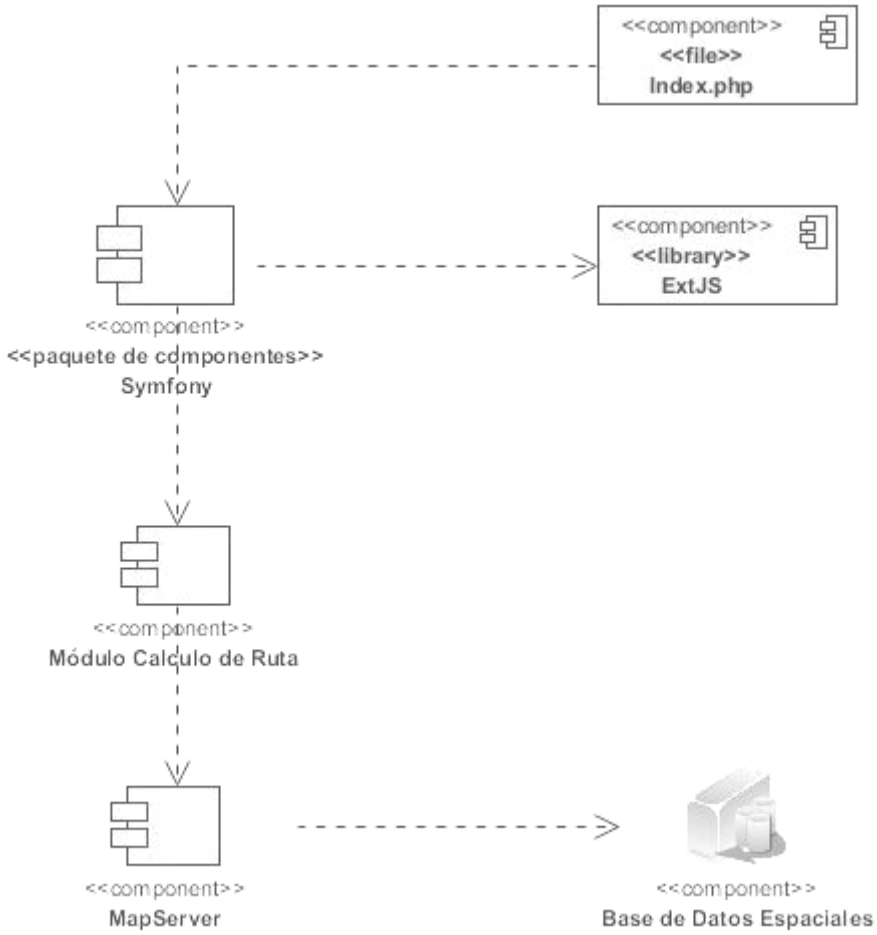


Figura 10 Diagrama de componentes del módulo cálculo de rutas.

4.6 Pruebas realizadas al sistema

Las pruebas del *software* son un elemento crítico para la garantía de calidad del mismo y representa una revisión final de las especificaciones, del diseño y la codificación. La creciente percepción del *software* como un elemento del sistema y la importancia de los costes asociados a un fallo del propio sistema, están motivando la creación de pruebas minuciosas y bien planificadas (44).

Con el objetivo de mostrar el correcto funcionamiento de la aplicación, comprobar que las operaciones internas se ajustan a las especificaciones planteadas así como los componentes internos, se realizaron pruebas de caja negra. Estas consideran la codificación dentro de sus parámetros a evaluar, es decir, que no están basadas en el consentimiento del diseño interno del programa. Se enfocan en los requisitos establecidos y en las funcionalidades del sistema.

Este tipo de prueba tiene como propósito verificar las relaciones de entrada y salida de una unidad. Su objetivo es verificar qué hace la unidad, pero sin averiguar cómo lo hace. Se llevan a cabo sobre la

interfaz del *software* y son completamente indiferentes al comportamiento interno y a la estructura del programa. Los casos de prueba de caja negra pretenden demostrar que: (45)

- ✓ Las funciones del *software* son operativas.
- ✓ Las entradas se aceptan de forma adecuada.
- ✓ Se produce una salida correcta.
- ✓ La integridad de la información externa se mantiene.

La prueba de caja negra intenta encontrar errores de las siguientes categorías:

- ✓ Funciones incorrectas o ausentes.
- ✓ Errores de interfaz.
- ✓ Errores en estructuras de datos o en accesos a bases de datos externas.
- ✓ Errores de rendimiento.

La técnica a utilizada dentro de las pruebas de caja negra es, partición de equivalencia, la cual permite definir las clases de equivalencia con la división de los campos de entrada en clases de datos válidos e inválidos para ejercitar las funciones del software y derivar casos de prueba. Todo ello centrándose principalmente en los requisitos funcionales del subsistema. El objetivo de esta técnica es reducir el posible conjunto de casos de prueba en uno más pequeño, para evaluar el software de forma eficiente.

Caso de prueba: Crear ruta por estructuras para vehículos.

Descripción general: El caso de uso se inicia cuando el usuario necesita conocer la ruta más corta de un lugar a otro de la UCI y para ello selecciona dos estructuras, una de origen y otra de destino.

Condiciones de ejecución: Ninguna.

Nombre de la sección	**Escenario de la sección**	**Descripción de la funcionalidad**	**Flujo Central**
SC1: Crear ruta por estructuras para vehículos.	EC 1.1: Crear ruta por estructuras correctamente.	El usuario selecciona una estructura origen, una destino y el sistema visualiza la ruta más corta entre ellas.	1. Dar clic en la opción “CALCULAR RUTA”. 2. Se muestra la interfaz “CALCULAR RUTA”. 3. Dar clic en la pestaña “POR ESTRUCTURA”. 4. Se muestra el interfaz de la pestaña. 5. Se seleccionan los datos de los campos. 6. Se pulsa el botón “Visualizar Ruta”. 7. Se muestra la ruta en el mapa.
	EC 1.2: Crear ruta por estructuras con fallo.	El usuario selecciona la misma estructura de origen, destino y el sistema muestra un mensaje de error “Las estructuras de origen y destino no pueden ser las	1. Dar clic en la opción “CALCULAR RUTA”. 2. Se muestra la interfaz “CALCULAR RUTA”. 3. Dar clic en la pestaña “POR

		mismas”.	ESTRUCTURAS”. 4. Se muestra el interfaz de la pestaña. 5. Se seleccionan los datos de los campos. 6. Se pulsa el botón “Visualizar Ruta”. 7. Se muestra un mensaje de error notificando que no puede seleccionar las mismas estructuras de origen y destino.

Tabla 3 Secciones a probar en el CU Crear ruta por estructuras para vehículos.

No.	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Tipo de estructura (origen)	ComboBox	No	Debe seleccionarse un tipo de estructura
2	Estructura (origen)	ComboBox	No	Debe seleccionarse una estructura
3	Tipo de estructura (destino)	ComboBox	No	Debe seleccionarse un tipo de estructura
4	Estructura (destino)	ComboBox	No	Debe seleccionarse una estructura

Tabla 4 Descripción de variables.

Id del Escenario	Escenario	1	2	3	4	Respuesta del Sistema	Resultado de la Prueba
EC 1.1	Crear ruta por estructuras correctamente.	V “Edificios Docentes”	V “Docente 6”	V “Plazas”	V “Plaza Niemeyer”	El sistema visualiza la ruta seleccionad	Satisfactorio

						a.	
EC 1.2	Crear ruta por estructuras con fallo.	V “Edificios Docentes”	V “Docente 6”	V “Edificios Docentes”	V “Docente 6”	El sistema muestra el mensaje de error “las estructuras de origen y destino no pueden ser las mismas”.	Satisfactorio

Tabla 5 Matriz de datos del CU Crear ruta por estructuras para vehículos.

El resultado de las pruebas realizadas al *software* fue satisfactorio ya que:

- Se comprobó que las respuestas del sistema son las esperadas y que el mismo cumple con las especificidades planteadas y que a su vez coinciden con las descripciones de los casos de uso planteados con anterioridad.
- Se comprobó que el sistema cumple con los objetivos propuestos.

4.7 Conclusiones parciales

En el presente capítulo se adoptó la arquitectura definida por el SIGUCI, facilitando la reutilización de componentes de *software* ya implementados, aumentando la productividad y rapidez en el desarrollo del módulo. Se facilita la comprensión y el mantenimiento del código debido a los patrones de arquitectura y diseño definidos, permitiendo la escalabilidad del mismo para futuras modificaciones. Los artefactos y diagramas generados referentes al flujo de trabajo de implementación favorecieron la compresión del proceso de desarrollo. La selección del método de prueba de caja negra y la técnica partición de equivalencia permitiendo detectar posibles no conformidades que posee el modulo desarrollado y verificar que la aplicación funciona correctamente.

Conclusiones

Producto de la investigación desarrollada y los resultados alcanzados se arriba a las siguientes conclusiones:

- Con el estudio de aplicaciones informáticas homólogas se identifican las principales tendencias en el desarrollo de SIG que cuenten con funcionalidades que permitan el cálculo de la ruta más corta. Aunque los sistemas estudiados no brindan una solución directa al problema de la investigación posibilita identificar funcionalidades básicas y comunes para la definición de los requisitos del producto final.

- El empleo del Proceso Unificado de Desarrollo (RUP) permitió durante el proceso de desarrollo del módulo, obtener la trazabilidad entre los artefactos generados. Además de garantizar una mejor organización y estructura de las partes de este, teniendo en cuenta elementos de calidad, rendimiento, reutilización y flexibilidad.

- Se logró finalizar con el desarrollo del módulo propuesto, verificando su correcto funcionamiento mediante el método de prueba de caja. Fue desarrollado con tecnologías libres y arquitectura propuesta por el sistema SIGUCI.

Por lo anteriormente expuesto, el marco teórico de la investigación, la propuesta de selección de herramientas y metodología y la implementación del módulo de cálculo de rutas constituyen los principales aportes del presente trabajo de diploma, trayendo consigo que el sistema SIGUCI le brinde a los usuario la posibilidad de conocer el camino más corto de un punto a otro en la Universidad de las Ciencias Informáticas de una forma rápida y precisa, ahorrándole tiempo y esfuerzo.

Recomendaciones

* Incorporar al sistema una cartografía de UCI más detallada y exacta con el objetivo de extender las funcionalidades del módulo de cálculo de rutas.

Referencias Bibliográficas

1. **Domingo Yagüez, Ing. Julio C. y Langhi, Ruben.** Sistema de Información Geográfica. *Sistema de Información Geográfica.* [En línea] 2002. http://www.inta.gov.ar/barrow/info/documentos/SIG/que_es_sig.htm.

2. **Reuter, Alfredo Favian.** Sistema de Información Geográfica(SIG). *Sistema de Información Geográfica(SIG).* [En línea] 2006. http://fcf.unse.edu.ar/archivos/series-didacticas/SD-25-SIG2-Reuter.pdf.

3. **Novua Alvarez, Maria, Orlando y Martinez.** SISTEMA DE INFORMACIÓN GEOGRÁFICA PARA EL ANÁLISIS. *SISTEMA DE INFORMACIÓN GEOGRÁFICA PARA EL ANÁLISIS.* [En línea] 2004. http://www.mappinginteractivo.com/plantilla-ante.asp?id_articulo=461.

4. **definicion.de.** definicion. [En línea] http://definicion.de/ruta/.

5. **Fallas, Jorge.** *Sistemas Integrados de Información geográfica.Conseptos Básicos de Cartografia.* Costa Rica : s.n., 2003.

6. **Montoya, Ing.Kalim Caro.** Scribd. *Scribd.* [En línea] http://es.scribd.com/doc/52147445/3/Teoria-de-datos-e-informacion.

7. *Algoritmos.Definicion.*

8. **Villalobos, A.R.** Grafos.Algoritmo de Floyd-Warshall. [En línea] http://personales.upv.es/arodrigu/grafos/FloydWarshall.htm.

9. —. Grafos. Algoritmo de Prim. [En línea] http://personales.upv.es/arodrigu/grafos/Prim.htm.

10. —. Grafos. Algoritmo de Ford-Fulkerson. [En línea] http://personales.upv.es/arodrigu/grafos/FordFulkerson.htm.

11. —. Grafos. Algoritmo de Kruskal. [En línea] http://personales.upv.es/arodrigu/grafos/Kruskal.htm.

12. —. Grafos. Algoritmo de Bellman-Ford. [En línea] http://personales.upv.es/arodrigu/grafos/Ford.htm.

13. **Bijit, P.L.S.** *Programación en Pascal Capítulo 25. Algoritmos heurísticos.* s.l. : UNIVERSIDAD TECNICA FEDERICO SANTA MARIA DEPARTAMENTO DE ELECTRONICA, 2003.

14. **Villalobos, A.R.** Grafos. Algoritmo de Dijkstra. [En línea] http://personales.upv.es/arodrigu/grafos/Dijkstra.htm.

15. **viamichelin.** [En línea] http://business.viamichelin.es/product/aplicaciones-moviles/viamichelin-rest-api.html.

16. **Ramos, Daniel Gómez.** [En línea] 2011. http://zaguan.unizar.es/TAZ/EINA/2011/6652/TAZ-PFC-2011-682.pdf.

17. **Pressman, Roger.** *Ingeniería del Software: Un enfoque práctico.* España : McGraw-Hill Companies, 2002. 5ta Edición.

18. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El proceso unificado de desarrollo de software.* Madrid : Addison Wesley, 2000.

19. **Kruchten, P.** *The Rational Unified Process: An Introduction.* s.l. : Addison Wesley, 2000.

20. **QUIÑONES, ERNESTO.** Introducción a postgresql. [En línea] 2007. http://www.postgresql.org.pe/articles/introduccion_a_postgresql.pdf.

21. **opengeo.** opengeo.org. *opengeo.org.* [En línea] http://opengeo.org/technology/postgis/.

22. **Kastl, Daniel y Junod, Frédéric.** workshop.pgrouting.org. [En línea] http://workshop.pgrouting.org.

23. **Castillo, C.** ¿Cómo funciona PHP? [En línea] 2001. http://www.tejedoresdelweb.com/307/article-1067.html.

24. **Popel, Dennis.** *Objects.A Beginner's Guide to PHP Data Objects, Database Connection Abstraction library for PHP 5.* BIRMINGHAM - MUMBAI : Packt Publishing, 2007. ISBN 978-1-847192-66-0.

25. **Fabien Potencier, François Zaninotto.** librosweb. *librosweb.* [En línea] [Citado el: 15 de Febrero de 2012.] http://www.librosweb.es.

26. **ExtJS.** extjs. [En línea] 2010. http://extjs.es.

27. **Giraldo, Luis y Zapata, Yuliana.** *Herramientas de desarrollo de ingeniería de SW para LINUX.* [Presentación] 2005.

28. **Fowler, Martin.** *Uml Gota a Gota.* Mexico : Addison Wesley Longman, 1999.

29. **Sierra, Daniel.** slideshare.net. [En línea] 15 de Noviembre de 2007. http://www.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml.

30. **Visual Paradingm.** Boost Productivity whith innovative and Intuitive Technologies. [En línea] Free Download Manager, 2007. http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M.

31. **NetBeans.** NetBeans. [En línea] 2010. http://netbeans.org/features/index.html.

32. **mapserver.org.** mapserver.org. *mapserver.org.* [En línea] http://mapserver.org/es/index.html.

33. **Cibernetia.** Cursos para empresa. *Ciberaula.* [En línea] 2010. http://linux.ciberaula.com/articulo/linux_apache_intro/..

34. **Booch, G, Rumbaugh, J y Jacobson, I.** *El Lenguaje Unificado de Modelado.* s.l. : Addison Wesley Iberoamericana, 1999.

35. **Bahit, Eugenia.** Arquitecta GLAMP & Agile Coach. *Agile Coaching.* [En línea] 2011. http://www.eugeniabahit.com/mvc/.

36. **Lago, Ramiro.** [En línea] http://www.proactiva-calidad.com/java/patrones/mvc.html de Abril de 2007.

37. **Liebener, L. y M.A. Rossi, Plinker.** *Relaciones entre los patrones de diseño.* Buenos Aires : s.n., 2003.

38. **Larman, C.** *UML y PATRONES.Introducción al análisis y diseño orientado a objetos.* 2004.

39. **Navarro, Alberto Limia.** *Sistema de descarga y procesamiento automatizado de patentes. Diseño del Sistema.* Ciudad de la Habana : s.n., 2007.

40. **VILLA CUESTA, MADELYS.** *Modelo para la ayuda a la toma de decisiones en la selección de patrones de desarrollo de software.* 2007.

41. **Villa, Madelys Cuesta.** *Modelo para la ayuda a la toma de decisiones en la selección de patrones de desarrollo de software.* 2007.

42. **Rumbaugh, J. y G., Booch.** *El Lenguaje Unificado de Modelado.* 2000.

43. **Cruz, Yhon.** sistems02.blogspot.com. [En línea] 18 de 3 de 2009. http://sistems02.blogspot.com/2009/03/exposicion-diagrama-de-componentes.html.

44. **Zenteno, Arturo Henry Torres.** *Método de pruebas de sistema basado en modelos navegacionales en un contexto MDWE.* X08100347-T.

45. **Jimenez, Darwin y Aguirre, Carlos Eduardo.** Modelo de pruebas de software. *Slideshare.* [En línea] 2008. [Citado el: 3 de Abril de 2012.] http://www.slideshare.net/dajigar/presentacion-pruebas-presentation.

46. **Zaldívar, Yoenis Pantoja.** Revista GeoNews. *Revista GeoNews.* [En línea] 19 de septiembre de 2009. http://revistageonews.wordpress.com/2009/09/19/modulo-de-analisis-de-la-plataforma-genesig/.

47. **Worsley, John C. y Drake, Joshua D.** *Practical PostgreSQL.* s.l. : O'Reilly, 2002.

48. **Wilfred, A.** *Proyectos profesionales. Programación PHP.* Madrid : Anaya Multimedia, 2002.

49. **Ullman, Larry.** *PHP 5 Advanced : visual quickpro guide.* California : Peachpit Press, 2007.

50. **Sommerville, Ian.** *Ingeniería del Software.* Madrid : Pearson Educación S.A., 2005. 7ma Edición.

51. **Regent.** Mapserver. [En línea] 2010. http://mapserver.org/en/trunk/es/about.html.

52. **Queraltó i Ros, Pau, Valls Dalmau, Francesc y Biere Arenas, Rolando.** *Herramienta de cálculo de rutas óptimas según parámetros de accesibilidad física en intirarios urbanos.* España : s.n., 2010.

53. **Potencier, Fabien y Zaninotto, Francois.** *Symfony la guía definitiva.* s.l. : Comunidad de desarrollo de Symfony, 2008.

54. **Olivares, Freddy Egdamar Paez.** egdamar877.blogspot.com. [En línea] 22 de 5 de 2009. http://egdamar877.blogspot.com/2009/05/expocicion.html.

55. **Matthew, Neil y Stone, Richard.** *Beginning Databases with PostgreSQL : from novice to professional.* New York : Springer-Verlag, 2005. 2da Edición.

56. **Martín, Manuel Martín.** *Manual PostGIS.*

57. **Langhi, R. y D.Y., Ing.Agr. Julio.** ¿Qué es un SIG? *Instituto Nacional de Tecnología Agropecuaria.* [En línea] 2002. http://www.inta.gov.ar/barrow/info/documentos/SIG/que_es_sig.htm..

58. **H. Canós, José, Letelier, Patricio y Penadés, Maria Carmen.** *Métodologías Ágiles en el Desarrollo de Software.* Valencia : s.n.

59. **Gutmans, Andi.** *PHP 5 Power programming.* Indianapolis : Prentice Hall, 2005.

60. *Ingeniería del Software Tema 3: Introducción a la Ingeniería de Requisitos.* **García Peñalvo, Dr. Francisco José, Conde González, Miguel Angel y Bravo Martín, Sergio.** Salamanca : Universidad de Salamanca, 2008. 3ro.

61. **Franco, Rodolfo.** Naturaleza de los Datos Espaciales. *Naturaleza de los Datos Espaciales.* [En línea] 2001. http://gemini.udistrital.edu.co/comunidad/profesores/rfranco/datos_espaciales.htm.

62. **EVA2.** Entorno Virtual de Aprendizaje. *Metodologías de desarrollo de software.pdf.* [En línea] http://eva.uci.cu/mod/resource/view.php?id=33747.

63. **EVA1.** Entorno Virtual de Aprendizaje. *Conferencia 1: Introducción a la Ingeniería de Software.* [En línea] http://eva.uci.cu/mod/resource/view.php?id=11361..

64. **cavsi.** www.cavsi.com. *www.cavsi.com.* [En línea] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.

65. **Casanova, Josep.** Desarrollo Web. *desarrolloweb.* [En línea] 9 de Septiembre de 2004. http://www.desarrolloweb.com/articulos/arquitectura-cliente-servidor.html.

66. **Cano, José Ignacio Barredo.** *Sistemas de Información Geográfica y evolución multicriterio en la ordenación del territorio.* 1996.

67. **Arias Marin, Marvin David.** Definición de lenguaje de programación. Tipos. Ejemplos. *catedraprogramacion.* [En línea] 16 de Octubre de 2008. [Citado el: 29 de Noviembre de 2011.] http://catedraprogramacion.foroactivo.net/t83-definicion-de-lenguaje-de-programacion-tipos-ejemplos.

Bibliografía

1. **Domingo Yagüez, Ing. Julio C. y Langhi, Ruben.** Sistema de Información Geográfica. *Sistema de Información Geográfica.* [En línea] 2002. http://www.inta.gov.ar/barrow/info/documentos/SIG/que_es_sig.htm.

2. **Reuter, Alfredo Favian.** Sistema de Información Geográfica(SIG). *Sistema de Información Geográfica(SIG).* [En línea] 2006. http://fcf.unse.edu.ar/archivos/series-didacticas/SD-25-SIG2-Reuter.pdf.

3. **Novua Alvarez, Maria, Orlando y Martinez.** SISTEMA DE INFORMACIÓN GEOGRÁFICA PARA EL ANÁLISIS. *SISTEMA DE INFORMACIÓN GEOGRÁFICA PARA EL ANÁLISIS.* [En línea] 2004. http://www.mappinginteractivo.com/plantilla-ante.asp?id_articulo=461.

4. **definicion.de.** definicion. [En línea] http://definicion.de/ruta/.

5. **Fallas, Jorge.** *Sistemas Integrados de Información geográfica.Conseptos Básicos de Cartografia.* Costa Rica : s.n., 2003.

6. **Montoya, Ing.Kalim Caro.** Scribd. *Scribd.* [En línea] http://es.scribd.com/doc/52147445/3/Teoria-de-datos-e-informacion.

7. *Algoritmos.Definicion.*

8. **Villalobos, A.R.** Grafos.Algoritmo de Floyd-Warshall. [En línea] http://personales.upv.es/arodrigu/grafos/FloydWarshall.htm.

9. —. Grafos. Algoritmo de Prim. [En línea] http://personales.upv.es/arodrigu/grafos/Prim.htm.

10. —. Grafos. Algoritmo de Ford-Fulkerson. [En línea] http://personales.upv.es/arodrigu/grafos/FordFulkerson.htm.

11. —. Grafos. Algoritmo de Kruskal. [En línea] http://personales.upv.es/arodrigu/grafos/Kruskal.htm.

12. —. Grafos. Algoritmo de Bellman-Ford. [En línea] http://personales.upv.es/arodrigu/grafos/Ford.htm.

13. **Bijit, P.L.S.** *Programación en Pascal Capítulo 25. Algoritmos heurísticos.* s.l. : UNIVERSIDAD TECNICA FEDERICO SANTA MARIA DEPARTAMENTO DE ELECTRONICA, 2003.

14. **Villalobos, A.R.** Grafos. Algoritmo de Dijkstra. [En línea] http://personales.upv.es/arodrigu/grafos/Dijkstra.htm.

15. **viamichelin.** [En línea] http://business.viamichelin.es/product/aplicaciones-moviles/viamichelin-rest-api.html.

16. **Ramos, Daniel Gómez.** [En línea] 2011. http://zaguan.unizar.es/TAZ/EINA/2011/6652/TAZ-PFC-2011-682.pdf.

17. **Pressman, Roger.** *Ingeniería del Software: Un enfoque práctico.* España : McGraw-Hill Companies, 2002. 5ta Edición.

18. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El proceso unificado de desarrollo de software.* Madrid : Addison Wesley, 2000.

19. **Kruchten, P.** *The Rational Unified Process: An Introduction.* s.l. : Addison Wesley, 2000.

20. **QUIÑONES, ERNESTO.** Introducción a postgresql. [En línea] 2007. http://www.postgresql.org.pe/articles/introduccion_a_postgresql.pdf.

21. **opengeo.** opengeo.org. *opengeo.org.* [En línea] http://opengeo.org/technology/postgis/.

22. **Kastl, Daniel y Junod, Frédéric.** workshop.pgrouting.org. [En línea] http://workshop.pgrouting.org.

23. **Castillo, C.** ¿Cómo funciona PHP? [En línea] 2001. http://www.tejedoresdelweb.com/307/article-1067.html.

24. **Popel, Dennis.** *Objects.A Beginner's Guide to PHP Data Objects, Database Connection Abstraction library for PHP 5.* BIRMINGHAM - MUMBAI : Packt Publishing, 2007. ISBN 978-1-847192-66-0.

25. **Fabien Potencier, François Zaninotto.** librosweb. *librosweb.* [En línea] [Citado el: 15 de Febrero de 2012.] http://www.librosweb.es.

26. **ExtJS.** extjs. [En línea] 2010. http://extjs.es.

27. **Giraldo, Luis y Zapata, Yuliana.** *Herramientas de desarrollo de ingeniería de SW para LINUX.* [Presentación] 2005.

28. **Fowler, Martin.** *Uml Gota a Gota.* Mexico : Addison Wesley Longman, 1999.

29. **Sierra, Daniel.** slideshare.net. [En línea] 15 de Noviembre de 2007. http://www.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml.

30. **Visual Paradingm.** Boost Productivity whith innovative and Intuitive Technologies. [En línea] Free Download Manager, 2007. http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M.

31. **NetBeans.** NetBeans. [En línea] 2010. http://netbeans.org/features/index.html.

32. **mapserver.org.** mapserver.org. *mapserver.org.* [En línea] http://mapserver.org/es/index.html.

33. **Cibernetia.** Cursos para empresa. *Ciberaula.* [En línea] 2010. http://linux.ciberaula.com/articulo/linux_apache_intro/..

34. **Booch, G, Rumbaugh, J y Jacobson, I.** *El Lenguaje Unificado de Modelado.* s.l. : Addison Wesley Iberoamericana, 1999.

35. **Bahit, Eugenia.** Arquitecta GLAMP & Agile Coach. *Agile Coaching.* [En línea] 2011. http://www.eugeniabahit.com/mvc/.

36. **Lago, Ramiro.** [En línea] http://www.proactiva-calidad.com/java/patrones/mvc.html de Abril de 2007.

37. **Liebener, L. y M.A. Rossi, Plinker.** *Relaciones entre los patrones de diseño.* Buenos Aires : s.n., 2003.

38. **Larman, C.** *UML y PATRONES.Introducción al análisis y diseño orientado a objetos.* 2004.

39. **Navarro, Alberto Limia.** *Sistema de descarga y procesamiento automatizado de patentes. Diseño del Sistema.* Ciudad de la Habana : s.n., 2007.

40. **VILLA CUESTA, MADELYS.** *Modelo para la ayuda a la toma de decisiones en la selección de patrones de desarrollo de software.* 2007.

41. **Villa, Madelys Cuesta.** *Modelo para la ayuda a la toma de decisiones en la selección de patrones de desarrollo de software.* 2007.

42. **Rumbaugh, J. y G., Booch.** *El Lenguaje Unificado de Modelado.* 2000.

43. **Cruz, Yhon.** sistems02.blogspot.com. [En línea] 18 de 3 de 2009. http://sistems02.blogspot.com/2009/03/exposicion-diagrama-de-componentes.html.

44. **Zenteno, Arturo Henry Torres.** *Método de pruebas de sistema basado en modelos navegacionales en un contexto MDWE.* X08100347-T.

45. **Jimenez, Darwin y Aguirre, Carlos Eduardo.** Modelo de pruebas de software. *Slideshare.* [En línea] 2008. [Citado el: 3 de Abril de 2012.] http://www.slideshare.net/dajigar/presentacion-pruebas-presentation.

46. **Zaldívar, Yoenis Pantoja.** Revista GeoNews. *Revista GeoNews.* [En línea] 19 de septiembre de 2009. http://revistageonews.wordpress.com/2009/09/19/modulo-de-analisis-de-la-plataforma-genesig/.

47. **Worsley, John C. y Drake, Joshua D.** *Practical PostgreSQL.* s.l. : O'Reilly, 2002.

48. **Wilfred, A.** *Proyectos profesionales. Programación PHP.* Madrid : Anaya Multimedia, 2002.

49. **Ullman, Larry.** *PHP 5 Advanced : visual quickpro guide.* California : Peachpit Press, 2007.

50. **Sommerville, Ian.** *Ingeniería del Software.* Madrid : Pearson Educación S.A., 2005. 7ma Edición.

51. **Regent.** Mapserver. [En línea] 2010. http://mapserver.org/en/trunk/es/about.html.

52. **Queraltó i Ros, Pau, Valls Dalmau, Francesc y Biere Arenas, Rolando.** *Herramienta de cálculo de rutas óptimas según parámetros de accesibilidad física en intirarios urbanos.* España : s.n., 2010.

53. **Potencier, Fabien y Zaninotto, Francois.** *Symfony la guía definitiva.* s.l. : Comunidad de desarrollo de Symfony, 2008.

54. **Olivares, Freddy Egdamar Paez.** egdamar877.blogspot.com. [En línea] 22 de 5 de 2009. http://egdamar877.blogspot.com/2009/05/expocicion.html.

55. **Matthew, Neil y Stone, Richard.** *Beginning Databases with PostgreSQL : from novice to professional.* New York : Springer-Verlag, 2005. 2da Edición.

56. **Martín, Manuel Martín.** *Manual PostGIS.*

57. **Langhi, R. y D.Y., Ing.Agr. Julio.** ¿Qué es un SIG? *Instituto Nacional de Tecnología Agropecuaria.* [En línea] 2002. http://www.inta.gov.ar/barrow/info/documentos/SIG/que_es_sig.htm..

58. **H. Canós, José, Letelier, Patricio y Penadés, Maria Carmen.** *Métodologías Ágiles en el Desarrollo de Software.* Valencia : s.n.

59. **Gutmans, Andi.** *PHP 5 Power programming.* Indianapolis : Prentice Hall, 2005.

60. *Ingeniería del Software Tema 3: Introducción a la Ingeniería de Requisitos.* **García Peñalvo, Dr. Francisco José, Conde González, Miguel Angel y Bravo Martín, Sergio.** Salamanca : Universidad de Salamanca, 2008. 3ro.

61. **Franco, Rodolfo.** Naturaleza de los Datos Espaciales. *Naturaleza de los Datos Espaciales.* [En línea] 2001. http://gemini.udistrital.edu.co/comunidad/profesores/rfranco/datos_espaciales.htm.

62. **EVA2.** Entorno Virtual de Aprendizaje. *Metodologías de desarrollo de software.pdf.* [En línea] http://eva.uci.cu/mod/resource/view.php?id=33747.

63. **EVA1.** Entorno Virtual de Aprendizaje. *Conferencia 1: Introducción a la Ingeniería de Software.* [En línea] http://eva.uci.cu/mod/resource/view.php?id=11361..

64. **cavsi.** www.cavsi.com. *www.cavsi.com.* [En línea] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.

65. **Casanova, Josep.** Desarrollo Web. *desarrolloweb.* [En línea] 9 de Septiembre de 2004. http://www.desarrolloweb.com/articulos/arquitectura-cliente-servidor.html.

66. **Cano, José Ignacio Barredo.** *Sistemas de Información Geográfica y evolución multicriterio en la ordenación del territorio.* 1996.

67. **Arias Marin, Marvin David.** Definición de lenguaje de programación. Tipos. Ejemplos. *catedraprogramacion.* [En línea] 16 de Octubre de 2008. [Citado el: 29 de Noviembre de 2011.] http://catedraprogramacion.foroactivo.net/t83-definicion-de-lenguaje-de-programacion-tipos-ejemplos.

Glosario de Términos

- WMS: Web Map Server
- WFS: Web Feature Service
- WCS: Web Coverage Service
- TCP: Transmission Control Protocol
- HTTP: Hypertext Transfer Protocol
- W3C: World Wide Web Consortium
- ADO: ActiveX Data Objects