UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 6

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS.

TÍTULO: Módulo de administración del Sistema de Información Geográfica para el transporte obrero de la Universidad de las Ciencias Informáticas.

AUTOR: Yeisy Legrá Matos

TUTOR: Ing. Carlos Enrique Ramírez Martín

Junio 2011. La Habana, Cuba.

¨**Año 53 de la Revolución**¨

"El Mundo está en las manos de aquellos que tienen el coraje de soñar y correr el riesgo de vivir sus sueños"

Paulo Coelho

Declaración de Autoría

Declaro que soy la única autora de este trabajo y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____	_____
Yeisy Legrá Matos	Ing. Carlos Enrique Ramírez Martín

Agradecimientos

A mis padres Anaevis Matos y Norberto Legrá, por su entrega y dedicación, a mi sobrino Ernesto por ser el regalo más bello que Dios me ha dado. A mi manita Mayumis Suarez por quererme y apoyarme tanto. A mis hermanos Yolexy Legrá y Osmanis Legrá por sus cuidados y amor. A mis abuelitos Obilio Matos y José Legrá donde quiera que estén. A mis abuelas Damaris Orduñez y Celina Lliber por su amor y oraciones. A mi buena amiga Norvelis Legrá Motos, por aguantarme tantos años y quererme tanto, gracias por ser tan especial. A mi tati Yuriesky Duarte por permitirme concer el verdadero amor. A mí querida tía Alis Matos, por su ayuda y enorme afecto. A mi tía Anais Matos y mi tío Infrain López por hacerme sonreír en los momentos difíciles y quererme como a una hija. A mi prima Yarinis López y Lilianni Cala por tantos momentos de alegría. A mi tío Domingo Legrá (Puchito), por ser tan bueno y quererme tanto, a mi tío Enris Matos por servirme de ejemplo, por sus consejos y cariño. A toda mi familia por su apoyo incondicional y por confiar siempre en mí, gracias por su cariño y amor. A mi Tutor Carlos, por su apoyo en la investigación, e inculcarme el deseo de hacer cosas con calidad, por su comprensión y dedicación durante el desarrollo del trabajo. Para todos aquellos que me apoyaron y ayudaron, mi eterno agradecimiento.

Dedicatoria

A mis padres Anaevis Matos y Norberto Legrá por su dedicación y enorme amor, por haber confiado siempre y darme todo su apoyo.

A mi sobrino Ernesto Legrá para que le sirva de ejemplo y guía para alcanzar todos sus sueños.

RESUMEN

Actualmente los Sistemas de Información Geográfica (SIG) cuentan con gran aceptación a escala mundial, ya que se considera que entre el 80 y el 90 por ciento de la toma de decisiones tiene involucrada un componente geoespacial. Probablemente el elemeto más importante de un SIG son los datos, los cuales necesitan a menudo ser periódicamente actualizados. Es por ello que una correcta gestión de la información que manejan los mismos, garantiza su eficiente funcionamiento.

La Universidad de las Ciencias Informáticas (UCI) cuenta con prestigio nacional e internacionalmente por su excelente desmpeño como organización educacional vinculada a la producción de software, siendo uno de sus principales objetivos la automatización de los procesos económico-sociales en Cuba. Teniendo en cuenta los elementos antes abordados, la investigación se centra en desarrollar un módulo para el Sistema de Información Geográfica de Rutas de transporte obrero de la UCI que permita administrar la información referente a las rutas de este transporte. La puesta en práctica de este sistema facilitará actualizar de manera sencilla, eficiente y con un alto nivel de seguridad, la información y funcionalidades que maneja el SIG.

En la investigación se abordan los resultados de un análisis de las especificaciones internacionales acerca del tema, así como las tecnologías actuales existentes y la selección de las más apropiadas para el desarrollo del sistema. Conjuntamente el proceso de desarrollo del software fue guiado por la metodología Agile Unified Process (AUP), mediante la cual se generaron los artefactos correspondientes a la misma posibilitando una mayor comprensión y entendimiento del sistema.

Palabras claves: Sistemas de Información Geográfica (SIG), administrar, transporte obrero de la UCI.

Índice de contenidos

Introducción

En la actualidad el auge de las Tecnologías de la Información y las Comunicaciones (TIC) han revolucionado el mundo, estas *son un conjunto de servicios, redes, software y dispositivos que tienen como fin la mejora de la calidad de vida de las personas dentro de un entorno, y que se integran a un sistema de información interconectado y complementario, y están destinadas a optimizar la comunicación humana* (Pérez, 2008).

La Universidad de las Ciencias Informáticas (UCI), es el motor impulsor de la aplicación de estas tecnologías en el país. Este centro cuenta con estudiantes y profesores que estudian y desarrollan las tecnologías informáticas. Su principal objetivo es la automatización de los procesos involucrados en el quehacer cotidiano, respondiendo a las necesidades que el momento histórico y la sociedad reclaman. En la UCI, específicamente en el Departamento de Geoinformática perteneciente a la Facultad 6, se desarrollan Sistemas de Información Geográfica (SIG).

Los SIG son un tipo especializado de sistemas que se distinguen por su capacidad de manejar información espacialmente referenciada y que permiten además su representación gráfica. Se dice que son herramientas, porque ayudan a la formación de elementos de juicio para la toma de decisiones luego de que se han aprovechado sus funciones de captura, almacenamiento, refinamiento, análisis y visualización de la información. Se define a un SIG como *un conjunto de métodos, herramientas y datos que están diseñados para actuar coordinada y lógicamente para capturar, almacenar, analizar, transformar y presentar toda la información geográfica y de sus atributos con el fin de satisfacer múltiples propósitos.* (Pérez, 2008)

La Universidad de las Ciencias Informáticas en su desempeño como organización educacional vinculada a la producción de software cuenta con un número considerable de estudiantes, y profesores tanto externos como internos. Con el fin de alcanzar un correcto funcionamiento laboral se dispone de un servicio de transportación para obreros y estudiantes que cuenten con la debida autorización. Para un eficaz funcionamiento de este servicio se hace imprescindible una correcta gestión de los itinerarios, rutas y paradas ubicadas en todo el territorio de la provincia de La Habana. Debido al creciente número de recorridos y a la carencia de un sistema capaz de brindar la información necesaria para la gestión de toda la información referente a la transportación de obreros de la UCI, a los implicados les resulta complejo la ubicación

de las diferentes paradas ya que las mismas son numerosas y existen disímiles asociaciones entre los recorridos y la paradas asociadas a cada uno de ellos. Con el objetivo de mejorar este servicio surge el proyecto SIG Rutas, el cual tiene como meta desarrollar un sistema que permita la interacción entre los usuarios que reciben su beneficio y la información asociada a cada recorrido y paradas de manera automática.

Con el fin de lograr una mejor organización y un óptimo funcionamiento del sistema, los datos asociados a las rutas y paradas se encuentran bajo constantes transformaciones, que van desde un simple cambio en la trayectoria de una ruta, hasta la compleja eliminación de una parada; por lo que se hace necesario gestionar dichos datos por un grupo de administradores. Estos serán los responsables de actualizar la información existente de las rutas y paradas de acuerdo a las necesidades de los trabajadores de transportación de la UCI. Conjuntamente se teme por la integridad y seguridad de los datos que se manejan debido a la repercusión que traería un mal uso de los mismos.

Una vez analizados todos estos elementos se define como **problema a resolver** la siguiente interrogante: ¿Cómo administrar los datos del sistema de información geográfica de rutas de transporte obrero de la Universidad de las Ciencias Informáticas?

Para dar solución a este problema se determinó como **objeto de estudio** los módulos de administración de los sistemas de información geográfica. Para luego incidir directamente en los módulos de administración del sistema de información geográfica de rutas de transporte obrero de la UCI, lo cual representa el **campo de acción** donde se enmarca la investigación.

La meta a alcanzar en el presente trabajo para solucionar el problema queda sujeta al siguiente **objetivo general**: Desarrollar un módulo para el sistema de información geográfica de rutas de transporte obrero de la Universidad de las Ciencias Informáticas, que permita administrar la información referente a las rutas de transporte obrero de la UCI.

Quedando definido como **idea a defender** en la presente investigación que: el desarrollo de un módulo para el sistema de información geográfica de rutas de transporte obrero de la UCI, permitirá administrar de manera óptima y eficiente los datos del sistema de información geográfica de rutas de transporte obrero de dicha universidad.

Para darle cumplimiento al objetivo general se han propuesto las siguientes **Tareas de investigación:**

- Caracterizar los módulos de administración de los Sistemas de información Geográfica.
- Fundamentar las tecnologías a utilizar para el desarrollo del módulo de administración de Sistemas de Información Geográfica.
- Seleccionar y argumentar la Metodología de Desarrollo de Software a usar en el proceso.
- Identificar las funcionalidades que debe brindar el módulo de administración del sistema de información geográfica de rutas de transporte obrero de la Universidad de las Ciencias Informáticas.
- Diseñar el módulo de administración del sistema de información geográfica de rutas de transporte obrero de la Universidad de las Ciencias Informáticas.
- Implementar las funcionalidades del módulo de administración del sistema de información geográfica de rutas de transporte obrero de la Universidad de las Ciencias Informáticas.
- Realizar las pruebas al módulo de administración del sistema de información geográfica de rutas de transporte obrero de la Universidad de las Ciencias Informáticas.

Durante el desarrollo de la investigación se emplearon distintos métodos que posibilitaron obtener la información necesaria para el desarrollo del sistema. Entre estos métodos se encuentran los teóricos que se relacionan a continuación.

Histórico-lógico: Se empleó durante la primera fase de la investigación, con el mismo fue posible desarrollar un estudio del estado del arte de la problemática analizada, permitiendo conocer basado en el análisis de otras soluciones: el surgimiento, la trayectoria y desarrollo de los SIG, así como las diferentes características de los módulos de administración de estos.

Análisis-sintético: Permitió el análisis de los SIG y sus módulos de administración, logrando desglosar el fenómeno para conocer sus características específicas. Además posibilitó realizar un estudio detallado de los conceptos relacionados con el tema, así como la revisión de los documentos existentes en el Departamento Geoinformática de los proyectos relacionados con la investigación.

Síntesis: Mediante el estudio de los SIG y los módulos de administración de los mismos el método brinda la posibilidad de integrar todo el conocimiento adquirido sobre estos durante el análisis, llegando así a revelar sus características generales. Además del análisis de los conceptos relacionados con este tema, así como los documentos existentes en el departamento Geoinformática, los cuales servirán de base para la investigación.

También se utilizó el siguiente método empírico:

Entrevista: Con este método se realizaron entrevistas a líderes y desarrolladores de los proyectos pertenecientes al departamento de Geoinformática, lo que permitió adquirir conocimientos e información acerca de la situación problemática y las particularidades del SIG Rutas.

El presente documento se encuentra estructurado en 4 capítulos; en el **capítulo 1**, se realiza un estudio de los principales conceptos referentes a los sistemas de información geográfica, haciendo énfasis en los módulos de administración de los mismos. Se ofrece un panorama o estado del arte a nivel mundial y nacional de los sistemas y proyectos que existen o están vinculados con el trabajo. En el **capítulo 2**, se seleccionan y describen las tecnologías a utilizar para desarrollar la solución propuesta, se realiza un análisis de las herramientas seleccionadas para la modelación e implementación del sistema. En el **capítulo 3** se describe el negocio a través de un Modelo de Dominio y se realiza un análisis del sistema que se quiere llevar a cabo. También se definen las principales funcionalidades descritas mediante las herramientas pertinentes. En el **capítulo 4,** se describen cómo fue construida la solución propuesta, se modelan los diagramas de clases, se plantea el modelo de datos, se especifican los principios seguidos en el diseño e implementación.

Capítulo 1: “Fundamentación Teórica”.

1.1 Introducción

Este capítulo comprende toda la teoría relacionada con los Sistemas de Información Geográfica especificando el módulo de administración de los mismos. También se describe detalladamente el objeto de estudio y el entorno en que se enmarca el proceso, se analizan otras soluciones nacionales e internacionales existentes y se aborda con mayor profundidad la situación problemática que da origen a este trabajo, mostrando la situación actual del dominio del problema, por qué, para qué y qué se va a realizar para darle solución al mismo. Se muestran las definiciones y conceptos que son de utilidad para el entendimiento de la investigación.

1.2 Características de los Sistemas de Información Geográfica (SIG).

Los SIG comenzaron a utilizarse como herramienta de mapeo y análisis en Geografía en la década de los 60 en Canadá. A partir de los 70 se produce una revolución de la tecnología SIG condicionada por diversos factores relacionados con el surgimiento, accesibilidad y bajo costo de las minicomputadoras, el desarrollo del software libre y la amplia disponibilidad de bases de datos en mapas digitales. Los SIG se convierten en un campo multidisciplinario contribuyendo a la planificación territorial, al marketing y a las investigaciones geológicas y climáticas, entre otras. Desde el surgimiento de los SIG se han propuesto varias definiciones dependiendo del grado de desarrollo de los mismos, entre las cuales se encuentra: *los SIG constituyen un sistema automatizado capaz de almacenar, manipular, analizar y visualizar grandes volúmenes de información referenciada geográficamente* (Torres, 2005).

Los SIG se consideraron como el componente tecnológico de una nueva disciplina denominada "ciencia de la información geográfica", referida al estudio de los aspectos genéricos relacionados con el uso de la tecnología SIG en cuanto a su implementación y capacidades potenciales (Serpa, 2001).

Un SIG se define como "*un sistema computacional para la entrada; manejo (almacenamiento y recuperación de información); manipulación, análisis; y representación de datos geográficos*". (Torres, 2005).

Un SIG puede entenderse como *un poderoso conjunto de herramientas para adquirir, almacenar, recuperar a voluntad, transformar y desplegar datos espaciales para determinados propósitos. Técnicamente se puede definir un SIG como una tecnología de manejo de información geográfica formada por equipos electrónicos (hardware) programados adecuadamente (software) que permiten manejar una serie de datos espaciales (información geográfica, datos geográficos) y realizar análisis complejos con estos siguiendo los criterios impuestos por el equipo científico (personal o equipo humano)* (Pérez, 2008).

1.2.1 Características de las base de datos en los SIG

Una base de datos es una colección de información organizada de forma que un programa de ordenador pueda seleccionar rápidamente los fragmentos de datos que necesite. Es un sistema de archivos electrónico, para su posterior uso en la recolección y manejo de datos. (Pérez, 2008).

Las bases de datos se utilizan normalmente para almacenar una variedad de información dependiendo del dominio de la aplicación elegida. Los datos necesitan a menudo ser periódicamente actualizados tanto dentro de su estructura y valor, como de los cambios en el dominio de la aplicación. (Torres, 2005). Probablemente el componente más importante de un SIG son los datos siendo así un aspecto fundamental dentro de los SIG la forma de almacenar la información. Si bien en el inicio de estos sistemas era habitual que la gestión de esta información se realizara mediante programas propios, la tendencia actual es la de desligar el producto SIG del gestor de la base de datos utilizado, de forma que sea posible utilizar cualquiera de los productos que para este fin existen en el mercado.

Las bases de datos de los SIG contienen datos gráficos y alfanuméricos, integrados para formar una completa fuente de información. La exactitud y el nivel de resolución son elementos importantes en el desarrollo de una base de datos de un SIG y vienen determinados por el uso al que va destinado el sistema. Así, un SIG diseñado para

aplicaciones de ingeniería requerirá, en general, un alto nivel de exactitud y una gran resolución. Sin embargo, sistemas pensados para planificaciones o análisis parcelarios no requieren ese alto nivel de exactitud y detalle, sobre todo teniendo en cuenta que el precio de una base de datos gráfica aumenta exponencialmente cuando se incrementa el nivel de resolución (Torres, 2005).

1.2.2 Tipos de datos trabajados en un SIG

Los datos en un Sistema de Información Geográfica pueden ser clasificados en: gráficos y alfanuméricos. Los datos gráficos *son descripciones digitales de las entidades del plano. Suelen incluir las coordenadas, reglas y símbolos que definen los elementos cartográficos en un mapa* (Torres, 2005). El SIG utiliza esos datos para generar un mapa o representación gráfica en una pantalla de ordenador o bien sobre papel. Para la representación de datos gráficos se utilizan tres tipos elementales de entidades:

Nodos o puntos: *Es un objeto sin dimensiones que representa una unión topológica o un punto terminal y que especifica una localización geométrica; en cualquier caso, se trata de la entidad básica para representar entidades con posición pero sin dimensión (al menos a la escala escogida)*. En el formato vectorial se les denomina puntos (Torres, 2005).

Líneas (o arcos): *Son objetos de una dimensión definidos por un nodo inicio y un nodo fin (Torres. 2005).*

Polígonos (o áreas): *Son objetos limitados y continuos de dos dimensiones (Torres. 2005).*

La información geográfica puede estar principalmente en formato raster y vectorial:

Formato vectorial: Es el formato de datos que utiliza puntos, líneas y polígonos para representar la información geográfica. Las unidades vectoriales están caracterizadas por el hecho de que su localización geográfica puede ser definida independientemente y de forma muy precisa, mediante sus relaciones topológicas. Las capas vectoriales son útiles para describir los distintos elementos de terreno, tales como carreteras, red hidrográfica, límites administrativos, y otros (Bover 2007).

Formato raster: La información geográfica también puede estar representada en conjuntos de unidades regulares constituido por celdillas o píxeles, en forma de mosaico. El píxel es la forma más simple y la malla regular en forma de mosaico se conoce como estructura raster. Cada uno de estos píxeles contiene un valor numérico que expresa una determinada característica del terreno en esa localización. Es un formato muy apropiado para la descripción de datos espaciales continuos como altitud, contaminación, temperatura, precipitación *(*Bover 2007).

Los datos alfanuméricos son descripciones de las características de las entidades gráficas. Generalmente son almacenados en formatos convencionales para este tipo de información y se han comenzado a utilizar junto con los sistemas de información geográfica los sistemas de gestión documental, que gestionan estos datos como imágenes gráficas en formato raster. La información alfanumérica y gráfica se encuentra completamente integrada, siendo esta integración, junto con la capacidad de gestión de ambos tipos de datos, lo que caracteriza a los Sistemas de Información Geográfica (Torres. 2005).

En una base de datos de un SIG se puede encontrar dos tipos de información alfanumérica:

Atributos alfanuméricos: Proporcionan información descriptiva sobre las características de las entidades gráficas. Se relacionan con dichas entidades a través de identificadores comunes que se almacenan tanto en el registro alfanumérico como en el gráfico (Torres. 2005).

Datos geográficamente referenciados: Mediante este tipo de datos se describen incidentes o fenómenos que se producen en una localización específica. A diferencia de los atributos estos datos no describen una entidad gráfica sino que proporcionan información (número de edificios permitidos en una zona, número de accidentes en un cruce, inspecciones de salud en un barrio, etc.) asociada a una localización geográfica. Este tipo de datos se almacena y gestiona de forma separada y no se relaciona directamente con las entidades geográficas de la base de datos del SIG (Torres. 2005).

Para mejorar el acceso a la información se establecen normalmente dos tipos de mecanismos:

Índices geográficos: Se utilizan en un SIG para seleccionar, relacionar y recuperar datos en función de su localización geográfica, de forma similar a como actúan los índices en una base de datos tradicional; no constituyen información en sí y únicamente sirven para mejorar los accesos.

Relaciones espaciales: Proporcionan la información sobre las relaciones entre las distintas entidades gráficas, como son la conectividad entre las líneas o la adyacencia en el caso de los polígonos. Esta información va a ser fundamental para determinadas aplicaciones tales como el análisis de redes, ya que proporcionan información sobre las interconexiones de los distintos elementos de la red. Este tipo de relaciones es otro de los aspectos diferenciadores de los SIG, que no suelen encontrarse en otros sistemas gráficos (Torres, 2005).

1.2.3 Modelo de datos

Un modelo de datos es un sistema formal y abstracto que permite representar la información del problema a resolver según reglas y convenios predefinidos. Los más usados en los SIG son:

Modelo relacional: En este modelo la tabla (filas y columnas) es la estructura de datos básica. Se crean diferentes tablas de datos sobre los elementos del territorio que se almacenan por separado en función de criterios temáticos. Las tablas se relacionan mediante indentificadores. Este modelo de datos es muy versátil y permite por ejemplo, realizar consultas según criterios, a varias tablas a la vez *(*Bover, 2007).

Modelo orientado a objetos: Este modelo toma como base la idea de objeto definida en el paradigma de programación orientada a objetos. Ahora, los objetos geográficos, además de unos atributos tienen unas operaciones que los definen. A este modelo se le pueden aplicar todas las características de la programación orientada a objeto como son la herencia, el polimorfismo, etc. *(*Bover, 2007).

1.2.4 Georreferenciación en los SIG

La georreferenciación se puede definir como aquel proceso mediante el cual se identifica una posición en la superficie terrestre (Torres, 2005).

Existen dos tipos de georreferenciación: la georreferenciación indirecta o discreta y la georreferenciación directa, esta última se basa en el uso de un sistema de coordenadas establecido para un determinado sistema de proyección. Su objetivo es resolver el problema de proyectar la superficie curva de la tierra en un sistema plano. Aunque todo sistema de proyección distorsiona la realidad, se puede mantener sin distorsión el área, (proyecciones equivalentes), las distancias (equidistantes) o los ángulos (conformes). Entre los sistemas de proyección globales (válidos en todo el globo terráqueo), el más utilizado es el correspondiente a la proyección UTM (Universal Transversal Mercator), que se obtiene proyectando sobre un cilindro cuya directriz es un meridiano terrestre (a lo largo del cual la distorsión es nula). En este caso, la georreferencia se expresa mediante un identificador de zona y dos coordenadas (x, y) en metros, según los ejes Este-Oste y Norte-Sur.

La georreferenciación indirecta o discreta se fundamenta en asociar al elemento que se representa una clave o índice, normalmente con significado administrativo dirección, código postal, etc.), que puede ser usada para la determinación de una posición, naturalmente con una precisión no siempre equivalente a la obtenida con georreferenciación directa. La virtud de este sistema es poder aprovechar de forma inmediata la gran cantidad de información disponible con georreferenciación directa (Torres, 2005).

1.3 Módulo de administración para SIG.

La disponibilidad y precisión de los datos afectan los resultados de cualquier análisis. Se requieren buenos datos de soporte para que el SIG pueda resolver los problemas y contestar a las preguntas de la forma más acertada posible. De ahí la importancia de una eficiente administración de estos.

1.3.1 Conceptos básicos

En su devenir histórico el conocimiento de administración se ha estructurada con una vasta producción de enunciados entre los cuales se encuentran:

Administración: Es el proceso de diseñar y mantener un entorno en el que, trabajando en grupos, los individuos cumplan eficientemente objetivos específicos. Esta definición básica debe ampliarse:

- Cuando se desempeñan como administradores, los individuos deben ejercer las funciones administrativas de plantación, organización, integración de personal, dirección y control.
- La administración se aplica a todo tipo de organizaciones.
- Se aplica a administradores de todos los niveles organizacionales.
- La administración persigue la productividad, lo que implica eficacia y eficiencia. (Emprendedores UCU n.d.).

Administración: Conjunto ordenado y sistematizado de principios, técnicas y prácticas que tiene como finalidad apoyar la consecución de los objetivos de una organización a través de la provisión de los medios necesarios para obtener los resultados con la mayor eficiencia, eficacia y congruencia; así como la óptima coordinación y aprovechamiento del personal y los recursos técnicos, materiales y financieros. En algunos tratados la dividen en: planificación, organización, dirección y control. Otros consideran cinco etapas del proceso administrativo: prever, organizar, dirigir, coordinar y controlar. (Definición n.d.) .

Administración: Proceso por el cual se mantiene un sistema a punto y operativo. Es una tarea de la que se encarga el administrador y sus posibles colaboradores. Abarca acciones tales como: configurar nuevos dispositivos, administrar cuentas, seguridad del sistema (Código Libre n.d.).

Partiendo de las definiciones de administración antes abordadas y llevándolas a al presente entorno de trabajo, se define administración como un proceso interactivo y dinámico que permite mantener y actualizar de manera constante un sistema con la mayor eficiencia y eficacia, combinando para ello un conjunto de técnicas y prácticas. Comprende acciones tales como: configurar nuevos dispositivos, administrar cuentas y seguridad del sistema, así como gestionar el negocio. La administración puede efectuarse mediante el personal encargado o de manera automática por el propio sistema.

1.4 Situación Problemática del módulo de administración.

En la actualidad a escala internacional, los problemas complejos generados a partir de la representación geoespacial de redes de transporte son resueltos mediantes el uso de herramientas SIG debido a sus grandes ventajas en este campo. Generalmente para una mayor organización los encargados de desarrollar el SIG dividen sus funcionalidades por módulos de acuerdo a las necesidades del negocio, incluyendo entre estos el módulo de administración. Este módulo juega un papel primordial en el funcionamiento del SIG ya que es el encargado de gestionar la información referente al negocio.

Actualmente la Universidad de las Ciencias Informáticas se yergue como una importante organización educacional vinculada al desarrollo de software. Para ejercerse como tal cuenta con una suma considerable de estudiantes y profesores internos y externos a la universidad. Con el objetivo de conservar su intachable reputación y mantener un correcto funcionamiento desde el punto de vista laboral, se dispone de un servicio de transportación para obreros y estudiantes que cuenten con la debida autorización. Para alcanzar un óptimo funcionamiento de este servicio se hace necesaria una correcta gestión de los itinerarios, rutas y paradas ubicadas en todo el territorio de la provincia de La Habana, en la cual se encuentra situada dicha universidad. Los trabajadores y estudiantes viven dispersos en todo el territorio, por lo que por concepto de ahorro de combustible al país, se han establecido puntos de recogidas que abarquen la región de manera que se beneficien con el servicio de transporte todos los implicados en el negocio.

Debido al creciente número de recorridos, rutas, paradas e itinerarios existentes en toda la provincia, al monto de usuarios que se benefician de este servicio y la privación de un sistema capaz de brindar la información necesaria para la gestión de toda la información referente a la transportación de obreros de la UCI, se hace necesario el desarrollo de un SIG que facilite georreferenciar espacialmente y representar en el mapa de la provincia de La Habana todas las rutas situadas en el mismo, que rigen el transporte obrero en la Universidad.

Por tal motivo surge el SIG Rutas que es un sistema capaz de brindar todo tipo de consultas a los usuarios concernientes al negocio del transporte obrero de la UCI,

posibilitando mayor comprensión por parte de los mismos por razón de la representación visual, a partir de la ubicación geográfica en el mapa. También se hace necesario gestionar el negocio, así como todos los datos asociados al mismo por un grupo de administradores, los cuales son los responsables de actualizar la información existente de las rutas y paradas de acuerdo a las necesidades de los trabajadores de transportación de la UCI.

Para dar respuesta a esta necesidad se hace inevitable el desarrollo de un módulo de administración para este SIG. EL módulo debe ser capaz de gestionar la seguridad del sistema de manera que los usuarios solo tengan acceso a las funcionalidades y a la información que se le sean accedidas. Debe permitir a los administradores cambiar cualquier dato referente al negocio, de tal modo que el SIG pueda actualizar su base de datos de manera constante logrando así una renovación de la información de manera fácil y eficaz.

1.5 Estado del arte de los módulos de administración para SIG.

A escala nacional e internacional son muchos los SIG que disponen de un módulo de administración ajustado a sus respectivas necesidades y a la complejidad de los datos que se manejan. En el presente epígrafe se hace un análisis de dichos módulos para buscar una base que sirva de apoyo para proponer la solución de la presente investigación, o comprobar si existe alguna semejante que permita enfrentar al problema científico planteado.

1.5.1 Soluciones existentes en Cuba

SAUXE es un paquete de soluciones integrales de gestión para las entidades presupuestadas y empresariales basada en los principios de independencia tecnológica y con funcionalidades generales de los procesos y las particularidades de la economía cubana. Ha sido desarrollado por estudiantes, profesores y especialistas de diversos organismos y entidades del país en la Infraestructura Productiva de la Universidad de las Ciencias Informáticas. (Informáticas, 2008).

SAUXE surgió con el desarrollo de los estándares para integrar aplicaciones de negocios ERP (Planificación de Recursos Empresariales) que están cada vez más inclinados a la integración con los Sistemas de Información Geográfica, ya que estos últimos ayudan a administrar los activos físicos de una organización y las compañías necesitan cada vez más adicionar información geoespacial sobre la empresa. Mediante el uso de este módulo las entidades pueden gestionar los procesos contables y financieros, logísticos, de administración, de capital humano, entre otros procesos administrativos y de toma de decisiones. Sus características funcionales fueron diseñadas de acuerdo con las políticas emitidas por los organismos rectores del Estado Cubano.

SAUXE pone al servicio de su entidad las potencialidades de la tecnología informática y provee facilidades para la integración de las diferentes áreas productivas y departamentos administrativos. Este módulo cuenta con varias aplicaciones, dentro de ellas la de seguridad, basada en la triple A (autenticación, autorización, Traceabilidad (Accounting)). En ella se gestiona el dominio, los roles y los usuarios donde se le asigna a cada dominio un rol y este a su vez se le asigna a un usuario. El rol cuenta con las funcionalidades a la cual va a tener acceso el usuario. El SAUXE también gestiona las aplicaciones con la cual el sistema se va a integrar, incluyendo las funcionalidades y las acciones que estas realizan y se le asignan estas funcionalidades al rol para limitar el acceso a la aplicación.

1.5.2 Soluciones existentes en el mundo

SIG GUADUA es un Sistema de Información Geográfica que ofrece datos generales del recurso guadua[1] y está conformado por nodos en tres departamentos de la Ecoregión integrado por sus Corporaciones Autónomas Regionales y el Sistema de Información Regional SIR de la Universidad Tecnológica de Pereir. Guadua es un sistema distribuido que busca satisfacer de manera eficiente, oportuna y confiable la demanda de datos e información georreferenciada del recurso guadua, cumpliendo con los estándares para la información geográfica existente y garantizando la participación efectiva de diferentes actores en el tema de la guadua.

[1] La Guadua es una gramínea y se considera y clasifica como un bambú leñoso que se encuentra ampliamente distribuido en América Latina desde la zona central del sur de México hasta la Argentina. Es una de las especies forestales más utilizadas en el proceso de construcción de viviendas, edificaciones, estructuras como puentes, cercos, puertas, conducción de aguas, etc.

SIG GUADUA cuenta con un módulo de administración para los usuarios delegados por las corporaciones, para ello se creó una aplicación llamada Sistema para la Consulta y Edición de Información del SIG-GUADUA (SASG). Esta aplicación funciona completamente vía Web y permite a los usuarios autorizados a usarla, editar y actualizar toda la información que existe en la base de datos geográfica que sustenta al SIG Guadua. Cabe mencionar que solo quienes posean un nombre de usuario y una clave de acceso secreta podrán ingresar al SASG para editar información (SIG GUADUA n.d.).

SIG para la gestión de obras y servicios de proyectos comunales en la República Bolivariana de Venezuela. Este sistema es una herramienta capaz de conseguir información, consultar, editar y generar reportes de información de manera rápida y sencilla, llevando un óptimo control y seguimiento de las obras y servicios a partir de una cartografía delimitada. Es una propuesta que contribuye al mejoramiento y calidad de proyectos comunitarios enmarcados en este ámbito.

El sistema está integrado por varios módulos, dentro de ellos el módulo de administración: que es la parte alfanumérica que permite cambiar cualquier dato de las comunicaciones de obra. Permitiendo de igual forma realizar cambios de datos alfanuméricos desde la parte gráfica, así como editar cualquier entidad de obra de cualquier capa del sistema. De esta manera el SIG puede actualizarse y extender su base de datos, desarrollándose con el uso de una hoja de registro en Excel, Dbase, Access y cualquier otro manejador. En este sentido es posible la constante modernización de la información de manera fácil, eficaz e interactiva, generando mapas temáticos (Juan, 2009).

El SIG para la Gestión de Rutas en Caminos no Cartografiados permite ampliar la cartografía estándar del territorio Catalán con las rutas que han seguido, por caminos no cartografiados, los vehículos de una empresa veterinaria (Bover, 2007). Este SIG para la organización de la información dispone de tres bases de datos que se crearon y gestionan desde GeoMedia (*GeoMedia es un SIG, creado por Intergraph Corporation, para los sistemas operativos Windows 2000, Windows XP y posteriores. Este software es una herramienta de examen, análisis, captura y mantenimiento de datos*.), aunque algunas de las funciones de gestión que requiere el usuario como la gestión de rutas, se implementaron con Visual Basic y fueron introducidas como nuevos comandos de

GeoMedia. Además, al ser un tipo de almacén Access, utilizan el gestor de bases de datos Microsoft Access que es *un sistema de gestión de bases de datos para uso personal o de pequeñas comunidades y empresas en las que el número de usuarios, que acceda simultáneamente a la base de datos, no debería superar la decena aproximadamente* (Bover 2007).

Conclusiones

Después del análisis de las soluciones existentes se concluye que a pesar de la eficiencia que pueden tener estas en sus respectivos entornos, no son viables para ser utilizadas en la solución del presente problema. La principal desventaja que muestran estos módulos es que exceptuando al SAUXE, no se encuentran bajo los principios de independencia tecnológica, característica fundamental que debe cumplir el módulo de administración para el SIG Rutas. Además estas soluciones no cuentan con la transparencia, las funcionalidades y la seguridad requerida. El SAUXE cumple parcialmente, pero no abarca todas las acciones que debe tener dicho módulo. Presenta errores que limitan el funcionamiento óptimo del sistema y no incluye la gestión del negocio. Por lo que para dar solución al problema que dio origen a la investigación se decidió realizar una personalización del SAUXE, adaptándolo a las condiciones del SIG Rutas. A partir del resultado del estudio abordado en este capítulo, se puede proceder a seleccionar las tecnologías necesarias y la metodología a utilizar para el desarrollo del sistema.

Capítulo 2: “Tecnologías utilizadas para la confección del sistema”

2.1 Introducción

En la actualidad existe una gran variedad tecnológica, que ofrecen excelentes posibilidades para que grandes y medianas empresas puedan dar soluciones efectivas a sus necesidades. En el capítulo se fundamentan las tecnologías utilizadas para la construcción del sistema que dará solución a la problemática que dio origen a la investigación. Este grupo de tecnologías lo conforman: la metodología para guiar el proceso de desarrollo del sistema, el lenguaje de modelado capaz de proporcionar el entendimiento del proceso de desarrollo a través de diagramas, las herramientas CASEs que soporten dicho modelado, el gestor de base de datos escogido, el lenguaje de programación, el entorno de desarrollo y el framework que soportará la aplicación.

2.2 Metodologías Existentes

Con el propósito de evitar pérdidas de tiempo y recursos, el proceso de desarrollo del software debe estar organizado o estructurado de forma correcta y disciplinada. Es preciso definir una estrategia para darle un orden a las tareas posibles a desarrollar, así como también llevar a cabo una guía de cómo efectuar las actividades. En fin, efectuar un *conjunto de procedimientos y pasos que se deben seguir para el desarrollo de un software, constituyendo los mismos una metodología de desarrollo de software* (Ortiz Fidalgo y Rios Morales 2010). Existen disímiles metodologías de desarrollo, todas ellas conforman dos grandes corrientes referentes a los procesos de desarrllo de software: las metodologías robustas o tradicionales y las metodologías ágiles.

Las **metodologías ágiles** son apropiadas para guiar proyectos de poco volumen, que requieran una rápida implementación. Dentro de estas metodologías se encuentran Agile Unified Process (AUP) y Extremme Programming (XP). Esta última, es una metodología centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. La misma tiene como herramienta principal, el desarrollo orientado a pruebas, que utiliza

las pruebas unitarias como eje de todo desarrollo. Las interacciones suelen ser muy cortas y se promueve a los programadores a buscar soluciones y experiencia con ellas, programar sin miedo a descomponer el sistema. AUP es una versión simplificada del Rational Unified Process (RUP), descrito en una forma simple, fácil de entender y brinda un enfoque de desarrollo de software utilizando técnicas ágiles y conceptos de RUP.

Las **metodologías robustas** pueden ser empleadas para guiar el proceso de desarrollo de proyectos grandes o pequeños, aunque son más apropiadas para proyectos grandes que por su importancia requieren una intensa etapa de análisis y diseño antes de la construcción del sistema. Ejemplo RUP: que es un proceso de desarrollo de software, que ofrece un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software. Está basado en componentes, dirigido por casos de uso y centrado en la arquitectura. El Proceso Unificado se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Cada ciclo concluye con una versión del producto para los clientes. Cada ciclo consta de cuatros fases, **Inicio, Elaboración, Construcción y Transición**. Cada fase se subdivide a su vez en iteraciones donde en cada una se realiza una evaluación para determinar si los objetivos de la fase fueron cumplidos.

2.2.1 Metodología de desarrollo seleccionada. Agile Unified Process (AUP)

El sistema a desarrollar es un sistema de poco volumen, que requiere una rápida implementación, en un corto y limitado tiempo por la necesidad que existe actualmente por parte de los administradores del SIG Rutas. Además, este proyecto es de vital importancia para lograr una correcta gestión y seguridad de los datos del SIG Rutas. Por todo lo antes expuesto se requiere una metodología capaz de guiar el proceso con precisión, que contribuya a obtener un software fiable, sin errores, se necesita una metodología que combine fortaleza y rapidez.

Para facilitar la selección de la metodología a emplear en la realización de la investigación, se efectuó una comparación entre las metodologías antes mencionadas.

	AUP	XP	RUP
Dirigido por casos de uso.	SI	NO	SI
Desarrollo iterativo e incremental.	SI	NO	SI
Participación activa del usuario.	NO	SI	NO
Adaptación del proceso.	SI	SI	SI
Centrado en la arquitectura.	SI	NO	SI

Tabla 1 Comparación entre AUP, XP, RUP

Teniendo en cuenta las necesidades del sistema a desarrollar y el resultado optenido de la tabla, se seleccionó para guiar el proceso la metodología Agile Unified Process (AUP). Esta se ajusta a las particularidades del proyecto porque combina características de la metodología ágil XP con los artefactos de RUP. No se podría elegir XP, porque el equipo de desarrollo no tiene experiencia en el trabajo con esta metodología y la misma precisamente se basa en la capacidad y madurez de los miembros del equipo. Además debido al bajo número de documentación a generar la misma se convierte en un proceso muy orientado a la programación. La metodología RUP serviría si se dispusiera de más tiempo para el desarrollo del sistema. Según Ivar Jacobson, Grady Booch y James Rumbaugh, en el libro “El Proceso Unificado de Desarrollo”, Capítulo 1, el ciclo de vida de ésta metodología es iterativo e incremental, que supone un gran esfuerzo que puede durar entre varios meses hasta posiblemente un año o más.

Dentro de las características particulares de AUP, se tiene que es una versión simplificada de la metodología RUP. La Fig. 2.1 muestra el ciclo de vida de esta metodología. AUP abarca siete flujos de trabajos, cuatro ingenieriles y tres de apoyo: Modelado, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyectos y Ambiente. El modelado agrupa los tres primeros flujos de RUP (Modelamiento del negocio, Requerimientos y Análisis y Diseño). Dispone de cuatro fases igual que RUP: Creación, Elaboración, Construcción y Transición.

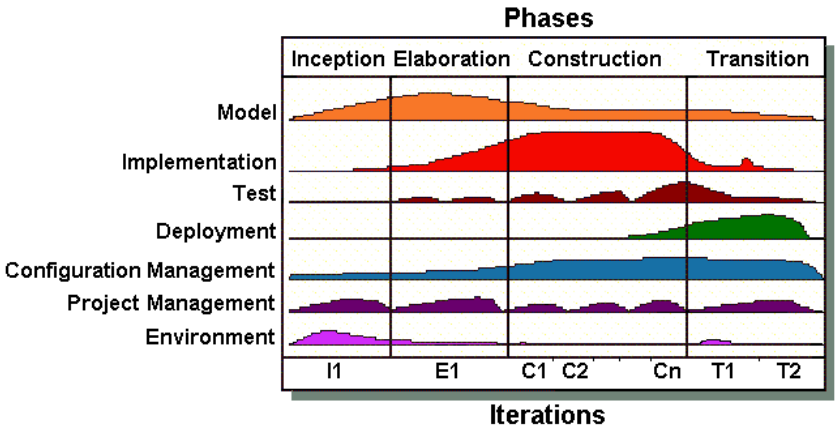


Fig 1 Esquema que muestra los flujos de trabajo y las fases de AUP.

El Modelado es el flujo de trabajo que tiene el objetivo de entender el negocio de la organización, el problema de dominio que se aborda en el proyecto y determinar una solución viable para resolver el problema de dominio. El flujo de trabajo Implementación tiene como objetivo transformar su (s) modelo (s) en código ejecutable y realizar un nivel básico de las pruebas, en particular, la unidad de pruebas. El flujo de trabajo de Prueba tiene como objetivo realizar una evaluación objetiva para garantizar la calidad. Esto incluye la búsqueda de defectos, validar que el sistema funciona tal como está establecido, verificando que se cumplan los requerimientos. Por último dentro de los flujos de trabajo ingenieriles se tiene el Despliegue, cuyo objetivo es el plan para la prestación del sistema y la ejecución de dicho plan, para que el sistema quede a disposición de los usuarios finales (Rdoriguez Donatien y Ramirez Martín 2009).

Esta versión ágil de la metodología RUP se basa en los siguientes principios:

- ***Simplicidad:*** Todo se describe concisamente utilizando poca documentación, no miles de ellas.

- ***Agilidad:*** El ajuste a los valores y principios de la Alianza Ágil[2].

- ***Centrarse en actividades de alto valor:*** La atención se centra en las actividades que en realidad lo requieren, no en todo el proyecto.

[2] La Alianza Ágil es una organización sin fines de lucro con membrecía global, comprometida con el avance del desarrollo ágil de principios y prácticas. Apoya a aquellos que aplican los principios ágiles y prácticas, para que el desarrollo del software sea más productivo, humano y sostenible.

- ***Herramienta de la independencia:*** Se puede usar cualquier conjunto de herramientas que se desee con el AUP. Se sugiere utilizar las herramientas más adecuadas para el trabajo, que a menudo son las herramientas simples o incluso herramientas de código abierto.

- ***Usted podrá adaptar este producto para satisfacer sus propias necesidades:*** La metodología AUP es un producto de fácil uso utilizando cualquier herramienta. No es necesario comprar una herramienta especial, o tomar un curso, para adaptar esta metodología.

2.3 Lenguaje de modelado y Herramientas.

El proceso de desarrollo de software ha tenido un gran avance en los últimos tiempos y las herramientas de modelado forman un componente muy significativo en el entorno de desarrollo, puesto que son esenciales para el análisis de sistemas. Las herramientas mejoran la forma en que ocurre el desarrollo y tiene influencia sobre la calidad del resultado final.

En la actualidad se cuenta con una serie de herramientas que se han creado con este fin. Algunas han sido mejoradas por parte de sus desarrolladores con el propósito de encontrar en ellas fiabilidad y eficiencia, entre las más comunes se encuentra: Visual Paradigm y Rational Rose Enterprise Edition. Estas herramientas usan el Lenguaje de Modelado UML para la especificación, la visualización, la construcción y la documentación de los artefactos de los sistemas de software.

2.3.1 Lenguaje de Modelado. UML V2.0

Es de vital importancia que tanto los programadores y analistas como los clientes y cualquier otra persona interesada, sean capaces de entender en un lenguaje visual y común el funcionamiento del sistema a partir de los esquemas. Además es necesaria la documentación de todo el ciclo de desarrollo del sistema, con vista a facilitar el desarrollo de futuras versiones o mejoras del mismo. Por lo que se requiere de un lenguaje de modelado que cubra estas necesidades.

UML (por sus siglas, del inglés **Unified Modeling Language**) es el lenguaje estándar de modelado para software. Permite la visualización, especificación, construcción y

documentación de los artefactos generados a través de diagramas exponiendo el significado de los mismos, así como las relaciones entre los diferentes componentes y objetos de manera estandarizada. Como dijeran Booch, Rumbaugh, y Jacobson en el libro “El Lenguaje Unificado de Modelado”, “UML es un lenguaje visual de modelado para visualizar, especificar, construir y documentar los artefactos de un sistema software”. El uso de UML para el modelado del sistema facilitó visualizar de manera fácil y abstracta las funciones del mismo, los procesos de negocio y los esquemas de bases de datos con los cuales el sistema interactúa.

2.3.2 Herramienta CASE. Visual Paradigm for UML 6.4

Durante la realización del proyecto la herramienta de modelado que se utilizó fue el Visual Paradigm pues a pesar de ser una herramienta privativa, la UCI paga la licencia de la misma. EL uso de esta herramienta permitió crear tipos diferentes de diagramas en un ambiente totalmente visual. Soportó el ciclo de vida completo del desarrollo del software: modelado, implementación, pruebas y despliegue. Además posibilitó acelerar la producción y mejorar la calidad del trabajo. No se utilizó Rational Rose Enterprise Edition porque a pesar de ser una herramienta muy utilizada es un software propietario y no es multiplataforma, al contrario de Visual Paradigm For UML Enterprise Edition que es una herramienta multiplataforma, característica que favorece al desarrollo de un producto que no se limita a ejecutarse sobre un sólo sistema operativo. Además algunas de sus versiones son gratuitas. A continuación se abordan las características fundamentales de Visual Paradigm que tributaron al desarrollo del sistema.

Visual Paradigm for UML es una herramienta CASE que soporta hasta la versión 2.1 de UML. Permite modelar los procesos del negocio, la base de datos y las clases del sistema de manera visual y soporta patrones de diseño para lograr mejores prácticas. Presenta gran usabilidad, porque los diagramas se agrupan por categorías, permitiendo al usuario una rápida localización de la información. Permite la configuración de los estilos y formatos de los diagramas incorporando imágenes y estereotipos. Así como la exportación de los diagramas en formato de imagen. Ofrece amplias características de modelado de casos de uso incluyendo la función completa de UML, diagrama de casos de uso y editor de flujo de eventos. También ofrece bondades como la creación de diagramas para UML 2.0. Cuenta con un diseño

centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad. Es muy sencillo de usar, fácil de instalar y actualizar.

2.4 Lenguaje de programación PHP 5

PHP (acrónimo de Hypertext Preprocessor) es un lenguaje "del lado del servidor" (esto significa que PHP funciona en un servidor remoto que procesa la página Web antes de que sea abierta por el navegador del usuario) especialmente creado para el desarrollo de páginas Web dinámicas. Este puede ser incluido con facilidad dentro del código HTML y permite una serie de funcionalidades extraordinarias.

PHP está dotado de un gran número de funciones que simplifican enormemente tareas habituales, como descargar documentos, enviar correos, trabajar con cookies, sesiones, etc. Dispone de una amplia gama de librerías y una conexión propia a todos los sistemas de base de datos. Además posee Licencia Open Source de Código Abierto que implica que el código fuente de PHP 5 es libre de ser descargado y completamente inspeccionado, puede ser utilizado, modificado y redistribuido sin coste alguno. Actualmente se encuentra en su versión estable 5.3.5, que incorpora una verdadera orientación a objetos y añade un control de errores muy mejorado, al estilo de los lenguajes de programación más avanzados.

Para un desarrollo eficiente del sistema a implementar se decidió usar PHP 5 como lenguaje de programación, por estar debidamente documentado, y porque no se necesita tecnología con gran capacidad de memoria para el desarrollo con el mismo. Es un lenguaje muy fácil de aprender y con su utilización se pueden programar de forma eficaz todas las funcionalidades que le dan solución al problema en cuestión.

2.5 Sistema Gestores de Bases de Datos.

Un Sistema Gestor de Bases de Datos (SGBD) permite: definir, crear y mantener las bases de datos. Además de minimizar las redundancias, garantiza la integridad, la seguridad y protección de los datos. Proporciona la manipulación de la información, permite un control centralizado, así como un acceso controlado a estas y un lenguaje de manejo de datos para la inserción, actualización, eliminación y consulta de información en estas bases de datos (Ortiz Fidalgo y Rios Morales 2010). Entre los

SGBD más comunes se encuentra Oracle, Microsoft SQL Server, MySQL y PostgreSQL.

2.5.1 Gestor de BD PostgreSQL

PostgreSQL es un servidor de base de datos de objeto relacional libre, el cual está liberado bajo la licencia BSD, es decir, que puede ser utilizado, modificado y distribuido por todo el mundo, libre de cargo para cualquier propósito, sea comercial, privado o académico. Se considera uno de los sistemas gestores de bases de datos de código abierto más avanzada del mundo.

PostgreSQL se ejecuta en casi todos los principales sistemas operativos: Linux, Unix, BSDs, Mac OS, Windows, entre otros. Posee un soporte nativo para los lenguajes más populares del medio: PHP, C, C++, Perl, Python, entre otros. PostgreSQL acerca los datos a un modelo objeto-relacional y es capaz de manejar complejas rutinas y reglas. Además soporta casi toda la sintaxis SQL (incluyendo sub-consultas, transacciones y funciones definidas por el usuario). Permite métodos almacenados, restricciones de integridad, vistas, etc. Aporta potencia y flexibilidad adicional como son las restricciones (constraints), disparadores (triggers), reglas (rules) e integridad (Ortiz Fidalgo y Rios Morales 2010).

Para el desarrollo del sistema se escogió PostgreSQL, pues es un potente motor de bases de datos, que tiene prestaciones y funcionalidades equivalentes a muchos gestores de bases de datos comerciales. Además PostgreSQL es más completo que MySQL ya que permite métodos almacenados, restricciones de integridad, vistas, etc. A pesar que en las últimas versiones de MySQL se han hecho grandes avances en ese sentido y de que MySQL posee diversas ventajas (alto rendimiento, su bajo coste, su facilidad de configuración y aprendizaje, su accesibilidad a código fuente). PostgreSQL es un sistema de bases de datos de mayor nivel que MySQL, a la altura de Oracle, Sybase o Interbase. También PostgreSQL ofrece una garantía de integridad en los datos mucho más fuerte que MySQL.

Otra característica fundamental para el sistema a desarrollar de PostgreSQL es su módulo PostGIS, debido que el sistema a desarrollar trabaja con la base de datos del SIG Rutas. Este módulo añade soporte de objetos geográficos a la base de datos

objeto-relacional PostgreSQL, convirtiéndola en una base de datos espacial para su utilización en Sistemas de Información Geográfica. Se publica bajo la licencia pública general de GNU. En la actualidad es un producto veterano que ha demostrado a través de la evolución de sus versiones la eficiencia alcanzada. Ha demostrado ser muy superior a la extensión geográfica de la nueva versión de MySQL y a juicio de muchos, es muy similar a la versión geográfica de la archiconocida Oracle. PostGIS ha sido certificado en 2006 por el Open Geospatial Consortium (OGC) lo que garantiza la interoperabilidad con otros sistemas. Utiliza formatos como el WKB (Well-Known Binary), aunque hasta la versión 1.0 se utilizaba la forma WKT (Well-Known Text).

2.6 Librerías a utilizar Ext JS 2.2

Ext JS es una librería Javascript ligera y de alto rendimiento, compatible con la mayoría de navegadores para crear páginas web y aplicaciones dinámicas, usando tecnologías como AJAX, DHTML y DOM. Proporciona una interfaz de usuario consistente y dispone de un conjunto de componentes (widgets) para incluir dentro de una aplicación web como: cuadros y áreas de texto, campos para fechas, campos numéricos, combos, checkboxs, editor HTML, elementos de datos (con modos de sólo lectura, datos ordenables, columnas que se pueden bloquear y arrastrar, etc.), árbol de datos, etc. Varios de estos componentes están capacitados para comunicarse con el servidor usando AJAX. De igual forma contiene numerosas funcionalidades que permiten añadir interactividad a las páginas HTML, como: cuadros de diálogo, quicktips para mostrar mensajes de validación e información sobre campos individuales. Ext JS es muy flexible y permite realizar de manera rápida interfaces consistente. La comunidad que está detrás de esta herramienta es muy grande y la documentación cada vez es más extensa. Además cuenta con varias licencias que se pueden utilizar de acuerdo al proyecto que se esté realizando (SG s.f.).

Para el desarrollo del sistema se escogió Ext JS como librería debido a todas las características antes expuesta, y a que la misma cuenta con una API fácil de usar y su Licencia es Open Source y Comercial. Conjuntamente provee una experiencia consistente sobre cualquier navegador, evitando el tedioso problema de validar que el código escrito funcione bien en cada uno de ellos. También existe un balance en la arquitectura Cliente – Servidor, permitiendo que el servidor, al tener menor carga,

pueda manejar más clientes al mismo tiempo, características fundamentales que tributa al sistema a desarrollar. Además el hecho de que la comunicación sea asíncrona (comunicación en tiempo diferido) le da la libertad al sistema de cargar información sin que el cliente se percate.

2.7 Servidor Web Apache 2.0.

Un servidor web sirve contenido estático a un navegador, carga un archivo y lo sirve a través de la red al navegador de un usuario. Este intercambio es mediado por el navegador y el servidor que hablan el uno con el otro mediante HTTP. Se pueden utilizar varias tecnologías en el servidor para aumentar su potencia más allá de su capacidad de entregar páginas HTML, éstas incluyen scripts CGI, seguridad SSL y páginas activas del servidor (ASP) entre otros.

Apache, es un servidor web hecho por excelencia, su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. Es una tecnología gratuita de código fuente abierto y multiplataforma. Es altamente configurable y de diseño modular, lo que permite ampliar las capacidades del servidor Web Apache de manera sencilla. Permite el trabajo con PHP y otros lenguajes de script, además tiene todo el soporte que se necesita para tener páginas dinámicas. Apache permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Se puede configurar, para que ejecute un determinado script cuando ocurra un error en concreto.

Apache es un servidor que puede ser adaptado a diferentes entornos y necesidades, además goza de diferentes módulos de apoyo y con la API de programación de módulos. De esta forma, la licencia de Apache es una descendiente de la licencias BSD (Berkeley Software Distribution), lo que permite la distribución de derivados de código abierto y cerrado a partir de su código fuente original. Partiendo de las características antes expuesta se escogió como Servidor Web, el Apache 2.0. Este satisface las particularidades que debe cumplir el Servidor Web, para desarrollar el sistema que dará solución a la problemática que dio origen a la investigación.

2.8 Zend Framework 1.0

Zend Framework (ZF) es un framework open source para PHP. Implementa el patrón MVC, es 100% orientado a objetos y sus componentes tienen un bajo acoplamiento por lo que los puedes usar en forma independiente. Un punto importante es que brinda un óptimo estándar de codificación. A su vez, cuenta con soporte para internalización y localización de aplicaciones (construir sitios multi_idioma, convertir formatos de fechas, monedas, etc.) según la región. Facilita el setup del proyecto brindando herramientas para crear la estructura de directorios, clases, por línea de comandos. Admite la integración con PHP Unit por medio de Zend_Test para facilitar el testing (Las pruebas de software) de las aplicaciones y posee adapters (El patrón Adapter se utiliza para transformar una interfaz en otra, de tal modo que una clase que no pudiera utilizar la primera, haga uso de ella a través de la segunda.) para gran cantidad de tipos de bases de datos. También brinda componentes para la autenticación y autorización de usuarios, envío de mensajes, creación de servidores web, etc (Maestrosdelweb s.f.).

Zend Framework se escogió como marco de trabajo para el desarrollo del sistema debido que el mismo permite la integración con el de Doctrine a través del componente Zend_Model que es una extensión realizada a Zend por grupo de la UCID y el equipo de arquitectura del proyecto ERP-Cuba, la cual encapsula todos los temas de conexión a la base de datos. Además ZF se integra con el Sistema Integral de Gestión de Seguridad (SIGES) a través del componente Zend_Security que es otra de las extensiones realizadas al framework de Zend. Este componente actúa como interfaz entre el framework de Zend y el SIGES, que es el que brinda los servicios de autenticación, autorización, auditoría, administración de conexiones y administración de perfiles.

2.9 Zend Studio for Eclipse

El IDE Eclipse es un entorno de desarrollo para crear aplicaciones clientes de cualquier tipo, se distribuye bajo licencia EPL (Eclipse Public License). Esta licencia es considerada como libre por la Free Software Foundation (FSF) y por la Open Source Initiative (OSI). La licencia EPL permite usar, modificar, copiar y distribuir nuevas versiones del producto licenciado. Eclipse está formado por su propio núcleo y por una variedad de plugins que yacen sobre él, este tipo de arquitectura le proporciona una flexibilidad y escalabilidad muy elevada. Todo componente con funcionalidad es

tratado como un plugin en Eclipse con neutralidad y adaptabilidad a cualquier tipo de lenguaje, por ejemplo C/C++, Cobol, C#, XML, etc.

Eclipse abarca todos los componentes de desarrollo necesarios que facilitan la extensibilidad de PHP. Proporciona funcionalidades tales como el coloreado de la sintaxis, asistente de código, plegado de código e integración con el modelo de proyecto. Inspecciona el código a través de las vistas outline del proyecto de fichero y la nueva vista de Explorador de PHP. Ayuda para eliminar errores de código de PHP adicionales. Permiten a los desarrolladores extender fácilmente la creación de nuevas herramientas de desarrollo orientadas a PHP. Cuenta con capacidades de desarrollo de PHP de Eclipse multi-idioma de apoyo y complemento en la extensión de tecnología.

Para una implementación eficiente de la aplicación planteada se escogió para desarrollar la misma el IDE Eclipse. Debido a que este proporciona todas las funcionalidades necesarias para aumentar la robustez de la aplicación a implementar, conjuntamente disminuye el ciclo de tiempo de la producción y le resulta fácil de usar a los programadores. La integración que provee de Zend Studio con Zend Framework está también en el contexto de las mejores prácticas.

2.10 Doctrine (ORM)

Un ORM o (Object Relation Mapper) es una técnica de programación que permite convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional. Es decir, las tablas de la base de datos pasan a ser clases y los registros pasan a ser objetos que se pueden manejar con facilidad.

Utilizar un ORM tiene una serie de ventajas que facilitan enormemente tareas comunes y de mantenimiento. Por ejemplo brindan una alta capacidad para la r**eutilización** (Permite llamar a los métodos de un objeto de datos desde distintas partes de la aplicación e incluso desde diferentes aplicaciones), **encapsulación** (La capa ORM encapsula la lógica de los datos pudiendo hacer cambios que afectan a toda la aplicación únicamente modificando una función), **portabilidad** (Utiliza una capa de abstracción que permite cambiar en mitad de un proyecto de una base de datos

MySQL a una Oracle, sin ningún tipo de complicación), **seguridad** (Los ORM suelen implementar mecanismos de seguridad que protegen a la aplicación de los ataques más comunes como inyecciones SQL).

Doctrine es un ORM para PHP 5 y posterior, se sitúa en la cima de una potente base de datos de capa de atracción (DBAL). Aparte de todas las ventajas que conlleva un ORM, uno de sus puntos fuertes es su lenguaje DQL (Doctrine Query Language). DQL es un lenguaje de consulta de objeto que permite formular consultas a objetos individuales o a una colección completa de objetos, utilizando la terminología de su modelo de dominio: la clase nombres, nombres de campo, las relaciones entre las clases, etc. Se trata de una herramienta de gran alcance para recuperar y manipular objetos sin romper la separación del modelo de dominio (nombres de campos, nombres de clase, etc) a partir del modelo relacional (nombres de tablas, nombres de columnas, etc).

Doctrine proporciona una alternativa SQL a los programadores que les permite mantener flexibilidad, sin requerir duplicación de código innecesario. Es una herramienta fácil de aprender que goza de gran popularidad entre los desarrolladores. También posee un proceso de de desarrollo continuo donde siempre se le adicionan nuevas características que les proporcionan más libertad en el modelado del campo.

Doctrine se escogió como ORM para desarrollar el sistema debido a la facilidad que proporciona el mismo para el manejo de los registros de la base de datos, permitiendo la reutilización de los métodos y una mayor organización desde el punto de vista de la arquitectura. También proporciona la emigración de manera sencilla a otras bases de datos en caso de posibles versiones del producto. Un punto significativo es que el uso de sentencias SQL mitiga la exposición a los ataques de inyecciones SQL garantizando la seguridad de los datos que maneja el sistema. Además es muy fácil de usar y de aprender, característica primordial debido a la falta de experiencia de los programadores.

Conclusiones

Teniendo en cuenta que para la selección de las tecnologías, se tuvo como principales elementos a considerar: la compatibilidad con el sistema operativo Linux y que fueran

herramientas y tecnologías libres, o de código abierto, se puede llegar a la conclusión de que se obtendrá un producto multiplataforma y totalmente libre. Para el mismo se definió AUP como metodología de desarrollo, UML v2.0 como lenguaje de modelado para especificar, construir y documentar los artefactos del sistema, Visual Paradigm v4.6 como herramienta CASE. El lenguaje a utilizar es PHP 5, utilizando Zend como Framework, Doctrine como ORM, el servidor será APACHE, ambiente de desarrollo Zend Studio for Eclipse y el sistema gestor de base de datos PostgreSQL.

Capítulo 3: “Presentación de la solución propuesta”

3.1 Introducción

En este capítulo se hace una descripción de la solución propuesta a través del modelo de dominio representando los principales conceptos asociados al sistema. Se detallan los requisitos funcionales y no funcionales del sistema, casos de uso que se generan a partir de los requisitos funcionales y una explicación de cada uno de ellos a través de las descripciones textuales.

3.2 Modelo de Dominio

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las “cosas” que existen o los eventos que suceden en el entorno en el que trabaja el sistema (Serrano Rosales y Rodríguez Vázquez 2008). Las clases del dominio aparecen en tres formas típicas:

- Objetos del negocio que representan conceptos que se manipulan en el negocio.
- Objetos del mundo real y conceptos de los que el sistema debe hacer un seguimiento.
- Sucesos que ocurrirán o han ocurrido.

El modelo de dominio se describe mediante diagramas UML (especialmente mediante diagrama de clases). Estos diagramas muestran las clases del dominio y cómo se relacionan unas con otras mediante asociaciones. El modelo de dominio ayuda además a los usuarios, clientes, desarrolladores y otros interesados a utilizar un vocabulario común.

El Modelo de Dominio se aplica cuando:

- Los flujos de información son difusos (múltiples orígenes, sólo eventos, sucesos).

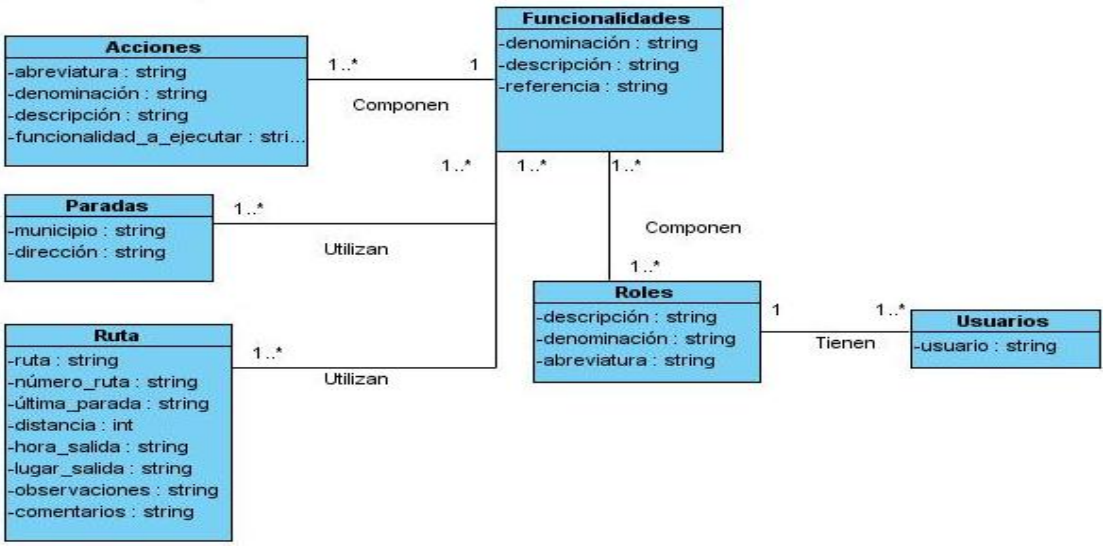
- Imposibilidad de determinar subsistemas (exceso de interconexiones).
- Solapamiento de responsabilidades.
- Múltiples responsabilidades.
- Difícil establecimiento de reglas de funcionamiento.

3.2.1 Diagrama de Clases del Modelo de Dominio

Después de un estudio de todo lo antes planteado y en base a un minucioso análisis de las condiciones del sistema a desarrollar, se llegó a la conclusión de que:

Al no tener una estructura bien definida de los procesos del negocio (fronteras bien establecidas, donde se logren ver claramente quiénes son las personas que lo inician, y a su vez que actividades realizan en cada uno de estos procesos), se planteó un modelo de dominio para que todo el interesado adquiera un entendimiento mínimo del contexto en que se emplaza el Sistema de Administración del SIG Rutas.

El mismo se realizó a través de un diagrama de clases de UML e identificando los conceptos representados en el diagrama en un glosario de términos.



Las **Funcionalidades** están compuestas por una o varias **Acciones** y estas a su vez, pertenecen a una única funcionalidad. El sistema por cuestiones de seguridad le asigna una o varias funcionalidades a un **Rol** determinado, que de igual manera le es asignado a un **Usuario** de acuerdo a los permisos que al mismo se le hayan conferido. Las funcionalidades también engloban la parte correspondiente a la gestión del negocio, por lo que utilizan la información socioeconómica relacionada a las **Paradas** y **Rutas** del transporte obrero de la UCI. A continuación se especifica la respectiva descripción de los conceptos asociados.

Funcionalidades: Conjunto de acciones que debe permitir realizar el sistema.

Acciones: Especificación de cada una de las acciones que debe tener una determinada funcionalidad.

Usuarios: Personas o sistemas que se benefician del sistema informático que posibilita administrar el SIG Rutas.

Roles: Engloban un conjunto de funcionalidades que se le asigna a un determinado usuario.

Paradas: Información socioeconómica relacionada a las paradas del transporte obrero de la UCI, en este caso es un conjunto de datos ubicados en la Base de Datos del SIG Rutas.

Rutas: Información socioeconómica relacionada a las rutas del transporte obrero de la UCI, en este caso es un conjunto de datos ubicados en la Base de Datos del SIG Rutas.

3.3 Requerimientos

A través de los años se ha podido confirmar que los requerimientos o requisitos son la pieza fundamental en un proyecto de desarrollo de software, ya que marcan el punto de partida para actividades como la planeación, básicamente en lo que se refiere a las estimaciones de tiempos y costos, así como la definición de recursos necesarios y la elaboración de cronogramas que será uno de los principales mecanismos de control con los que se contará durante la etapa de desarrollo.

Entiéndase por requerimientos: *"condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo"* (Rumbaugh, Ivar y Grady 2005). Los requerimientos se clasifican en funcionales y no funcionales: los requerimientos funcionales *"son capacidades o condiciones que el sistema debe cumplir, se mantienen invariables sin importar con que propiedades o cualidades se relacionen por lo que no alteran la funcionalidad del producto"* (Rumbaugh, Ivar y Grady 2005). Los requerimientos no funcionales *"son las propiedades o cualidades que el sistema debe tener"* (Rumbaugh, Ivar y Grady 2005).

3.3.1 Requerimientos funcionales

Los requerimientos funcionales son de vital importancia para un eficaz desarrollo del sistema. A través de estos, se puede expresar una especificación más detallada de las responsabilidades que debe cumplir el producto. Con ellos, se pretende determinar de manera clara y concisa lo que debe hacer el sistema siguiendo un enfoque funcional.

Se definieron 6 requerimientos funcionales que quedaron expresados como funcionalidades del sistema. Estos se recogen en la siguiente tabla:

RF 1.	**El sistema debe permitir gestionar parada**
RF1.1.	El sistema debe permitir modificar los datos de una parada
RF1.2.	El sistema debe permitir eliminar una parada.
RF 2.	**El sistema debe permitir modificar los datos de una ruta**
RF 3.	**El sistema debe permitir gestionar acción**
RF3.1.	El sistema debe permitir adicionar una acción
RF3.2.	El sistema debe permitir modificar los datos de una acción
RF3.3.	El sistema debe permitir eliminar una acción
RF 4.	**El sistema debe permitir gestionar usuario**
RF4.1	El sistema debe permitir activar usuario
RF4.2.	El sistema debe permitir desactivar usuario
RF 4.3	El sistema debe permitir adicionar usuario

RF 4.4	El sistema debe permitir modificar los datos de un usuario
RF4.5.	El sistema debe permitir eliminar un usuario
RF4.6	El sistema debe permitir asignarle un rol a un determinado usuario.
RF 5.	**El sistema debe permitir gestionar rol**
RF5.1.	El sistema debe permitir regular acciones.
RF5.2.	El sistema debe permitir adicionar rol.
RF5.3.	El sistema debe permitir modificar los datos de un rol.
RF5.4.	El sistema debe permitir eliminar rol.
RF 6.	**El sistema debe permitir gestionar funcionalidad**
RF6.1.	El sistema debe permitir adicionar una funcionalidad.
RF6.2.	El sistema debe permitir modificar los datos de una funcionalidad.
RF6.3.	El sistema debe permitir eliminar una funcionalidad.

Tabla 2 Listado de los requerimientos funcionales del Sistema de Administración del SIG Rutas

3.3.1.1 Descripción de los requerimientos funcionales

A continuación se describen los requerimientos funcionales más importantes del Sistema de Administración del SIG Rutas (Admin Rutas). Sus descripciones tienen gran importancia porque ayudan a comprender el funcionamiento total del sistema.

El sistema debe permitir gestionar parada

Este requerimiento permite adiconar una parada, así como modificar y eliminar los datos de la misma, a partir de la selección de esta.

El sistema debe permitir modificar ruta

Este requerimiento permite modificar los datos de una ruta a partir de la selección de la misma y la modificación de los datos que la conforman.

El sistema debe permitir gestionar usuario

Este requerimiento permite: adicionar, modificar y eliminar los datos de un usuario, además de asignarle un rol, activar y desactivar a los mismos.

El sistema debe permitir gestionar rol

Este requerimiento permite: adicionar, modificar y eliminar un determinado rol, además de regular las acciones de este.

3.3.2 Requerimientos no funcionales

Los requerimientos no funcionales son fundamentales en el éxito del producto, debido a que forman una parte significativa del mismo. Estos requisitos constituyen rasgos fundamentales que hacen que el sistema sea usable, rápido, confiable y agradable para los usuarios. De ahí la importancia de su correcta descripción para el funcionamiento óptimo de la aplicación. A continuación se exponen los más importantes para el desarrollo del sistema que dará solución al problema que dio inicio a la investigación.

- ***Apariencia y usabilidad***

El sistema será utilizado por aquellas personas (administradores, instaladores, visualizadores) que tengan el permiso necesario para trabajar con la información que se maneja o para navegar en la web. Además contará con un manual de usuario, con instrucciones de tipo paso a paso, para entender el trabajo del sistema. La estructura de las interfaces serán claras, se contará con un menú general que le permitirá al usuario acceder de manera organizada a las diferentes funcionalidades, conjuntamente le brindará la opción de abrir varias funcionalidades a la vez, permitiendo minimizarlas en caso de querer trabajar con ellas en otro momento. La interfaces de cada una de las funcionalidades estarán bien distribuidas para que los usuarios sepan en cada momento qué acción realizar, a través de controles que puedan ser fácilmente identificados con cada una de las acciones previstas en el sistema.

Se prevé que la usabilidad de este producto sea elevada, o sea: que cuente con un alto nivel de aceptación por los usuarios finales, porque será sencillo, agradable, bien organizado y con un alto nivel de seguridad. Además el software tendrá siempre visible la opción de ayuda, lo que posibilitará un mejor aprovechamiento por parte de los

usuarios de sus funcionalidades. Las técnicas empleadas en su confección permiten que el sistema sea utilizado por usuarios con conocimientos mínimos de informática. Los obliga a realizar una funcionalidad y después otra, siguiendo una secuencia, de esta forma se evita que tengan mucha información para manipular y no se pierdan en la navegación por el sistema. Se desarrollará una interfaz de fácil uso y se combinará los colores de manera que sea agradable a la vista del usuario sin dejar que se pierda la seriedad y profesionalidad del sistema.

- ***Eficiencia***

El tiempo de respuesta será directamente proporcional a la cantidad de información a procesar, entre mayor cantidad de información mayor será el tiempo de procesamiento y viceversa. Aunque la velocidad no es lo más importante de este sistema, sino la calidad de la respuesta, debido a que el mismo maneja información socioeconómica de gran importancia.

- ***Fiabilidad***

La herramienta de implementación a utilizar tendrá soporte para recuperación ante fallos y errores. La información manejada por el sistema estará protegida de acceso no autorizado y divulgación. Para esto se restringirá el acceso a la aplicación, siendo obligatorio autenticarse antes de entrar al sistema, y los usuarios tendrán acceso a la información según los permisos de los mismos.

- ***Soporte***

La aplicación recibirá mantenimiento en el período de tiempo determinado por el equipo de desarrollo y los clientes.

- ***Seguridad***

Disponibilidad, Integridad y Confidencialidad

La información procesada por el sistema estará disponible todos los días y horas laborables. Esto se garantiza porque se tendrá implementado un plan de contingencias ante posibles fallos. Se tendrá una copia de seguridad fuera del nodo central en caso

de incendio y el ordenador en el que esté el servidor deberá contar con fuente de alimentación (bakup) en caso de inconvenientes con el fluido eléctrico. Al servidor, sólo tendrán acceso las personas autorizadas. Además el sistema garantiza un alto nivel de seguridad basada en la triple A (Autenticación, Autorización, Traceabilidad (Accounting)), gestionando dominio, roles y usuarios.

- ***Requisitos para la documentación de usuarios en línea y ayuda del sistema***

El software tendrá siempre la posibilidad de ayuda disponible para cualquier tipo de usuario y un manual para los mismos, con instrucciones de tipo paso a paso, para entender el trabajo del sistema y lograr una mayor explotación de las funcionalidades del mismo.

- ***Hardware***

  Para determinar los requerimientos de Hardware se partió de los requerimientos mínimos que necesita el sistema para su ejecución. Teniendo en cuenta que el framework utilizado es Zend y que el sistema sigue una arquitectura cliente /servidor, se recomienda:

Para las máquinas clientes:

- Se requiere tengan tarjeta de red.
- Al menos 128 MB de memoria RAM.
- Se requiere al menos 100 MB de disco duro.
- Procesador 512 MHz como mínimo.

Para los servidores:

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

- ***Software***

La construcción de la aplicación funcionará bajo los conceptos de arquitectura cliente/servidor. Por tanto, el servidor del usuario final debe tener como requerimientos mínimos de software:

Para las máquinas clientes:

- Un navegador web que cumpla los estándares w3c

Para los Servidores:

- Sistemas operativos GNU/Linux
- Servidor Web Apache 2.0 o superior, con módulo PHP 5 configurado con la extensión pgsql incluida.
- PostgreSQL como Sistema Gestor de Base de Datos incluyendo su módulo PostGIS debido a que los datos con los que se trabajan poseen información geoespacial.

- ***Requisitos de Licencia***

De acuerdo a los tipos de licencias de los componentes y herramientas que se proponen a utilizar para el desarrollo del Admin Rutas se puede catalogar legalmente esta arquitectura de modelo libre. Permitiendo la utilización, modificación y distribución de las mismas por terceros sin necesidad de obtener la autorización de sus respectivos titulares.

- ***Requisitos Legales, de Derecho de Autor y otros***

- El sistema debe ajustarse y regirse por la ley, decretos leyes, decretos, resoluciones y manuales (órdenes) establecidos, que norman los procesos que serán automatizados.
- La mayoría de las herramientas de desarrollo son libres y del resto, las licencias están avaladas.
- Como producto, el Admin Rutas se distribuye amparado bajo las normativas legales establecidas en el registro comercial emitido por las entidades jurídicas de la Universidad de las Ciencias Informáticas.

3.4 Casos de uso del sistema

Luego de establecer los requerimientos funcionales, éstos se agruparon en Casos de Uso del Sistema (CUS). Estos se definen como: “fragmentos de funcionalidad del sistema que proporciona al usuario un resultado importante” (Rumbaugh, Ivar y Grady 2005).

Concentrar la cantidad posible de requerimientos funcionales de la forma más fragmentada posible en casos de uso, trajo consigo importantes beneficios desde el punto de vista de la programación, debido a que se minimizó el grado de dificultad de los mismos a la hora de implementarlos.

3.4.1 Diagrama de casos de uso del sistema.

Un diagrama de Casos de Uso del Sistema *describe parte del modelo de casos de uso y muestra un conjunto de casos de uso y actores con una asociación entre cada par actor/caso de uso que interactúan (Ortiz Fidalgo y Rios Morales 2010).*

Un actor es un rol que cumple un usuario, puede intercambiar información o puede ser un recipiente pasivo de información y representa a un ser humano, a un software o a una máquina que interactúa con el sistema (Serrano Rosales y Rodríguez Vásquez 2008).

Actor del sistema.	Descripción
Administrador	Actor que interactúa directamente con el sistema, es el que inicializa todos los CUS entre los que se encuentran: “Gestionar Rol”, “Modificar Ruta”, “Gestionar Parada”, “Gestionar Usuario”.

Tabla 3 Descripción del Actor del Sistema.

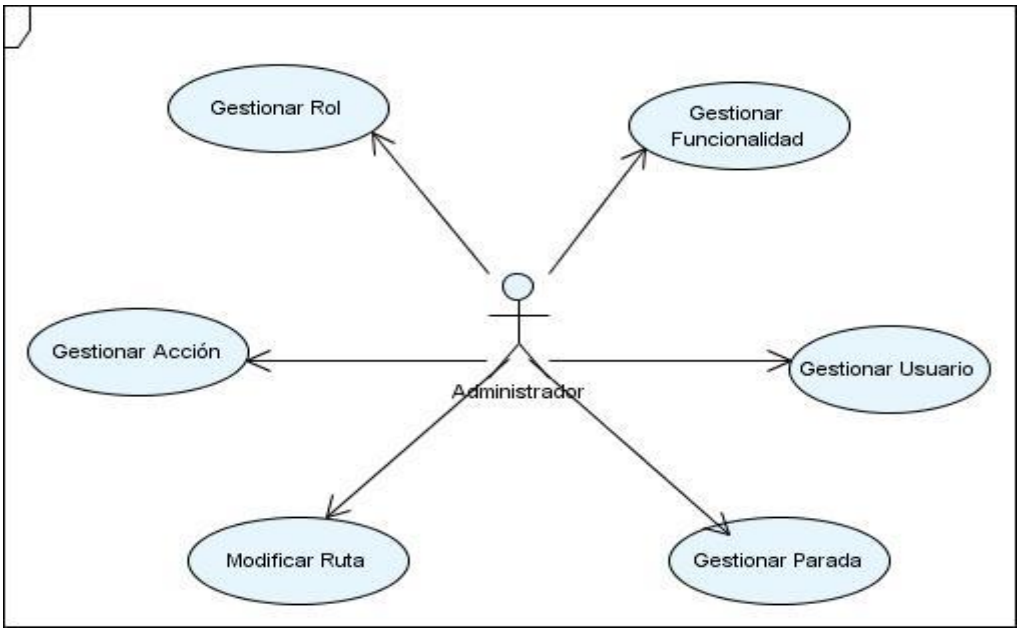


Fig2 Diagrama de Casos de Uso del Sistema

3.4.2 Clasificación de los CUS

Los CUS se clasifican en varias categorías: *los críticos son los más importantes para los usuarios porque cubren las principales tareas o funciones que el sistema ha de realizar y definen la arquitectura básica* (Rumbaugh, Ivar y Grady 2005). En el caso del Admin Rutas se determinaron como críticos por su elevado nivel de responsabilidad con el sistema los siguientes casos de uso: “Gestionar Rol, “Modificar Ruta”, “Gestionar Parada”, “Gestionar Usuario”. De los casos de uso críticos identificados se consideraron todos arquitectónicamente significativos.

Los casos de uso arquitectónicamente significativos, son aquellos que ayudan a mitigar los riesgos principales y a cubrir las funcionalidades significativas del sistema. Deben ser los más importantes para los usuarios y es necesario desarrollarlos en las primeras iteraciones. Para determinar los casos de uso arquitectónicamente significativos, se tuvo en cuenta que cumplieran con los siguientes criterios:

- Casos de uso con dificultad de desarrollo.
- Casos de uso imprescindibles para la puesta en marcha del sistema.
- Organización del desarrollo incremental.

- Disponibilidad de equipo de desarrollo.

El resto de los casos de uso mostrados en el diagrama de CUS, se clasificaron como casos de uso auxiliares. Los casos de uso auxiliares *no son claves para la arquitectura y completan casos de uso críticos o secundarios* (Rumbaugh, Ivar y Grady 2005).

3.4.3 Descripción de los CUS críticos arquitectónicamente significativos.

En este epígrafe se expondrán las breves descripciones de los CUS críticos arquitectónicamente significativos. Para una información más detallada de todos los CUS (ver Anexos Figuras desde la página 65 a la 89).

Gestionar Rol: El caso de uso se inicia cuando el administrador desea eliminar, modificar, adicionar, o regular las acciones de un determinado rol. Como precondición para ejecutar el caso de uso se tiene que, el administrador debe estar autenticado previamente. El sistema le muestra al administrador una interfaz que contiene una tabla con todos los roles registrados en el sistema y le permite realizar las siguientes operaciones con los mismos: Eliminar, Modificar, Adicionar, Regular Acciones. El administrador selecciona la opción deseada y el sistema en cada caso le muestra una interfaz que le permita realizar la operación seleccionada.

Gestionar Usuario: El caso de uso se inicia cuando el administrador desea eliminar, modificar, adicionar, activar, desactivar y asignarle un rol a uno o varios usuarios. Como precondición para ejecutar el caso de uso se tiene que, el administrador debe estar autenticado previamente. El sistema le muestra al administrador una interfaz que contiene una tabla con todos los usuarios registrados en el sistema y le permite realizar las siguientes operaciones con los mismos: Eliminar, Modificar, Activar, Desactivar, Adicionar, Asignarle un rol. El administrador selecciona la opción deseada y el sistema en cada caso le muestra una interfaz que le permita realizar la operación seleccionada.

Gestionar Parada: El caso de uso se inicia cuando el administrador desea eliminar o modificar una parada. Como precondición para ejecutar el caso de uso se tiene que, el administrador debe estar autenticado previamente. El sistema le muestra al Administrador las opciones de: Modificar Parada y Eliminar Parada. El administrador

selecciona la opción deseada y el sistema en cada caso le muestra una interfaz que le permita realizar la operación seleccionada.

Modificar Ruta: El caso de uso se inicia cuando el administrador desea modificar los datos de una ruta. Como precondición para ejecutar el caso de uso se tiene que, el administrador debe estar autenticado previamente. El sistema le muestra una interfaz que le permite realizar la operación.

Conclusiones

Siguiendo la metodología AUP, se presentó la descripción de la parte del Sistema de Administración para el SIG Rutas, correspondiente al flujo de trabajo de modelado. La especificación del Modelo de Dominio arrojó como resultado que todo el interesado pueda adquirir un entendimiento mínimo del contexto en que se emplaza el Admin Rutas. Se logró especificar de manera detallada las responsabilidades con la que debe cumplir el producto a través de los requisitos, minimizando el grado de dificultad a la hora de implementarlos mediante la definición de los Casos de Usos del Sistema.

Capítulo 4: “Construcción de la solución propuesta.”

4.1 Introducción.

Luego de un entendimiento de las funcionalidades y de los procesos del sistema, como parte del flujo de trabajo de modelado de la metodología de desarrollo AUP, se puede proseguir a realizar el diseño y como parte del flujo de trabajo de implementación, la construcción del Admin Rutas. En el siguiente capítulo se explica la arquitectura empleada y el conjunto de patrones de diseño usados para garantizar la viabilidad y calidad requerida del sistema a construir. De igual manera se analiza la fase de implementación a partir de los resultados del diseño, describiendo el estado actual del sistema en término de paquetes.

4.2 Descripción de la Arquitectura.

Definir una buena arquitectura ofrece los beneficios de tomar decisiones tempranas. Según plantea Barry Boehm, en el libro “Engineering Context (for Software Architecture)”, en 1995: “Si un proyecto no ha logrado una arquitectura del sistema, incluyendo su justificación, el proyecto no debe empezar el desarrollo en gran escala. Si se especifica la arquitectura como un elemento a entregar, se puede usar a lo largo de los procesos de desarrollo y mantenimiento.”

La arquitectura es una vista estructural de alto nivel, que define estilo o combinación de estilos para una solución. Se puede decir que la arquitectura es esencial para éxito o fracaso de un proyecto. Además la arquitectura de un software es necesaria para comprender el sistema, organizar el desarrollo del mismo, fomentar la reutilización y controlar la evolución del proyecto (Pressman, 2005).

Los estilos arquitectónicos, sirven para sintetizar estructuras de soluciones. Pocos estilos abstractos encapsulan una enorme variedad de configuraciones concretas, definen los patrones posibles de las aplicaciones. Conjuntamente permiten evaluar arquitecturas alternativas con ventajas y desventajas conocidas ante diferentes conjuntos de requerimientos no funcionales.

Para la realización del Admin Rutas se definió utilizar el estilo de Llamada y Retorno, que permite construir una estructura de programa relativamente fácil de modificar y ajustar a escala. Dentro de este, el patrón de diseño Modelo Vista Controlador (MVC), en estructura en capas. Donde todo el proceso está dividido en 3 partes, el Modelo, la Vista y el Controlador y para cada capa se encuentra un Framework que se encarga de su gestión. A modo de ejemplo: Se utiliza el framework Zend para todo el trabajo de caché, sección, registro, configuración, auditoría, autenticación, autorización, administración de conexiones y administración de perfiles, etc. Para la capa de acceso a datos se utiliza el doctrine con todas las bondades que el mismo brinda, así como postgre SQL para la capa de datos y Ext.js para la vista.

El patrón MVC se centra en la separación del modelo y la vista, mientras que el controlador es el encargado de relacionarlos. Su principal característica es aislar la vista del modelo. El patrón MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de la aplicación, es decir, se tratan como entidades separadas. Esto hace que cualquier cambio producido en el Modelo se refleje automáticamente en cada una de las Vistas.

La Vista será responsable de mostrar, mediante el uso de interfaces gráficas, la información almacenada en el modelo. Además será la encargada de recibir información del usuario mediante acciones efectuadas por el mismo a través de dichas interfaces. El Controlador especificará todas las reglas de negocio que permitirán ejecutar la totalidad de las operaciones que son ordenadas desde los objetos gráficos que se han definido previamente en el componente interfaz. Para ello se debe procesar la información que toma del paquete Vista para devolverle finalmente una respuesta, de modo que el usuario sea capaz de visualizar el resultado de la operación que ha ordenado. Además, se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones.

El Modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes. También realiza el encapsulamiento de los datos almacenados en objetos accesibles por los otros dos paquetes del sistema: Controlador y Vista.

Entre las ventajas y desventajas que este estilo presenta están las siguientes:

Ventajas.

- Permite tener un completo control sobre el comportamiento de una aplicación.
- Soporte de vistas múltiples, es decir, dado que la vista se encuentra aislada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente.
- Las interfaces tienden a cambiar más rápido que las reglas de negocios. Agregar nuevos tipos de vistas no afectan al modelo.
- Permite un diseño modular, posibilitando que los diseñadores y los desarrolladores trabajen conjuntamente (Rdoriguez Donatien y Ramirez Martín 2009).

Desventajas.

- Puede aumentar un poco la complejidad de la solución, porque está guiada por eventos, puede ser algo más difícil de depurar.
- El tiempo de desarrollo de una aplicación que implementa el patrón MVC es mayor, por el alto nivel de complejidad que este posee, al introducir nuevos niveles de indirección y por lo tanto aumenta ligeramente la complejidad de la solución.
- MVC es un patrón de diseño orientado a objetos por lo que su implementación es sumamente costosa y difícil en lenguajes que no siguen este paradigma.
- Si el modelo experimenta cambios frecuentes, la vista podría desbordarse con un diluvio de requerimientos de actualizaciones (Rdoriguez Donatien y Ramirez Martín 2009).

4.2.1 Patrones.

En el libro **"The Timeless Way of Building",** el Arquitecto Christopher Alexander define el concepto de patrón de la siguiente manera: "Cada patrón es una regla de 3 partes, que expresa una relación entre un contexto, un problema y una solución. Como un elemento en el mundo, cada patrón es una relación entre un contexto, un sistema

de fuerzas que ocurren repetidamente en ese contexto y una configuración espacial que permite que esas fuerzas se resuelvan entre sí."

Del mismo modo expone que: "Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo siquiera dos veces de la misma forma".

A modo de complemento un patrón es una representación o estructura determinable e identificable en diferentes dominios. Los patrones ofrecen soluciones a un problema recurrente, identificando los subsistemas, componentes y las unidades de colaboración entre ellos, al mismo tiempo, ayudan a diseñar un sistema en un breve tiempo. A continuación se describen los principales patrones que se utilizaron en el desarrollo del Admin Rutas.

Data Access Object (DAO): permite abstraer y encapsular todos los accesos a la fuente de datos. Permite además manejar la conexión con la fuente de datos para obtener y almacenar los mismos. Este patrón se evidencia en la capa de acceso a datos, ejemplo en la clase **CAD_Ruta.**

Patrones Creacionales: ofrecen soporte a una de las tareas más comunes dentro de la programación orientada a objetos, la instanciación. Estos ayudan a crear un sistema independiente de cómo se crean, se componen y se representan sus objetos.

- **Registry**

Permite mantener de forma global objetos que han sido registrados. Para la puesta en práctica de este patrón se utilizó la personalización que lleva por nombre ¨Conceptos Globales¨, que no son más que contenedores, que agrupan conceptos que son comunes para los subsistemas o componentes. Además se realizó la extensión Zend Registro al framework Zend, permitiendo el mismo, a través de este componente, almacenar todos los objetos en registros que pueden ser utilizados de manera global, facilitando conjuntamente que las instancias de las diferentes clases se creen a la par que cargue la aplicación.

- **Singleton**

Garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a esta instancia, característica que tributa al óptimo funcionamiento del sistema. Par la puesta en práctica de este patrón se crearon instancias únicas de: las clases, las secciones, los conceptos globales. Estos son almacenados en los registros del componente Zend Registro y pueden ser utilizados de manera global desde el mismo instante que se ejecuta la aplicación.

Patrones de diseño **GRASP** (**G**eneral **R**esponsibility **A**ssignment **S**oftware **P**arterns), describen los principios esenciales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

- **Experto.**

Asigna una responsabilidad al experto en información. Está enfocado en resolver el problema de ¿Cuál es el principio fundamental en virtud del cual se asignan las responsabilidades en el diseño orientado a objetos? La solución se basa en asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

En la aplicación se utilizan extensiones del Zend como Zend_Model la cual encapsula todos los temas de conexión a la base de datos. Además Zend se integra con SIGES que brinda los servicios de autenticación, autorización, auditoría, administración de conexiones y administración de perfiles.

- **Patrón Bajo Acoplamiento.**

Es la meta principal que es preciso tener presente durante el diseño. Es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño. Estimula asignar una responsabilidad de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento.

- **Patrón Alta Cohesión.**

Plantea que una clase tiene responsabilidades moderadas en un área funcional y colabora con las otras para llevar a cabo las tareas. O sea indica la coherencia que debe existir en la información almacenada en una clase. La cohesión es una medida

de la fuerza con la que se relacionan las clases y el grado de focalización de las responsabilidades de un elemento, Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable, una clase con baja cohesión hace muchas cosas no relacionadas o hace demasiado trabajo.

Para el diseño de la arquitectura se tuvo como factor primordial, eliminar el mayor grado de dependencia entre los módulos. Para ello la aplicación está agrupada en componentes según los procesos que se realizan. Por lo que cada una de las clases que implementa dicho componente está relacionada sobre la misma área funcional. Es decir se crearon componentes basado en la interrelación o la similitud existente entre los requisitos. En caso de que uno de estos componentes o subsistemas necesite la información de otro, se garantiza la comunicación entre los mismos a través del método de programación: Inversión de Controles (IOC). Este método facilita la integración a nivel de módulos y subsistemas, garantizando de esta forma el Bajo Acoplamiento y la Alta Cohesión.

- **Creador**

Asignar responsabilidades para la creación de objetos. El patrón trata de resolver el problema de diseño de quién debería ser responsable de crear una nueva instancia de alguna clase.

Este patrón se evidencia en las clases del paquete Domain, las que crean los objetos de tipo Doctrine_Query, posibilitando el acceso a la información almacenada a nivel de datos.

- **Controlador**

Asigna la responsabilidad a una clase de manejar mensajes correspondientes a eventos en un sistema. Dicha clase es la encargada de recibir los datos del usuario y enviarlos a las distintas clases. La aplicación del patrón conlleva a separar la lógica de negocios de la capa de presentación. Esto facilita la centralización de actividades (validaciones, seguridad, entre otras cosas). Si se aplica estos principios, el controlador

no realiza las actividades mencionadas sino que las delega en otras clases con las que mantiene un modelo de alta cohesión.

4.3 Diseño del Sistema.

El diseño de software se encuentra en el núcleo técnico de la ingeniería del software y se aplica independientemente del modelo de diseño de software que se utilice. Una vez que se analizan y especifican los requisitos del software, el diseño del software es la primera de las tres actividades técnicas; diseño, generación de código y prueba; que se requieren para construir y verificar el software. (Pressman, 2005)

Para lograr una implementación con calidad del Admin Rutas, se modelaron las páginas, sus enlaces y todo el contenido dinámico tanto del lado del servidor, como del cliente a través del Diagrama de Clases del Diseño. A continuación se propone una Vista General de dicho diagrama, basado en la arquitectura Modelo Vista Controlador. En el mismo se figuran los paquetes fundamentales y el conjunto de clases utilizadas directamente entre el resto de las contenidas.

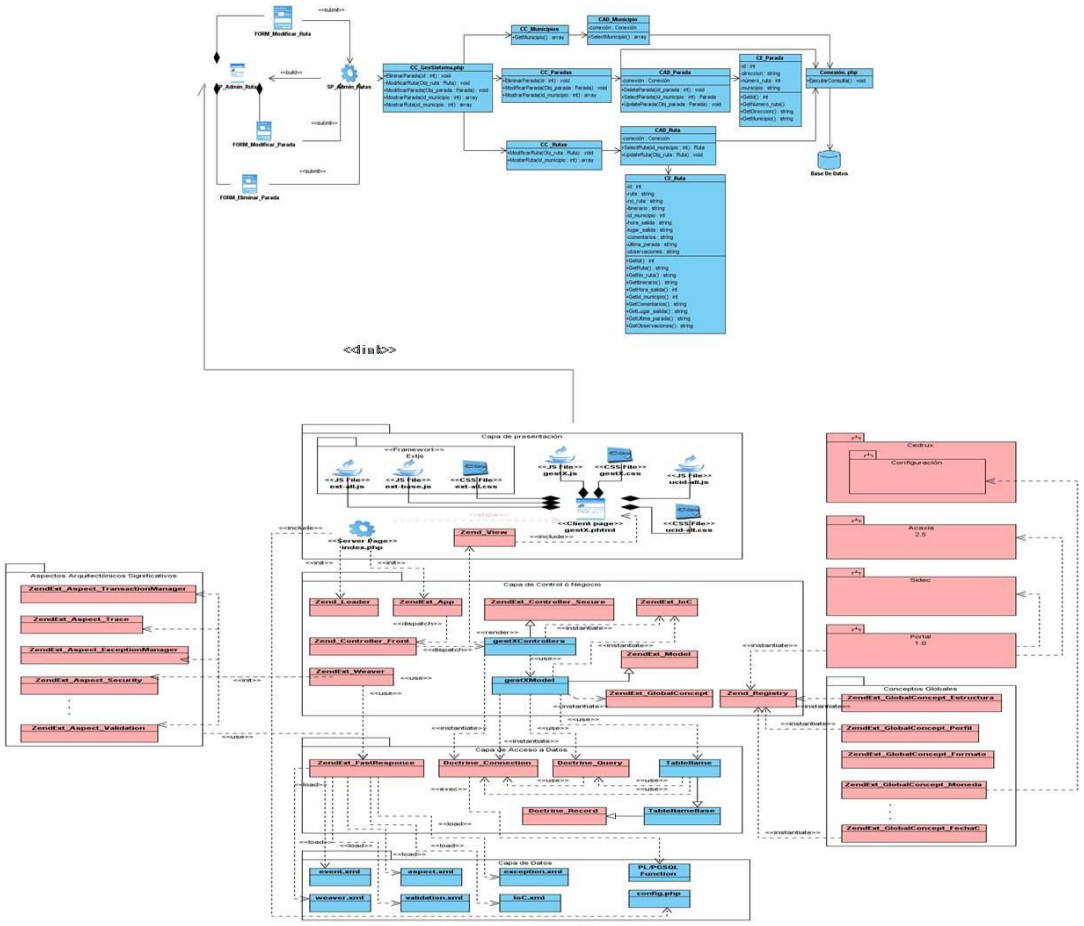


Fig. 3 Vista General del Diagrama de Clases del Diseño.

4.3.1 Diseño de la Base de Datos

Para el óptimo funcionamiento del Admin Rutas se trabaja con dos Base de Datos: la de gestión de la información del propio sistema y la del SIG Rutas. La Base de Datos (BD) del Admin Rutas se encuentra normalizada con el objetivo de evitar ambigüedades e inconsistencia en los datos.

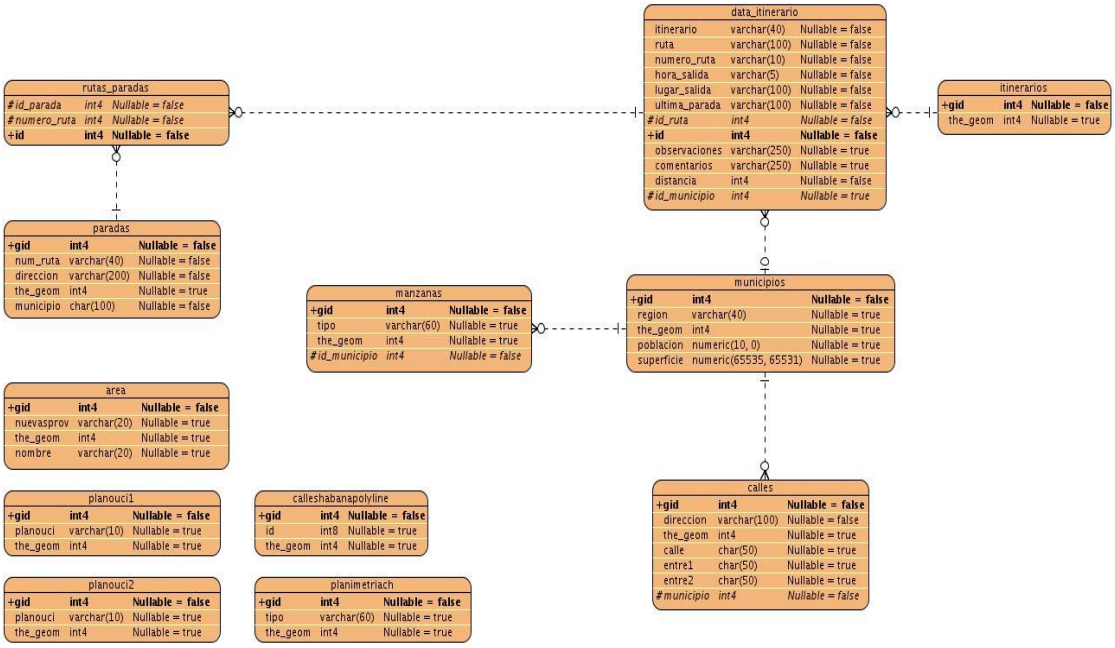


Figura.4 Modelo Entidad-Relación del SIG Rutas.

A continuación se describe una de las tablas más importantes del modelo, la cual se relaciona con el resto de las tablas.

Nombre	**Paradas**	
Descripción	Contiene la información necesaria para la identificación de las paradas en el mapa, así como su ubicación geográfica.	
Atributos	Tipo	Descripción
gid	int4	Constituye la llave primaria de la tabla.
num_ruta	varchar (40)	Número de la ruta a la cual pertenece.
dirección	varchar (200)	Dirección en la que está enclavada la parada.
the_geom	int4	Coordenadas espaciales del punto de referencia.
municipio	char (100)	Municipio al cual pertenece la ubicación de la parada.

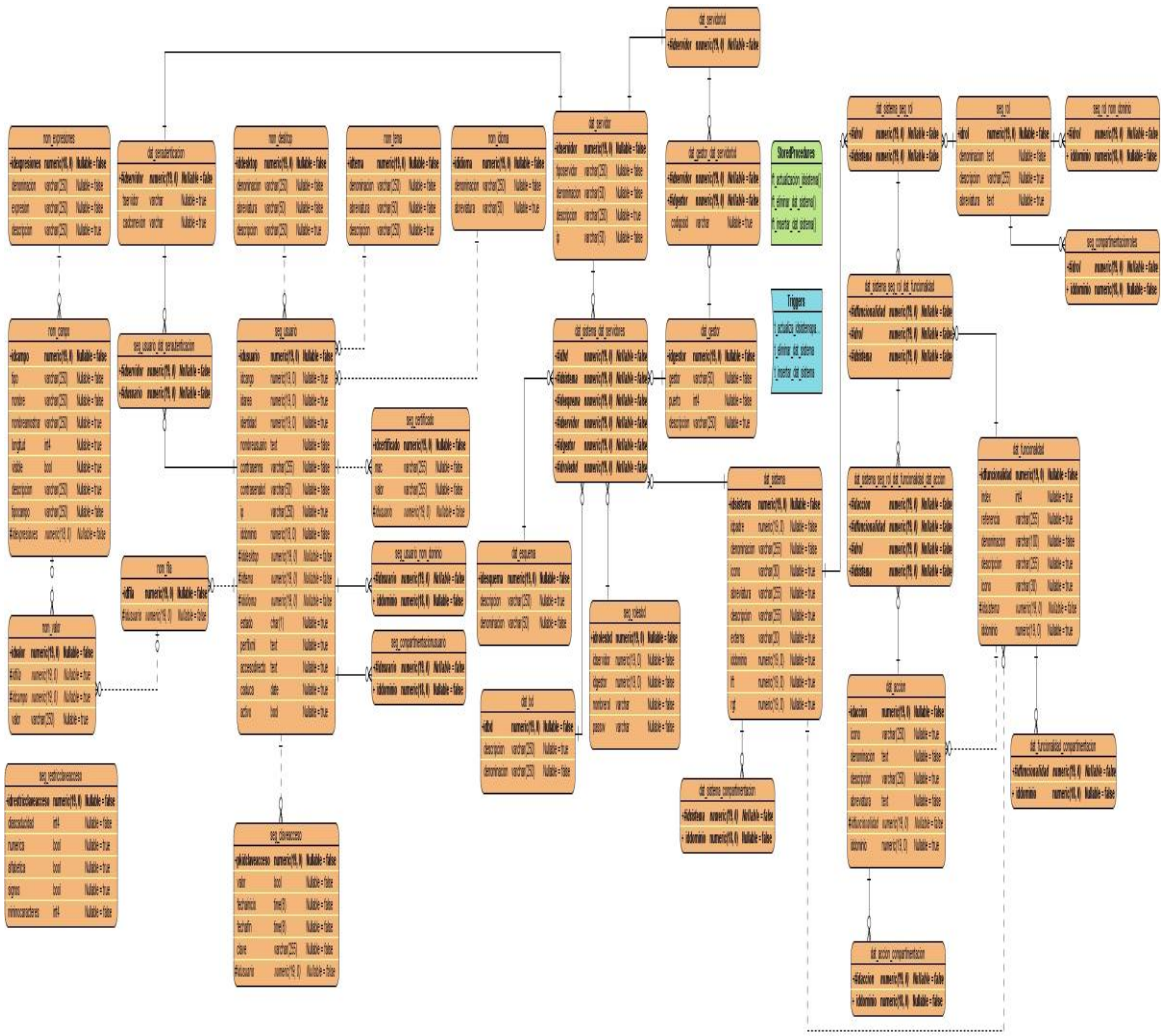


Fig. 5: Modelo Entidad-Relación correspondiente al Admin Rutas.

4.4 Modelo de Implementación.

La implementación empieza con el resultado del diseño y se implementa en términos de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares. El modelo de implementación está compuesto por un conjunto de subsistemas y componentes que constituyen la composición física de la implementación del sistema.

El objetivo esencial del modelo de implementación, es representar cómo las clases del modelo de diseño se implementan en términos de componentes, tales como ejecutables, ficheros de código fuentes o tablas de una base de datos. A continuación

se presentan los paquetes que engloban los componentes utilizados en el desarrollo del Admin Rutas. El paquete Acaxia y Portal representan subsistemas, que cuentan con su respectiva documentación ubicada en el repositorio del departamento de Geoinformática.

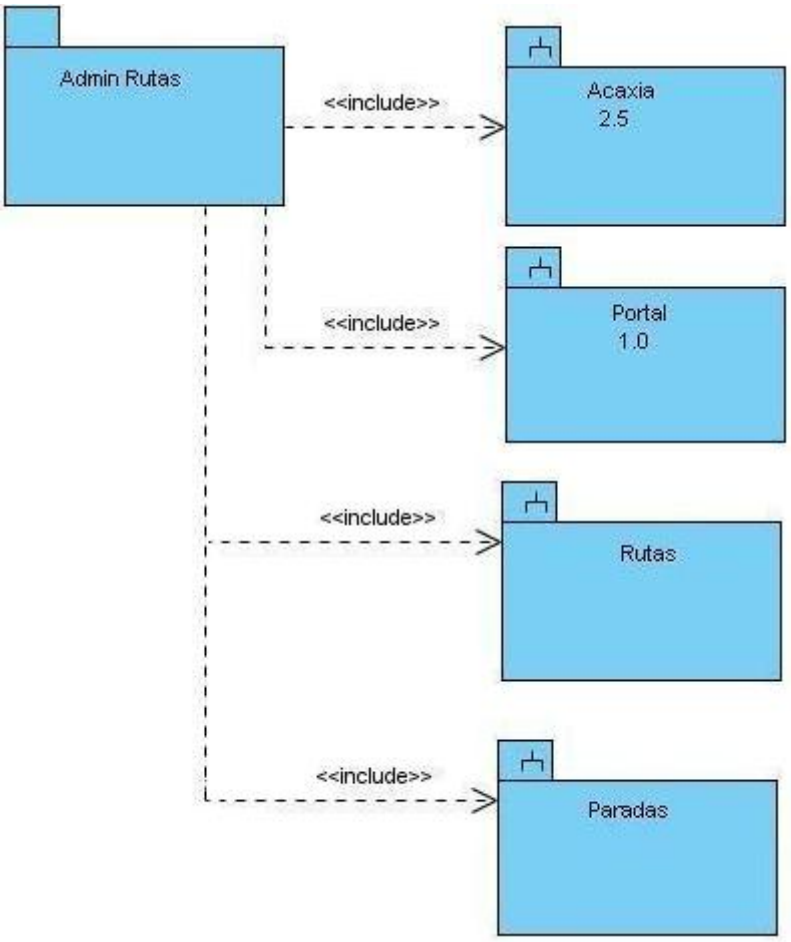


Fig. 6: Diagrama de Paquetes.

4.4.1 Diagrama de Despliegue.

Para lograr un entendimiento de la arquitectura de la aplicación en tiempo de ejecución se desarrolló el Diagrama de Despliegue. En este se evidencia la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos. Describe el camino en el cual los componentes específicos corresponden al despliegue de la aplicación.

El diagrama está representado por cuatro nodos (PC cliente, servidor web y dos servidores de base de datos). La PC cliente se comunica con el Servidor Web a través del protocolo HTTP. El servidor web haciendo uso del protocolo ADO establece comunicación con los servidores de base de datos. En la siguiente figura se muestra el Diagrama de Despliegue.

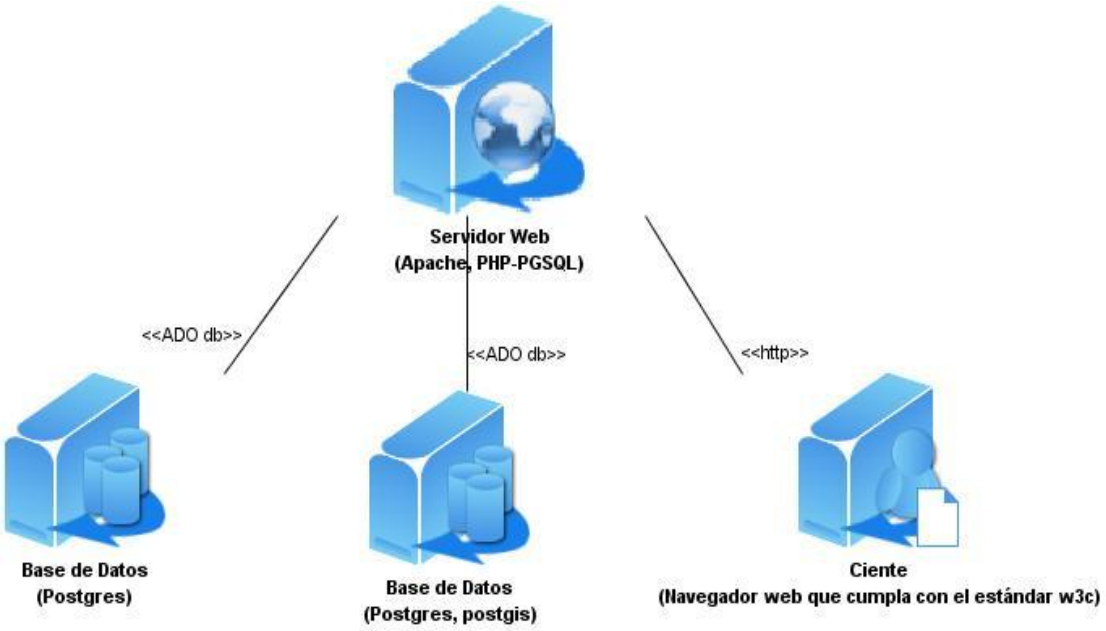


Fig. 7: Diagrama de despliegue.

4.5 Pruebas del sistema propuesto

Las pruebas pertenecen al penúltimo flujo de trabajo ingenieril de AUP, pero esto no significa que sea lo último que se realice. Lo que implica que se pueden ir realizando pruebas desde la fase de inicio del software hasta la fase de construcción, siendo esta última fase donde tiene mayor volumen el flujo de trabajo de prueba. Las pruebas fundamentales son denominadas: pruebas de caja blanca y pruebas de caja negra.

Las pruebas de caja blanca son: *el tipo de pruebas de software que se realiza sobre las funciones internas de un módulo. Las pruebas de caja blanca están dirigidas a las funciones internas.*

Las pruebas de caja negra son: *el tipo de pruebas de software que ejercitan los requisitos funcionales desde el exterior del módulo. En este tipo de prueba solo interesa su forma de interactuar con el medio que le rodea, entendiendo "qué es lo que hace", pero sin dar importancia a "cómo lo hace". Por tanto, deben estar muy bien definidas sus entradas y salidas.*

Con el objetivo de garantizar la calidad del producto y verificar el cumplimiento de los requisitos que se plantearon inicialmente, al sistema propuesto se le realizaron pruebas de caja negra. Se diseñaron seis Casos de Prueba, que arrojaron como resultado en la primera iteración **cinco** No Conformidades (NC) significativas y **una** NC no significativa, después de una segunda iteración se obtuvieron como resultado cero NC. A continuación se muestra un caso de prueba realizado al caso de uso **Gestionar Rol.**

Descripción de las variables.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Denominación	Campo de texto	No	Campo editable para registrar la deneminación del rol.
2	Abreviatura	Campo de texto	No	Campo editable para registrar la abreviatura de la denominación del rol.
3	Descripción	Campo de texto	Si	Campo editable para registrar la descripción del rol.
4	Sistemas registrados	Árbol de selección.	No	Árbol de selección donde se escogen el o las funcionalidades del sistema a las que va a tener acceso.
5	Sistemas y roles	Árbol de selección.	No	Árbol de selección donde se escogen el o las funcionalidades del sistema que se desean regular.
6	Acciones autorizadas	Tabla de selección.	Si	Tabla donde se listan las acciones autorizadas del rol seleccionado.
7	Acciones no autorizadas	Tabla de selección.	Si	Tabla donde se listan las acciones no autorizadas del rol seleccionado.

SC Adicionar rol

Escenario	Descripción	Denominación	Abreviatura	Descripción	Sistemas registrados	Respuesta del sistema	Flujo central
EC 1.1 Adicionar rol correctamente.	Se realiza la acción de adicionar un rol introduciendo los datos de manera correcta.	V Probador	V Prob	V Usuario con permisos de prueba.	NA		1. Inicio. 2. Seguridad. 3. Configurar usuarios. 4. Roles. 5. Adicionar. 6. Aceptar o Aplicar.
EC 1.2 Llevar a cabo la acción de Adicionar rol introduciendo datos incorrectos.	Realizar Adicionar rol introduciendo datos incorrectos.	I 123	V Prob	V Usuario con permisos de prueba.	NA		1. Inicio. 2. Seguridad. 3. Configurar usuarios. 4. Roles. 5. Adicionar. 6. Aceptar o Aplicar.
		V Probador	I 1234	V Usuario con permisos de prueba.	NA		
EC 1.3 Llevar a cabo la acción de Adicionar rol dejando datos incompletos.	Realizar Adicionar rol dejando datos necesarios en blanco.	I	V Prob	V Usuario con permisos de prueba.	NA		1. Inicio. 2. Seguridad. 3. Configurar usuarios. 4. Roles. 5. Adicionar. 6. Aceptar o Aplicar.
EC 1.4 Cerrar la acción Adicionar rol.	Terminar la acción de adicionar rol.	V Probador	V Prob	V Usuario con permisos de prueba.	NA		1. Inicio. 2. Seguridad. 3. Configurar usuarios. 4. Roles. 5. Adicionar. 6. Cerrar.

SC Modificar rol

Escenario	Descripción	Denominación	Abreviatura	Descripción	Sistemas registrados	Respuesta del sistema	Flujo central
EC 1.1 Modifiar un rol correctamente.	Se realiza la acción de Modificar un rol de manera correcta.	V Probador	V Prob1	V Usuario con permisos de prueba.	NA		1. Inicio. 2. Seguridad. 3. Configurar usuarios. 4. Roles 5. Seleccionar el rol. 6. Modificar. 7. Aceptar.
EC 1.2 Llevar a cabo la acción de Modificar rol introduciendo datos incorrectos.	Realizar Adicionar rol introduciendo datos incorrectos.	I 123	V Prob	V Usuario con permisos de prueba.	NA		1. Inicio. 2. Seguridad. 3. Configurar usuarios. 4. Roles 5. Seleccionar el rol. 6. Modificar. 7. Aceptar.
		V Probador	I 1234	V Usuario con permisos de prueba.	NA		
EC 1.3 Llevar a cabo la acción de Modifiar rol dejando datos incompletos.	Realizar Adicionar rol dejando datos necesarios en blanco.	I	V Prob	V Usuario con permisos de prueba.	NA		1. Inicio. 2. Seguridad. 3. Configurar usuarios. 4. Roles 5. Seleccionar el rol. 6. Modificar. 7. Aceptar.
EC 1.4 Cerrar la acción Modifiar rol.	Terminar la acción de adicionar rol.	V Probador	V Prob	V Usuario con permisos de prueba.	NA		1. Inicio. 2. Seguridad. 3. Configurar usuarios. 4. Roles 5. Seleccionar el rol. 6. Modificar. 7. Cerrar.

SC Eliminar rol

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Eliminar un rol correctamente.	Se realiza la acción de Eliminar un rol de manera correcta.		1. Inicio. 2. Seguridad. 3. Configurar usuarios. 4. Roles. 5. Seleccionar el rol. 6. Eliminar. 7. Aceptar. 8. Aceptar.

SC Regular acciones

Escenario	Descripción	Sistemas y roles	Acciones autorizadas	Acciones no autorizadas	Respuesta del sistema	Flujo central
EC 1.1 Regular acciones de manera correcta.	Se realiza la acción de Regular acciones de manera correcta.	NA	NA	NA		1. Inicio. 2. Seguridad. 3. Configurar usuarios. 4. Roles. 5. Selecciona un rol. 6. Regular acciones. 7. Cerrar
EC 1.4 Cerrar la acción Regular acciones.	Terminar la acción de Regular acciones.	V Probador	V Prob	V Usuario con permisos de prueba.		1. Inicio. 2. Seguridad. 3. Configurar usuarios. 4. Roles. 5. Selecciona un rol. 6. Regular acciones. 7. Cerrar

Conclusiones

Como resultado de este capítulo se definió la arquitectura que posibilitó organizar el desarrollo del sistema alcanzando un alto nivel de reutilización y control de la evolución del mismo; de igual manera los patrones de diseño utilizados fomentaron la escalabilidad del producto. La representación de las clases de diseño y sus relaciones en el diagrama de clases posibilitó una implementación con calidad del Admin Rutas. Se representó las agrupaciones lógicas en la que se encuentra dividido el sistema a través del Diagrama de Paquetes y se alcanzó un entendimiento de la arquitectura en tiempo de ejecución a través del Diagrama de Despliegue. La puesta en práctica de las pruebas de caja negra al sistema arrojó como resultado: que el mismo posee la calidad requerida y abarca el cumplimiento de todos los requerimientos que inicialmente fueron propuestos, implicando que se alcanzara un sistema útil y listo para ser desplegado.

Conclusiones generales

Con el estudio realizado en la investigación y la puesta en práctica para el desarrollo de la misma se arribó a las siguientes conclusiones:

- El estudio de los conceptos y teorías necesarias facilitó la comprensión de las temáticas tratadas.
- El análisis de las soluciones existentes a escala mundial, arrojó como resultado que las mismas no eran viables para dar respuesta al problema que dio origen a la investigación. Por lo que se realizó una personalización del SAUXE, adaptándolo a las condiciones del SIG Rutas.
- Se obtuvo todos los artefactos generados durante el proceso de desarrollo propuesto por la metodología AUP. Alcanzando no solo un mejor entendimiento de las responsabilidades que debe cumplir el producto, sino también que sirvieron de soporte para el desarrollo del sistema.
- Se obtuvo un sistema multiplataforma y totalmente libre, amigable, sencillo, de cómoda manipulación para los usuarios finales. Un sistema capaz de actualizar de manera fácil, eficiente y con un alto nivel de seguridad la información y funcionalidades que maneja el SIG Rutas.
- El análisis de los resultados arrojados por las pruebas de caja negra y por las revisiones realizadas por los profesores del proyecto Aplicativos SIG al sistema, convergen a la eficacia y calidad del sistema.

.

Recomendaciones

Para extender la investigación realizada se recomienda:

- El desarrollo de un módulo que les permita a los usuarios editar la cartografía desde el Admin Rutas de acuerdo a los permisos que le hayan conferido a estos.
- El Admin Rutas puede extenderse a otros SIG que necesiten la gestión de la información que manejan de manera segura.

BIBLIOGRAFÍA

1. Bover, Miguel García. "Construcción de un SIG para la Gestión de Rutas en Caminos no Cartografiados." Trabajo final de carrera, 2007.
2. Código Libre. http://www.codigolibre.org/index.php (accessed 11 27, 2010).
3. Definición. www.definicion.org/diccionario/172 (accessed 11 28, 2010).
4. Informáticas, Universidad de las Ciencias. "Subsistema Estructura y Composición." Manual de Usuario, 2008.
5. Juan Malle, Osman Medina y Marbelys Cánchica. SIG para la gestión de obras y servicios de proyectos comunales. República Bolivariana de Venezuela, Revista Generación Digital, 2009.
6. Pérez, Yasmany Zapata. Estrategia para la replicación de datos espaciales en Sistemas de Información Geográfica. Habana, Cuba: Universidad de las Ciencias Informáticas, 2008.
7. Serpa, Ivette Molina. Los Sistemas de Información Geográfica en Epidemiología. La Habana, Cuba: Instituto de Medicina Tropical "Pedro Kourí", 2001.
8. SIG GUADUA. http://www.sigguadua.gov.co (accessed Noviembre 26, 2010).
9. Torres., Alexander Rodríguez. Sistema de Información Geográfica de la UCI basado en. Ciudad de la Habana: Universidad de la Ciencias Informáticas, 2005.
10. Jacobson, Ivar, Grady Booch, and James Rumbauch. El Proceso Unificado de Desarrollo de Software. Félix Varela, 2004.
11. José H.Canós, Patricio Letelier y M. Carmen Penadés. "Métodologías Ágiles en el Desarrollo de Software." http://www.willydev.net/descargas/prev/TodoAgil.pdf (accessed Enero 21, 2010).
12. Mendoza, Luis E, Anna Grimán, and María Pérez. "Especialización de MSF para el Desarrollo basado en Componentes de un Sistema Colaborativo."

http://www.lisi.usb.ve/publicaciones/07%20integracion%20de%20sistemas/integracion_23.pdf (accessed Diciembre 15, 2009).

13. Valverde Rebaza, Carlos Jorge, and Sarah Dámaris Amaro Calderón. Metodoloías Ágiles. Trujillo, 2007.

14. Ortiz Fidalgo, Jackeline, and Fredy Hector Rios Morales. "Desarrollo de un componente se software para importar reportes desde documentos Excel a Sistemas Gestores de Bases de Datos en el CEINPET." Trabajo de Diploma, Habana, 2010.

15. Serrano Rosales, Carlos Luis, and Solangel Rodríguez Vásquez. "Desarrollo de Biblioteca de Métodos Numéricos (BMN), referente a Sistemas de Ecuaciones Lineales, Sistemas de Gran Dimensión y Poco Densos e Integración Numérica. Trabajo de diploma, Habana: UCI, 2008.

16. Pérez Zapata, Yasmany. Estrategia para la replicación de datos espaciales en Sistemas de Información Geográfica. Trabajo de Diploma, Habana: UCI, 2008.

17. Rdoriguez Donatien, Ariagna, and Carlos Enrique Ramirez Martín. Sistema para la Identificación de Aguas en Pozos Petroleros (SIAPP). Trbajo de diploma, Habana: UCI, 2009.

18. Ing. Agr. Néstor Di Leo. Zona Educativa. ¿Qué es un SIG o GIS? [Online] junio 10, 2002. [Cited: noviembre 28, 2010.] http://www.fcagr.unr.edu.ar/mdt/GTS/Zonaedu/Zonaedu.htm.

19. Rumbaugh, James, Ivar Jacobson, and Grady Booch. El Lenguaje Unificado de Modelado.Manual de Referencia. Madrid: Pearson Educación., 2005.

20. Pressman, Roger. Ingeniería de Software. Un enfoque práctico. Vol I, vol II. Editorial Félix Varela. La Habana, Cuba. 2005.

21. Alexander, Christopher. 1979. The Timeless Way of Building. 1979. 0-19-502402-8.

Anexos

Anexo 1. Descripción de los casos de uso del sistema

Anexo 1 .1 Gestionar Parada

Caso de Uso	Gestionar Parada
Actor	Administrador
Precondiciones	El administrador tiene que estar autenticado.
Referencias	RF1
Prioridad:	Crítica
Resumen	El Caso de uso se inicia cuando el administrador desea eliminar, o modificar los datos de una parada.
Flujo Normal de Eventos	
Acción del actor	**Respuesta del Sistema**
	1. El sistema le muestra al Administrador las opciones de: Modificar Parada y Eliminar Parada.
2. El Administrador selecciona la opción deseada.	3. En caso de que el Administrador selecciona la opción: a) "Modificar Parada": ver sección" Modificar Parada". b) "Eliminar Parada": ver sección "Eliminar Parada".
Sección Eliminar Parada	

Acción del actor	Respuesta del Sistema
	1. El Sistema muestra una lista de paradas.
2. El Administrador selecciona la parada a eliminar. 2.1 El Administrador selecciona la opción Aceptar. En caso contrario ver flujo alterno 1 ***Anulación de Eliminación de Parada.***	3. El sistema elimina la parada.

Prototipo de Interfaz de Usuario (1)

Flujo Alterno 1 Anulación de Eliminación de Parada.

Acción del Actor	Respuesta del Sistema
	1. El sistema no elimina la parada seleccionada.

Sección Modificar Parada

Acción del actor	Respuesta del Sistema
	1. El Sistema muestra una lista de municipios(A).

2. El administrador selecciona el municipio al que pertenece la parada a modificar.	3. El sistema muestra una lista con todas las paradas pertenecientes al municipio seleccionado (B).
4. El administrador selecciona la parada a modificar.	5. El sistema muestra una interfaz (2) con los datos de la parada a modificar: • Número de Ruta (C) • Dirección (D) • Municipio (E)
6. El administrador modifica los campos y selecciona la opción Aceptar. En caso contrario: ver flujo alterno 1 ***Anulación de Modificación de Parada.***	*7. El Sistema verifica que no existan campos vacios, en caso contrario: ver flujo alterno 2* ***Error de Modificación de Parada.*** *7.1 El Sistema modifica la parada seleccionada.*

Prototipo de Interfaz de Usuario (2)

Flujo Alterno 1 Anulación de Modificación de Parada	
Acción del Actor	**Respuesta del Sistema**

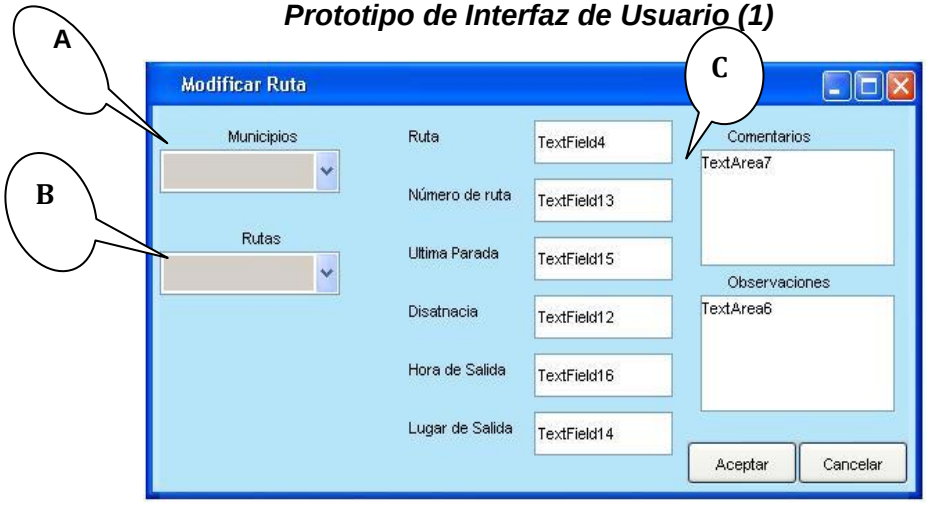
	2. El sistema no modifica la parada seleccionada.
Flujo Alterno 2 Error de Modificación de Parada.	
Acción del Actor	**Respuesta del Sistema**
	1. El sistema muestra un mensaje de error.
Prototipo de Interfaz de Usuario	
	
Poscondiciones	Se modificó o eliminó la parada correctamente.

Anexo 1.2 Modificar Ruta

Caso de Uso:	Modificar Ruta
Actores:	administrador
Resumen:	El caso de uso se inicia cuando el administrador desea cambiar los datos de una ruta
Precondiciones:	El administrador tiene que estar autenticado.
Referencias:	RF2
Prioridad:	Crítica
Flujo Normal de Eventos.	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción modificar ruta.	2. El Sistema muestra una interfaz (1) para modificar la

	ruta. 2.1 El Sistema muestra una lista de municipios (A).
3. El administrador selecciona el municipio al que pertenece la ruta.	4. El sistema muestra una lista con todas las rutas pertenecientes al municipio seleccionado (B).
5. El administrador selecciona la ruta a modificar.	6. El sistema habilita el panel con todos los datos de la ruta a modificar (C).
7. El administrador modifica los campos y selecciona la opción aceptar. En caso contrario: ver flujo alterno 1 ***Anulación de Modificación Ruta.***	8. El Sistema verifica que no existan campos vacios, ni error al llenar los datos. En caso contrario: ver flujo alterno 2 ***Error de Modificación de Ruta.*** 8.1 El Sistema guarda los cambios realizados.

Prototipo de Interfaz de Usuario (1)



Flujo Alterno 1 Anulación de Modificación Ruta.	
Acción del Actor	**Respuesta del Sistema**

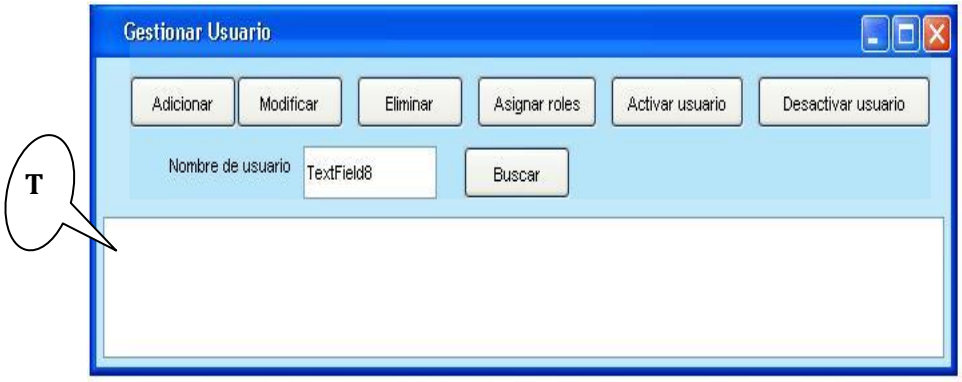
	1. El sistema no modifica la ruta seleccionada.
Flujo Alterno 2 Error de Modificación de Ruta.	
Acción del Actor	**Respuesta del Sistema**
	1. El sistema muestra un mensaje de error.
Prototipo de Interfaz de Usuario	
	
Poscondiciones	La ruta se modificó correctamente.

Anexo 1.5 Gestionar Usuario

Caso de Uso:	Gestionar Usuario
Actores:	administrador
Resumen:	El caso de uso se inicia cuando el administrador desea eliminar, modificar, adicionar, activar, desactivar y asignarle un rol a uno o varios usuarios.
Precondiciones:	El administrador tiene que estar autenticado.
Referencias:	RF4
Prioridad:	Crítica
Flujo Normal de Eventos.	
Acción del Actor	**Respuesta del Sistema**

1. El administrador selecciona la opción Gestionar Usuario	2. El sistema le muestra al administrador una interfaz (1) que contiene una tabla (T) con todos los usuarios registrados en el sistema y le permite realizar las siguientes operaciones con los mismos: Eliminar, Modificar, Activar, Desactivar, Adicionar, Asignarle un rol
1. El Administrador selecciona la opción deseada.	2. En caso de que el Administrador selecciona la opción: a) ”Eliminar Usuario” ver sección”Eliminar Usuario”. b) ”Modificar Usuario” ver sección “Modificar Usuario”. c) .“Adicionar Usuario” ver sección “Adicionar Usuario”. d) ”Activar Usuario” ver sección”Activar Usuario”. e) ”Desactivar Usuario” ver sección”Desactivar Usuario”. f) ”Asignar Rol a Usuario” ver sección”Asignar Rol a Usuario”.

Prototipo de Interfaz de Usuario (1)



Sección Adicionar Usuario	
	1. El sistema muestra una interfaz (2) para Adicionar el usuario, con los siguientes campos: • Usuario (A) • Rango IP (B)
2. El administrador llena los campos y selecciona la opción Aceptar o Aplicar. En caso contrario ver flujo alterno 1 ***Anulación del Proceso de Adicionar Usuario.***	3. El sistema verifica que no existan campos vacios. 3.1 El sistema adiciona el usuario y envía un mensaje informando que se realizó bien el proceso. En caso contrario ver flujo alterno 2 ***Error al Adicionar Usuario.***

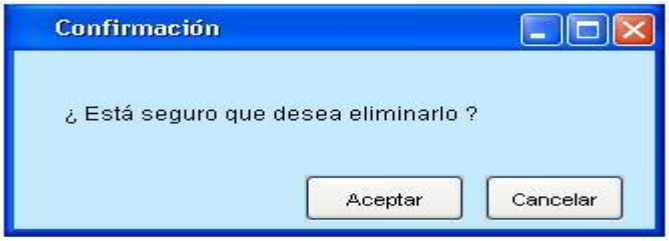
Prototipo de Interfaz de Usuario (2)

Flujo Alterno 1 Anulación del Proceso de Adicionar Usuario.	
Acción del Actor	**Respuesta del Sistema**
	1. El sistema no adiciona el usuario.

Flujo Alterno 2 Error al Adicionar Usuario.	
Acción del Actor	**Respuesta del Sistema**
	1. El sistema muestra un mensaje notificando el error.

Prototipo de Interfaz de Usuario

Sección Eliminar Usuario	
1. El administrador selecciona de la tabla el usuario a eliminar.	2. El sistema muestra un mensaje de información con la opción de "Aceptar" o "Cancelar".
3. El administrador selecciona la opción aceptar. En caso contrario: ver flujo alterno 1 ***Anulación del proceso de***	4. El Sistema elimina el usuario y envía un mensaje confirmando la realización satisfactoria de

Eliminar el Usuario	dicho proceso.
Prototipo de Interfaz de Usuario 	
Flujo Alterno 2 Anulación del Proceso de Eliminar Usuario.	
Acción del Actor	**Respuesta del Sistema**
	1. El sistema no elimina el usuario.
Sección Modificar Usuario.	
2. El administrador selecciona de la tabla el usuario a modificar.	3. El sistema muestra una interfaz (3) para Modificar los datos del usuario, con los siguientes campos: • Usuario (A) • Rango IP (B)
4. El administrador llena los campos y selecciona la opción Aceptar. En caso contrario ver flujo alterno 1 ***Anulación del Proceso de Modificar Usuario.***	5. El sistema verifica que no existan campos vacios, ni errores al llenar los datos. 5.1 El sistema modifica el usuario y envía un mensaje confirmando la realización satisfactoria de dicho proceso. En caso contrario ver flujo alterno 2 ***Error al Modificar Usuario.***

Prototipo de Interfaz de Usuario (3)

Flujo Alterno 1 Anulación del Proceso de Modificar Usuario.

Acción del Actor	Respuesta del Sistema
	1. El sistema no modifica el usuario.

Flujo Alterno 2 Error al Modificar Usuario.

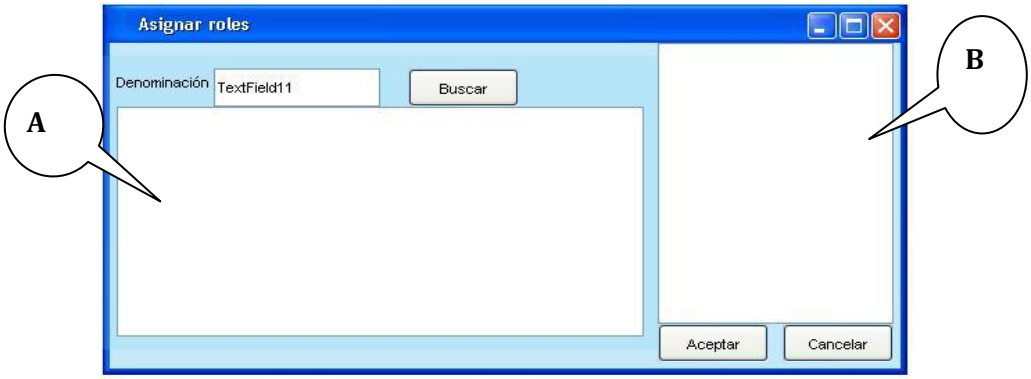
Acción del Actor	Respuesta del Sistema
	1. El sistema muestra un mensaje notificando el error.

Prototipo de Interfaz de Usuario

Sección Asignar Rol	
1. El administrador selecciona de la tabla (T) el usuario al que se le va asignar el rol.	2. El sistema muestra una interfaz (4) para Asignar el Rol que contiene: una tabla (A) con los roles y la descripción de los mismos.

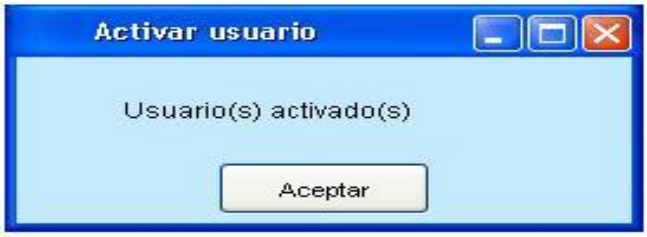
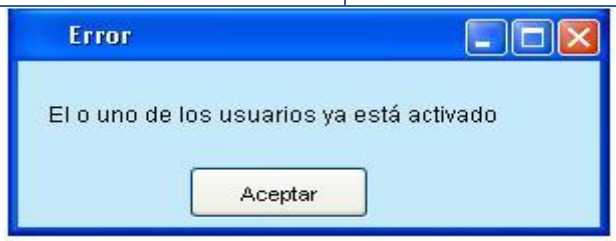
3. El administrador escoge el rol de la tabla (A).	4. El sistema le muestra en un árbol (B) las funcionalidades a las que tiene permiso dicho rol.
5. El administrador selecciona la opción Aceptar. En caso contrario ver flujo alterno 1 ***Anulación del Proceso de Asignación de Rol.***	6. El sistema le asigna el rol al usuario y envía un mensaje confirmando la realización satisfactoria de dicho proceso.

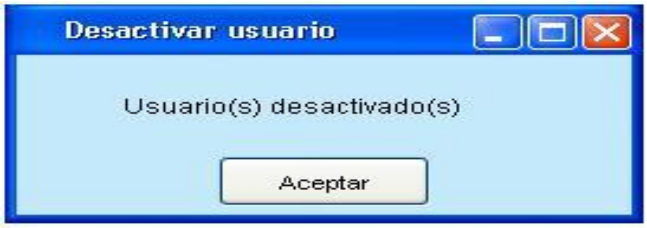
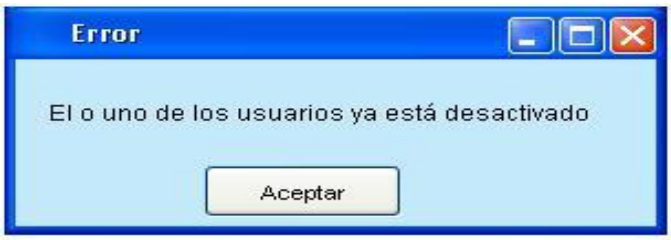
Prototipo de Interfaz de Usuario (4)



Flujo Alterno 1 Anulación del Proceso de Asignación de Rol.

Acción del Actor	**Respuesta del Sistema**
	1. El sistema no asigna el rol al usuario.
Sección Activar Usuario	
1. El administrador selecciona de la tabla el usuario o los usuarios a activar.	2. El sistema comprueba si el usuario(s) ya está activado. 2.1El Sistema activa el usuario(s) y envía un mensaje confirmando la realización satisfactoria de dicho proceso. En caso contrario: ver flujo alterno 1

	Error al Activar Usuario.
Prototipo de Interfaz de Usuario  	
Flujo Alterno 1 Error al Activar Usuario.	
Acción del Actor	**Respuesta del Sistema**
	1.El sistema envía un mensaje de error
 	
Sección Activar Usuario	
1. El administrador selecciona de la tabla (T) el usuario o los usuarios a desactivar.	2. El sistema comprueba si el usuario(s) ya está desactivado. 2.1 El Sistema desactiva el usuario(s) y envía un mensaje informando que se realizó bien dicho proceso. En caso contrario: ver flujo alterno 1 ***Error al Desactivar Usuario***.

<table>
<tr><td colspan="2">Prototipo de Interfaz de Usuario
 
</td></tr>
<tr><td colspan="2">Flujo Alterno 1 Error al Desactivar Usuario.</td></tr>
<tr><td>Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td></td><td>1. El sistema envía un mensaje de error</td></tr>
<tr><td colspan="2">Prototipo de Interfaz de Usuario
 
</td></tr>
<tr><td>Poscondiciones</td><td>El usuario se eliminó, modificó, adicionó, activó, desactivó y se le asignó un rol correctamente.</td></tr>
</table>

Anexo 2. Entrevista #1

Esta entrevista fue realizada al especialista general Darien García Tejo con el objetivo de obtener un conocimiento sobre los patrones utilizados en el desarrollo del SAUXE.

1. ¿Qué Arquitectura se tuvo en cuenta para la construcción del SAUXE?
2. ¿Qué patrones se utilizaron en el desarrollo del SAUXE?
3. ¿Cómo se evidencian en el sistema la aplicación de estos patrones?
4. ¿En qué consisten las extensiones realizadas al Framework Zend?
5. ¿Cómo aplican el patrón singleton en el sistema?