Universidad de las Ciencias Informáticas
Facultad 4
Departamento de informática

Servicio de camino mínimo sobre un Sistema de Información Geográfica basado en software libre

Tesis en opción al título
de
Máster en Informática Aplicada

Autor
Lic. Rafael Rodríguez Puente
Tutor
Dra. Tatiana Delgado Fernández

Ciudad de la Habana, julio de 2007
"Año 49 de la Revolución"

Agradecimientos

Agradecimientos

Le agradezco a mis familiares y en especial a mi madre, Olga Puentes González, por haberme guiado por el camino correcto.

A mi novia y amiga Baby.

A Tatiana, por haberme introducido en el mundo de la geomática, por aceptar ser mi tutora y por revisar la tesis con tan poco tiempo.

A Karel, por su ayuda al iniciarme con la plataforma de desarrollo.

Por último, muchas gracias a todo aquel que se preocupó por la realización de este trabajo, y a los que, de una forma u otra, estuvieron atentos al resultado del mismo. A todo el que me brindó su apoyo.

Dedicatoria

A mi madre, Olga Puentes González

A mi hija, Betsy Rodríguez Hernández

Síntesis

SÍNTESIS

Los Sistemas de Información Geográfica (SIG) son sistemas Informáticos que se usan para almacenar y manipular información relacionada estrechamente con su posición geográfica. Esta tecnología se ha desarrollado tan rápidamente en las dos décadas pasadas que ya es aceptada como una herramienta esencial para el uso efectivo de dicha información.

El presente trabajo tiene como uno de sus objetivos desarrollar un Sistema de Información Geográfica que brinde las funcionalidades básicas(acercar, alejar, información de un punto), además, se desarrollará un módulo que permita hacer una búsqueda del camino más corto entre dos direcciones postales, y mostrará el mismo sobre el mapa.

Es importante resaltar que este módulo sólo necesita como entrada un mapa, el mismo puede estar en cualquiera de los formatos vectoriales soportados por Mapserver (ESRI shapefiles, PostGIS, ESRI ArcSDE, GML entre otros), de forma tal que se podrá añadir esta funcionalidad a cualquier SIG que disponga de un mapa en alguno de los formatos antes mencionados, sin necesidad de hacer ningún cambio en el código del módulo, o de generar datos adicionales de entrada.

También se incluye un capítulo, en el que se especifica una posible forma de

manipular mapas muy grandes, obteniendo una representación muy compacta de un grafo que tenga una gran cantidad de vértices y aristas (varios millones), lo que facilitaría el trabajo con el mapa; esto se logra representando el mapa a través de un grafo, y con el uso de gramáticas de grafo [Ehrig1992] [Gaurav2004] [Ehrig1980].

TABLA DE CONTENIDOS

SÍNTESIS.....I
INTRODUCCIÓN.....1
1. FUNDAMENTACIÓN TEÓRICA.....6
1.1 Introducción.....6
1.2 Sistemas de Información Geográfica.....6
1.2.1 Contexto tecnológico que propicia la evolución de las tecnologías geoespaciales.....9
1.2.2 Características de las Arquitecturas modernas de los SIG.....10
1.2.3 La construcción de bases de datos geográficas.....11
1.2.4 Topologías, modelos de datos y tipos de SIG.....12
1.2.4.1 Los SIG Vectoriales.....12
1.2.4.2 Los SIG Raster.....14
1.2.5 ¿Para que podemos utilizar un SIG?.....15
1.2.6 Sistemas de Información Geográfica sobre web con funcionalidad de búsqueda de caminos mínimos.....16
1.2.7 Servidor de mapas.....17
1.3 Gramáticas de grafo.....18
1.3.1 Conceptos.....19
1.3.2 Tipos de gramáticas de grafo.....20
1.3.2.1 Reemplazo de nodos.....21
1.3.2.2 Reemplazo de aristas.....22
1.4 Mecanismos de reescritura de grafos.....22
1.4.1 Terminología.....23
1.4.2 Clasificación de los mecanismos de empotrado.....24
1.4.3 Consideraciones prácticas para seleccionar un mecanismo de reescritura.....30
1.4.3.1 Potencia del empotrado.....31
1.4.3.2 Propiedades formales.....31
1.4.3.3 Legibilidad y manejabilidad intelectual.....31
1.4.3.4 Eficiencia de la aplicación de las reglas.....32
1.5 Reducción de grafos.....33
2. SELECCIÓN DE LA PLATAFORMA DE DESARROLLO.....34
2.1 Introducción.....34
2.2 Php.....34
2.3 Java.....38
2.4 Python.....41
2.5 Zope como servidor de aplicaciones.....44

3. DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA....57
Introducción....57
3.1 Descripción del Modelo de Dominio....57
3.2 Especificación de Requisitos....59
3.2.1 Requerimientos funcionales....59
3.2.2 Requerimientos no funcionales....60
3.3 Descripción del Sistema propuesto....61
3.4 Modelo de casos de uso del sistema....61
3.4.1 Definición de los actores del sistema....61
3.4.2 Diagrama de casos de uso. Especificación....62
3.4.3 Expansión de los casos de uso....65
3.5 Modelo de Diseño....69
3.5.1 Diagramas de Clases del Diseño....69
3.5.2 Diagramas de interacción....69
3.6 Principios de diseño....70
3.6.1 Interfaz de la aplicación....70
3.7 Modelo de despliegue....71
3.8 Módulo map2Graph....71
4. ALGORITMO DE REDUCCIÓN DE GRAFOS. CAMINO MÍNIMO....74
Introducción....74
4.1 Conceptos....76
4.1.1 Reducción de un grafo....77
4.1.2 Reescritura de grafos....78
4.2 Algoritmos propuestos....79
4.2.1 Reducir el grafo....80
4.2.2 Búsqueda de rutas mínimas....81
4.2.3 Beneficios....83
CONCLUSIONES....84
RECOMENDACIONES....86
REFERENCIAS BIBLIOGRÁFICAS....87
BIBLIOGRAFÍA....89
ANEXO I. DIAGRAMAS DE CLASES....92
ANEXO II. DIAGRAMAS DE SECUENCIA....96
ANEXO III. DIAGRAMA DE DESPLIEGUE....101

ÍNDICE DE ILUSTRACIONES

Ilustración 1: División en capas de un mapa....11
Ilustración 2: Formación de líneas en la topología arco-nodo....13
Ilustración 3: Formación de polígonos en la topología arco-nodo....14
Ilustración 4: Organización de la información en el modelo de datos raster....15
Ilustración 5: Ejemplo de mecanismo de empotrado de Schneider....27
Ilustración 6: Arquitectura de Zope....55
Ilustración 7: Objetos del dominio....58
Ilustración 8: Diagrama de casos de uso 62
Ilustración 9: Diagrama de casos de uso....63
Ilustración 10: Vista principal del sistema....71
Ilustración 11: Grafo a reducir....77
Ilustración 12: Grafo reducido....77
Ilustración 13: Diagrama de clases del módulo mapObj....92
Ilustración 14: Diagrama de clases web del módulo browser....93
Ilustración 15: Diagrama de clases del módulo Map2Graph....94
Ilustración 16: Módulos utilizados....95
Ilustración 17: Diagrama de secuencia del caso de uso Localizar Entidad....96
Ilustración 18: Diagrama de secuencia del caso de uso Buscar Camino....96
Ilustración 19: Diagrama de secuencia del caso de uso Alejar....97
Ilustración 20: Diagrama de secuencia del caso de uso Centrar 98
Ilustración 21: Diagrama de secuencia del caso de uso Obtener Información....99
Ilustración 22: Diagrama de secuencia del caso de uso Mover....100
Ilustración 23: Diagrama de secuencia del caso de uso Vista Inicial....100
Ilustración 24: Modelo de despliegue....101

ÍNDICE DE TABLAS

Tabla 1: Bases de datos soportadas por PHP.....37
Tabla 2: Actores del sistema.....62
Tabla 3: Especificación del caso de uso Realizar Operaciones Básicas.....63
Tabla 4: Especificación del caso de uso Localizar Entidad.....64
Tabla 5: Especificación del caso de uso Buscar Camino.....64
Tabla 6: Expansión del caso de uso Localizar Entidad.....65
Tabla 7: Expansión de caso de uso Realizar Operaciones Básicas.....68
Tabla 8: Expansión del caso de uso Buscar Camino.....68

INTRODUCCIÓN

Los Sistemas de Información Geográfica (SIG) son sistemas Informáticos que se usan para almacenar y manipular información relacionada estrechamente con su posición geográfica. Esta tecnología se ha desarrollado tan rápidamente en las dos décadas pasadas que ya es aceptada como una herramienta esencial para el uso efectivo de dicha información.

Desde su surgimiento y hasta la fecha los SIG han evolucionado por varias etapas en correspondencia con el propio desarrollo de las Tecnologías de la Información y las Comunicaciones. Los primeros SIG desarrollados entre las décadas del sesenta y el ochenta estaban orientados a un proyecto, donde toda la información se almacenaba en una única computadora, ejecutándose el SIG también en ella. Después estos sistemas fueron ampliando su conectividad dentro de la empresa, en intranets corporativas, y finalmente, fue necesario que surgieran enfoques orientados al Web para satisfacer las demandas de toda la sociedad.

Los Sistemas de Información Geográfica están de moda hoy en día. Más allá de la espectacularidad de sus resultados, o la amplitud de su campo de aplicaciones, esta herramienta se ha popularizado básicamente porque se considera que entre el ochenta y el noventa por ciento de toda la información involucrada en la toma de decisiones de la sociedad a nivel global, tiene una componente espacial. Se trata de una disciplina joven y ciertamente compleja, un nuevo paradigma que forma parte del ámbito más extenso de los Sistemas de Información (SI). Es una herramienta multipropósito con aplicaciones en los campos más diversos.

En la actualidad estos sistemas son una herramienta fundamental en la transferencia del conocimiento del mundo real a modelos, que serán utilizados posteriormente en el análisis y toma de decisiones en sus diversas aplicaciones

dentro de actividades como: la gestión de recursos naturales y medio ambiente, la planificación urbana, el mantenimiento de redes hidráulicas, eléctricas, telefónicas y alcantarillados, por citar algunos ejemplos.

La reciente y amplia introducción de los SIG en el mundo, y los resultados que con estos se puede alcanzar, ha creado una necesidad de conocer y profundizar en estas tecnologías.

Después de analizar algunos de los sistemas existentes, así como las necesidades actuales de información para resolver determinados problemas, el presente trabajo comprende, como principal objetivo, desarrollar Sistema de Información Geográfica, que sea accesible desde la web, y que aporte información al proceso de la toma de decisiones, teniendo en cuenta las especificaciones de de OpenGIS[1] para un servidor de mapas web.

La utilización de la aplicación presupone lograr un impacto en el ámbito digital Universitario con vistas a posteriores desarrollos de nuevos servicios utilizando la misma plataforma que desarrollará este proyecto.

Problema de investigación

El Sistema de Información Geográfica de la UCI está implementado utilizando el servidor de mapas de código abierto MapServer, este servidor de mapas no cuenta con herramientas analíticas para resolver problemas típicos de optimización y de redes, como son el problema de transporte y la ruta más corta entre dos lugares, por lo que el SIG de la UCI no cuenta con las funcionalidades antes mencionadas.

Hipótesis

Si se desarrolla un Sistema de Información Geográfica basado en web, que permita representar automáticamente un mapa a través de un grafo, y además, brinde la posibilidad de mostrar gráficamente el camino más corto

1"*Consorcio sin ánimo de lucro formado por organizaciones publicas y privadas creado en 1994 y cuyo fin es la definición de estándares abiertos e interoperables dentro de los Sistemas de Información Geográfica. Persigue acuerdos entre las diferentes empresas del sector que posibiliten la interoperación de sus sistemas de geoprocesamiento y facilitar el intercambio de la información geográfica en beneficio de los usuarios*"

entre dos lugares:

- La dirección de transporte y los usuarios en general, contarán con una herramienta que les permitirá hacer búsquedas del camino más corto entre dos lugares de la universidad.
- Se obtendrá una base (la representación en un grafo del mapa) para poder desarrollar herramientas que permitan hacer análisis de redes sobre un mapa.
- Se podrá contar con servicios de valor añadido en el SIG de la UCI.

Objetivo general

Desarrollar un Sistema de Información Geográfica basado en web, que brinde la posibilidad de obtener gráficamente el camino mas corto entre dos direcciones postales[2].

Objetivos específicos

- Desarrollar un Sistema de Información Geográfica que cuente con las operaciones básicas: acercar, alejar, mover, información de un punto, línea o polígono y mostrar mapa completo.
- Desarrollar un módulo para la búsqueda del camino más corto entre dos direcciones postales así como su representación en un mapa, teniendo como única entrada el mapa en cualquiera de los formatos soportados por MapServer.
- Desarrollar un algoritmo que me permita representar un mapa extenso[3] en una estructura de datos.

Para darle cumplimiento a los objetivos, la tesis se ha estructurado en cuatro capítulos. El primer capítulo se centra en exponer el marco teórico, brindando información sobre los SIG y estándares relacionados con los mismos, además

2 En este caso sólo se brindará la posibilidad de escoger esquinas o intersecciones entre calles, por ejemplo: desde L y 23 hasta 29 y 23.

3 De millones de objetos.

se hace referencia a las compañías líderes en el tema, y a los diferentes tipos de mapas que se pueden utilizar en un SIG. El segundo capítulo está dedicado al estudio de distintas plataformas de desarrollo de aplicaciones web, y queda expuesta la selección de una de ellas para darle cumplimiento a los objetivos planteados. El tercer capítulo muestra el diseño general del sistema implementado, en el mismo se pueden apreciar algunos de los artefactos propuestos por la metodología de desarrollo de software Rational Unified Process (RUP), así como la secuencia de pasos utilizados para obtener una representación del mapa en una estructura de datos. El cuarto capítulo está relacionado con la búsqueda de caminos en mapas extensos, el mismo queda dividido en dos partes esenciales, algoritmo para compactar un grafo, y una variación del algoritmo Dijkstra para hacer búsquedas del camino mínimo entre dos nodos, en un grafo reducido con el algoritmo propuesto.

Objeto de investigación

- Sistemas de Información Geográfica que brinden el servicio de búsqueda del camino más corto entre dos direcciones.
- Gramáticas de grafo
- Mecanismos de reescritura de grafos
- Reducción de grafos

Campo de investigación

- Búsqueda del camino mínimo entre dos lugares haciendo uso de un Sistema de Información Geográfica.

Resultados esperados

- Se espera obtener un Sistema de Información Geográfica que le brinde la posibilidad al usuario de obtener gráficamente, el camino más corto entre dos direcciones.
- Con la generación automática de una estructura de datos que

posibilite la búsqueda del camino más corto entre dos direcciones en un mapa, se brindará un nuevo servicio a los usuarios del sistema, además, se creará un componente que será el propio Sistema de Información Geográfica, de forma tal que será muy sencillo añadir un SIG a un portal y/o intranet de una entidad. Además, creará una base, que se podrá utilizar para dar solución a algunos problemas de optimización (i.e. problema de transporte), así como cualquier problema que involucre la utilización de rutas en un mapa.

Novedad científica

- Desarrollo de un módulo que representa un mapa[4] a través de un grafo de forma automática.
- Desarrollo de un módulo que brinda la posibilidad de hacer búsquedas del camino más corto entre dos direcciones, como un valor añadido de MapServer.
- Desarrollo de un algoritmo de reducción de grafos haciendo uso de las gramáticas de grafo, así como métodos de reescritura de grafo.

Métodos de investigación

Análisis y síntesis: Este método fue utilizado para analizar la situación problémica y determinar posibles variantes de solución.

Hipotético – deductivo: Permitió, a partir de la hipótesis, elaborar conclusiones acerca de la factibilidad de la utilización de un Sistema de Información Geográfica que brinde la posibilidad de hacer búsquedas de caminos mínimos.

Criterio de experto: Para asumir determinados criterios que son imprescindibles para realizar el desarrollo del sistema.

4 *En este caso, sólo se tiene en cuenta la capa calles, porque es la única que es imprescindible para hacer búsquedas de camino mínimo sobre el mapa, por lo que la estructura de datos que se obtiene, es básicamente la representación de las calles del mapa.*

CAPÍTULO

1. FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo, será realizado un análisis de cómo se encuentran en el mundo aquellas tecnologías que serán necesarias para la construcción del sistema que se pretende desarrollar. Además, se llevará a cabo el estudio de los conceptos necesarios para una buena comprensión del resto del trabajo.

1.2 Sistemas de Información Geográfica.

El uso de los Sistemas de Información Geográfica ha aumentado enormemente en las décadas de los ochenta y los noventa y como consecuencia, estos sistemas han pasado del total desconocimiento a la práctica cotidiana en el mundo de los negocios, en las universidades y en los organismos gubernamentales, usándose para resolver problemas diversos. Es lógico, por tanto, que hayan sido propuestas varias definiciones.

Una definición precisa es la siguiente:

Un SIG se define [Aronoff1987] como un sistema computacional para la entrada; manejo (almacenamiento y recuperación de información); manipulación, análisis; y representación de datos geográficos. Actualmente, nuevas modificaciones a estos conceptos clásicos de SIG se han producido para destacar el papel de la diseminación de los datos como una función ineludible de los sistemas de información geográfica en ambientes distribuidos y globales de acceso de datos y en el entorno WWW [Delgado2000].

Otras definiciones de SIG:

- Un sistema para capturar, almacenar, comprobar, integrar, manipular, analizar y visualizar datos que están espacialmente referenciados a la tierra [Chorley1987].
- Sistemas automatizados para la captura, almacenamiento, composición, análisis y visualización de datos espaciales [Clarke1990].
- Un sistema de hardware, software y procedimientos diseñados para soportar la captura, gestión, manipulación, análisis, modelado y visualización de datos espacialmente referenciados para resolver problemas complejos de planeamiento y gestión [Cowen1989].

Desde un punto de vista práctico un Sistema de Información Geográfica es un sistema informático capaz de realizar una gestión completa de datos geográficos referenciados. Por referenciados se entiende que estos datos geográficos o mapas tienen unas coordenadas geográficas reales asociadas, las cuales nos permiten manejar y hacer análisis con datos reales como longitudes, perímetros o áreas. Todos estos datos alfanuméricos asociados a los mapas más los que queramos añadirle los gestiona una base de datos integrada al SIG; estas bases de datos no son como las conocidas normalmente pues tienen características especiales dado su contenido.

La reciente y amplia introducción de los SIG en el mundo, y los resultados que con estos se puede alcanzar, ha creado una necesidad de conocer y

profundizar en estas tecnologías. Los directivos de los organismos empresariales y del estado están siendo instados a tomar decisiones sobre la introducción de la tecnología SIG y establecer directrices para su uso. Se realizan programas para convertir datos de mapas a formato digital para el uso de los SIG. Los estudiantes y educadores que usan información geográfica están ganando acceso a la tecnología SIG que puede ser usada para incrementar la profundidad y amplitud de sus análisis.

La tecnología ha creado un excitante potencial para la información geográfica al poder ser usada más sistemáticamente y por una gran diversidad de disciplinas. Sin embargo, la facilidad con que un SIG puede manipular información geográfica también ha creado una mayor dificultad. Los usuarios no familiarizados con los SIG o la naturaleza de la información geográfica pueden producir fácilmente tantos análisis válidos como inválidos. Válidos o no, los resultados tienen un aire de precisión asociado con sofisticados gráficos de ordenador y tablas numéricas. Un mejor entendimiento de la tecnología SIG por los usuarios y directivos es crucial para el uso apropiado de esta tecnología.

Un SIG está diseñado para la colección, almacenamiento y análisis de objetos y fenómenos donde la localización geográfica es una característica importante o crítica para el análisis. Por ejemplo, para la localización óptima de un parque de bomberos o para conocer los lugares donde la erosión del suelo es más severa, debemos utilizar información geográfica. En cada caso, debemos tener en cuenta qué es y dónde está.

Mientras gestionar y analizar datos que están referidos a una localización geográfica son funciones clave en un SIG, el poder del sistema es más visible cuando la cantidad de datos implicados es demasiado grande para poder ser manejada manualmente. Puede haber cientos o miles de entidades a considerar, o cientos de factores asociados con cada entidad o lugar. Estos datos pueden existir como mapas, tablas de datos, o incluso como listas de nombres y direcciones. Volúmenes de datos tan grandes no son gestionados eficientemente usando métodos manuales. Sin embargo, cuando estos datos

se han introducido a un SIG, pueden ser fácilmente manipulados y analizados en formas que serían demasiado costosas -en tiempo o dinero- o prácticamente imposibles de hacer usando métodos manuales.

En el contexto actual de la sociedad de la información, resulta clave el componente espacial de ésta, debido al análisis estratégico que se puede llevar a cabo con la utilización de los SIG, hecho que puede significar mayor eficacia y eficiencia en los procesos de toma de decisiones a los que se enfrenta a diario cualquier país tanto a nivel de Gobierno como en los diferentes sectores industriales, sin excluir al propio ciudadano. Todos ellos requieren, en este proceso, manejar información relativa al *dónde* ocurren los fenómenos.

1.2.1 Contexto tecnológico que propicia la evolución de las tecnologías geoespaciales.

Las tendencias más importantes de la Tecnología Geoespacial, que se han evidenciado en numerosas publicaciones internacionales de los últimos tiempos, muestran según la proyección ofrecida por la revista GEOWorld en Diciembre de 1999 los 3 aspectos claves siguientes :

- Impacto de Internet.
- Interoperabilidad y Estándares.
- Soluciones empresariales.

En este mismo contexto se manifestó la Vicepresidenta Ejecutiva de la División de Soluciones de Mapificación y SIG de INTERGRAPH en su intervención sobre la Evolución de las Tecnologías Geoespaciales en la quinta Conferencia de la Infraestructura Global de Datos Espaciales, celebrada en mayo de 2001 en Colombia.

1.2.2 Características de las Arquitecturas modernas de los SIG.

En la actualidad, las interfaces, basadas en las especificaciones OpenGIS, permiten a los servidores de datos de Internet ser consultados desde puestos remotos y extraer sólo la información específica requerida. Ya no es necesario preocuparse por los diferentes formatos de datos y por su conversión, gracias a una arquitectura abierta en virtud de la cual los servidores de datos pueden manipular los mismos en su formato nativo. En un entorno de interoperabilidad y estándares, tampoco hay que preocuparse por el software que se esté utilizando. Es posible, también, servir mapas en Web gracias a la filosofía adoptada basada en metadatos y catálogos de datos geográficos. Con todas estas características presentes, la tendencia apunta a la integración de las tecnologías espaciales con el resto de las Tecnologías de la Información y las Comunicaciones, lo cual es particularmente apreciado en el manejo de recursos de empresas.

Las principales características de la arquitectura actual de los SIG pueden ser resumidas en los aspectos siguientes:

- Filosofía cliente-servidor, que permite una variedad de datos y accesos a grupos de usuarios de la Empresa, una Intranet o Internet.
- Interoperabilidad (Especificaciones estándares ISO TC 211, OpenGIS; COM, CORBA y Java) que permite integrar funcionalidad de varios vendedores.
- Arquitectura abierta permite acceso de datos, sin necesidad de conversión de formatos.
- Metadatos y catálogos de datos espaciales para el uso compartido de información geográfica a través de Intranets e Internet.
- Uso de XML (Extensible Markup Language) y GML para comunicar la información geográfica entre sistemas heterogéneos.
- Servidores de Mapas en Web.

- Integración dentro de la Tecnología de Información de una Empresa.

1.2.3 La construcción de bases de datos geográficas

La construcción de una base de datos geográfica implica un proceso de abstracción para pasar de la complejidad del mundo real a una representación simplificada asequible para el lenguaje de los ordenadores actuales. Este proceso de abstracción tiene diversos niveles y normalmente comienza con la concepción de la estructura de la base de datos, generalmente en capas, como se muestra en la figura; en esta fase, y dependiendo de la utilidad que se vaya a dar a la información a compilar, se seleccionan las capas temáticas a incluir.

Pero la estructuración de la información espacial procedente del mundo real en capas conlleva cierto nivel de dificultad. En primer lugar, la necesidad de abstracción que requieren las máquinas implica trabajar con primitivas básicas de dibujo, de tal forma que toda la complejidad de la realidad ha de ser reducida a puntos, líneas o polígonos.

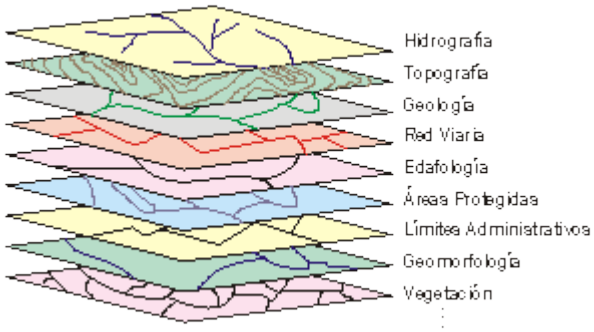

Ilustración 1: División en capas de un mapa

En segundo lugar, existen relaciones espaciales entre los objetos geográficos que el sistema no puede obviar; es lo que se denomina topología, que en realidad es el método matemático-lógico usado para definir las relaciones espaciales entre los objetos geográficos.

Aunque a nivel geográfico las relaciones entre los objetos son muy complejas, siendo muchos los elementos que interactúan sobre cada aspecto de la realidad, la topología de un SIG reduce sus funciones a cuestiones mucho más sencillas, como por ejemplo conocer el polígono (o polígonos) a que pertenece una determinada línea, o bien saber qué agrupación de líneas forman una determinada carretera.

Existen diversas formas de modelar estas relaciones entre los objetos

geográficos o topología. Dependiendo de la forma en que ello se lleve a cabo se tiene uno u otro tipo de Sistema de Información Geográfica dentro de una estructura de dos grupos principales :

- SIG Vectoriales [SIGVectoriales].
- SIG Raster [SIGRaster].

No existe un modelo de datos que sea superior a otro, sino que cada uno tiene una utilidad específica.

1.2.4 Topologías, modelos de datos y tipos de SIG

En función del modelo de datos implementado en cada sistema, podemos distinguir dos grandes grupos de Sistemas de Información Geográfica: SIG Vectoriales y SIG Raster.

Aunque veremos posteriormente las diferencias entre ambos con más detalle, adelantaremos que los SIG vectoriales utilizan vectores (básicamente líneas), para delimitar los objetos geográficos, mientras que los raster utilizan una retícula regular para documentar los elementos geográficos que tienen lugar en el espacio.

1.2.4.1 Los SIG Vectoriales

Son aquellos Sistemas de Información Geográfica que para la descripción de los objetos geográficos utilizan vectores definidos por pares de coordenadas relativas a algún sistema cartográfico.

Con un par de coordenadas y su altitud gestionan un punto (e.g. un vértice geodésico), con dos puntos generan una línea, y con una agrupación de líneas forman polígonos. De todos los métodos utilizados para formar topología vectorial la forma más robusta es la topología arco-nodo, cuya lógica de funcionamiento se explica en la ilustración 2.

La topología arco-nodo basa la estructuración de toda la información

geográfica en pares de coordenadas, que son la entidad básica de información para este modelo de datos. Con pares de coordenadas (puntos) forma vértices nodos, y con agrupaciones de éstos puntos forma líneas, con las que a su vez puede formar polígonos.

Hemos visto en la ilustración 2 cómo se forman las líneas a partir de puntos (pares de coordenadas). Veamos ahora cómo se forman los polígonos a partir de la agrupación de líneas en la ilustración 3.

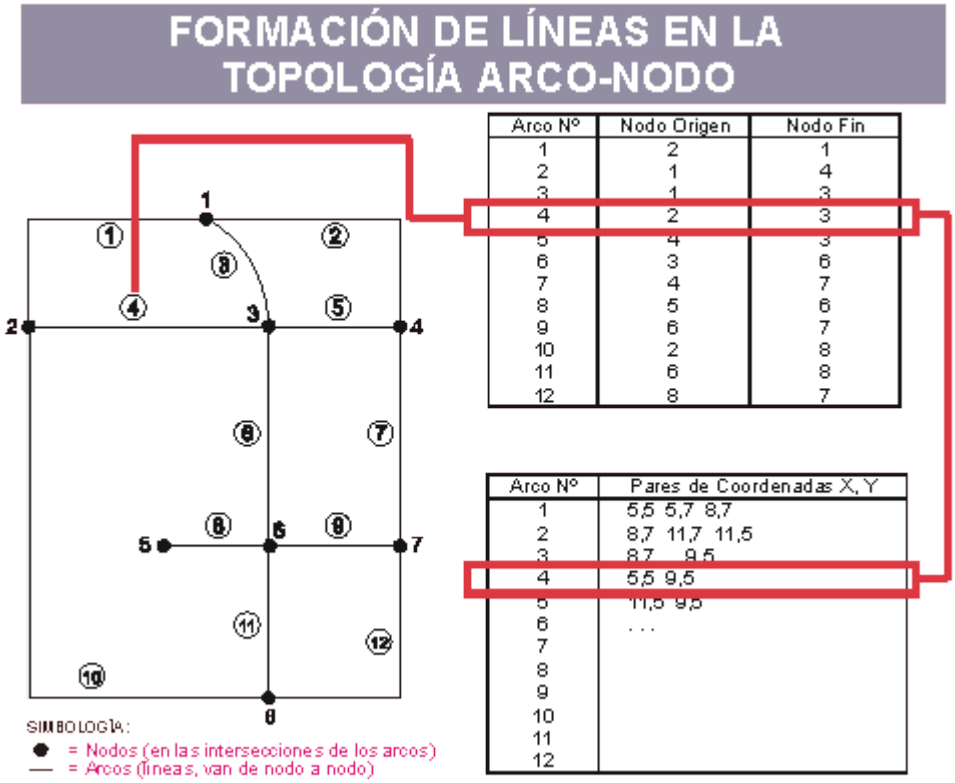

Ilustración 2: Formación de líneas en la topología arco-nodo

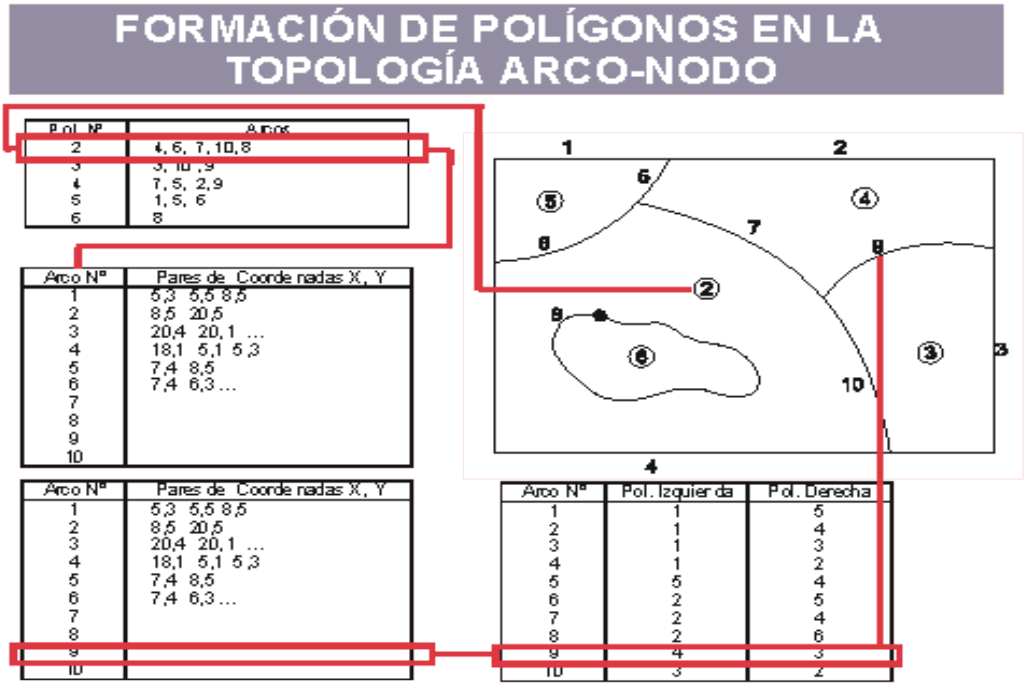

Ilustración 3: Formación de polígonos en la topología arco-nodo

En general, el modelo de datos vectorial es adecuado cuando trabajamos con objetos geográficos con límites bien establecidos, como pueden ser fincas, carreteras, etc.

1.2.4.2 Los SIG Raster

Los Sistemas de Información Raster basan su funcionalidad en una concepción implícita de las relaciones de vecindad entre los objetos geográficos. Su forma de proceder es dividir la zona de afección de la base de datos en una retícula o malla regular de pequeñas celdas (a las que se denomina pixel) y atribuir un valor numérico a cada celda como representación de su valor temático. Dado que la malla es regular (el tamaño del pixel es constante) y que conocemos la posición en coordenadas del centro de una de las celdas, se puede decir que todos los pixel están georreferenciados.

Lógicamente, para tener una descripción precisa de los objetos geográficos contenidos en la base de datos el tamaño del pixel ha de ser reducido (en función de la escala), lo que dotará a la malla de una resolución alta. Sin

embargo, a mayor número de filas y columnas en la malla (más resolución), mayor esfuerzo en el proceso de captura de la información y mayor costo computacional a la hora de procesar la misma.

Ilustración 4: Organización de la información en el modelo de datos raster

No obstante, el modelo de datos raster es especialmente útil cuando tenemos que describir objetos geográficos con límites difusos, como puede ser la dispersión de una nube de contaminantes, o los niveles de contaminación de un acuífero subterráneo, donde los contornos no son absolutamente nítidos; en esos casos, el modelo raster es más apropiado que el vectorial.

1.2.5 ¿Para que podemos utilizar un SIG?

Hasta ahora hemos descrito un SIG por medio de definiciones de tipo formal y por medio de su capacidad para satisfacer demandas espaciales, relacionando conjuntos de datos por medio de su localización geográfica. Igualmente un SIG puede describirse también enumerando el tipo de demandas a las que puede responder. Un SIG suficientemente sofisticado puede responder a cinco preguntas genéricas:

Localización: ¿Qué hay en ...?

La primera de las preguntas se refiere a identificar que es lo que se encuentra en una localización determinada. La localización puede describirse de varias formas, por ejemplo, por su topónimo, por su código postal, o por

referencias geográficas como latitud y longitud.

Condición: ¿Dónde se encuentra ...?

La segunda demanda es la inversa de la primera y requiere un análisis espacial. En lugar de identificar lo que se encuentra en un punto. Lo que se busca es un lugar que reúna ciertas condiciones (por ejemplo, un terreno sin bosque, que tenga un área mayor de 2000 metros cuadrados, que esté a menos de 100 metros de una carretera y al que sus condiciones geotécnicas le permitan soportar edificios.

Tendencia: ¿Qué ha cambiado desde ...?

Esta pregunta involucra a las dos anteriores y su respuesta establece que diferencias ocurren en un área determinada a través del tiempo.

Distribución: ¿Qué patrones de distribución espacial existen?

Esta pregunta es más compleja. Se plantea al querer determinar, por ejemplo, si el cáncer es una causa importante de mortalidad entre las personas que residen en las proximidades de una central nuclear. O también, al querer conocer cuántas situaciones anormales se producen en una determinada distribución espacial y donde se localizan.

Modelación: ¿Qué sucede si ...?

Cuestión que se plantea al intentar conocer que pasa en un sistema cuando ocurre un hecho determinado, por ejemplo, que le sucede a un sistema viario si construimos una carretera, o que sucedería si se produjera un determinado vertido tóxico en la red de suministro de agua potable. Las respuestas requieren, además de la información geográfica, otras informaciones adicionales, como pueden ser determinadas leyes científicas.

1.2.6 Sistemas de Información Geográfica sobre web con funcionalidad de búsqueda de caminos mínimos

En la red de redes existen publicados una gran cantidad de SIG, implementados para utilizar a través de la web, que brindan la funcionalidad de búsqueda de caminos mínimos, como son:

- El callejero de páginas amarillas.es, disponible en http://callejero.paginasamarillas.es/
- El Callejero LaNetro.com, disponible en http://callejero.lanetro.com/
- Callejero, mapas y fotos de satélites, disponible en http://www.elmundo.es/callejero/
- Callejero Terra, disponible en http://callejero.terra.es/
- El país.com Callejero, disponible en http://www.elpais.com/callejero/

Sólo por mencionar una muestra; todos estos SIG nos brindan una amplia cantidad de información sobre las rutas, cada uno en un país determinado y algunos de ellos en todo el mundo, utilizando los servicios que brinda Google, además, permiten hacer búsquedas del camino más corto para llegar de un lugar a otro, incluso permiten especificar si se está viajando en carro o a pie, entre otras muchas posibilidades.

Como se puede apreciar, existen varios sistemas que brindan una amplia cantidad de información, el problema que se presenta es el siguiente: no se ha publicado, en la bibliografía consultada, la forma en que se maneja tal cantidad de datos, ni los algoritmos utilizados para el análisis de las rutas, además, estos SIG están desarrollados sobre tecnología propietaria. Este no es un problema trivial; para hacer búsquedas de camino mínimo, es necesario tener una estructura de datos (grafo) que represente la información de alguna red de transporte (las calles de un mapa), por lo que un grafo que represente las calles de nuestro país, tendría varios millones de nodos, por lo que hay que prestar especial atención a su procesamiento.

1.2.7 Servidor de mapas

Para el desarrollo de un SIG en la actualidad, es muy importante disponer de un servidor de mapas, ya que éste facilita el acceso a los datos (mapas) de forma transparente, esto quiere decir que el desarrollador del SIG, no tiene que preocuparse por la forma en que está implementado el acceso a los datos, o

por el formato en que se encuentren los mismos. Para la selección del servidor de mapas, se tuvo en cuenta que fuera software libre, siguiendo la política de de desarrollo de software del país y que cumpliera con las especificaciones del consorcio OpenGIS para el servicio WMS [OGC:WMS].

Teniendo en cuenta lo anterior, se decidió utilizar MapServer, ya que además de cumplir con lo mencionado, es el servidor de mapas de código abierto más popular en Internet y existe una amplia experiencia en el desarrollo de SIG utilizando el mismo.

MapServer es un servidor de mapas de código abierto, utilizado para la creación de aplicaciones SIG en Internet/Intranet con el fin de visualizar, consultar y analizar información geográfica a través de la red.

Sus características principales son:

- Se ejecuta bajo plataformas Linux y Windows.
- Formatos vectoriales soportados: ESRI shapefiles [ESRI:SHP], PostGIS [PostGIS], ESRI ArcSDE, GML [GML] y otros muchos vía OGR [OGR].
- Formatos raster soportados: JPG, PNG, GIF, TIFF/GeoTIFF [TIFF], EPPL7 y otros vía GDAL [GDAL].
- Fuentes TrueType [TType].
- Configuración "al vuelo" vía URL.

Además tenemos que mencionar, que Mapserver tiene como deficiencia la falta de herramientas para hacer análisis de redes, imposibilitando de esta forma añadir funcionalidades de búsqueda de camino mínimo o de solución a problemas de optimización.

1.3 Gramáticas de grafo

Las gramáticas de grafo proveen una generalización natural de la teoría de lenguajes formales basada en cadenas y de la teoría de la reescritura de términos.

Sin embargo, al contrario de las gramáticas de cadena, tenemos varias formas de definir las gramáticas de grafo, atendiendo al modo de reescritura.

Las gramáticas de grafo surgieron como una vía de generalización de las gramáticas de cadena, la idea principal fue extender la concatenación de cadenas al ”engomado” (o empotrado) de grafos.

Los grafos pueden ser utilizados para representar diversas entidades, como son:

- Componentes químicos.
- Redes de transporte.
- WWW.
- Mapas (en formato vectorial) de ciudades.
- etc.

Debido a la amplia aplicación de los mismos, dada su capacidad de representar datos de dos o más dimensiones, surge el objeto de estudio “gramáticas de grafo”.

1.3.1 Conceptos

Definición: Un grafo G = (V, E), consiste en un conjunto finito de vértices(o nodos) V, y una relación binaria sobre V que son las aristas que conforman el conjunto E. Cada arista es un par de vértices, o sea, $E = \{(v1, v2)|v1, v2 \in V\}$.

Si el grafo es no dirigido, el orden del par que forma cada arista, sería irrelevante.

Definición: Dos grafos G = (V, E), y G = (V , E) son isomorfos si hay una función biyectiva $f : V \rightarrow V$ tal que $(u, v) \in E$ si y sólo si $(f (u), f (v)) \in E$.

Los grafos isomorfos son esencialmente ”iguales” con la excepción de que sus vértices tienen distintos nombres.

Regla de reescritura:

$g_l ::= g_r$: un subgrafo isomorfo a g_l es sustituido por otro isomorfo a g_r

- Información de empotrado, puede ser textual o gráfica, los modelos de engomado la especifican con un isomorfismo de engomado.
- Condición de aplicación, son restricciones sobre la aplicación de la regla, es opcional.
- Función de transferencia de atributos, es una función que asigna atributos, es opcional.

Definición: Una gramática de grafo o web es una 6-tupla G = (Σ, Δ, Γ, Ω, P, S), donde:

- Σ es el alfabeto de las etiquetas no terminales de los nodos.
- Δ es el alfabeto de las etiquetas terminales de los nodos.
- Γ es el alfabeto de las etiquetas no terminales de las aristas.
- Ω es el alfabeto de las etiquetas terminales de las aristas.
- P es el conjunto finito de producciones de la gramática o reglas de reescritura.
- S es el conjunto de grafos iniciales, que normalmente consisten en un nodo con una etiqueta no terminal.

Una producción de una gramática G tiene la forma X → (D, E), donde:

- X es un símbolo no terminal.
- D es un grafo.
- E es la información de empotrado.

1.3.2 Tipos de gramáticas de grafo

Los grafos están formados por dos tipos de objetos: nodos y aristas; los objetos pudieran estar etiquetados. Los nodos y las aristas pueden ser reemplazados por grafos. En otras palabras, cuando se define una gramática de grafo, es natural escoger entre reemplazo de nodos o reemplazo de aristas. En general, una producción tiene la forma X → (D, E), aunque debemos aclarar, que el conjunto de reglas de reescritura E, depende del tipo de gramática.

Para un grafo dado H, ¿cómo una producción puede ser aplicada a cualquier nodo o arista m de H con etiqueta X?.

La aplicación de una producción consiste en eliminar m del grafo H, y conectar D con la vecindad de m, según se especifique en E. Después de escoger entre nodos o aristas, debemos decidir como conectar D a la vecindad de m, o cómo empotrar D en H; en otras palabras, debemos seleccionar el mecanismo de empotrado de la gramática de grafo.

A continuación, veremos dos tipos de gramáticas de grafo libres de contexto, una basada en en reemplazo de nodos, y la otra en reemplazo de aristas.

1.3.2.1 Reemplazo de nodos

Un tipo muy simple de gramáticas de grafo libres de contexto que usa reemplazo de nodos son las gramáticas de grafo B-NCE[5] . Estas gramáticas generan grafos no dirigidos con nodos etiquetados.

El mecanismo de empotrado conecta D con H estableciendo aristas entre D y la vecindad de m (los nodos adyacentes a m), dependiendo de las etiquetas de estas vecindades. Para ser más preciso, una producción en una gramática B-NCE, es de la forma X → (D, E), donde X es un nodo(no terminal) etiquetado, D es un grafo no dirigido (con nodos terminales y no terminales) y E es un conjunto finito de instrucciones de conexión, una instrucción de conexión es un par (σ, x), donde σ es la etiqueta de un nodo terminal, y x es un nodo de D.

La aplicación de la producción a un nodo m (con etiqueta X) de un grafo H consiste en eliminar m y todas sus aristas incidentes del grafo H, luego añadimos D a H(sin aristas que los conecten), y construimos todas las aristas {w,x} tal que E contiene una instrucción de conexión (σ, x), y w es un vecino de m en H con etiqueta σ.

Como σ es la etiqueta de un nodo terminal (esta es la restricción de

5 *B-NCE: Empotrado de vecindad controlada, bajo la restricción de limite o frontera [Blostein1994](Neigbourhood Controled Embedding, under the Boundary restriction)*

frontera), podemos asumir que no existen aristas entre nodos no terminales en D, es decir, cada nodo no terminal, está ”rodeado” de nodos terminales, dicho de otra forma, no existen dos nodos no terminales que sean adyacentes; esto nos asegura que el proceso de reescritura es siempre libre de contexto.

1.3.2.2 Reemplazo de aristas

Un tipo simple de gramática de grafo libre de contexto que usa reemplazo de aristas son las gramáticas ER[6].

Estas gramáticas generan grafos dirigidos con las aristas etiquetadas. El mecanismo de empotrado conecta D con H identificando los nodos fuente y destino de la arista m con dos nodos particulares de D. Para ser mas preciso, una producción en una gramática ER es de la forma X → (D, E), donde X es una arista(no terminal) etiquetada, D es un grafo dirigido (con aristas etiquetadas terminales y no terminales), y E es un par ordenado (d_1 ,d_2), donde d_1 y d_2 son nodos de D. Sea m una arista del grafo H, con etiqueta X, nodo fuente v_1 , y destino v_2 . La aplicación de una producción a m consiste en eliminar m, añadir D al grafo inicial H(sin conectarlos), y luego, identificar d_1 con v_1 y d_2 con v_1.

1.4 Mecanismos de reescritura de grafos

Los grafos proveen una expresiva y versátil representación de datos. Usualmente, los nodos o vértices, representan objetos o conceptos, y las aristas representan las relaciones entre los mismos. Además, podemos representar información adicional añadiendo atributos a los nodos o a las aristas. Debido al extendido uso de los grafos para representar datos, es natural que las formas de manipular grafos sean elementos básicos de variados cómputos y/o algoritmos que utilicen los mismos como representación de los datos involucrados para darle solución al problema que se está tratando.

Las transformaciones sobre los grafos, pueden representarse implícitamente, embebidas en el programa que, entre otras cosas, construye o modifica el

6 *Gramáticas de reemplazo de aristas, del inglés, Edge Replacement Grammars.*

grafo. De forma alternativa, las transformaciones pueden representarse explícitamente, utilizando reglas de reescritura claras y bien delimitadas que modifiquen el grafo. El uso explicito de las reglas de reescritura de grafos, nos otorga varias ventajas. La reescritura de grafos provee una representación abstracta y de alto nivel de una solución a un problema computacional.

La reescritura de grafos tiene el potencial de ser muy útil en una gran variedad de aplicaciones. Sin embargo, es difícil para los diseñadores de software evaluar los méritos del uso de la reescritura de grafos para solucionar varios de sus problemas de software. La literatura de reescritura de grafos es extensa y altamente técnica. Aquí se presenta una visión general de la reescritura de grafos, enfatizando en su aplicación a problemas prácticos.

Una regla de reescritura de grafo es aplicada a un grafo para sustituir un subgrafo por otro. En este documento trataremos los mecanismos de reescritura de grafos secuenciales, aunque las reglas de reescritura se pueden aplicar también en paralelo. La reescritura secuencial y en paralelo es revisada por [Nagl1979].

1.4.1 Terminología

La terminología para la reescritura de grafos no está estandarizada, en este documento, se utilizarán los términos que se definen a continuación:

Regla de reescritura:

- $g_l ::= g_r$, un subgrafo isomorfo a g_l es sustituido por otro isomorfo a g_r
- Información de empotrado, puede ser textual o gráfica, los modelos de engomado la especifican con un isomorfismo de engomado.
- Condición de aplicación, son restricciones sobre la aplicación de la regla, es opcional.
- Función de transferencia de atributos, es una función que asigna atributos, es opcional.

g: Grafo al cual le vamos a aplicar la regla

g_l^{host}*:* subgrafo a ser reemplazado.

g_r^{host}: subgrafo usado para reemplazar a g_l^{host} .

Resto del grafo: $g\text{-}g_l^{host}$

1.4.2 Clasificación de los mecanismos de empotrado

Una regla de reescritura, provee información de empotrado, la cual es usada para conocer la forma de conectar el grafo g_r^{host}, con el resto del grafo. La principal problemática es convertir las aristas pre-empotradas en aristas post-empotradas.

El mecanismo de empotrado, debe especificar para cada posible arista pre-empotrada lo siguiente:

- La dirección de la arista pre-empotrada.
- La etiqueta de la arista pre-empotrada.
- El vértice en el cual termina la arista(cero, uno o más vértices).

Algunos mecanismos de empotrado, permiten una especificación no restringida de las aristas post-empotradas, aunque también existen mecanismos que si establecen una restricción. *La selección de un mecanismo, nos pone en la disyuntiva siguiente: usamos un mecanismo complejo pero con pocas reglas de escritura, o uno de muchas reglas de escritura que sean simples*.

Nagl desarrolló la siguiente clasificación de los mecanismos de empotrado [Nagl1979] [Nagl1987], aquí veremos una lista en orden descendente de complejidad:

- No restringido: Expresión, Diagramático.
- Preserva la orientación y la etiqueta
- Profundidad 1
- Simple

- Elemental
- Análogo
- Invariante

No restringido

Podemos encontrar una secuencia de aristas, comenzando con una arista pre-empotrada, condicionada por la orientación y las etiquetas de las mismas, que terminan en un conjunto de nodos del resto del grafo.

Las direcciones y las etiquetas de las arista post-empotradas pueden ser escogidas libremente.

A continuación, mostramos dos ejemplos de mecanismos de empotrados no restringidos.

Preserva la orientación y la etiqueta

Como en el caso anterior, pero cuando seguimos una arista con cierta orientación y etiqueta, todas las aristas post-empotradas, deben tener la misma orientación y etiqueta

Profundidad 1

Como en el caso no restringido, pero los vértices de las aristas post-empotradas que pertenezcan al resto del grafo, quedan restringidos a los adyacentes de g_l^{host}.

Cada especificación de empotrado es una colección de cinco elementos (n,n',w_1,w_2,v)

donde:

- n: un vértice de g_l
- n':un vértice de g_r
- w_1, w_2: aristas etiquetadas
- v: un vértice del resto del grafo

Para aplicar la regla:

Buscamos una arista pre-empotrada de etiqueta w_1 y que conecte el nodo n de g_l^{host}, con un nodo del *resto del grafo* con etiqueta v.

Se transforma en una arista post-empotrada de etiqueta w_2 y que conecte el nodo n' de g_r^{host}, con el mismo nodo del *resto del grafo* con etiqueta v.

La arista pre-empotrada que no aparezca en ninguna especificación de empotrado, no aparecerá como arista post-empotrada.

Si una arista pre-empotrada aparece en varias especificaciones, se convertirá en varias aristas post-empotradas.

Simple

Cumple con la clasificación Profundidad 1, y preserva la orientación y la etiqueta.

Elemental

Como el caso anterior (Simple), pero el empotrado no depende de las etiquetas de los vértices (nodos) del resto del grafo.

Cada especificación de empotrado, está compuesta por pares de la forma (n,n’) donde:

- n, nodo de g_l.
- n’ , nodo de g_r.
- El resto del grafo, permanece invariable.

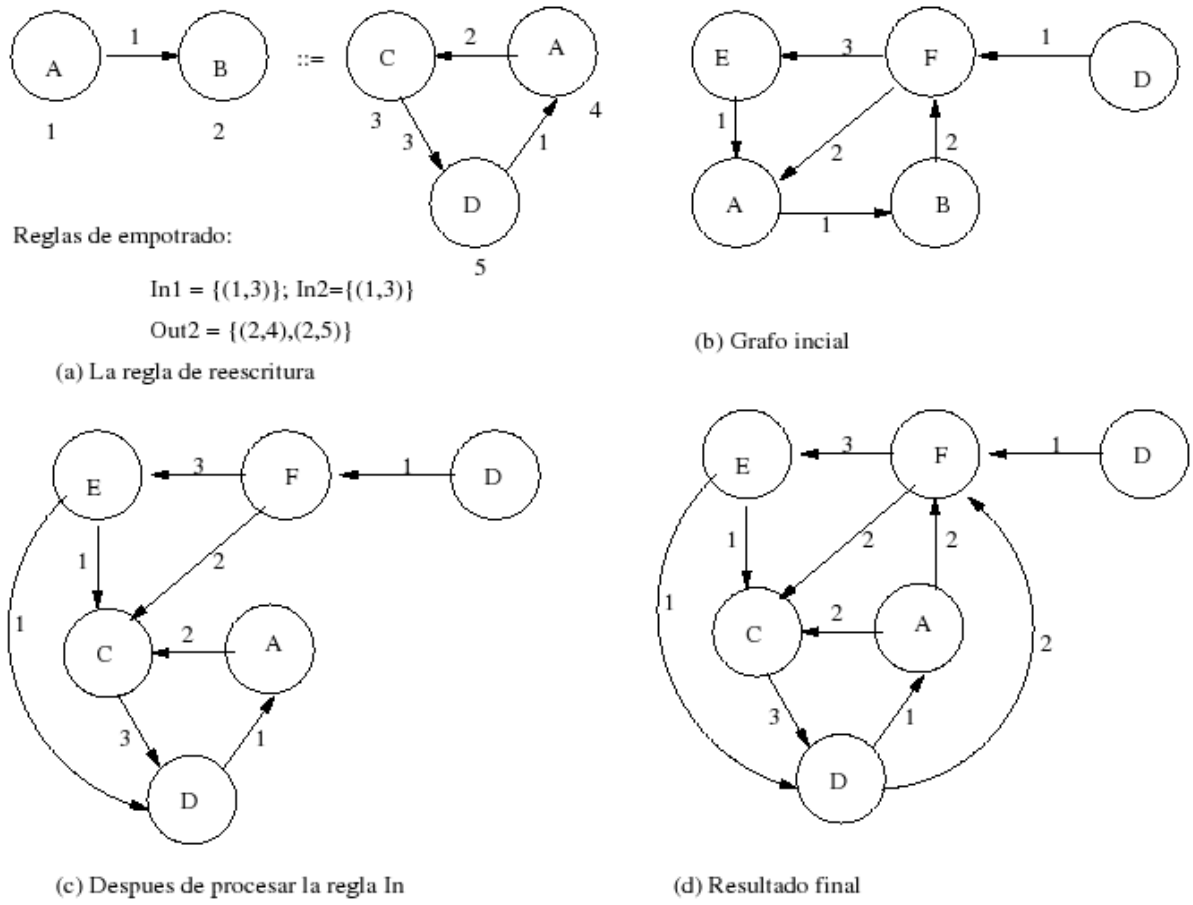

Ilustración 5: Ejemplo de mecanismo de empotrado de Schneider

En la Ilustración 5 se muestra un ejemplo de mecanismo de empotrado. En la regla de reescritura (a), las etiquetas de los nodos están dentro de los mismos, mientras que las denotaciones están en el exterior. La especificación de empotrado consiste de de dos conjuntos: Ini y Outi , los cuales especifican la transformación de las aristas con etiqueta i. Cada conjunto Ini y Outi está formado por pares de la forma (n,n'), donde n representa un nodo de g_l y n' representa uno de g_r . Cada par representa una arista pre-empotrada (la arista i) que conecta a un nodo n en g_l^{host} , y será transformada en una arista post-empotrada conectada a un nodo n' en g_r^{host} . La orientación, etiquetas, y los puntos finales en el resto del grafo, no varían. La notación se simplifica si las

aristas no tienen etiquetas (se utilizará un sólo conjunto In y uno Out) o si las aristas no son dirigidas (se utilizaría un sólo conjunto en lugar de dos: In y Out)

Análogo

Elemental, y el empotrado es independiente de la orientación y la etiqueta de las aristas pre empotradas. Ejemplo: Aplicación del empotrado de Schneider [Schneider1993] a un grafo no dirigido y sin etiquetas.

Invariante

Aquí existe un mapeo entre los nodos de g_l y g_r tal que g_r^{host} se encarga directamente de las aristas pre-empotradas de g_l^{host} . Este es el único tipo de empotrado que no permite la división y contracción de aristas, o sea, el número de aristas pre-empotradas es igual al número de aristas post-empotradas.

El modelo de engomado, provee un importante uso del empotrado invariante. El mismo tiene una fuerte base matemática, con teoremas muy útiles concernientes al orden de dependencia y al paralelismo en la aplicación de las reglas.

En el método de engomado de la reescritura de grafos, una regla de isomorfismo de engomado, actúa como especificación de empotrado. En este caso, los nodos y las aristas pueden ser utilizados como puntos de engomado, en este documento, sólo trataremos el caso de los nodos.

La regla de reescritura designa un subconjunto de nodos de gl como puntos o nodos de engomado, el isomorfismo de engomado establece una correspondencia entre estos nodos y un conjunto de nodos de g_r . Para aplicar una regla siempre debemos chequear la condición de engomado: cada arista pre-empotrada debe estar conectada a un nodo de engomado en g_l^{host}. (Si hay una arista pre-empotrada que no esté conectada a un punto de engomado de g_l^{host} , entonces la regla no se aplica, o queda prohibida).

El isomorfismo de engomado provee un empotrado invariante, especificando

cuál nodo de g_r^{host} hereda las aristas de cada nodo de engomado de g_l^{host}. Debido a la simplicidad del empotrado invariante, un paso de la reescritura de grafo no necesita eliminar una arista pre-empotrada y sustituirla con una post-empotrada. En lugar de eso, se elimina g_l^{host} excepto los nodos de engomado y se añade g_r^{host} , excepto sus nodos de engomado. Esto mantiene las aristas empotradas invariantes.

Las clasificaciones vistas anteriormente, caracterizan la clase de los lenguajes generados por una gramática de grafo. Dado un tipo particular de una producción (como libre de contexto o dependiente de contexto), se origina una jerarquía de lenguajes de grafo, basada en el mecanismo de empotrado [Nagl1987]:

$$expression - T \supseteq preserva.orientacion.etiqueta - T, prof undidad1 - T \supseteq$$

$$simple - T \supseteq elemental - T \supseteq analogo - T \supseteq invariante - T$$

Las igualdades se cumplen cuando las producciones son del tipo no restringido.

Este es el momento apropiado para revisar más consideraciones sobre los variados mecanismos de reescritura de grafos:

- La notación “expression” provee empotrado no restringido.
- La notación “diagramatic” provee empotrado no restringido. Las notaciones X, Y y Δ, utilizan un contexto “requerido” para marcar las partes del grafo comunes a g_l y g_r , un contexto “opcional” para marcar el empotrado, y un contexto “prohibido” para marcar la estructura del grafo host prohibida(la cual es, de otra forma, incluida como parte de la condición de aplicación)
- NCE (Neighbourhood Controlled Embedding) y NCL (Node Label Control) proveen empotrado del tipo profundidad 1 (Depth 1), y tienen importantes propiedades teóricas.
- La notación de Scheneider provee un empotrado elemental.

- Los modelos de engomado utilizan el empotrado invariante, el cual se especifica a través de isomorfismos de engomado. Estos modelos incluyen el método algebraico, sistemas de reemplazo de aristas y sistemas de reemplazo de hiperaristas.
- La reescritura de grafo estructurada es un modelo abarcador que combina empotrado y engomado en un sólo sistema.

Dada la diversidad de los mecanismos de reescritura, es importante considerar algunos elementos prácticos relacionados con la selección de un mecanismo particular.

1.4.3 Consideraciones prácticas para seleccionar un mecanismo de reescritura

Un sólo mecanismo de reescritura no es adecuado para todas las aplicaciones. La selección de un mecanismo particular, depende en una gran medida de la aplicación, de la disponibilidad de herramientas y de los gustos del diseñador del sistema. Análogamente, una gran variedad de lenguajes de programación siguen teniendo uso generalizado, a pesar de la existencia de lenguajes de programación de propósito general.

Muchos factores son relevantes cuando se selecciona un mecanismo de reescritura, incluyendo la potencia del empotrado, propiedades formales, legibilidad y manejabilidad intelectual, eficiencia de la aplicación de las reglas y las herramientas para desarrollar y depurar reglas. A continuación se mostrarán algunas experiencias obtenidas al aplicar las reglas de reescrituras al reconocimiento de diagramas.

A pesar de que no se puede hacer una recomendación definitiva, el uso práctico de la reescritura de grafos puede ser promovido aclarando los tópicos relacionados con la selección del mecanismo de reescritura.

1.4.3.1 Potencia del empotrado

Los mecanismos de empotrado complejos, permiten una inspección y manipulación significativa del grafo durante la aplicación del empotrado. Los mecanismos de empotrado más restringidos, tales como el empotrado invariante y los modelos de engomado, no son convenientes para expresar ciertas operaciones de grafo comunes. Por ejemplo, consideremos la operación de eliminar un nodo con etiqueta A del grafo host. Todas las aristas incidentes en el nodo en cuestión deben ser eliminadas también. Esta operación es fácilmente llevada a cabo utilizando un empotrado elemental, pero es difícil cumplirla con un empotrado invariante.

La selección de un mecanismo de empotrado, involucra la decisión de escoger pocas reglas de reescritura complejas, o muchas reglas simples. Muchas aplicaciones de reescritura de grafos, pueden tener un nivel natural de complejidad del mecanismo de empotrado.

1.4.3.2 Propiedades formales

Las propiedades formales tienen importancia práctica en la reescritura de grafos. La robusta subestructura teórica de los modelos de engomado pueden ofrecer ventajas significantes. Por ejemplo, la reescritura algebraica de grafos permite una fácil construcción de pruebas de integridad en un sistema de bases de datos (sistema de transacción bibliotecaria de [Ehrig1980], sin embargo, el modelo de engomado es apropiado sólo si el empotrado invariante es suficiente para expresar convenientemente la reescritura de grafo necesitada.

1.4.3.3 Legibilidad y manejabilidad intelectual

La legibilidad es una consideración muy importante, con efectos en la manejabilidad intelectual, tiempo de desarrollo del sistema, fácil mantenimiento y depurado. Empotrados complejos, son muy difíciles de especificar, entender y depurar. La notación diagramática muestra las aristas

de empotrado gráficamente como aristas entre el grafo de contexto y g_r.

Debido a esto, si utilizamos empotrados muy complejos, pueden ser entendidos fácilmente con la notación diagramática. La presentación visual puede ser simplificada evitando la duplicaciones de subgrafos comunes a g_r y a g_l. Según experiencias, mecanismos de empotrados tan complejos como el de Schneider [Schneider1993], pueden ser comprendidos fácilmente utilizando la notación diagramática.

La manejabilidad intelectual es una importante consideración. Algunas aplicaciones necesitan empotrados complejos, y otras no. En el uso práctico de la reescritura de grafos, como se ha visto, las reglas de reescritura tienden a ser bastante simples, con empotrados sencillos. Generalmente, las mayores dificultades no radican en la formulación de una regla individual, sino en la estructuración de varias reglas que interactúan entre si.

1.4.3.4 Eficiencia de la aplicación de las reglas

La reescritura de grafos tiene reconocidos problemas de eficiencia. La aplicación de una regla de reescritura, requiere que el subgrafo g_l^{host} sea localizado en el grafo host o inicial, lo cual, involucra pruebas de isomorfismo de subgrafo.

Puesto que la forma general de un isomorfismo de subgrafo es un problema NP-completo, la reescritura de grafos puede ser computacionalmente costosa.

Sin embargo, a menudo es posible expresar un cómputo utilizando subgrafos g_l pequeños. Los nodos y aristas etiquetados, y la dirección de las aristas, reducen drásticamente el espacio de búsqueda para los subgrafos isomorfos. Además de esto, algunos sistemas de reescritura de grafos, tienen ciertas frases que aparecen frecuentemente en las condiciones de aplicación; esto también lo podemos explotar para reducir el espacio de búsqueda para los subgrafos isomorfos que cumplan con la condición de aplicación.

1.5 Reducción de grafos

Existen algunos sistemas para el análisis y diseño de grafos con una gran cantidad de usuarios y con un alto grado de madurez [Grafos], y están documentadas una serie de soluciones a problemas que implican reducción de grafos [MRG], por ejemplo, reducción de expresiones representadas como grafos [REG], obtención de arboles de expansión [ArbolExp], entre otras; pero todas utilizan el grafo reducido sin tener en cuenta la forma inicial del gafo, o sea, los análisis y algoritmos se aplican al grafo reducido en todo momento, obviando de esta forma el grafo original.

En el caso de los análisis sobre mapas, sería deseable tener una representación de grafo de los objetos del mapa implicados en una análisis determinado, para poder aplicar desde algoritmos simples como búsquedas de camino mínimo, hasta otros un poco mas complejos como el que da solución al problema del viajante, o al problema de optimización “Flujo máximo de costo mínimo”. Esto tiene sus inconvenientes, por ejemplo, si se necesita hacer un análisis determinado sobre un mapa con un número elevado de objetos geográficos, como podría ser el mapa de Cuba llevado al detalle de las casas y las calles, se tendría un grafo que no se podría cargar en memoria de una sola vez, además, si se desea hacer un análisis de una región del mapa, habría que tener el grafo completo en memoria, aunque se fuera a utilizar sólo una parte, es por esto que es conveniente definir un algoritmo que permita obtener un grafo reducido, teniendo como entrada un grafo cualquiera.

Pero el problema no termina aquí, ya se tiene el grafo reducido, pero se quiere hacer un análisis de una determinada región del mapa (una región pequeña, de forma tal que se encuentre completamente reducida en el nuevo grafo), por ejemplo, encontrar un camino entre dos lugares que pertenecen al mismo reparto (el grafo está reducido por municipios, o sea, cada nodo en el grafo representa un municipio), se necesitaría un mecanismo que permita expandir la parte del grafo que se desea analizar, pero sólo esta parte, y mantener los restantes nodos del grafo reducidos.

CAPÍTULO

2. SELECCIÓN DE LA PLATAFORMA DE DESARROLLO

2.1 Introducción

El objetivo de este capítulo es mostrar algunas de las características principales de las plataformas estudiadas.

Para seleccionar las plataformas y lenguajes a estudiar, se tuvo en cuenta como requisitos indispensables que fueran multiplataforma y software libre, siguiendo la política que plantea el país sobre la migración a software libre.

2.2 Php

PHP (acrónimo de "Hypertext Preprocessor") es un lenguaje de "código abierto" interpretado, de alto nivel, embebido en páginas HTML y ejecutado en el servidor [PHPDocs].

Muy pocas veces se vio tanta expectativa ante el lanzamiento de una nueva versión (la versión 5) de PHP, esto se debe a que las nuevas características que incluye PHP5 son cruciales para que los desarrolladores y los Ingenieros en Sistemas comiencen a tomar mucho más en serio al lenguaje y que sea más aceptado para desarrollos medianos y grandes, ofreciendo la posibilidad de escribir códigos más legibles y eficientes, así como el uso de patrones de

diseño que en versiones anteriores no se podían utilizar.

Las características que propician la Programación Orientada a Objetos (POO) bajo PHP, han sido extensamente revisadas y mejoradas, más que nada oyendo las necesidades que los desarrolladores clamaban como fundamentales: objetos por referencia, clases abstractas, interfaces, variables y funciones privadas y protegidas, son sólo algunas de ellas, que harán que PHP sea aún mucho más poderoso y flexible que antes.

¿Qué se puede hacer con PHP?

Con el lenguaje PHP podemos procesar la información de formularios, generar páginas con contenidos dinámicos, o enviar y recibir cookies y de forma general, podemos tener las funcionales que se puedan desarrollar con un script CGI[7].

Existen tres campos en los que se usan scripts escritos en PHP.

- Scripts del lado del servidor. Este es el campo más tradicional y el principal foco de trabajo. Se necesitan tres cosas para que esto funcione: El intérprete PHP (como CGI o módulo del servidor web), un servidor web y un navegador. Es necesario correr el servidor web con PHP instalado [PHPInstall]. El resultado del programa PHP se puede obtener a través del navegador, conectándose con el servidor web.

- Scripts en la línea de comandos. Puede crear un script PHP y correrlo sin ningún servidor web o navegador. Solamente necesita el intérprete PHP para usarlo de esta manera [PHPLineaComandos]. Este tipo de uso es ideal para scripts ejecutados regularmente desde cron[8] (en *nix o Linux) o el Planificador de tareas (en Windows). Estos scripts también pueden ser usados para tareas simples de procesamiento de texto.

- Escribir aplicaciones de interfaz gráfica (aplicaciones de escritorio).

7 *CGI es la abreviatura de Common Gateway Interface, en español Interfaz común de pasarela, es un mecanismo de comunicación entre el servidor web y una aplicación externa.*

8 *Un administrador regular de procesos en segundo plano que ejecuta programas a intervalos regulares [WikiCron].*

> Probablemente PHP no sea el lenguaje más apropiado para escribir aplicaciones gráficas, pero se puede utilizar para desarrollar aplicaciones de escritorio, haciendo uso de PHP-GTK [PHPGTKUserGuide]. También es posible escribir aplicaciones independientes de una plataforma. PHP-GTK es una extensión de PHP, no disponible en la distribución principal.

PHP puede ser utilizado en los principales sistemas operativos del mercado, incluyendo Linux, muchas variantes Unix (incluyendo HP-UX, Solaris y OpenBSD), Microsoft Windows, Mac OS X, RISC OS entre otros. PHP soporta la mayoría de servidores web de hoy en día, como son: Apache, Microsoft Internet Information Server, Personal Web Server, Netscape e iPlanet, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd entre otros. PHP tiene módulos disponibles para la mayoría de los servidores, para aquellos otros que soporten el estándar CGI, PHP puede usarse como procesador CGI.

Por consiguiente, cuando se utiliza PHP, se tiene la libertad de elegir el sistema operativo y el servidor que se va a utilizar. También tiene la posibilidad de usar programación estructurada y/o programación orientada a objetos. Aunque no todas las características estándar de la programación orientada a objetos están implementadas en la versión actual de PHP, muchas bibliotecas y aplicaciones (incluyendo la biblioteca PEAR) están escritas íntegramente usando programación orientada a objetos.

Los resultados de la ejecución de un script PHP no están limitados a páginas HTML. Entre las habilidades de PHP se incluyen: creación de imágenes, archivos PDF y películas Flash (usando libswf y Ming). También puede presentar otros resultados, como XHTM y archivos XML. PHP puede autogenerar éstos archivos y almacenarlos en el sistema de archivos en vez de presentarlos en la pantalla.

Quizás la característica más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos. Escribir una interfaz vía web para una base de datos es una tarea simple con PHP. Las siguientes bases de datos están soportadas actualmente:

Adabas D	Ingres	Oracle (OCI7 and OCI8)
dBase	InterBase	Ovrimos
Empress	FrontBase	PostgreSQL
FilePro (read-only)	mSQL	Solid
Hyperwave	Direct MS-SQL	Sybase
IBM DB2	MySQL	Velocis
Informix	ODBC	Unix dbm

Tabla 1: Bases de datos soportadas por PHP

También contamos con una extensión DBX de abstracción de base de datos que permite usar de forma transparente cualquier base de datos soportada por la extensión. Adicionalmente, PHP soporta ODBC (el estándar abierto de conexión con bases de datos), por lo que puede conectarse a cualquier base de datos que soporte dicho estándar.

PHP también cuenta con soporte para comunicarse con otros servicios usando protocolos tales como LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (en Windows) y muchos otros. También se pueden crear sockets puros. PHP soporta WDDX para el intercambio de datos entre lenguajes de programación en web. Y en términos de interconexión, PHP puede utilizar objetos Java de forma transparente como objetos PHP y la extensión de CORBA puede ser utilizada para acceder a objetos remotos.

PHP tiene características muy útiles para el procesamiento de texto, desde expresiones regulares POSIX extendidas [ERPOSIX] hasta procesadores de documentos XML. Para procesar y acceder a documentos XML, el lenguaje soporta los estándares SAX y DOM. Puede utilizar la extensión XSLT para transformar documentos XML.

Para terminar, PHP cuenta con otras extensiones muy interesantes: las funciones del motor de búsquedas mnoGoSearch, funciones para pasarelas de IRC, utilidades de compresión (gzip, bz2), conversión de calendarios, traducción, etc.

2.3 Java

Java es un *lenguaje orientado a objetos* [Java], esto significa que posee ciertas características que hoy en día se consideran estándares en los lenguajes orientados a objeto:

- Objetos
- Clases
- Métodos
- Subclases
- Herencia simple
- Enlace dinámico
- Encapsulamiento

La Simplicidad de Java

Java ha sido diseñado para eliminar las complejidades de otros lenguajes como C y C++. Si bien posee una sintaxis similar a C, con el objeto de facilitar la migración de C hacia a Java, Java es semánticamente muy distinto a C:

- Java no posee aritmética de punteros: La aritmética de punteros es el origen de muchos errores de programación que no se manifiestan durante la depuración y que una vez que el usuario los detecta son difíciles de resolver.
- No se necesita hacer delete: Determinar el momento en que se debe liberar el espacio ocupado por un objeto es un problema difícil de resolver correctamente. Esto también es el origen de errores difíciles de detectar y solucionar.
- No hay herencia múltiple: En C++ esta característica da origen a muchas situaciones de borde, donde es difícil predecir cuál será el resultado. Por esta razón en Java se opta por herencia simple, que es mucho más sencilla de aprender y dominar.

Java posee bibliotecas de clases estándares

Toda implementación de Java debe tener las siguientes bibliotecas de clases:

- Manejo de archivos
- Comunicación de datos
- Acceso a la red Internet
- Acceso a bases de datos
- Interfaces gráficas

La interfaz de programación de estas clases es estándar en todas las implementaciones, es decir, en todas las implementaciones de Java, las operaciones se invocan con el mismo nombre y los mismos argumentos.

Java es multiplataforma

Los programas en Java pueden ejecutarse en cualquiera de las siguientes plataformas, sin necesidad de hacer cambios:

- Windows/95 y /NT
- Power/Mac
- Unix (Solaris, Silicon Graphics, ...)
- Linux

La compatibilidad es total:

- A nivel de fuentes: El lenguaje es exactamente el mismo en todas las plataformas.
- A nivel de bibliotecas: En todas las plataformas están presentes las mismas bibliotecas estándares.
- A nivel del código compilado: el código intermedio que genera el compilador es el mismo para todas las plataformas. Lo que cambia es el intérprete del código intermedio.

El Look-and-Feel[9]

Lo único que varía de acuerdo a la plataforma es el *look-and-feel*. Un programa en Windows/95 tendrá el aspecto característico de esta plataforma (en cuanto a la forma de los botones, barras de deslizamiento, menús, etc.). El *mismo programa* en Unix tendrá el aspecto característico de Motif y en Power/Mac se verá como un programa para Macintosh.

Sin embargo, en el código que escriben los programadores, no es necesario tener presente las características de ninguna de estas plataformas. Es la implementación de la interfaz gráfica estándar de Java la que se encarga de desplegar las ventanas con el *look-and-feel* de la plataforma local.

Java es flexible

Java combina flexibilidad, robustez y legibilidad gracias a una mezcla de chequeo de tipos durante la compilación y durante la ejecución. En Java se pueden tener punteros a objetos de un tipo específico y también se pueden tener punteros a objetos de cualquier tipo.

El programador usa punteros de tipo específico en la mayoría de los casos con el fin de ganar legibilidad y en unos pocos casos usa punteros a tipos desconocidos cuando necesita tener flexibilidad.

Java administra automáticamente la memoria

En Java los programadores no necesitan preocuparse de liberar la memoria cuando ya no lo necesitan, es el recolector de basuras el que determina cuando se puede liberar la memoria ocupada por un objeto.

Un recolector de basuras es un gran aporte a la productividad. Se ha estudiado en casos concretos que los programadores han dedicado un cuarenta por ciento del tiempo de desarrollo a determinar en qué momento se puede liberar la memoria ocupada por un objeto.

Este porcentaje de tiempo aumenta a medida que se incrementa la

9 *El Look-and-Feel es la apariecia que se proporciona a los diferentes componentes: botones, cajas de texto, cajas de selección, listas, etc.*

complejidad del software en desarrollo. Es relativamente sencillo liberar correctamente la memoria en un programa de 1000 líneas. Sin embargo, es difícil hacerlo en un programa de 10000 líneas y se puede postular que es imposible liberar correctamente la memoria en un programa de 100000 líneas.

Es inevitable que la fase de prueba dejará pasar errores en el manejo de memoria que sólo serán detectados más tarde por el usuario final. Probablemente se incorporarán otros errores en la fase de mantenimiento.

Resumen

Java es un lenguaje que ha sido diseñado para producir software:

- Confiable: Minimiza los errores que se escapan en la fase de prueba.
- Multiplataforma: Los mismos binarios funcionan correctamente en Windows/95 y /NT, Unix/Motif y Power/Mac.
- Seguro: Los applets recuperados por medio de la red no pueden causar daño a los usuarios.
- Orientado a objetos: Beneficioso tanto para el proveedor de bibliotecas de clases como para el programador de aplicaciones.
- Robusto: Los errores se detectan en el momento de producirse, lo que facilita la depuración.

2.4 Python

Python es un lenguaje de programación dinámico orientado a objetos, que puede ser utilizado para muchos tipos de desarrollos de software. El mismo ofrece un soporte robusto para la integración con otros lenguajes y herramientas, incluye múltiples librerías estándares y puede aprenderse en pocos días. Muchos programadores de Python reportan ganancias sustanciales en la productividad y sienten el aumento en la calidad del desarrollo con un código al que se le puede dar mantenimiento de una forma más sencilla [PythonSitioOficial].

Python corre sobre Windows, Linux/Unix, Mac OS X, OS/2, Amiga, Palm Handhelds, y teléfonos móviles Nokia y también ha sido reescrito para las máquinas virtuales de java y .NET.

Si se quiere utilizar Python en un sistema operativo que no esté listado en el párrafo anterior, sólo se necesita un compilador de C para compilar el código fuente del lenguaje, y posteriormente podremos utilizarlo.

Python es distribuido bajo licencia open source, la cual lo hace libre, incluso para productos comerciales.

La Fundación de Software de Python (PSF) mantiene y protege los derechos de propiedad intelectual detrás de Python, financia la conferencia de PyCon, y financia los permisos y los otros proyectos en la comunidad de Python [PSF] .

La PSF no sería posible sin la generosa ayuda financiera de los siguientes patrocinadores:

- ActiveGrid, patrocinador desde enero de 2005, representada por Peter Yared.
- ActiveState representa por David Ascher.
- Advanced Simulation Technology Inc. (ASTi), desde enero de 2002, representada por Patrick Gaffney.
- Array BioPharma, desde octubre de 2003, representada por Daniel Weaver.
- BizRate.com, desde enero de 2004, representada por Henri Asseily.
- CCP Games, desde septiembre de 2005, representada por Kristján Valur Jónsson.
- Enthought, desde enero de 2004, representada por Travis Vaught.
- Google, desde abril de 2004, representada por Chris DiBona.
- Hostway Corporation, desde abril de 2002, representada por Jason Abate.

- IronPort Systems.
- Merfin LLC, desde julio de 2002, representada por Reggie Dugard.
- O'Reilly & Associates, Inc., desde abril de 2002, representada por Betsy Waliszewski.
- Open Source Applications Foundation, desde octubre de 2003, representada por Ted Leung.
- Opsware, desde enero de 2004, representada by Ray Suorsa.
- Strakt Holdings, Inc., desde julio de 2002, representada por Laura Creighton.
- ZeOmega, desde enero de 2004, representada por Sathya Rangaswamy.
- Zope Corporation representada por Jim Fulton.

Las siguientes compañías, están pendientes como miembros patrocinadores:

- cPacket Networks, desde junio de 2006, representada por Rony Kay.
- Madison Tyler LLC, desde julio de 2006, representada por John Benediktsson.
- Microsoft, desde agosto de 2006, representada por Jim Hugunin.
- Tabblo, desde abril de 2006, representada por Antonio Rodriguez.

A continuación se enumera algunas características del lenguaje:

- Sintaxis clara y legible.
- Fuerte capacidad de introspección.
- Orientación a objeto intuitiva.
- Completa modularidad, soportando paquetes jerárquicos.
- Manejo de errores basado en excepciones.
- Tipos de datos dinámicos de alto nivel.

1. Librerías estándares de gran alcance y terceros módulos para virtualizar cada tarea.
2. Se pueden escribir fácilmente extensiones y módulos en C y C++, en java para Jython, o en .NET para IronPython.
3. Empotrable dentro de aplicaciones como una interfaz de script (interface scripting).

2.5 Zope como servidor de aplicaciones

El acrónimo "Zope" significa Z Object Publishing Environment (la "Z" no significa nada en particular). La mayoría del código de Zope ha sido escrito en Python, un lenguaje de scripts con las partes más criticas de rendimiento escritas en C [Zope] [Zope3].

¿Por qué usar Zope a cambio de otros servidores de aplicaciones?

Si se necesita desarrollar aplicaciones web, Zope nos puede ayudar a crearlas con menos costos y mucho mas rápido que con otro servidor de aplicaciones. Esto es debido a características propias de Zope:

- Zope es gratuito y es distribuido bajo licencia de software libre, a diferencia de muchos servidores de aplicaciones comerciales, las cuales son relativamente costosas.
- Zope por sí mismo es una plataforma inclusiva. Trae todos los componentes necesarios para comenzar a desarrollar una aplicación. No es necesario conseguir una licencia extra para complementar Zope (por ejemplo una base de datos relacional) como requisito para comenzar a desarrollar una aplicación. Esto hace que Zope sea muy fácil de instalar. Muchos de los otros servidores de aplicaciones tienen costos "ocultos" que requieren de una licencia costosa o de una configuración complicada de herramientas de terceros para iniciar a desarrollar una aplicación.
- Zope permite y apoya los desarrollos de terceros para ser distribuidos como aplicaciones listas para usar, al mismo tiempo, cuenta con una

gran variedad de servicios integrados y módulos disponibles para su uso inmediato. La mayoría de esos componentes, como Zope en si, son gratis y de código abierto. La popularidad de Zope es debida a una gran comunidad de desarrolladores. Muchos de los otros servidores de aplicaciones no cuentan con el apoyo de terceros o cuentan con muy pocos plugins.

- Las aplicaciones creadas en Zope pueden escalarse linealmente con el uso de Zope Enterprise Objects (ZEO) solución para cluster [ZEO]. Usando ZEO, puede servir una aplicación Zope basándose en múltiples computadoras sin necesidad de modificar significativamente (sí es necesario cambiar algo) el código aplicación. Muchos servidores de aplicaciones no poseen esta escalabilidad de una manera transparente.
- Zope permite a los desarrolladores crear aplicaciones web utilizando únicamente un navegador Web, puede ser Mozilla, Internet Explorer, Netscape, OmniWeb, Konqueror, Opera, son todos compatibles para mostrar y manejar el entorno de desarrollo de Zope (Zope Management Interface también conocido como ZMI). Zope también permite delegar funciones de desarrollo de la aplicación a otros desarrolladores a través de Internet de una manera muy segura usando la misma interfaz. Muy pocos de los demás servidores de aplicaciones proponen este nivel de funcionalidad.
- Zope provee un granular y extensible entorno de desarrollo. Puedes integrar fácilmente Zope con diversos sistemas de autenticación y autorización como: LDAP, WindowsNT y RADIUS simultáneamente, usando módulos pre-fabricados. Muchos de los otros servidores de aplicaciones solo ofrecen algunos de estos sistemas de autenticación y autorización.
- Zope permite que equipos de desarrolladores colaboren entre sí efectivamente. Entornos colaborativos requieren herramientas que permitan a los usuarios trabajar sin que las modificaciones que realicen

unos interfieran en el trabajo de los demás, por eso Zope cuenta con opciones para: deshacer, versiones, historial y otras herramientas que les permiten a los desarrolladores trabajar seguros y recuperarse de los errores. Muchos de los otros servidores de aplicaciones no ofrecen este tipo de características.

- Zope corre en las plataformas de sistemas operativos mas difundidas: Linux, Windows NT/2000/XP, Solaris, FreeBSD, NetBSD, OpenBSD, Mac OS X, Windows 98/Me (recomendado solo para propósitos de desarrollo). Muchos de los demás servidores de aplicaciones requieren un sistema operativo determinado, que ellos escogen según el tipo de licencia.

- Zope puede ser complementado usando el lenguaje interpretado de scripts Python, el mismo es popular, fácil de aprender y promueve un rápido desarrollo. Muchas librerías están disponibles para Python las cuales pueden ser usadas en el desarrollo de aplicaciones. Muchos de los demás servidores de aplicaciones deben ser extendidos usando lenguajes compilados como Java, lo cual limita la velocidad de desarrollo. Muchos de los demás servidores de aplicaciones usan lenguajes poco conocidos y que no tienen muchas librerías listas para su uso.

Se puede consultar los casos de estudio de Zope Corporation (versión en inglés) en línea, para ver ejemplos de aplicaciones que han sido creadas con este servidor de aplicaciones [ZopeCaseStudies].

Términos de uso, licenciamiento e introducción a la comunidad Zope

Zope es gratuito. Se puede utilizar Zope para crear y ejecutar aplicaciones web sin pagar licenciamiento o derechos de uso. Incluso se puede incluir Zope en los productos y aplicaciones desarrollados sin pagar algún tipo de derechos.

Zope es distribuido bajo la licencia de software libre, la Zope Public License o ZPL [ZPL]. Los términos de licenciamiento ZPL estipulan que estás en la capacidad de obtener y modificar el código fuente de Zope.

La ZPL es diferente de otras licencias populares de software libre, como la

Licencia Pública GNU [GPL]. Los términos de licenciamiento GPL especifican que si intentas redistribuir una aplicación licenciada bajo GPL y modificas o complementas la aplicación de alguna manera, entonces tienes que contribuir tus modificaciones al licenciante original. Esto no es requerido por las aplicaciones bajo ZPL. Puedes modificar y redistribuir Zope sin contribuir tus modificaciones de vuelta a Zope Corporation mientras cumplas con todos los términos de licenciamiento.

Debemos tener en cuenta que la ZPL ha sido certificada [CERTZPL] como compatible con Open Source Definition (OSD) [OpenSourceDefinition] por la iniciativa de código libre [OpenSource] y es catalogada como compatible con la licencia GPL [GPLCompatibles] por la fundación de software libre (Free Software Foundation FSF) .

La comunidad de desarrolladores es responsable del mantenimiento y prolongación de Zope. Muchos de los miembros de las comunidades son consultores profesionales, desarrolladores y webmasters los cuales desarrollan aplicaciones usando Zope en su propio beneficio, otros son estudiantes y aficionados al desarrollo de sitios Web. Zope Corporation es un miembro de esta comunidad y controla la distribución original de Zope y permite que los desarrolladores modifiquen libremente el contenido del código fuente.

La comunidad Zope brinda ocasionalmente conferencias pero gasta la mayoría de su tiempo discutiendo sobre Zope en las muchas listas de correo y sitios Web. Se puede obtener una gran cantidad de información acerca de Zope en las listas de correo de Zope [ZopeMailingLists].

Zope Corporation gana rentas con el uso de Zope para la creación de aplicaciones web para consumidores, por capacitaciones a los desarrolladores, por venta de soporte contratado con compañías, y por hacer hosting a páginas Web; esto significa que no se obtiene ninguna ganancia por la distribución del servidor de aplicaciones.

Conceptos y arquitectura de Zope

El framework Zope **[FW]** tiene diferentes conceptos fundamentales, los cuales se podrán comprender a medida que se vaya adquiriendo más experiencia.

Zope es un framework

Zope ayuda a los desarrolladores en la mayoría de los detalles concernientes al desarrollo de aplicaciones web como la persistencia de los datos, integridad de los datos y control de acceso, permitiendo un mejor enfoque en el problema a resolver. También les permite utilizar servicios para la construcción de aplicaciones web de una manera más rápida que otros lenguajes y entornos de trabajo, permite además, desarrollar la lógica del negocio por medio del lenguaje Python y también provee un soporte adicional para Perl, provee dos soluciones para la creación de plantillas, una basa en XML: Zope Page Template (ZPT) y la otra basada en HTML: Document Template Markup Language (DTML).

Orientación a objetos

En comparación con sistema de archivos como PHP, Zope es una plataforma de desarrollo web altamente orientada a objetos. La orientación a objetos es un concepto comúnmente utilizado en diferentes lenguajes de programación, incluyendo Python que es el lenguaje en el que se implementó Zope.

Publicación por objetos

La tecnología sobre la cual fue desarrollado Zope está basada en la misma que la utilizada en la web, la orientación por objetos. Una URL a un recurso Web en realidad es una ruta a un objeto que se encuentra dentro de un conjunto de contenedores y el protocolo HTTP permite un camino para el envío del mensaje al objeto y la recepción de la respuesta.

Zope posee una estructura de objetos jerárquica, la cual permite que un sitio típico este compuesto por objetos, los cuales pueden contener otros objetos (estos a su vez pueden contener otros objetos, y así sucesivamente). Las URLs

apuntan naturalmente a los objetos que se encuentran dentro de la jerarquía del entorno de Zope basado en sus nombres. Por ejemplo la URL "/Marketing/index.html" puede ser usada para acceder al objeto documento llamado "index.html" que se encuentra localizado en la carpeta "Marketing".

La finalidad de Zope es la publicación de los objetos. La forma en que este los publica es conceptualmente en linea recta.

- Su navegador envía una petición al servidor Zope. Esta petición especifica una URL de la forma "protocolo://host:puerto/ruta?queryString" ej: "http://www.zope.org:8080/Resources?batch_start=100".
- Zope separa la URL en sus componentes "host", "puerto", "ruta" y "queryString" (http://www.zope.org, 8080, /Resources y ? batch_start=100, respectivamente).
- Luego localiza los objetos en su base de datos orientada a objetos correspondiente a la ruta (/Resources).
- Ejecuta el objeto utilizando la lista de parámetros "queryString", la cual puede modificar el comportamiento del objeto. Esto permite que el objeto se comporte de diferentes formas dependiendo de los valores suministrados.
- Si al ejecutar un objeto es retornado un valor, este valor es enviado de vuelta al navegador. Comúnmente un objeto de Zope retorna HTML, datos de un archivo o datos de una imagen.
- Los datos son interpretados por el navegador y desplegados.

“Ejecutar” objetos a través de direcciones URL no es un idea nueva. Los servidores web como Apache y Microsoft IIS realizan lo mismo. Estos trasladan las URLs a archivos y directorios que se encuentran en un sistema de archivos. Zope realiza algo similar, mapea las URL a objetos que se encuentran en su base de datos.

Las URLs de los objetos están basadas en su ruta. Están compuestas por los

identificadores de la carpeta donde están contenidos y por el identificador del objeto, separado por el carácter slash('/'). Por ejemplo, si se tiene una carpeta en la raíz llamada Bob, su ruta es /Bob. Pero si Bob se encuentra dentro de otra carpeta llamada Uncles, entonces su URL va a ser /Uncles/Bob.

También podrá haber otras carpetas dentro de la carpeta Uncles, por ejemplo: si estuvieran las carpetas Rick, Danny y Louis, se podría acceder a estas de la siguiente forma:

- /Uncles/Rick
- /Uncles/Danny
- /Uncles/Louis

La URL de un objeto en su parte más simple se encuentra compuesta por su host, puerto y ruta por lo cual para ver el objeto con la ruta /Bob que se encuentra en el servidor Zope http://localhost:8080, la URL deber ser http://localhost:8080/Bob. Visitar una URL de un objeto de Zope directamente es definido como "llamado del objeto a través de la web", esto se debe a que la petición de evaluación de un objeto se realiza a través de una petición HTTP y el resultado de la evaluación es retornado al navegador web.

Para una explicación más detallada acerca de como Zope realiza la publicación de los objetos, se puede ver el capitulo [ObjectPublishing] del libro Zope Developer's Guide.

A través de la administración web

Para crear y trabajar con los objetos de Zope, se puede utilizar un navegador web para acceder a la interfaz de administración de Zope (Zope management interface). Toda la administración y el desarrollo de aplicaciones podrá realizarse completamente utilizando únicamente un navegador web. La interfaz de administración provee una vista parecida al Explorador de Windows, del sistema de objetos de Zope. A través de esta un desarrollador podrá crear y escribir objetos o definir nuevos tipos de objetos, sin necesidad de acceder al

sistema de archivos del servidor web.

Los objetos pueden colocarse en cualquier parte de la jerarquía de objetos. Los administradores pueden trabajar con esos objetos seleccionando las pestañas que representan diferentes tipos de "vista" de un objeto. Estas vistas varían dependiendo del tipo de objeto. Un objeto Zope llamado "Method DTML", por ejemplo, tiene una pestaña "Edit" la cual le permite editar la fuente del documento, mientras que un "Database Connection" provee algunas vistas que permiten la modificación de la cadena de conexión. Todos los objetos poseen una vista "Security" que permite manejar el control de acceso al objeto.

Seguridad y delegación de seguridad

Una de las cosas que diferencia a Zope de otros servidores de aplicaciones es que fue diseñado desde el comienzo para ajustarse no solo con los objetos web, sino también con el modelo de desarrollo web. Actualmente las mejores aplicaciones web requieren la participación de varias personas a través de una organización que tiene diferentes áreas de conocimiento. Zope está específicamente diseñado para acomodarse a este modelo, permitiendo a los administradores del sitio delegar de una forma segura el control a los expertos en diseño, en las bases de datos y en el administrador de contenidos.

Un buen web, generalmente, requiere la colaboración de muchas personas de la organización, desarrolladores de aplicaciones, expertos en SQL, administradores de contenidos y frecuentemente de los usuarios finales. En un sitio web convencional, el mantenimiento y la seguridad pueden rápidamente convertirse en un problema. ¿Cuánto control se le debe dar al administrador de contenidos?. ¿Como darle acceso al administrador de contenidos sin que afecte la seguridad?. ¿Qué acerca del código SQL embebido en un archivo ASP en el cual se esta trabajando (código que probablemente exponga el usuario de su base de datos)?

Los objetos en Zope proveen un conjunto de permisos mucho mas complejos que los sistemas convencionales basados en archivos. Estos permisos varían

con cada objeto, dependiendo de las capacidades del objeto. Esto permite la implementación bien lograda de un control de accesos. Por ejemplo, puede hacer un control de accesos en el cual los administradores de contenidos puedan utilizar objetos "SQL Method", pero que no puedan ver ni cambiar su contenido. También puede realizar restricciones en las cuales un usuario pueda crear cierta clase de objetos, por ejemplo "Folders" y "DTML Documents" pero no "SQL Methods" u otros objetos.

Zope provee la capacidad de administrar usuarios a través de la web vía "User Folders", las cuales son carpetas especiales que contiene la información de los usuarios. Muchos complementos de Zope están disponibles para extender los tipos de "User Folders" que contienen la información de los usuarios de fuentes externas como bases de datos relacionales o directorios LDAP. La capacidad de agregar "User Folders" puede ser delegada a usuarios dentro de una subcarpeta, esencialmente permitiendo delegar la creación y la administración de subsecciones de un sitio web a usuarios poco confiables sin preocuparse de que puedan cambiar los objetos que no estén dentro de su carpeta.

Persistencia de objetos nativos y transacciones

Los objetos de Zope se encuentran almacenados en una base de datos orientada a objetos de alto rendimiento conocida como Zope Object Database (ZODB). Cada petición web es tratada como una transacción aparte por la base de datos. Si algún error ocurre en la aplicación durante la petición, cualquier cambio realizado durante la misma sera automáticamente deshecho. La base de datos permite deshacer en varios niveles, permitiendo al administrador deshacer cambios con solo realizar clic sobre el botón correspondiente. El entorno de trabajo de Zope permite que todo lo concerniente a la persistencia y transacciones sea completamente transparente para el desarrollador.

Adquisición

Uno de los aspectos mas poderosos es la adquisición, el cual se especifica a

través de los siguientes enunciados:

- Los objetos están contenidos dentro de otros objetos (como lo son las carpetas).
- Los objetos pueden adquirir los atributos y el comportamiento de sus contenedores.

El concepto de adquisición trabaja con todos los objetos de Zope, y provee una potente vía para centralizar los recursos comunes. Una consulta SQL común o un modelo HTML, por ejemplo, puede ser definida en una carpeta y los objetos que se encuentran dentro de una subcarpeta pueden utilizarlos automáticamente a través de la adquisición. Si la consulta necesita ser cambiada, únicamente es necesario realizar los cambios en un solo lugar sin necesidad de tener que preocuparse por los objetos que la utilizan.

Como los objetos en un comienzo adquieren lo que se encuentra en su mismo nivel de la jerarquía de contenido y realizan una adquisición de sus contenedores superiores, es fácil especializar áreas del sitio con un mínimo esfuerzo. Por ejemplo, si se tiene un objeto “folder” llamado "Deportes" en el cual se encuentran contenidos temas relacionados con deportes, se tendrá que crear un nuevo encabezado (header) y pie de página (footer) en la carpeta de deportes en la cual se encuentra un tema para las noticias relacionadas con deportes. El contenido de la carpeta deportes y sus subcarpetas utilizaran el encabezado y pie de página específicos para deportes en vez de los que se encuentran en la carpeta de nivel superior del sitio.

Zope es extensible

Zope es altamente extensible y usuarios avanzados pueden crear nuevos tipos de objetos Zope, cualquiera puede escribir nuevos complementos para Zope en Python o construyéndolos completamente por medio de la web. Zope provee una serie de componentes preconstruidos para ayudar a los desarrolladores, incluyendo una robusta colección de clases que toman en cuenta los detalles necesarios para implementar nuevos objetos Zope.

Una cantidad de productos complementarios que están disponibles proveen cualidades como arrastrar tópicos de discusión, datos de publicación de escritorio, herramientas XML y de implementación de comercio electrónico. Muchos de estos productos han sido escritos por los miembros altamente activos de la comunidad Zope y la mayoría son también de código libre.

Componentes fundamentales de Zope

Zope se basa en diferentes componentes que colaboran para facilitar la construcción de aplicaciones web. Los componentes fundamentales de Zope son mostrados en la siguiente figura, y explicados posteriormente.

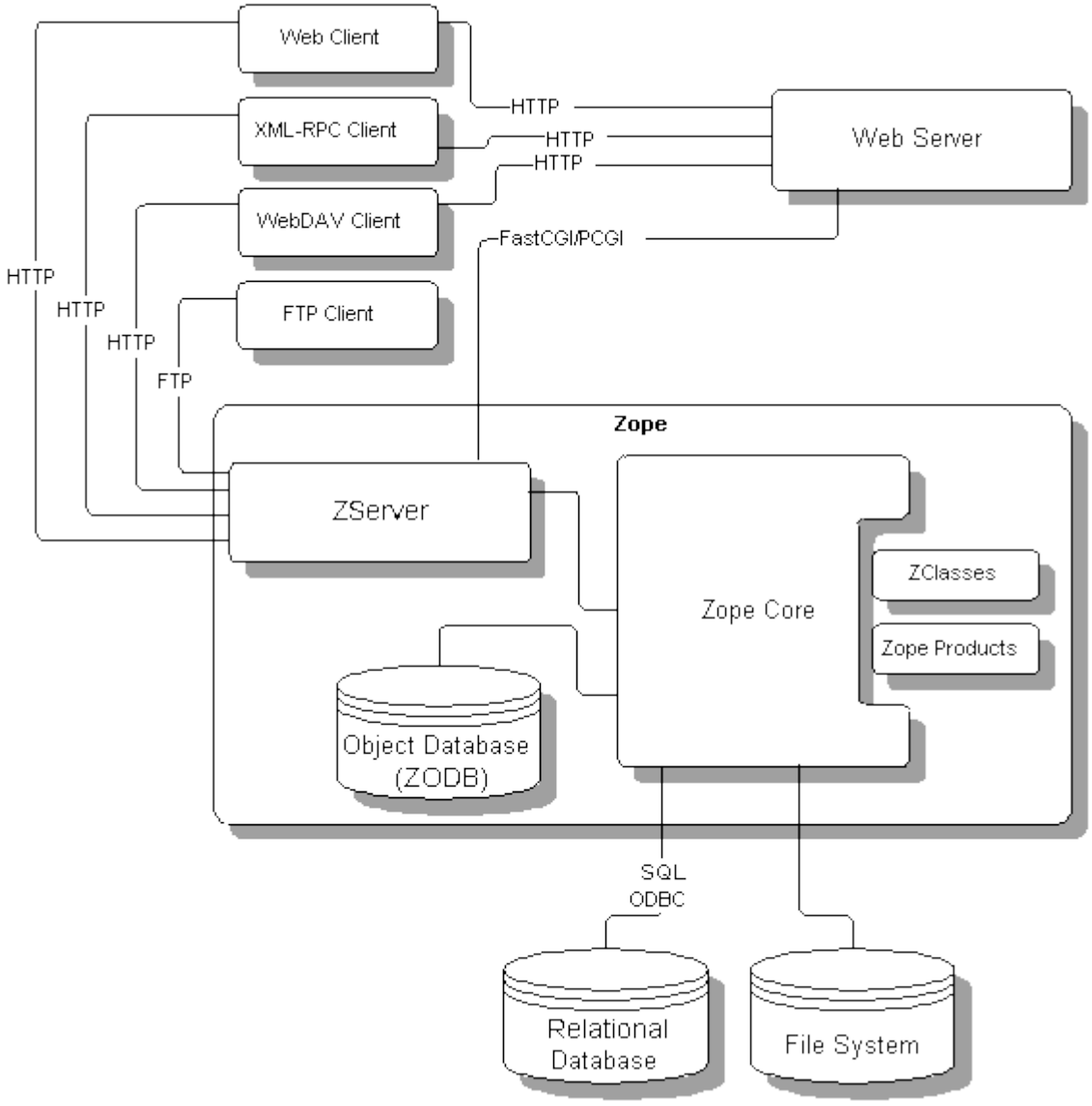

Ilustración 6: Arquitectura de Zope

- ZServer: Zope trae consigo un servidor web que sirve contenido a los usuarios. Este servidor web también sirve contenido Zope vía FTP, WebDAV, y XML-RPC (un procedimiento remoto de fácil llamada).
- Servidor Web: En caso de que se tenga instalado y debidamente configurado un servidor web, como Apache o Microsoft IIS, se puede integrar Zope a los mismos. Además, Zope se puede integrar con cualquier otro servidor que soporte Common Gateway Interface (CGI).
- Núcleo de Zope: Es el motor que coordina la función, conduciendo la

interfaz de configuración y la base de datos de objetos.

- Base de Datos de Objetos: Cuando se utiliza Zope, generalmente se trabaja con objetos que son guardados en la Base de Datos de Objetos de Zope.
- Base de Datos Relacional: No es imprescindible almacenar la información utilizada en una aplicación en la base de datos de Zope. Zope trabaja con otras bases de datos relacionales como Oracle, PostgreSQL, Sybase, MySQL entre otras.
- Sistema de archivos: Zope puede trabajar con documentos y otros archivos almacenados en su sistema de archivos.
- ZClasses: Zope permite a los administradores añadir nuevos tipos de objetos a Zope usando la interfaz de administración. ZClasses son esos tipos de objetos.
- Productos: Zope también permite a los administradores añadir nuevos tipos de objetos.

Resumen

Según lo analizado en este capítulo, teniendo en cuenta además, que el Lenguaje Java tiene altos requerimientos de hardware, y el lenguaje PHP no es recomendado para aplicaciones grandes[10], se seleccionó como lenguaje de desarrollo Python, y como plataforma de desarrollo web Zope, dada las facilidades de desarrollo y características explicadas anteriormente.

10 *Sólo comienza a evaluarse la posibilidad de uso para desarrollos grandes a partir de la versión 5, por lo que no podemos decir que es utilizado para este tipo de aplicaciones.*

CAPÍTULO

3. DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

Introducción

En el presente capítulo se describen los requisitos funcionales y no funcionales que debe tener el sistema que se propone, lo que nos permite comprender el sistema de forma general, e identificar mediante un Diagrama de Caso de Uso, las relaciones de los actores que interactúan con el sistema, y las secuencias de acciones con las que interactúan.

Además se mostrarán algunos diagramas de secuencia que servirán de apoyo para la comprensión del funcionamiento del sistema.

En el desarrollo de este sistema se utilizó como metodología de desarrollo Rational Unified Process (RUP) [Jacobson2000], por lo que los artefactos descritos en este capítulo, están definidos como parte de la metodología RUP; y como lenguaje para describir los artefactos el Lenguaje de Modelado Unificado (UML) [Rumbaugh1998].

3.1 Descripción del Modelo de Dominio

Los usuarios que interactúan son el administrador del sistema y un usuario

estándar, estos acceden según su rol en el sistema a los diferentes mapas que estén disponibles en el mismo.

Un mapa está compuesto por varias capas y cada una de estas por un conjunto de elementos de una misma geometría, es decir, un mapa lo componen varias figuras diferentes como puntos, líneas y polígonos, agrupadas por capas independientes que están compuestas por elementos de un mismo tipo.

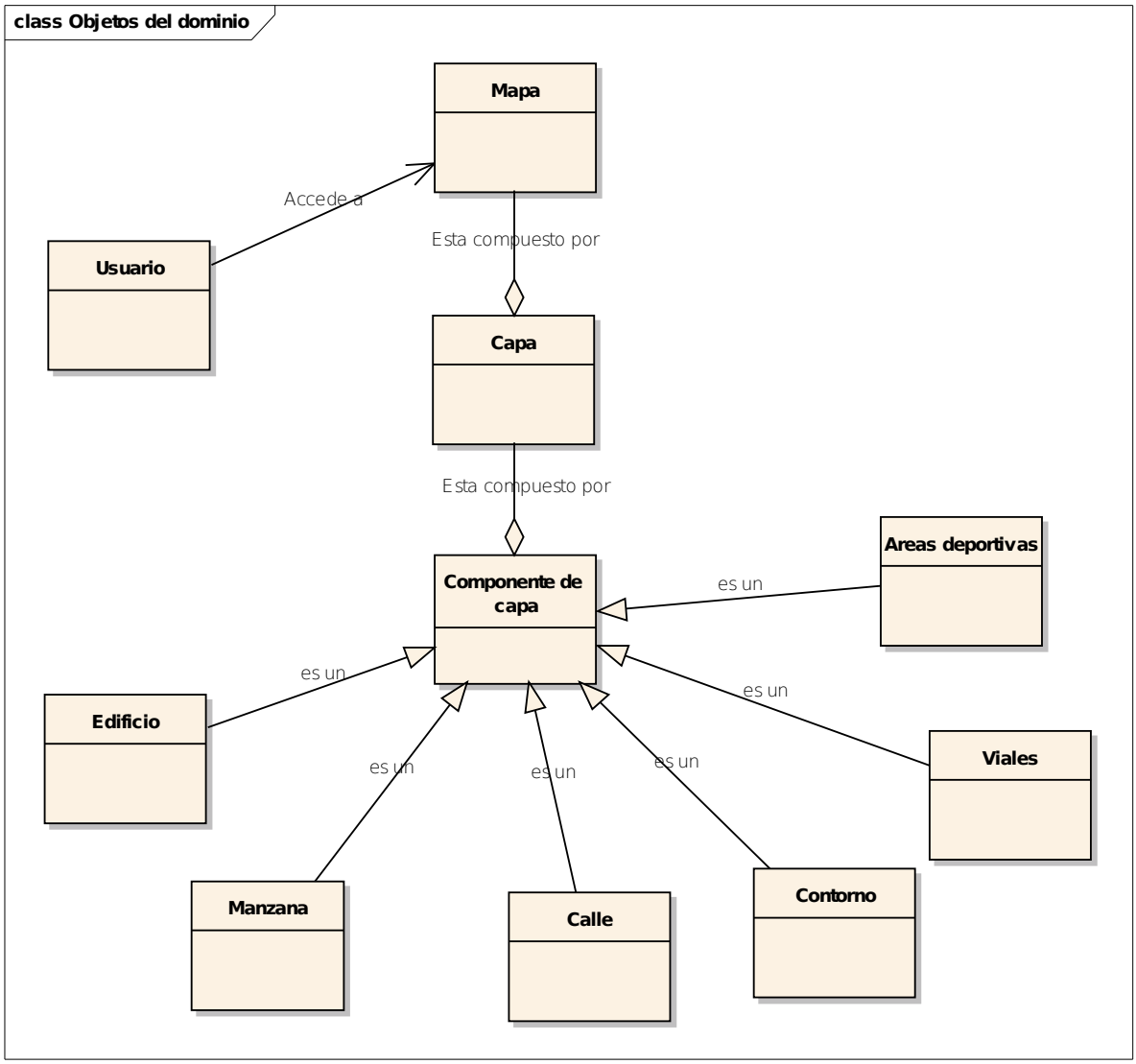

Ilustración 7: Objetos del dominio

3.2 Especificación de Requisitos.

A continuación se enumeran los requisitos funcionales y no funcionales del sistema, para una mejor comprensión del mismo. Además, se muestra un análisis de los principales casos de uso del sistema.

3.2.1 Requerimientos funcionales

R1: Mostrar mapa

- Mostrar un Mapa

R2: Acercar mapa

1. Acercar mapa.

R3: Alejar mapa

- Alejar mapa.

R4: Centrar un área del mapa

1. Centrar un área del mapa.

R5: Mover una región del mapa

1. Mover una región del mapa.
   1.1 Mediante el Mouse.

R6: Mostrar mapa inicial

- Mostrar el mapa inicial.

R7: Obtener información

1. Obtener información del(de los) componente(s) de capa relacionado con el punto donde se realice un clic.

R8: Mostrar coordenadas geográficas

1. Obtener coordenadas mediante el movimiento del Mouse sobre el mapa.

R9: Mostrar Leyenda

1. Mostrar leyenda.

R10: Localizar Entidad

1. Efectuar una búsqueda de un elemento en el mapa, los mismos pueden ser:

- Un edificio.

R11: Buscar camino

1. Busca el camino mas corto entre dos lugares existentes en el mapa, mostrados en un Combobox.

3.2.2 Requerimientos no funcionales

Interfaz externa

1. Diseño sencillo con mayor uso del mouse y con pocas entradas por teclado, interfaz web amigable y que permita a usuarios no familiarizados un uso extensivo de la aplicación.

Portabilidad

2. Sistema multiplataforma.

Hardware

3. Requerimientos mínimos de hardware, puede estar instalada en una PC Pentium, 128 Mb de RAM y una tarjeta de red de 50 - 100Mbps, es importante destacar que los accesos al sistema dependen del buen funcionamiento de la red.

Software

4. Como el sistema es multiplataforma se puede tener instalado en el servidor cualquier sistema operativo, los lenguajes de programación serán Python y JavaScript, como servidor web Apache2.x y como servidor de aplicaciones Zope3.
5. Cualquier navegador, Firefox recomendado.

Seguridad

6. El usuario anónimo sólo puede realizar las acciones básicas.
7. Permitir autenticar un usuario.
8. Garantizar que las funcionalidades del sistema se muestren de acuerdo al nivel de usuario que esté activo.

Funcionalidad

9. Mínima cantidad de páginas para ejecutar todas las funciones posibles.

3.3 Descripción del Sistema propuesto

Para cumplir con los objetivos del presente trabajo y teniendo en cuenta cada uno de los requerimientos antes planteados se ha decidido que el sistema a desarrollar debe tener tres módulos: el módulo de administración del sistema, el módulo para la visualización de mapas y el módulo que obtiene una representación del mapa en un grafo, y hace búsquedas de camino más corto, en adelante módulo Map2Graph.

También se han definido dos roles principales para el uso del sistema y sus funcionalidades, ellos son: usuario, que puede solamente visualizar los mapas y realizar operaciones básicas sobre dicho mapa y administrador del sistema que básicamente puede adicionar mapas al SIG o cambiar la configuración del mismo.

Destacamos que hemos dividido el trabajo en dos ciclos uno para crear el módulo de visualización de mapas y otro para la creación del módulo Map2Graph, no se dedica un ciclo para el módulo de administración, ya que las funcionalidades implementadas inicialmente son de acceso público y el servidor de aplicaciones brinda una serie de funcionalidades sobre el control de acceso a la aplicación para el caso de la modificación de los mapas.

3.4 Modelo de casos de uso del sistema

En este epígrafe enumeraremos los actores del sistema así como daremos una breve descripción de sus principales casos de uso.

3.4.1 Definición de los actores del sistema

Actores	Descripción
Administrador	Tiene los privilegios necesarios para utilizar todas las funcionalidades que brinda el sistema. Puede crear nuevos roles, modificar mapas, además de ejecutar las funcionalidades básicas sobre dichos mapas y las búsquedas.
Usuario	Es la persona que utiliza el sistema para visualizar los mapas y realizar las funciones básicas sobre este.

Tabla 2: Actores del sistema

3.4.2 Diagrama de casos de uso. Especificación.

A continuación se presentan los casos de uso determinados para satisfacer los requerimientos funcionales de sistema:

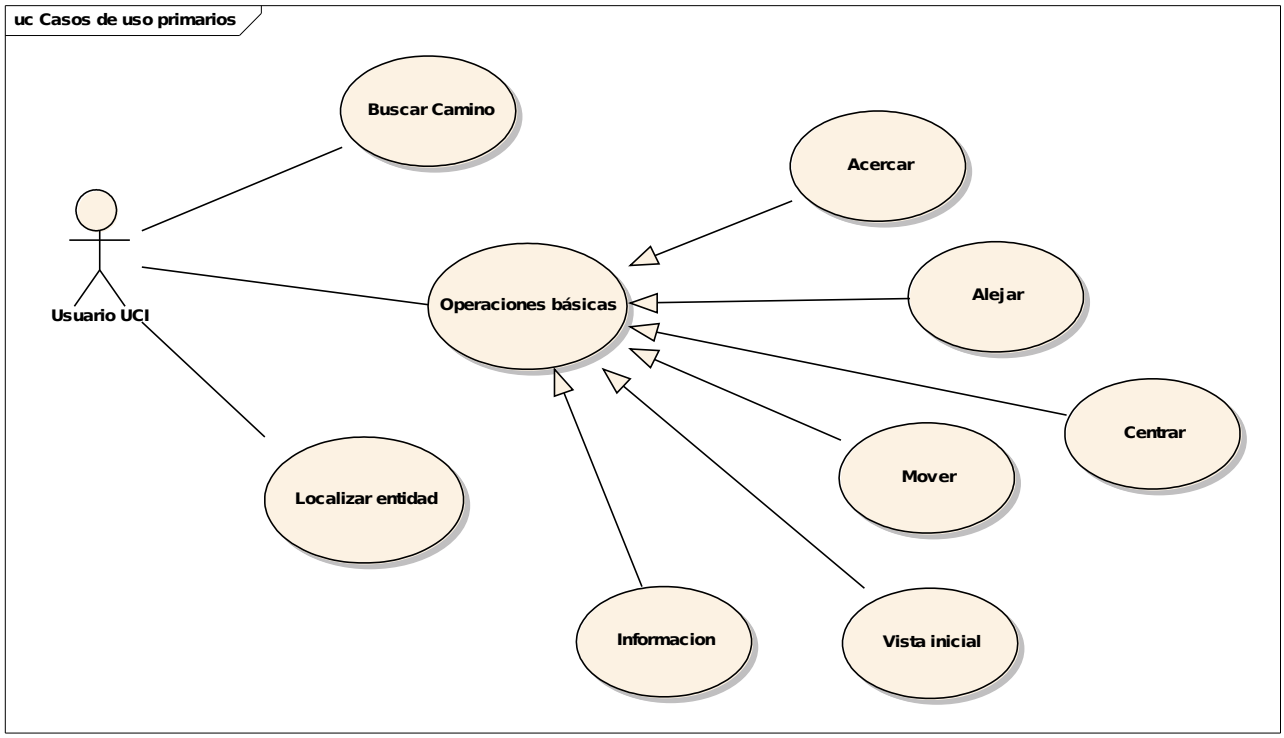

Ilustración 8: Diagrama de casos de uso

CU -1	Realizar Operaciones Básicas
Propósito	Realizar operaciones básicas que permitan al usuario trabajar con el mapa que se muestra.
Actores:	Administrador, Usuario
Resumen:	El CU se inicia cuando el usuario desea hacer alguna operación sobre el mapa que se muestra. El sistema identifica cual operación va a realizar el usuario sobre el mapa y da respuesta de esta.
Referencias:	R2, R3, R4, R5, R6, R7, R8, R9, R10, R11

Tabla 3: Especificación del caso de uso Realizar Operaciones Básicas

CU - 2	Localizar Entidad
Propósito:	Localizar cualquier entidad de la capa edificios que exista dentro del mapa.
Actores:	Administrador, Usuario.
Resumen:	El CU se inicia cuando el usuario elige la opción Buscar. El sistema automáticamente llena los campos de parámetro de la búsqueda y el nombre del mapa y realiza la búsqueda.
Referencias:	R1

Tabla 4: Especificación del caso de uso Localizar Entidad

CU - 3	Buscar camino
Propósito:	Localizar cualquier entidad de la capa edificios que exista dentro del mapa.
Actores:	Administrador, Usuario.
Resumen:	El CU se inicia cuando el usuario elige la opción Buscar. El sistema automáticamente llena los campos de parámetro de la búsqueda y el nombre del mapa y realiza la búsqueda.
Referencias:	R11

Tabla 5: Especificación del caso de uso Buscar Camino

3.4.3 Expansión de los casos de uso

Caso de Uso Localizar Entidad.		
Propósito:	Localizar cualquier entidad de la capa edificios que exista dentro del mapa.	
Actores:	Administrador, Usuario.	
Resumen:	El CU se inicia cuando el usuario elige la opción Buscar. El sistema automáticamente llena los campos de parámetro de búsqueda y el nombre del mapa y realiza la búsqueda.	
Referencias:	R1	
Acción del actor:		**Respuesta del sistema:**
1 - El usuario selecciona el nombre del edificio que desea buscar en el Combobox NoEdificio.		3.1 - El sistema muestra el nombre del parámetro escogido.
4 - El usuario desplaza el Mouse sobre el botón Buscar.		4.1 - El sistema busca según el nombre seleccionado el objeto geográfico que se corresponda.
		4.2 - El sistema muestra de colo rojo el lugar en el mapa donde se encuentra el edificio buscado.

Tabla 6: Expansión del caso de uso Localizar Entidad

El caso de uso Realizar Operaciones Básicas, está dividido en varias secciones, como se muestra en la tabla 7.

Caso de Uso Realizar Operaciones Básicas	
Propósito	Realizar operaciones básicas que permitan al usuario trabajar con el mapa que se muestra.
Actores:	Administrador, Usuario
Resumen:	El CU se inicia cuando el usuario desea hacer alguna operación sobre el mapa que se muestra. El sistema identifica cual operación va a realizar el usuario sobre el mapa y da respuesta de esta.
Referencias:	R2, R3, R4, R5, R6, R7, R8, R9, R10, R11

Acción del actor:	**Respuesta del sistema:**
1 - Realiza una acción básica sobre el mapa.	1.1 - El sistema ejecuta alguna de las siguientes acciones: • Acercar por Rectángulo (Ir a Sección Acercar por Rectángulo). • Alejar Mapa (Ir a Sección Alejar Mapa). 4. Centrar un área del mapa (Ir a Sección Centrar). • Obtener Información (Ir a Sección Obtener Información) • Mover región del mapa (Ir a Sección Mover Mapa) • Mostrar el mapa inicial. (Ir a Sección Mapa Inicial) • Obtener coordenadas en Movimiento.(Ir a Sección Coordenadas en Movimiento) •
Sección Acercar por rectángulo	
1 - El usuario selecciona el botón Acercar por rectángulo.	

2 - El usuario da clic sobre las coordenadas iniciales de uno de los puntos del rectángulo y luego mueve el mouse hasta las coordenadas finales que desea.	3.1 - Muestra con color rojo el rectángulo que se está seleccionando. 3.2 - Amplia el mapa en la pantalla usando las coordenadas determinadas por el rectángulo rojo.
Sección Alejar Mapa	
2 - El usuario selecciona la opción alejar y presiona el botón izquierdo del mouse en la posición que desea alejar.	2.1 - Determina coordenadas del clic. 2.2 - Reduce el mapa en la pantalla usando las coordenadas.
Sección Centrar	
2 - El usuario selecciona la opción centrar y presiona el botón izquierdo del mouse	3.1 - Centra la vista del mapa que se está mostrando.
Sección Obtener Información	
2 - El usuario selecciona el botón Obtener Información.	
3 - El usuario presiona el botón izquierdo del mouse sobre la región del mapa que desea identificar.	3.1 – El sistema muestra una ventana con la información del objeto que contiene la coordenada del clic.
Flujo Alternativo	
Acción 3.1	El sistema muestra una ventana informando al usuario que en esas coordenadas no hay capas que brinden información
Sección Mover Mapa	
2 - El usuario selecciona el botón Mover Mapa.	
3 - El usuario hace clic con el botón izquierdo del mouse sobre una zona del mapa y comienza a mover el mouse en	3.1 El sistema va a mover el mapa en la misma dirección que vaya el mouse hasta que el usuario haga clic nuevamente.

cualquier dirección que desee.	
Sección Mostrar el mapa inicial.	
2 - El usuario presiona el botón izquierdo del mouse sobre el botón Mapa Inicial.	2.1 – El mapa vuelve a la vista inicial, o sea, la que apareció por primera vez.
Sección Coordenadas en Movimiento	
3 – El usuario desplaza el Mouse sobre el área donde esta el mapa.	3.1- El sistema muestra las coordenadas por las que se mueve el mouse en la barra de estado.

Tabla 7: Expansión de caso de uso Realizar Operaciones Básicas

Caso de Uso Buscar Camino.	
Propósito:	Buscar el camino más corto entre dos localizaciones del mapa.
Actores:	Administrador, Usuario.
Resumen:	El CU se inicia cuando el usuario elige la opción Buscar Camino. El sistema automáticamente llena los campos de parámetro de búsqueda y el nombre del mapa y realiza la búsqueda.
Referencias:	R11
Acción del actor:	**Respuesta del sistema:**
1 - El usuario presiona el botón izquierdo del Mouse sobre el Combobox Desde.	1.2 – El sistema despliega el combobox Desde.
2 - El usuario selecciona el origen del camino a buscar.	2.1 - El sistema muestra el nombre del origen escogido en el combobox Desde.
3 - El usuario presiona el botón izquierdo del Mouse sobre el Combobox Hasta.	3.1 - El sistema despliega el combobox Desde.
4 - El usuario selecciona el destino del camino a buscar.	4.1 - El sistema muestra el nombre del destino escogido en el Combobox Hasta.
4 - El usuario presiona el botón izquierdo del Mouse sobre el botón Buscar Camino.	4.1 - El sistema busca el camino según los parámetros seleccionados (origen-destino).
	4.3 - El sistema muestra con una línea de un color aleatorio el camino mas corto “desde” origen a “destino”.

Tabla 8: Expansión del caso de uso Buscar Camino

3.5 Modelo de Diseño.

En la fase de diseño se modela el sistema de manera que soporte todos los requisitos, tanto funcionales como no funcionales. La esencia de esta fase es la elaboración de diagramas de interacción, que muestran gráficamente como los objetos se comunican entre ellos a fin de cumplir con los requerimientos. Estos diagramas permiten la realización de los diagramas de clases del diseño, los cuales resumen la definición de las clases que serán implementadas.

3.5.1 Diagramas de Clases del Diseño

Como ya explicamos anteriormente la aplicación fue diseñada usando una arquitectura de tres capas. Teniendo en cuenta dicha arquitectura y atendiendo a las funcionalidades del sistema se definieron dos paquetes principales: Map2Graph, para la obtención de una representación del mapa en una estructura de datos; y mapObj, que se encarga de representar el mapa gráficamente al usuario del sistema, y de las operaciones que se pueden realizar sobre el mismo

Dentro del paquete Map2Graph, se encuentra un paquete de Acceso a Datos que contiene las clases para todo el trabajo con los datos que usa la aplicación (el mapa en formato shape) y que le permite a esta abstraerse del origen de los datos.

Recordemos que los diagramas de clases del diseño resumen la definición de las clases que se van a implementar en el software. Ver Anexo I.

3.5.2 Diagramas de interacción

Los diagramas de interacción se dividen en dos tipos de diagramas de UML, los diagramas de secuencia y los diagramas de colaboración. Para modelar los aspectos dinámicos de este sistema se utilizaron diagramas de secuencia por cada caso de uso. Ver Anexo II.

3.6 Principios de diseño.

El diseño de la interfaz de la aplicación es un elemento muy importante dado que este tiene que basarse en las necesidades del usuario que puede o no estar familiarizado con la informática, por lo tanto debe ser lo más amigable y comprensible posible.

Para esto nos hemos propuesto las siguientes metas.

- Que para cada usuario solo sean mostradas y accesibles las opciones que le brinda su rol.
- Que cualquier persona que acceda a la aplicación requiera mínimo dominio de la informática para su utilización.

3.6.1 Interfaz de la aplicación.

La página principal de la aplicación, se concibe como un portal, donde podemos ver divididas las funcionalidades del sistema como sigue:

- Mapa: Muestra al usuario el mapa resultante de las operaciones especificadas.
- Herramientas: Para seleccionar la operación que se desee realizar sobre el mapa.
- Leyenda: Muestra la leyenda del mapa.
- Búsqueda: Muestra en el mapa el objeto que se escoja.
- Buscar camino: Muestra en el mapa el camino mas corto desde un origen a un destino.

Estas funcionalidades, se encuentran claramente dividas, como se muestra en la ilustración 9.

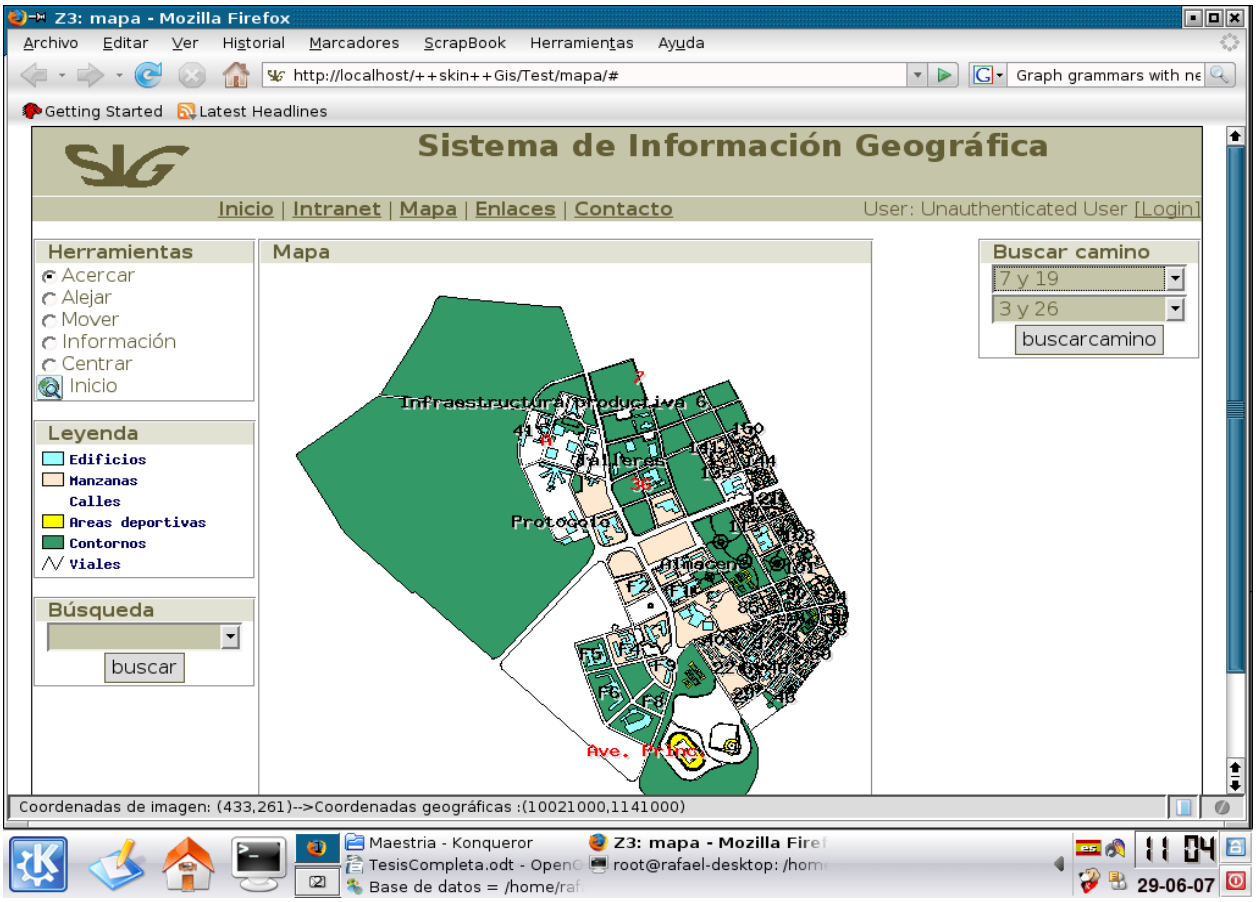

Ilustración 9: Vista principal del sistema

3.7 Modelo de despliegue.

El modelo de despliegue describe la distribución física del sistema, muestra como están distribuidos los componentes de software entre los distintos nodos de cómputo. Permite comprender la correspondencia entre la arquitectura software y la arquitectura hardware. Ver Anexo III.

3.8 Módulo map2Graph

Este módulo es el encargado de crear una estructura de datos que represente a un mapa dado, para lo que se necesita únicamente un mapa en formato shape, es imprescindible que el mapa cuente con una capa que represente las calles que existan en la zona geográfica en cuestión, y que

dicha capa esté construida correctamente; debemos enfatizar, que el objetivo de este módulo, es ejecutarlo una sola vez para cada mapa, al instalar el sistema o al añadir un nuevo mapa al mismo[11]; en esa primera ejecución, se obtiene un grafo representativo del mapa, siguiendo los pasos que se muestran a continuación:

1. Se obtiene un un diccionario, donde cada elemento es una lista -una por cada calle-, construida de la siguiente forma:
   [ShapeObj, cant, *ShapeObjL]
   donde:
   - ShapeObj, representa una calle.
   - cant, es la cantidad de calles que se cruzan con la calle ShapeObj.
   - *ShapeObjL es la lista de calles que se cruzan con la calle ShapeObj.
2. Luego se calcula la distancia correspondiente desde el comienzo de cada una de las calles, hasta cada uno de los puntos donde se cruza con otra calle.
3. Se ordenan los elementos de la lista según la distancia calculada, para obtener el orden de los intersectos sobre la calle, con esto se logra conocer cuales intersectos son adyacentes, esto es imprescindible para crear el grafo.
4. Se recorre cada lista de elementos, creando un nodo para cada elemento, y una arista para cada par de elementos consecutivos, con atributo nombre -nombre de la calle- y peso -distancia entre los dos elementos, calculada en el paso 2-.
5. Se adicionan los nodos y las aristas a un grafo vacío
6. Se retorna el grafo.

El diagrama de clases de este módulo podemos verlo en el Anexo I. Básicamente se tiene una clase Mapa, de la que heredan dos clases

11 *En caso de que se modifique el mapa luego de haber generado su representación, habría que utilizar nuevamente la funcionalidad que brinda el módulo map2Graph.*

representativas de dos tipos de mapas, Map_SHP y Map_PGSQL_POSTGIS, en este trabajo sólo se implementó la clase Map_SHP, dado que es el formato de mapa que más abunda -soportado por ESRI, compañía líder en sistemas de información geográfica-, para añadirle soporte al módulo para cualquier otro tipo de mapa, según el diseño del mismo, implicaría cambios de código mínimos, como se puede apreciar en el diagrama de clases antes mencionado.

Este módulo tiene una gran importancia y le brinda un valor añadido a cualquier sistema de información geográfica que lo utilice, ya que la verdadera utilidad de los SIG está donde es prácticamente imposible el manejo de los datos por las personas, y sería muy engorroso o prácticamente interminable, para una persona crear un grafo que represente un mapa sin la utilización de un medio informático.

Se le dedica una especial atención a este tipo de representación, ya que es la base para hacer búsquedas del camino más corto entre dos lugares, para resolver el conocido problema de transporte (Búsqueda de flujo máximo de costo mínimo para una red de distribución), para problemas de tipo ”el problema del viajante”, entre otros. A estos problemas nos enfrentamos diariamente en la vida cotidiana, la mayoría de las personas, y las empresas(nacionales e internacionales), por lo que este módulo representa la base para darle una solución automatizada a los problemas anteriores.

Es necesario mencionar, que en estos momentos, el país no cuenta con una herramienta de este tipo.

CAPÍTULO

4. ALGORITMO DE REDUCCIÓN DE GRAFOS. CAMINO MÍNIMO

Introducción

La mayor utilidad de un Sistema de Información Geográfica esta estrechamente relacionada con la capacidad que posee éste de construir modelos o representaciones del mundo real a partir de las bases de datos digitales, esto se logra aplicando una serie de procedimientos específicos que generan más información para el análisis.

Una de las aplicaciones de un SIG, es la búsqueda de rutas mínimas en mapas, o sea, un SIG puede responder a la pregunta: ¿Cómo llego desde el lugar X al lugar Y, de la forma más rápida posible?, esta pregunta es respondida por un SIG, en forma de imagen, o sea, se muestra la ruta para llegar desde el lugar X al Y en un mapa.

Además, en muchas empresas, se necesitan los SIG para planificar el transporte de sus productos, por ejemplo, el Instituto Postal Telegráfico de Venezuela (IPOSTEL), necesita un SIG que calcule y le muestre como debe distribuir la flota de vehículos para el transporte de bultos postales, lo cual, va a disminuir el tiempo que un paquete demora en llegar al destinatario.

Hasta el momento, este tipo de problemas está resuelto, pero sólo parcialmente, porque en la mayoría de los sistemas desarrollados, en el caso de las rutas mínimas, sólo se tiene en cuenta los puntos principales de una ciudad, o las ciudades principales de un país, y por tanto, se obvia la mayoría de los objetos (que pueden ser casas, calles, hospitales, escuelas) del mapa, lo cual se traduce en restricciones sobre los usuarios del sistema de información geográfica.

En el primer ejemplo (transporte de bultos postales por IPOSTEL), se planifica cómo se utiliza la flota de vehículos, los mismos distribuyen los bultos postales por todo el país, pero llegan solamente hasta las oficinas postales, que son pocas, por lo que no se tiene en cuenta la ruta que debe seguir un cartero para ir desde la oficina postal hasta la casa de los destinatarios, para entregarle su correspondencia o bulto postal. Teniendo en cuenta esto, podemos apreciar que el proceso se puede automatizar y optimizar más.

Estas restricciones están dadas en gran medida, debido a que para resolver estos problemas, debemos transformar el mapa en un grafo, para después, hacer los análisis sobre rutas mínimas y/o problemas de transporte, entre otros. El problema realmente es la dimensión que alcanzaría este grafo, ya que en un mapa nos podemos encontrar con varios millones de objetos, por ejemplo, un mapa de Cuba, que es un país pequeño, que esté a nivel de casas, o sea, que en el mapa, haya un objeto geográfico por cada una de las casas que hay en el país, cuando buscamos el grafo que representa este mapa, nos sería prácticamente imposible cargarlo en memoria.

El problema que se trata de resolver, es el de representar un mapa completo (todos los objetos necesarios para hacer análisis de redes) a través de un grafo, ya que en la actualidad, no se cuenta, en el país, con un SIG basado en web que tenga esta capacidad, en sentido general (nacional e internacionalmente), para hacer esto, se crea un grafo en el cual quedan representados los puntos principales del mapa, que pueden ser las ciudades más importantes, los aeropuertos, etc., esta solución se ha dado generalmente,

porque un mapa puede llegar a tener decenas de millones de objetos, luego, si se tiene un grafo que tenga decenas de millones de vértices, sería prácticamente imposible cargarlo en memoria de una sola vez, y si le aplicamos algún algoritmo, por ejemplo, de búsqueda de camino mínimo, el tiempo de corrida del mismo sería muy elevado.

Por lo cual, se presenta en este capítulo, una propuesta de algoritmo para reducir un grafo, y a su vez, un algoritmo para hacer búsquedas de camino mínimo en el grafo reducido; para posteriormente, aplicarlos a los Sistemas de Información Geográfica, ya que para analizar y manipular los datos de un mapa, generalmente se hace a través de un grafo, en el cual, los vértices representan los objetos del mapa, y las aristas las relaciones entre estos.

Con esto lograremos una mayor calidad en los servicios que brinda un SIG, ya que no sólo se utilizaría en la búsqueda de caminos mínimos, sino, en cualquier operación que traiga consigo un procesamiento del mapa a nivel de todos los objetos que lo componen.

Es necesario mencionar que se llevó a cabo esta tarea, empleando conceptos de la teoría de grafos, teoría de redes, gramáticas de grafo y la reescritura de grafos, para finalmente obtener dos algoritmos que nos permiten reducir sustancialmente un grafo y además hacer búsquedas de caminos mínimos.

El resultado de este capítulo está listo para ser aplicado en cualquier Sistema de Información Geográfica.

4.1 Conceptos

En esta sección examinaremos algunos conceptos referidos a conjuntos de vértices de un grafo que poseen determinadas propiedades. Mediante la búsqueda de dichos conjuntos pueden ser resueltos ciertos tipos de problemas de la vida práctica.

4.1.1 Reducción de un grafo

Un método utilizado con el fin de estudiar de una forma más sencilla las propiedades de un grafo G, consiste en construir, a partir de éste, un grafo más simple denotado G_r, descomponiendo G en subgrafos y considerando éstos como vértices de G_r. Estos vértices se unirán por arcos deducidos de G de acuerdo con ciertas reglas.

Consideremos el grafo G=(V,A), y sea p una propiedad mediante la cual se define una partición P en V, lo cual permite descomponer G en subgrafos de acuerdo con dicha partición. Sean $A_1,...,A_s$, las clases de la partición P.

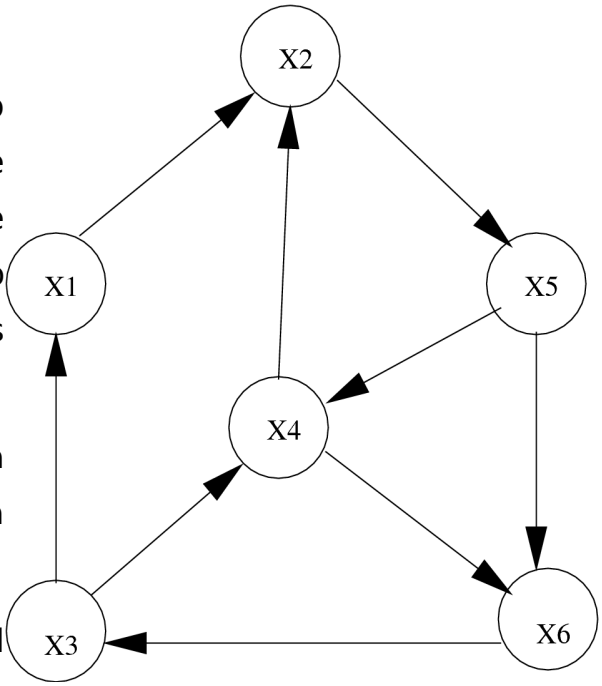

Ilustración 10: Grafo a reducir

El grafo reducido $G_r=(C,R)$ según la propiedad p se construye de la manera siguiente:

- Asignamos a cada conjunto A_i, el vértice $C_i \in C$. De esta forma G_r tendrá S vértices.
- $C_i,C_j \in R$ si y solamente si $(x,y) \in A$ con $x \in A_i$ e $y \in A_j$.

Ejemplo: Consideremos el grafo de la ilustración 10:

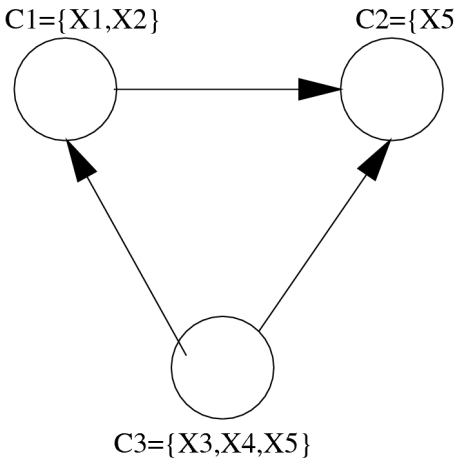

Ilustración 11: Grafo reducido

Si consideramos la propiedad p1, la cual determina la partición P en la que:

$A_1=\{ x_1,x_2\}$

$A_2=\{ x_5\}$

$A_3=\{ x_3,x_4,x6\}$

Obtendremos el grafo reducido que se muestra en la ilustración 11

Resulta evidente que considerando diferentes propiedades para definir la partición, obtendremos diferentes grafos reducidos, por tanto, es importante especificar la propiedad basada en la cual se va a realizar la reducción.

4.1.2 Reescritura de grafos

Una regla de reescritura de grafo es aplicada a un grafo para sustituir un subgrafo por otro [Rozemberg1997] [Janssens1982] [Kreowski1990i].

La terminología para la reescritura de grafos no está estandarizada, aquí utilizaremos los términos que se definen a continuación:

Regla de reescritura:

- $g_l ::= g_r$, un subgrafo isomorfo a g_l es sustituido por otro isomorfo a g_r
    - Información de empotrado, puede ser textual o gráfica, los modelos de engomado (Gluing Models [Blostein1994]) la especifican con un isomorfismo de engomado.
    - Condición de aplicación, son restricciones sobre la aplicación de la regla, es opcional.
    - Función de transferencia de atributos, es una función que asigna atributos, es opcional.
- G: Grafo al cual le vamos a aplicar la regla.
- g_l^{host}: subgrafo a ser reemplazado.
- g_r^{host}: subgrafo utilizado para reemplazar a g_l^{host}.
- Resto del grafo: $g-g_l^{host}$.

Una regla de reescritura, provee información de empotrado, la cual es usada para conocer la forma de conectar el grafo g_r^{host} con el resto del grafo. La principal problemática es convertir las aristas pre-empotradas en aristas post-empotradas [Rozemberg1997].

Algunos mecanismos de empotrado, permiten una especificación no restringida de las aristas post-empotradas, aunque también existen

mecanismos que si establecen una restricción [Rozenberg1986].

La selección de un mecanismo, nos pone en la disyuntiva siguiente: usamos un mecanismo complejo pero con pocas reglas de escritura, o uno de muchas reglas de escritura que sean simples [Blostein1994].

Después de una revisión de los mecanismos de empotrado existentes, aquí aplicaremos una variante del mecanismo de empotrado profundidad 1, utilizando el modelo NCE[12]. En lugar de especificar la etiqueta de la arista en la información de empotrado, especificaremos el peso de la misma.

4.2 Algoritmos propuestos

Los Sistemas de Información Geográfica generalmente tienen la tarea de buscar rutas mínimas en un determinado mapa, hasta el momento, la mayoría de los SIG existentes, llevan a cabo esta tarea de la siguiente forma:

Lo primero es definir puntos esenciales o principales sobre el mapa, por ejemplo, principales ciudades, aeropuertos, etc., y luego representar esos puntos con un grafo, en el cual los puntos serán vértices del grafo, y el o los caminos entre los dichos puntos serán las aristas; como solamente están representados ciertos puntos del mapa, sólo podemos buscar rutas en las que el origen y el destino sean puntos principales para el SIG en cuestión.

En cierta medida, se hace de esta forma, porque en un mapa nos podemos encontrar con millones de objetos, si se toma como referencia el mapa de Cuba, que como sabemos, es un país pequeño, el mismo contaría con al menos varios millones de objetos, por ejemplo, representaría a todas las casas, hospitales, escuelas, hoteles, áreas deportivas, calles, caminos, por lo que podemos darnos cuenta de la dimensión del mismo.

Cargar un grafo en memoria que represente este mapa, sería prácticamente imposible, dada la cantidad de memoria que necesitaríamos, por lo que se propone en este documento un algoritmo que reduce significativamente un

12 *Neighborhood Controlled Embedding: Empotrado con vecindad controlada.*

grafo, así como la complejidad de la búsqueda de rutas mínimas.

4.2.1 Reducir el grafo

Entrada del algoritmo:

1. Un grafo que representa un mapa. G=(V,E)
2. Una propiedad p por la cual se agruparán los vértices del grafo[13]

Salida:

1. Un grafo que representa al mapa[14]
2. Un conjunto de reglas de reescritura de grafos

Para aplicar el algoritmo, seguiremos los siguientes pasos:

- Creamos las particiones según la propiedad escogida, o sea, agrupamos los vértices que cumplan la propiedad p, por ejemplo, los vértices que representan objetos que estén en mismo municipio.
- Construimos el grafo reducido G_r.
- Por cada A_i , se escribe una regla de reescritura, de la siguiente forma:
    - g_l es un nodo (lo llamaremos nodo no terminal), con una etiqueta que identifique la propiedad p, es decir, si la propiedad p fuera código postal, un nodo g_l pudiera tener la etiqueta 10400, si todos los objetos en A_i cumplen que su código postal es 10400.
    - g_r sería el subgrafo dado por $g_l=(A_i,E_i)$, donde $(u,v) \in E_i$ si y sólo si $\exists (u,v) \in E$ y $u,v \in A_i$
    - Como especificación de empotrado, se utilizará pares de vértices (vp,v,c,u), si y sólo si $\exists (u,v) \in E$ de costo c, u ϵ Resto del grafo y v ϵ g_r^{host}, vp es el valor de la propiedad que identifica a A_i. Se crearán dos conjuntos formados por tuplas con la forma anterior, uno para

13 *La propiedad puede ser repartos, código postal, municipios, provincias, en casos de mapas, o una característica que defina a un grupo de nodos en general.*

14 *Con una significante reducción de la cantidad de vértices que conforman el grafo de entrada.*

las aristas que entran en el nodo vp, y otras para las que salen del mismo.

- Si se quiere seguir compactando el grafo, se escoje otra propiedad y se repite el proceso.

4.2.2 Búsqueda de rutas mínimas

El primer paso para la búsqueda es definir una función de distancia $fd(V_1,V_2,V)$, que calcule la distancia mínima desde el vértice V_1 hasta el V_2 pasando por V, para lo que se deben cumplir las condiciones:

V es un nodo no terminal

- $\exists (V_1,V_1')|V_1' \in V$
- $\exists (V_2,V_2')|V_2' \in V$

Para buscar una ruta mínima, podemos aplicar varios algoritmos, en este caso, aplicaremos el algoritmo Dijkstra [Sedgewick1983], con una pequeña variación:

- Sea G=(V,E) un grafo dirigido y etiquetado.
- Sean los vértices A (origen) y Z (destino)
- Sea un conjunto N un conjunto vacío
- Sea un vector D, con tantas dimensiones como elementos tiene V, y que ''guarda'' las distancias entre a y cada uno de los vértices de V.
- Sea P un vector con las mismas dimensiones que D, y que conserva la información sobre qué vértice precede a cada uno de los vértices en el camino.

El algoritmo para determinar el camino de longitud mínima entre los vértices a y z es:

1. N={A}
2. Para todos los nodos v

3. si v es adyacente a A
    - D(v)=c(A,v)
    - P(v)=A
4. en caso contrario
    - D(v) = infinito
5. Loop
    - Buscar un nodo w que no este en N tal que Dw es mínima
    - N = N U w
    - Si D(v) > D(w) + c(w,v)
        - P(v)=w
    - D(v) = min (D(v), D(w)+c(w,v))
6. Hasta que todos los nodos estén en N

La variación al algoritmo estaría, en que cada vez que calculamos distancia de un nodo V_x a otro nodo V_y adyacente a V_x, y V_x es un vértice no-terminal, la distancia se calcula utilizando la función fd(V_1,V_2,V), definida anteriormente, donde V sería V_x, V_1 sería el vértice que precede a V, y se puede encontrar en el conjunto P, y V_2 sería V_y.

Al terminar este algoritmo, en D_z estará guardada la distancia mínima entre A y Z. Por otro lado, mediante el vector P se puede obtener el camino mínimo: en P_z estará y, el vértice que precede a z en el camino mínimo; en P_y estará el que precede a y, y así sucesivamente, hasta llegar al vértice A.

Este algoritmo se puede optimizar más aún, si en cada iteración de reducción del grafo, guardamos las distancias entre cada tres vértices (u,v,z) (Ver definición de la función fd), esto traerá como consecuencia una alta complejidad para encontrar cada posible trío de vértices en cada iteración, pero como la reducción del grafo se realiza una sola vez, nos beneficiaría en la búsquedas de caminos mínimos, que si realizan muchas veces, ya que cada

vez que apliquemos Dijkstra, y nos encontremos con un nodo no-terminal, no tendremos que buscar el camino en el subgrafo correspondiente al vértice no-terminal, porque ya lo tendríamos calculado de antemano.

4.2.3 Beneficios

Con la implementación y posterior aplicación en un sistema de información de geográfica de los algoritmos presentados, se puede lograr una mayor calidad en los servicios que brinda un SIG, ya que no sólo se utilizaría en la búsqueda de caminos mínimos, sino, en cualquier operación que traiga consigo un procesamiento del mapa a nivel de todos los objetos que lo componen.

Se puede tener en cuenta el siguiente ejemplo práctico, se quiere visitar a un familiar, pero nunca se ha visitado su casa (puede vivir en cualquier lugar), solamente habría que utilizar la funcionalidad de búsqueda de camino del SIG, para obtener la ruta que se debe seguir para llegar desde una dirección determinada hasta la dirección del familiar que se desea visitar, de esta forma, el SIG mostrará la respuesta en forma de mapa, como resultado, se dispondría de una guía para llegar sin problemas hasta el destino, en este caso, la casa del familiar, además, mostrará la ruta más corta. Esto estaría limitado solamente por el mapa que esté disponible para el SIG.

Al obtener los resultados expuestos en el presente capítulo, podemos afirmar que la teoría de grafos y de reescritura de grafos, está estrechamente vinculada con los sistemas de información geográfica.

El algoritmo planteado permitirá a los sistemas de información geográfica brindar una mayor cantidad de servicios y con más calidad, al poder manipular todos los objetos representados en un mapa utilizando una forma compactada o reducida.

CONCLUSIONES

Los Sistemas de Información Geográfica tienen una gran importancia en la actualidad, permitiendo a los usuarios de los mismos, obtener información georeferenciada de forma visual (imágenes o gráficos), brindada por un servicio remoto, lo que permite al usuario final, abstraerse del origen de los datos, y combinar o compartir información a través de los SIG. Estos sistemas permiten además, contar con información, en ocasiones imprescindibles, para el proceso de toma de decisiones.

Como resultado del trabajo realizado, se llegó a las siguientes conclusiones:

1. Se desarrolló un Sistema de Información Geográfica basado en web, que cuenta con las operaciones básicas de todo SIG. Además, permite hacer búsquedas del camino más corto entre dos lugares del mapa, y localizar las construcciones representadas en el mapa en cuestión.
2. El software SIG se encuentra en una etapa de masificación, apoyado por proyectos de código abierto que amplían las posibilidades de implementar una solución de este tipo, en periodos de tiempo mucho más cortos que los necesitados hasta el momento.
3. El SIG está construido en su totalidad por herramientas libres, lo que reduce el costo de desarrollo, impulsando a las empresas(entidades) a adquirir software de este tipo.
4. El SIG presenta características generales que permiten ser ampliadas en el futuro para resolver problemas específicos, y cuenta con una documentación detallada de lo que se ha implementado hasta el momento, por lo que constituye una base para futuros desarrollos.
5. El mantenimiento y ampliación de los mapas soportados por el SIG es configurable por medio del sistema, permitiendo aumentar las posibilidades de uso del mismo.
6. Se pudo confirmar la calidad de MapServer, ya que con el presente

sistema se muestra, algunas de las posibilidades que nos brinda este servidor de mapas libre.

7. Se implementó un módulo que le da el valor añadido a los SIG que utilizan MapServer como servidor de mapas, de hacer búsquedas del camino más corto entre dos lugares.

8. Este sistema es extensible, en estos momentos se está desarrollando un módulo para crear gráficos estadísticos y mapas temáticos, el cual contribuirá datos al SIG desarrollado.

9. Se necesita disponibilidad de una mayor cantidad de cartografías para aumentar las funcionalidades del SIG.

10. Es necesaria la combinación de las bases de datos geográficas con las bases de datos de los organismos o entidades para proveer una mayor gama de servicios en un Sistema de Información Geográfica.

11. Un SIG puede contribuir significativamente al ahorro de combustible, utilizándolo para obtener la ruta más corta entre dos direcciones, y en general para planificar redes de distribución de productos o personas.

12. Podremos avanzar en el desarrollo de Sistemas de Información Geográfica en la medida que seamos capaces de compartir información cartográfica, ya que sin abundante información de este tipo, un SIG carece de sentido, debido a que su mayor aplicación es dónde existe gran cantidad de información, la cual, debido al volumen de la misma, es prácticamente imposible de manejar manualmente por las personas.

RECOMENDACIONES

1. Identificar las necesidades de los usuarios actuales del SIG para de esta forma proveer al mismo de una variedad de servicios que tributen a los usuarios del sistema.
2. Incluir en el SIG el módulo de creación de gráficos estadísticos y mapas temáticos para dotar al mismo de una mayor funcionalidad.
3. Llevar la cartografía a un nivel superior de detalle, para aumentar la información que en este momento se brinda a los usuarios, así como la calidad de la misma.
4. Desarrollar un módulo para la optimización del uso de flotas de transporte[15] utilizando el módulo Map2Graph desarrollado en el presente trabajo.

15 *Se refiere al problema de optimización "Flujo máximo de costo mínimo".*

REFERENCIAS BIBLIOGRÁFICAS

[**Ehrig1992**]: Ehrig, H. K. H. A.. *Introduction to graph grammars with applications to semantic networks*. *International Journal of Computers and Mathematical Applications*, 1992.

[**Gaurav2004**]: Gaurav, C. & Frank, D.. *Grammatical Methods in Computer Vision: An Overview*. , 2004.

[**Ehrig1980**]: Ehrig, H. K.. *Applications of graph grammar theory to consistency, synchronization, and scheduling in data base systems*. *Information Systems*, 1980.

[**Aronoff1987**]: Aronoff, S.. *Geographical Information Systems: A management perspective*. *WDL Pub.*, 1987.

[**Delgado2000**]: Delgado, T.. *Infraestructura Cubana de Datos Geoespaciales: Una necesidad nacional para la integración y diseminación de datos geoespaciales*. *Memorias del II Congreso InternacionalGeomática 2000*, 2000.

[**Chorley1987**]: Chorley, R.. *Handling Geographic Information*. , 1987.

[**Clarke1990**]: Clarke, M.. *Geographical information systems and model based analysis:towards effective*
decision support systems. , 1990.

[**Cowen1989**]: Cowen D.. *Lectura NCGIA*. , 1989.

[**SIGVectoriales**]:
http://www.geogra.uah.es/gisweb/1modulosespanyol/IntroduccionSIG/GISModule/GIST_Vector.htm

[**SIGRaster**]:
http://www.geogra.uah.es/gisweb/1modulosespanyol/IntroduccionSIG/GISModule/GIST_Raster.htm

[**OGC:WMS**]: Open Geospatial Consortium Inc.. *OpenGIS Web Map Server Implementation Specification*. , 2006.

[**ESRI:SHP**]: *http://es.wikipedia.org/wiki/Shapefile*

[**PostGIS**]: *http://es.wikipedia.org/wiki/PostGIS*

[**GML**]: *http://es.wikipedia.org/wiki/Geography_Markup_Language*

[**OGR**]: *http://es.wikipedia.org/w/index.php?title=OGR&action=edit*

[**TIFF**]: *http://es.wikipedia.org/wiki/TIFF*

[**GDAL**]: *http://es.wikipedia.org/w/index.php?title=GDAL&action=edit*

[**TType**]: *http://es.wikipedia.org/wiki/TrueType*

[**Blostein1994**]: Blostein D., Fahmy H., Grbavec A.. *Practical Use of Graph Rewriting*. , 1994.

[**Nagl1979**]: Nagl, M.. *A tutorial and bibliographical survy on graph grammars*. , 1979.

[**Nagl1987**]: Nagl, M.. *Set theoretic approaches to graph grammars*. , 1987.

[**Schneider1993**]: Schneider, H.. *On categorical graph grammars integrating structural transformations and operations on labels*. , 1993.

[**Grafos**]: *http://personales.upv.es/arodrigu/grafos/index.htm*
[**MRG**]: *http://www-2.dc.uba.ar/materias/so/datos/cap10.pdf*
[**REG**]: *http://www.sc.ehu.es/jiwlucap/ILNITema3.pdf*
[**ArbolExp**]: *http://www.algoritmia.net/articles.php?id=18*
[**PHPDocs**]: *http://www.php.net/docs.php*
[**PHPInstall**]: *http://www.php.net/manual/es/install.php*
[**WikiCron**]: *http://es.wikipedia.org/wiki/Cron*
[**PHPLineaComandos**]: *http://www.php.net/manual/es/features.commandline.php*
[**PHPGTKUserGuide**]: *http://gtk.php.net/manual1/es/userguide.php*
[**ERPOSIX**]: *http://www.tin.org/bin/man.cgi?section=7&topic=regex*
[**Java**]: *http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java*
[**PythonSitioOficial**]: *http://python.org/*
[**PSF**]: *http://www.python.org/psf/*
[**Zope**]: *http://www.zope.org/*
[**Zope3**]: *http://z3lab.org/*
[**ZEO**]: *http://www.zope.org/Documentation/Books/ZopeBook/2_6Edition/ZEO.stx*
[**ZopeCaseStudies**]: *http://www.zope.com/customers/case_studies.html*
[**ZPL**]: *http://www.zope.org/Resources/ZPL*
[**GPL**]: *http://www.gnu.org/copyleft/gpl.es.html*
[**CERTZPL**]: *http://www.opensource.org/licenses/zpl.php*
[**OpenSourceDefinition**]: *http://www.opensource.org/docs/definition.html*
[**OpenSource**]: *http://www.opensource.org/*
[**GPLCompatibles**]: *http://www.gnu.org/philosophy/license-list.html#GPLCompatibleLicenses*
[**ZopeMailingLists**]: *http://www.zope.org/Resources/MailingLists*
[**FW**]: *http://es.wikipedia.org/wiki/Framework*
[**ObjectPublishing**]: *http://www.zope.org/Documentation/Books/ZDG/ObjectPublishing.stx*
[**Jacobson2000**]: JACOBSON, I., BOCH, g., RUMBAUGH, J.. *El Proceso Unificado de Desarrollo de Software*. , 2000.
[**Rumbaugh1998**]: RUMBAUGH, J., JACOBSON, I., BOOCH, G.. *El Lenguaje Unificado de Modelado. Manual de Referencia.* , 1998.
[**Rozemberg1997**]: Rozenberg G., S. A.. *Handbook of Formal Languages*. , 1997.
[**Janssens1982**]: Janssens D.. *Graph grammars with neighbourhood-controlled embedding*. , 1982.
[**Kreowski1990i**]: Kreowski, H. R.. *On structured graph grammars i*. *Informations Sciences*, 1990.
[**Rozenberg1986**]: Rozenberg G., W. E.. *Boundary NLC graph grammars - basic definitions, normal form, and complexity*. , 1986.
[**Sedgewick1983**]: Sedgewick R.. *Algorithms*. *ADDISON-WESLEY*, 1983.

BIBLIOGRAFÍA

[**Bunke1989**]: Bunke, H. H.. *A parser for context free plex grammars*. *Proc. 15th Intl. Workshop on Graphic Theoretic Concepts in Computer Science*, 1989.

[**Bunke1992**]: Bunke, H. H.. *Syntactic analysus of context-free plex languages for pattern recognition*. *Structured Document Image Analysis, Eds. Baird, Bunke, Yamamoto, Springer*, 1992.

[**Egar1992**]: Egar, J. M. P. A.. *Automated interpretation of diagrams for specification of medical protocols*. *AAAI Symposiumï¿½ Reasoning with Diagrammatic Rpresentation, Stanford Univiersity*, 1992.

[**Fahmy1993**]: Fahmy, H. B.. *A graph grammar programming style for recognition of music notation*. *Machine Vision and Applications*, 1993.

[**Fu1982**]: Fu. *Syntactic pattern recognition and applications*. *Prentice Hall*, 1982.

[**Gemma2001**]: Gemma, S. A.. *Un modelo sintáctico para la representación, segmentación y reconocimiento de símbolos texturados en documentos gráficos*. , 2001.

[**Goldfarb1992**]: Goldfarb. *Transformation system are more economicaland informative class description than formal grammars*. *11th International Conference on Pattern Recognition, Delft, Netherlands*, 1992.

[**Gottler1992**]: Gottler H.. *Diagram editors= graps + attributes + graph grammars*. *International Journal of Man-Machine Studies*, 1992.

[**Gottler1982**]: Gottler H., B.. *On the generative power of sequential and parallel programmed graph grammars*. *Computing*, 1982.

[**Haken**]: Haken, L. B.. *Shared processing in automatic diagram recognition and generation: a case study of music notation*. , .

[**István2003**]: István, J.. *CONTEXT-FREE GRAPH GRAMMAR INDUCTION USING THE MININUM DESCRIPTION LENGTH PRINCIPLE*. , 2003.

[**Janssens**]: Janssens D., R. G.. *Graph grammars with neighbourhood-controlled embedding*. , 1982.

[**Kreowski1990**]: Kreowski, H. R.. *On structured graph grammars ii*. *Informations Sciences*, 1990.

[**Martín-Vide2004**]: Martín-Vide C., Mitrana V., Paun G.. *Formal Languages and Applications*. *Springer*, 2004.

[**Mauss1992**]: Mauss, J. K.. *A heuristic driven parser based on graph grammars for feature recognition in cim*. *Proc. International Workshop on Structural and Syntactic Pattern Recognition, Bern Switzerland*, 1992.

[**Pfaltz1969**]: Pfaltz, J. R.. *Web grammars*. *Proc. 1st Int. Joint Conf. on Artificial Intelligence, Washinton*, 1969.

[**Pfaltz1972**]: Pfaltz, J. R.. *Web grammars and picture description*. *Computer Graphics and Image Processing*, 1972.

[**Rozenberg1986**]: Rozenberg G., W. E.. *Boundary NLC graph grammars - basic definitions, normal form, and complexity*. , 1986.

[**Collin1993**]: Collin S., T. K.. *Don't tell mom i'm doing document analysis, she believes i'm in the computer vision field*. *Proc. Second International Conference on Document Analysis and Recognition, Tsukuba, Japan*, 1993.

[**Shailesh**]: Shailesh, P. D., Fang, H. & Dr., T.O.. *Inferring the structure of Graph Grammar from Data*.

[**Sindre1993**]: Sindre, G. J. G. B.. *Onion graphs aesthetics and layout*. *Proc. IEEE Symposium on Visual Languages, Bergen, Norway*, 1993.

[**Strzalkowski1990**]: Strzalkowski, T.. *Reversible logic grammars for natural language parsing and generation*. *Canadian Computational Intelligence Journal*, 1990.

Anexos

ANEXO I. DIAGRAMAS DE CLASES

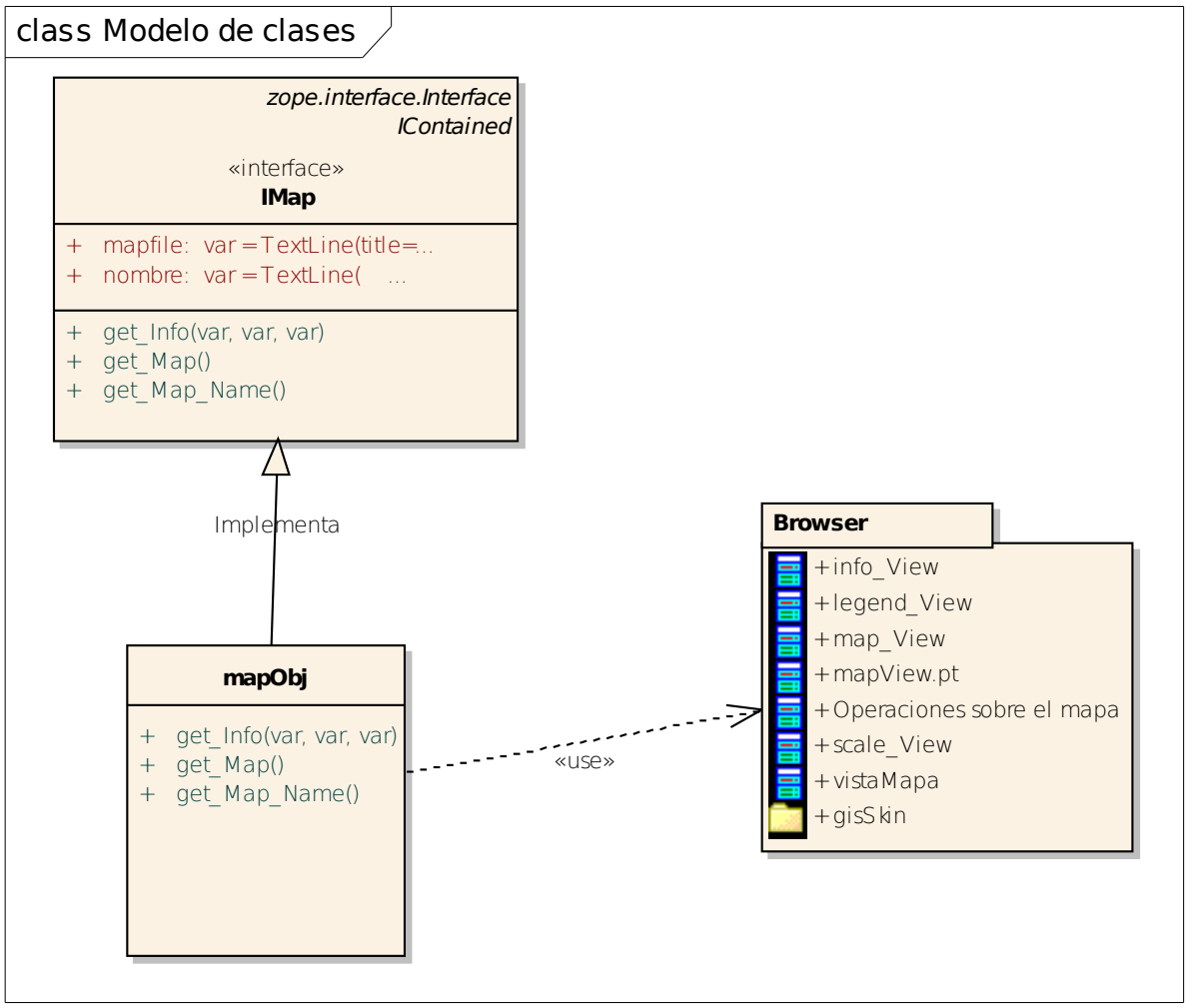

Ilustración 12: Diagrama de clases del módulo mapObj

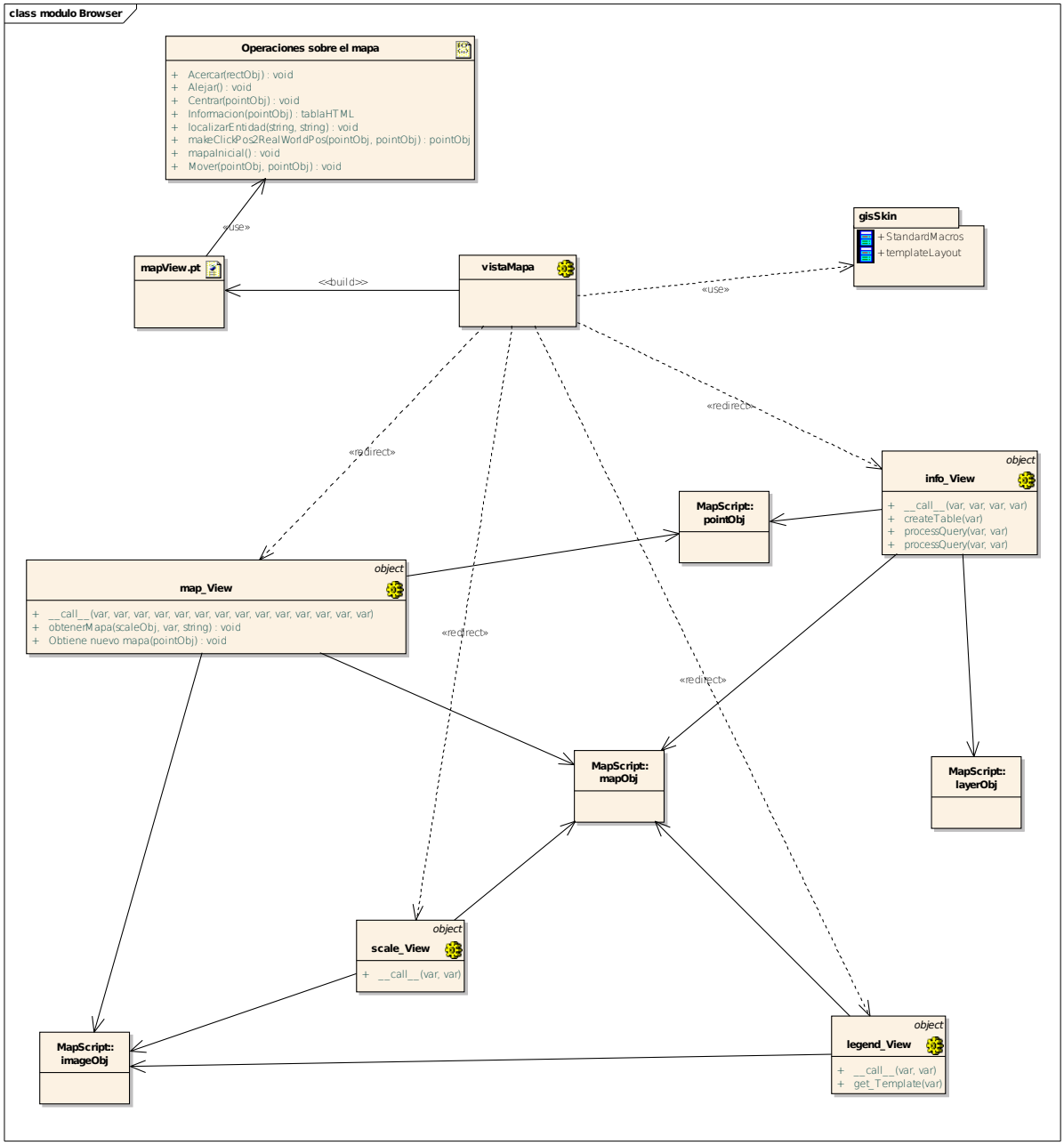

Ilustración 13: Diagrama de clases web del módulo browser

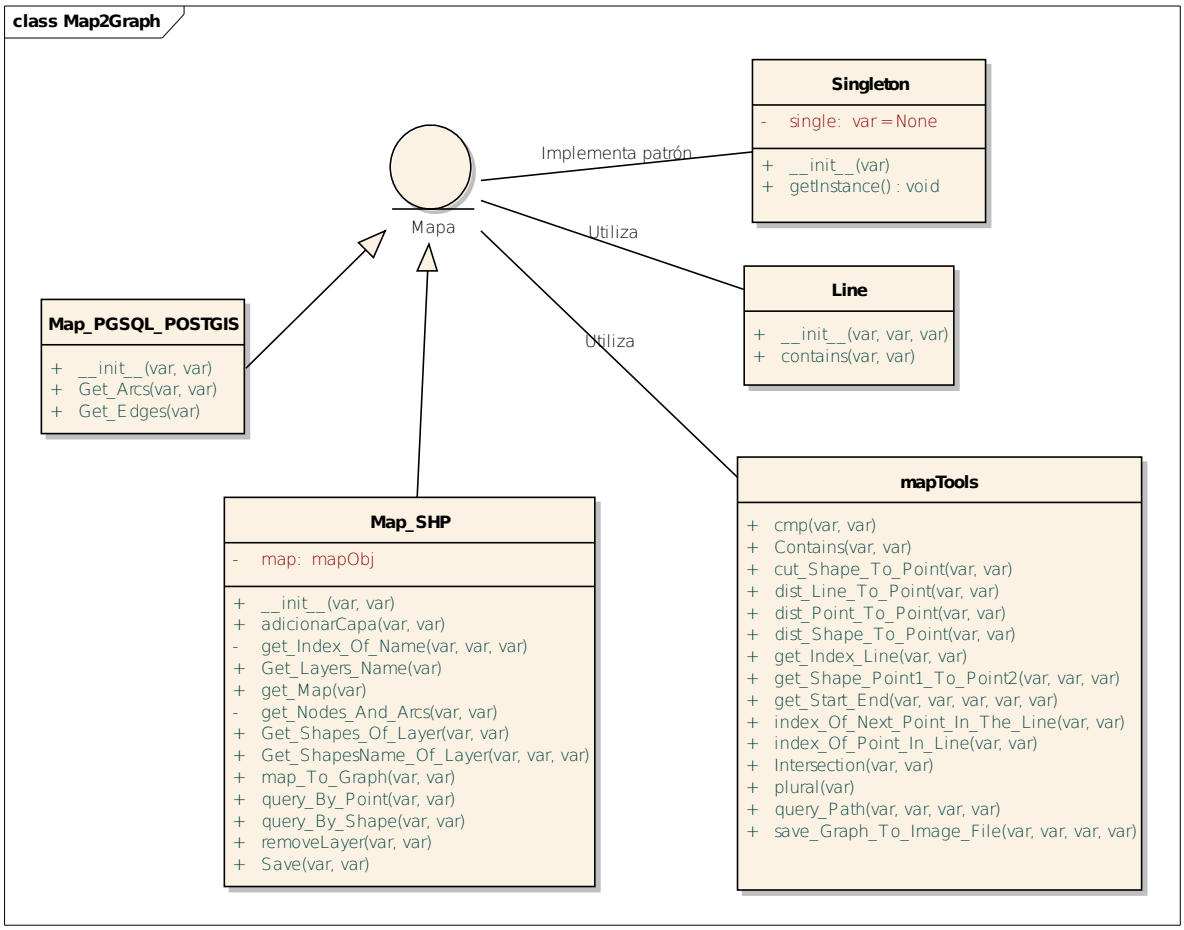

Ilustración 14: Diagrama de clases del módulo Map2Graph

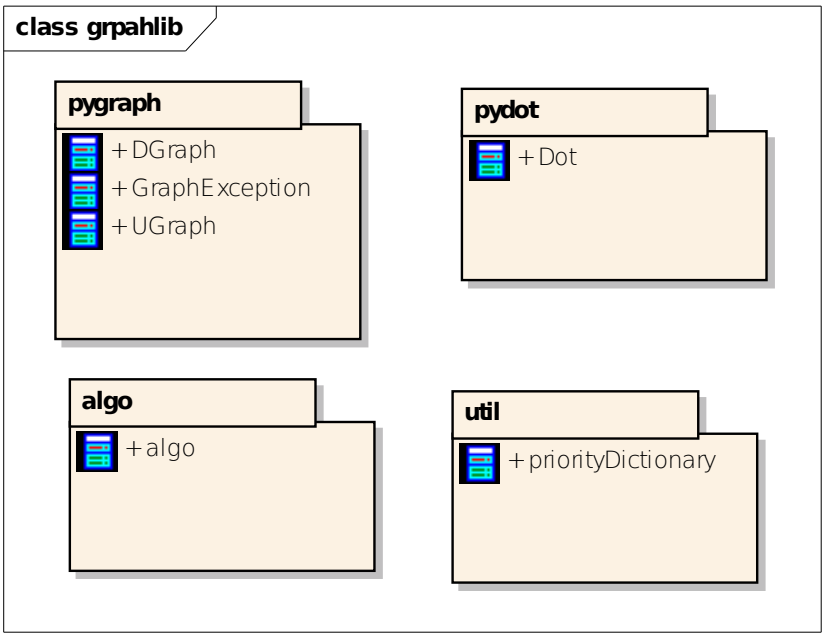

Ilustración 15: Módulos utilizados

ANEXO II. DIAGRAMAS DE SECUENCIA

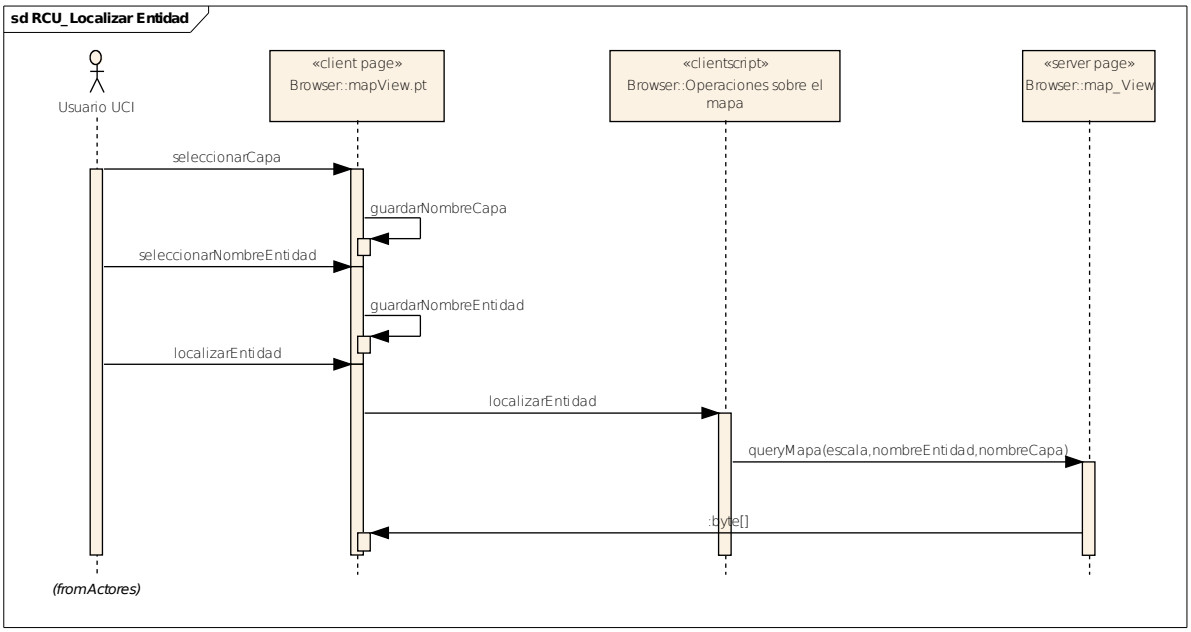

Ilustración 16: Diagrama de secuencia del caso de uso Localizar Entidad

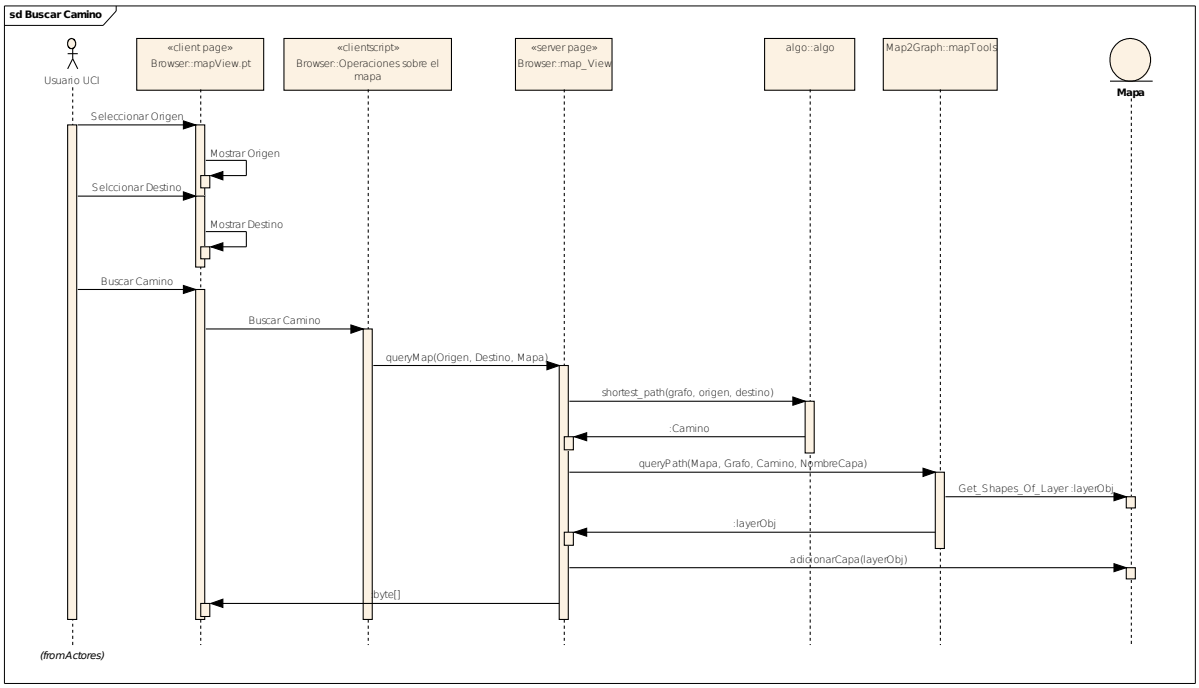

Ilustración 17: Diagrama de secuencia del caso de uso Buscar Camino

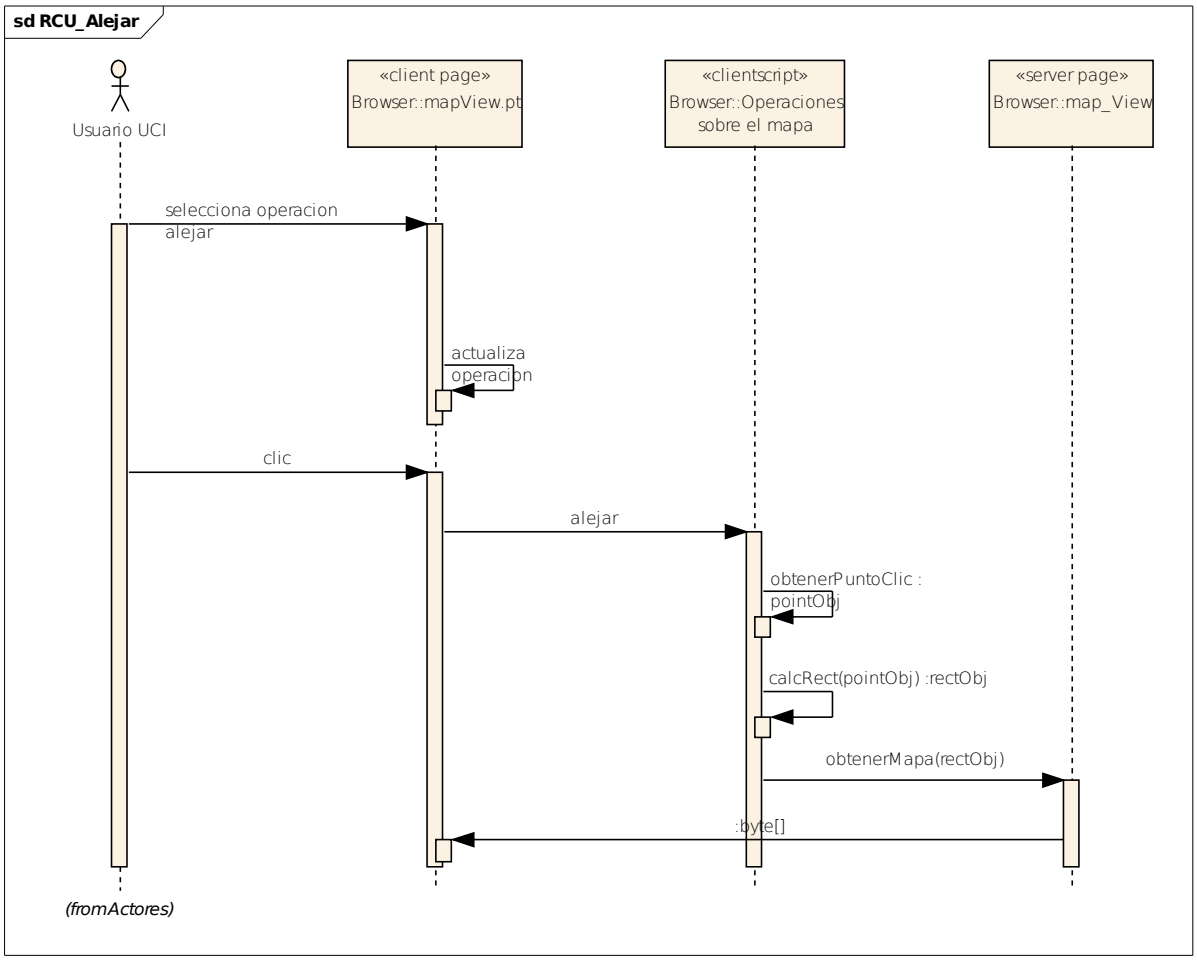

Ilustración 18: Diagrama de secuencia del caso de uso Alejar

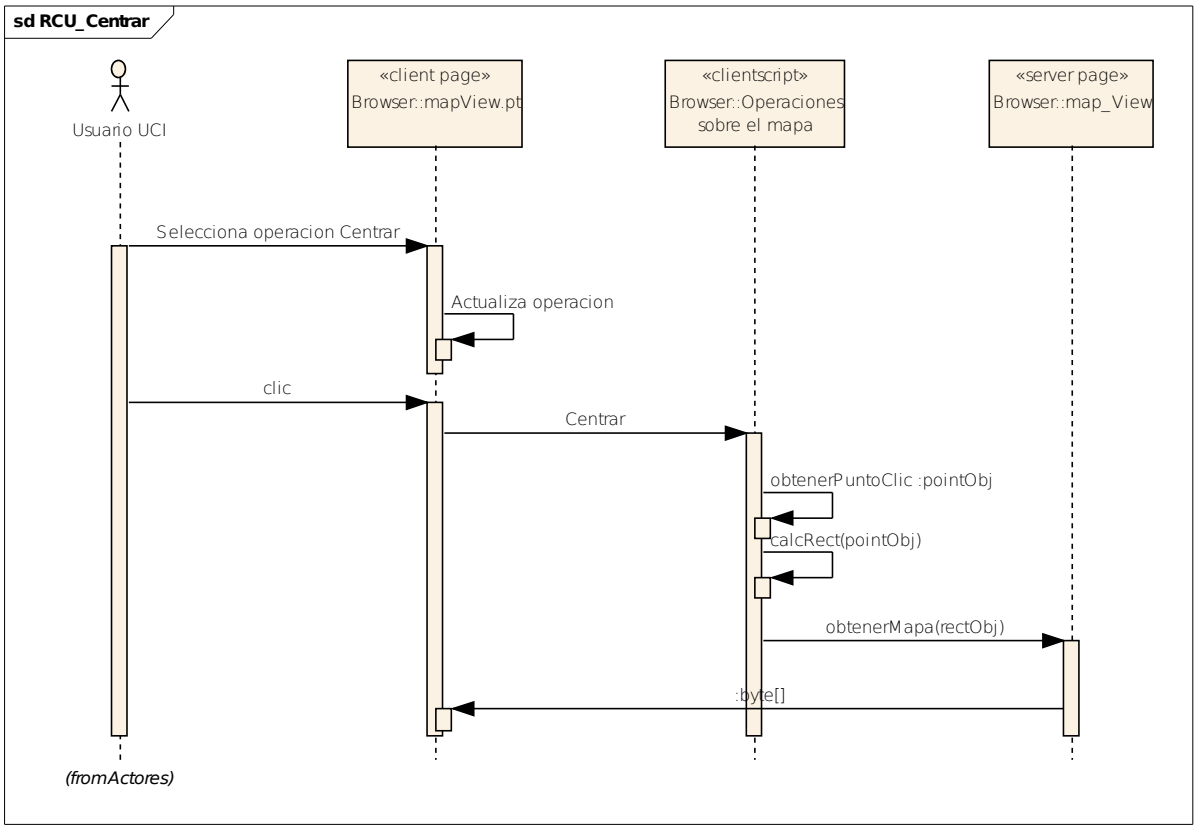

Ilustración 19: Diagrama de secuencia del caso de uso Centrar

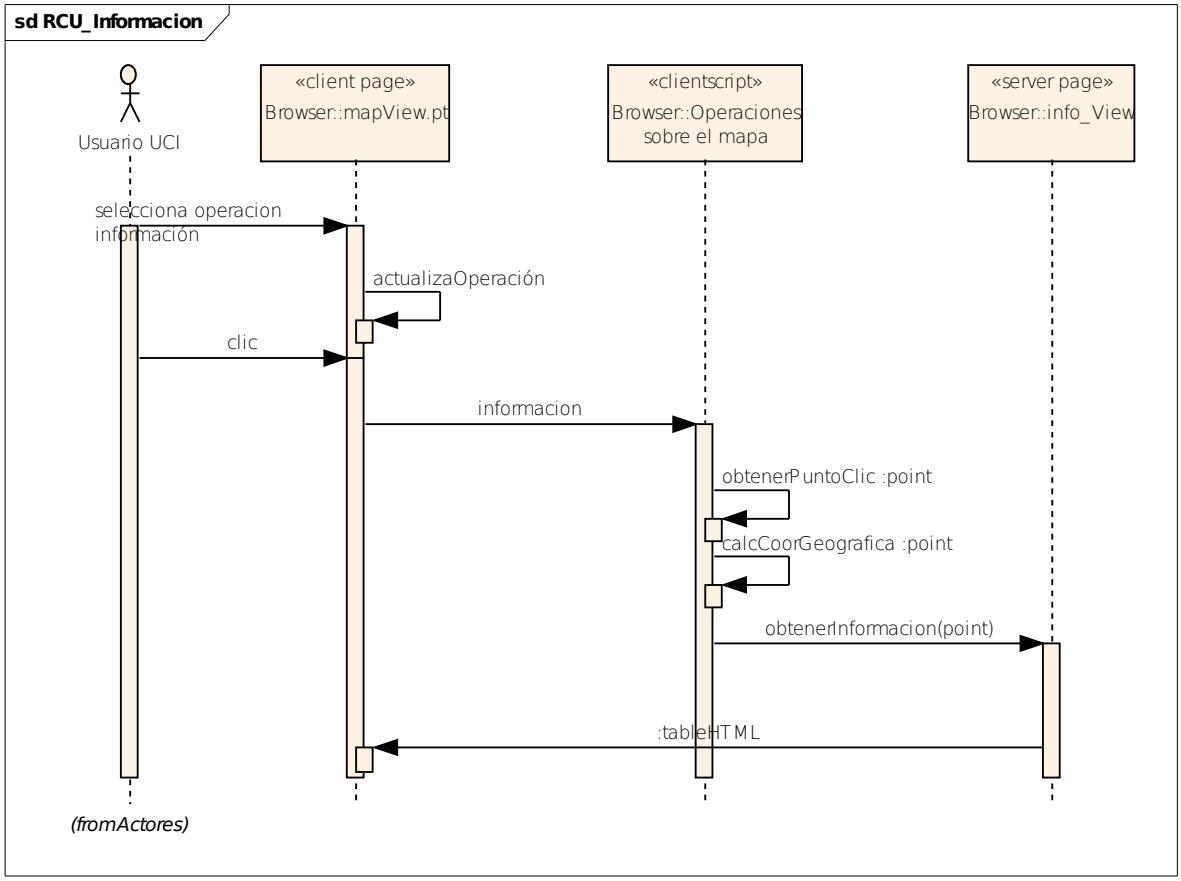

Ilustración 20: Diagrama de secuencia del caso de uso Obtener Información

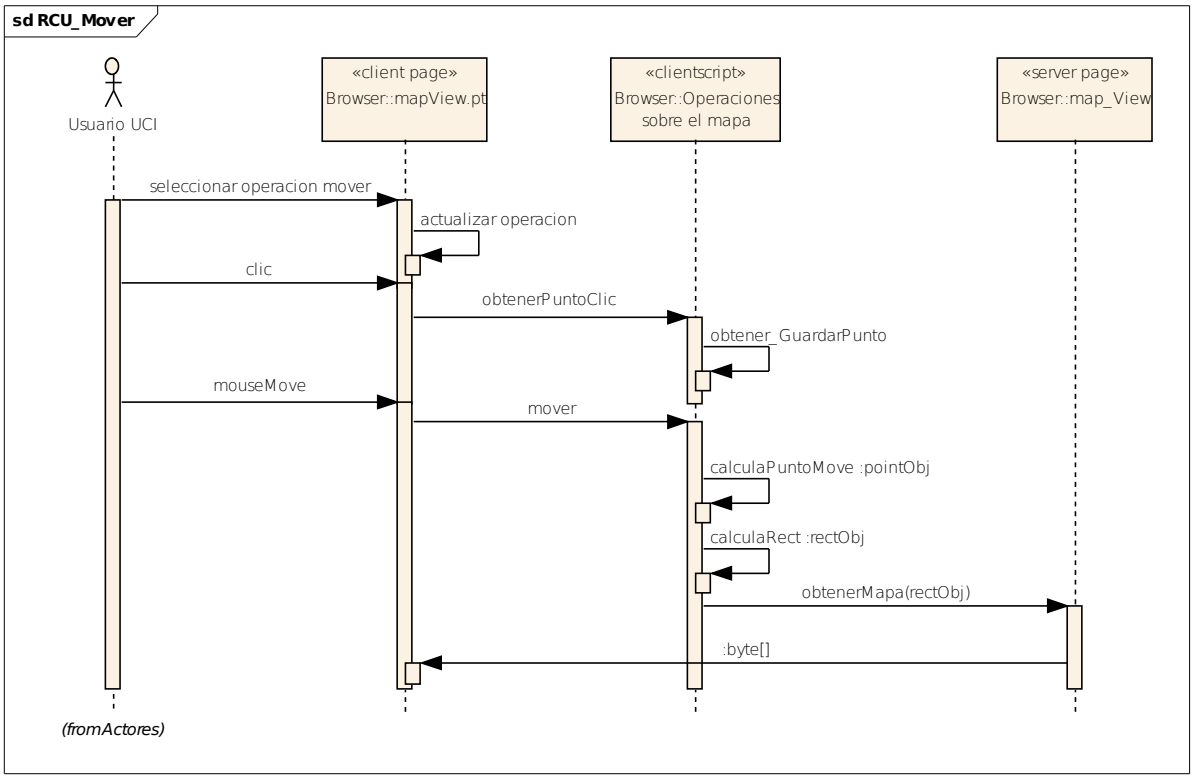

Ilustración 21: Diagrama de secuencia del caso de uso Mover

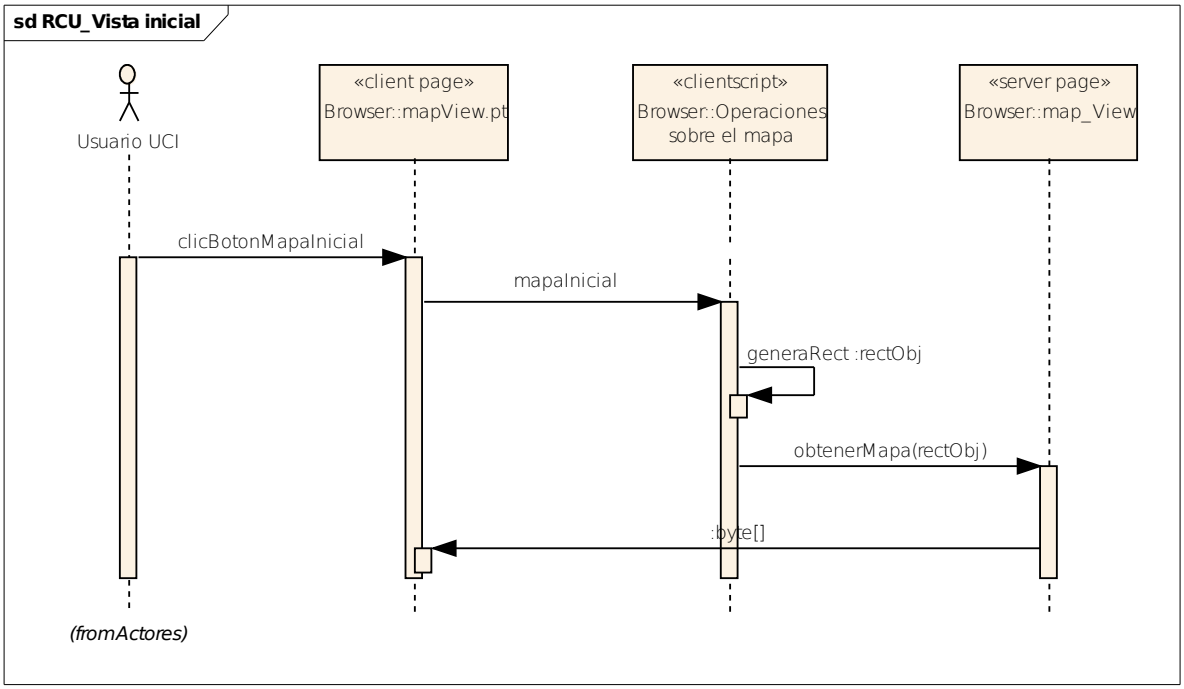

Ilustración 22: Diagrama de secuencia del caso de uso Vista Inicial

ANEXO III. DIAGRAMA DE DESPLIEGUE

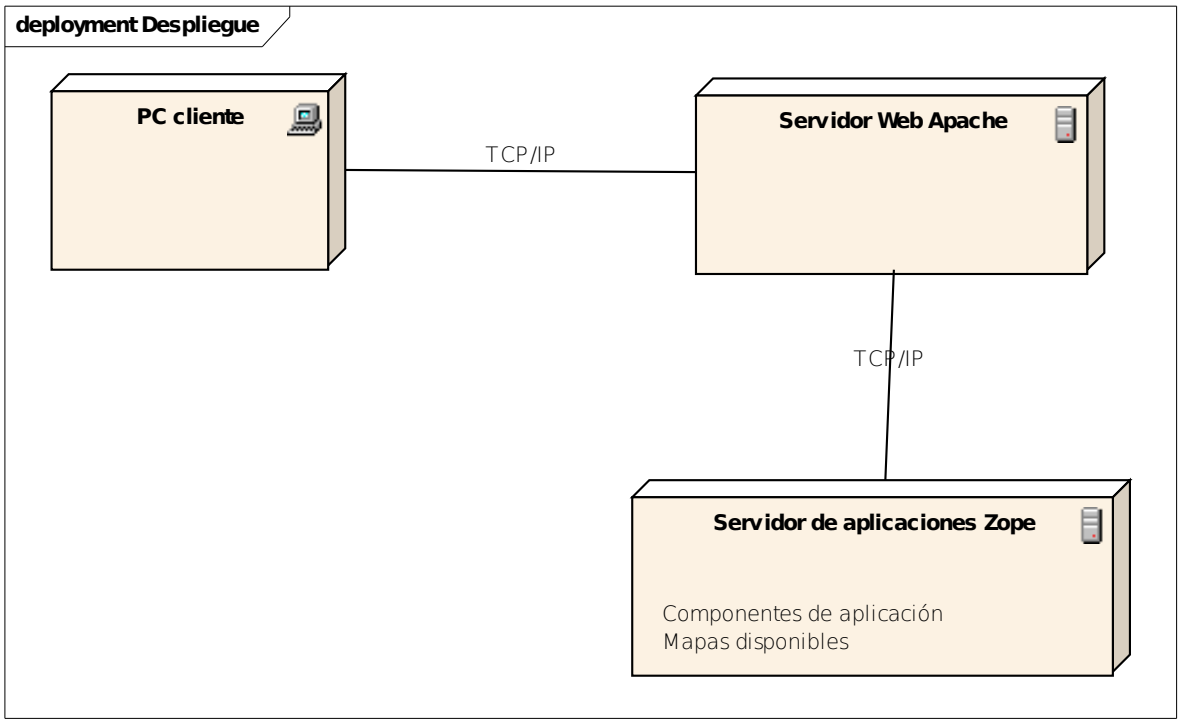

Ilustración 23: Modelo de despliegue