Universidad de las Ciencias Informáticas

Facultad 6

Título:

"Diseño de la arquitectura de software del Proyecto de Sistemas de Información Geográfica para dispositivos móviles."

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas

Autor: Raisa Ortega Baez

Tutor: Ing. Liester Cruz Castro

"Año 53 de la Revolución"

Declaración de autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 6 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.
Para que así conste firmamos la presente a los ____ días del mes de ____del año 2011.

_____	_____
Raisa Ortega Baez	Liester Cruz Castro
Autor	Tutor

Opiniones y avales

Centro de Desarrollo de Software "Geoinformática y Señales Digitales"

La Habana, 28 de mayo de 2011
"Año 53 de la Revolución"

De: Líder del proyecto: Odiel Estrada Molina

A: Miembros del Tribunal # 24.

Facultad 6.

Por medio del presente documento certifico que la investigación realizada por la estudiante Raisa Ortega Báez tutorada por el ingeniero Liester Cruz Castro, tributa al proyecto de software: Sistemas de Información Geográfica para Dispositivos Móviles. La investigación realizada le permitió a la estudiante obtener e implementar una arquitectura para el Producto MóvilMap perteneciente al proyecto de desarrollo de software.

A continuación se exponen los elementos que amparan la afirmación anterior:

1. Se identificaron algunos de los Casos de Usos Críticos del producto MóvilMap.
2. Se obtuvo los principales artefactos de software correspondiente a la línea base de la Arquitectura.
3. Se implementó en el proyecto de desarrollo de software la arquitectura definida por la estudiante y su tutor, permitiendo obtener en tiempo el producto de software.

Finalmente, se solicita adjuntar este documento al expediente de Tesis de la estudiante.

Para que así conste, firmo la presente, en un ejemplar, en la Universidad de las Ciencias Informáticas a los 30 días del mes de Mayo del año 2011.

Ing. Odiel Estrada Molina
Líder del proyecto

Resumen

El centro de Geoinformática y Señales Digitales (GEySED), de la facultad 6 de la Universidad de las Ciencias Informáticas, se ha propuesto desarrollar un Sistema de Información Geográfica para Dispositivos Móviles (SIG-Móviles). Esta investigación surge producto a la necesidad de conformar una Arquitectura de Software robusta y confiable para dicho sistema, la cual debe proporcionar los elementos puntuales para el desarrollo de la aplicación.

Para alcanzar los objetivos propuestos en la investigación se realizó el análisis de sistemas existentes con arquitecturas similares a SIG-Móviles, los principales conceptos, estilos y patrones arquitectónicos. Se hace además un estudio de las principales tendencias, metodologías y técnicas actuales con el fin de obtener una arquitectura que responda a las buenas prácticas de programación, así como la definición de la arquitectura diseñada y la propuesta de evaluación para la misma, garantizando una solución robusta que carezca de complejidad y con mejoras en cuanto a propiedades como la escalabilidad, seguridad y la reusabilidad.

Palabras Claves: Arquitectura de software, Móviles, SIG.

Índice de tablas

Índice de figuras

Índice

Introducción

Los avances de las Tecnologías de la Información y las Comunicaciones (TIC) han incidido positivamente en el progreso de la economía, la política y la sociedad mundial. Estos traen consigo un conjunto de mejoras y aportes a los procesos de gestión de la información, asumiendo de tal forma un papel importante en las esferas de investigación, formación y producción.

Las empresas que se dedican al desarrollo de software en el mundo, emplean herramientas computacionales con el objetivo de aumentar la producción de sistemas informáticos; para estos casos,cuentan con recursos humanos y tecnológicos capaces de brindar una mayor solidez en las operaciones que realizan, y a su vez que dichas aplicaciones sean más rápidas y seguras, facilitando la toma de decisión a través de los sistemas de información. Las aplicaciones de software en el mercado mundial presentan una demanda de forma permanente, lo cual requiere que estas aplicaciones informáticas posean un alto número de funcionalidades, servicios y además, muestren un grado de flexibilidad y rendimiento elevado. Las características anteriormente planteadas no son posibles de alcanzar si el software no está desarrollado con una arquitectura capaz de soportarla.

Cuba no está exenta a las exigencias que impone el mercado mundial con respecto a la producción de software, pues con capacidad de comercio limitada, ha venido desarrollando desde hace más de 10 años la industria nacional del software,a pesar de que en la actualidad los principales productores y consumidores de sistemas informáticos son los países desarrollados. Hoy se puede hablar de que existe soberanía tecnológica, pues aquellas restricciones que impone el bloqueo han obligado a la industria nacional del software a desarrollar sus sistemas basados en tecnologías libres. Con limitaciones económicas, pocos recursos y esfuerzo propio, las aplicaciones de “código abierto”, desarrolladas en el país,son la vía de escape para poder comercializar los productos internacionalmente.

La calidad se impone para desarrollar software competitivo, para ello fue creada la Universidad de las Ciencias informáticas (UCI), institución que junto a otras empresas productoras de software del país se ha venido vinculando satisfactoriamente en el desarrollo de sistemas informáticos para la exportación y en

mayor medida para darle solución a los problemas de informatización que requieren tanto la economía como la sociedad cubana.

La UCI entre sus principales pilares, tiene como objetivo el desarrollo de software para la exportación y el consumo nacional, para ello está dividida en centros de desarrollo que se especializan en determinadas áreas que deben ser informatizadas. El centro de desarrollo Geoinformática y Señales Digitales (GEySED), en una de sus aristas de trabajo, es el encargado del desarrollo de sistemas informáticos para el área del conocimiento de la geoinformática. Entre los proyectos de desarrollo de software del Departamento de Geoinformática se encuentra el Sistema de Información Geográfica para Dispositivos Móviles (SIG-Móviles), compuesto por un grupo de desarrolladores de aplicaciones informáticas especialistas en el tema.

Esta línea de investigación, producción e innovación perteneciente al Centro GEySED enfoca su razón de ser en el desarrollo de aplicaciones con capacidad de gestionar información relevante a los usuarios de dispositivos móviles sobre datos georeferenciados. Los objetivos que se persiguen con este sistema son proporcionar servicios de acceso a la información geográfica, para su consulta, análisis y visualización sobre mapas, mediante una interfaz de usuario sencilla y de fácil manejo que pueda ser utilizada por usuarios no especializados en tecnología de Sistemas de Información Geográfica (SIG). Además de integrar la información socioeconómica existente (recursos humanos, activos fijos, entidades de servicios, lugares de interés), con la información geográfica asociada. Estas funcionalidades deben ser soportadas por los SIG-Móviles y solamente es posible teniendo en cuenta que no se afecten factores considerables en la arquitectura en sí a la hora de desarrollar el software. Sin lugar a dudas la reutilización de los subsistemas que se definen y la escalabilidad de la aplicación son elementos notables a tener en cuenta a la hora de desarrollar un sistema que pone en evidencia un grupo de cambios y mejoras con el transcurso del tiempo.

El desarrollo de estas aplicaciones carecen de reutilización de susbsistemas ya desarrollados, estos tienen poca documentación o nula, lo que hace que el código generado en cada proyecto apenas se pueda utilizar, no existe una línea base que guíe el desarrollo basado en los distintos procesos identificados, lo que provoca constantes cambios por parte del equipo de trabajo, no existen vistas

generales que representen los componentes desarrollados, por lo que los trabajadores no tendrán una visión de lo que se quiere realizar, estas deficiencias conllevan a que el proyecto no se termine en el tiempo planificado y que el costo del mismo aumente; por todo esto se identifica el siguiente **problema a resolver**:

Mejorar el proceso de desarrollo de software que se ejecuta en el proyecto SIG-Móviles.

Para darle respuesta al problema planteado, se define como **objeto de estudio:** El proceso de diseño de una arquitectura basada en tecnologías de software libre y como **campo de acción:** El diseño y descripción de la arquitectura base para el desarrollo del proyecto Sistemas de Información Geográfica para Dispositivos Móviles.

Como **idea a defender** se plantea que; con el diseño de una arquitectura robusta y flexible se garantizará una mejora en el proceso de desarrollo de software del proyecto Sistemas de Información Geográfica para Dispositivos Móviles.

Para ello se ha definido el siguiente **objetivo general:** Diseñar la arquitectura base para el Sistema de Información Geográfica para Dispositivos Móviles que permita responder a las necesidades trazadas en el mismo.

Para darle cumplimiento al objetivo se definieron las siguientes **tareas de investigación**:

- Analizar sistemas informáticos con características similares a los existentes en el proyecto Sistemas de Información Geográfica para Dispositivos Móviles.
- Valorar las diferentes propuestas arquitectónicas reflejadas en los sistemas analizados.
- Analizar estilos y patrones arquitectónicos actuales.
- Analizar las herramientas a utilizar para el desarrollo de las aplicaciones informáticas en el proyecto Sistemas de Información Geográfica para Dispositivos Móviles.
- Seleccionar las herramientas a utilizar para el desarrollo de las aplicaciones informáticas en el proyecto Sistemas de Información Geográfica para Dispositivos Móviles.

- Plantear la propuesta arquitectónica para el proyecto Sistemas de Información Geográfica para Dispositivos Móviles.
- Especificar mediante el estudio de varias metodologías de prueba de arquitectura cuál es aplicable a la arquitectura de software definida.
- Proponer una evaluación a la arquitectura de software definida.

Para dar cumplimiento a las tareas de la investigación, se hace necesario el uso de diferentes métodos de investigación como los que se presentan a continuación:

Como **métodos teóricos** que se emplean para el desarrollo del trabajo de diploma:

- Analítico-Sintético: Permite a través de la investigación realizar un análisis detallado de bibliografías y documentos para extraer los elementos más importantes y establecer el diseño de la arquitectura, método empleado en el análisis del estado del arte.
- Histórico-Lógico: Para analizar la trayectoria completa del objeto, así como las etapas principales de su desenvolvimiento. Este método fue empleado para analizar las tendencias de la arquitectura de software, estilos y patrones arquitectónicos que son la base del desarrollo de la aplicación.
- Inductivo-deductivo: Permite llegar a la conclusión mediante la generalización del conocimiento adquirido desde el análisis atravesando de lo general a lo particular, este método fue utilizado a la hora de arribar a las diferentes conclusiones parciales por parte del autor.
- Modelación: Es el que permite la creación de modelos, propuestas, alternativas, estrategias y así llegar a modelar el diseño de la arquitectura propuesta. Este método fue empleado en la realización de los diagramas representativos de las vistas arquitectónicas del sistema.

Como **método empírico** utilizado se presenta:

- Observación: Empleado para alcanzar una caracterización detallada de las soluciones existentes para sistemas similares, con la perspectiva de valorar si a partir de ellas se pueden aminorar los principales problemas del centro GEySED.

Estos métodos permiten la creación de modelos, propuestas, alternativas, estrategias.

Con la realización de este trabajo se espera como **resultado** lograr una definición de herramientas a utilizar y una documentación completa asociada al diseño de la arquitectura para un sistema que controle el flujo de información que se genera a partir del proceso de producción en el proyecto SIG-Móviles.

El presente trabajo de diploma está estructurado en cuatro capítulos:

- El **Capítulo 1** con la **Fundamentación Teórica,** se presentaron los principales conceptos asociados al objeto de estudio, así como lo referente a los patrones, estilos arquitectónicos, herramientas y tecnologías a usar para el desarrollo del sistema, de forma tal que en los capítulos posteriores se pudo decidir cuál era la más adecuada para el proyecto.

- El **Capítulo 2** con la **Propuesta de Arquitectura y Tecnologías a Utilizar** después de haber hecho un análisis y una representación de los elementos en el capítulo anterior, a través de ellos se precisó finalmente lo conveniente y lo más correcto para el proyecto.

- El **Capítulo 3** con la **Descripción de la Arquitectura de Software** basado en los estudios realizados en capítulos anteriores y las necesidades del negocio que exige la empresa, se propone el diseño de la arquitectura, en función de proporcionar una visión general arquitectónica completa del sistema, mediante una serie de vistas arquitectónicas diferentes, para representar los aspectos del sistema .

- El **Capítulo 4** con la **Propuesta de Evaluación de la Arquitectura** una vez presentado el diseño de la arquitectura del capítulo anterior, se propone como realizar la evaluación de la arquitectura a través de técnicas y métodos para conocer los problemas presentes en el sistema y de igual forma corregirlos a tiempo.

Capítulo I: Fundamentación Teórica

Introducción

En el presente capítulo se especifican los principales elementos teóricos que se tienen en cuenta para lograr un mejor entendimiento y comprensión de la investigación. Se hace además una descripción general del proyecto SIG-Móviles y del estado actual del problema que se ha planteado.

Se realiza un estudio detallado de los patrones y estilos arquitectónicos existentes, además de las principales herramientas, tecnologías, lenguajes de programación y metodologías de desarrollo que pudieran ser utilizadas para dar solución a dicho problema.

1.1 Conceptos asociados al dominio del problema.

Es necesario definir un conjunto de conceptos que se encuentran asociados al dominio del problema, y que deben ser descritos para lograr un mejor entendimiento de la investigación. A continuación se puntualizan los de más importancia:

SIG: Siglas en español que se refieren a Sistemas de Información Geográfica, además puede encontrarse las siglas en inglés GIS (*Geographic Information Systems*).

Según el programa de estudio de Maestría de Diseño Urbano de la Universidad de Costa Rica, SIG es un software o programa informático, proyecto y sistema de información orientado fundamentalmente a la gestión o planificación territorial y campo o ambiente de trabajo de múltiples disciplinas profesionales orientado fundamentalmente al análisis espacial dentro de la ordenación del territorio.

Se pueden encontrar definiciones incompletas como la de Joaquín Bosque Sendra donde plantea en el libro SIG segunda edición en el año 1997, que SIG es una tecnología informática para gestionar y analizar información espacial (Bosque, Joaquín,1997). Rubén Martínez Marín en su libro “Topografía y Sistemas

de Información" publicado en el año 2000 define como SIG a sistemas informáticos capaces de ensamblar, almacenar, manipular y representar en una pantalla gráfica, la información geográficamente referenciada, es decir, información identificable por su posición real sobre la superficie terrestre.(Martínez, Rubén, 2000)

La definición anterior es parcial, considerando que autores como Flor Álvarez Taboada (Álvarez, Flor, 2009), Nieves Lantada Zarzosa y M. Amparo Núñez Andrés (Lantada Nieves, 2004), coinciden con el criterio de este autor de que un SIG es la combinación de componentes: personas especializadas, datos descriptivos y espaciales, métodos analíticos, hardware y software; organizados para analizar, gestionar y visualizar todo tipo de información referenciada geográficamente. Son sistemas que permiten realizar operaciones de consulta y análisis sobre bases de datos espaciales, a la vez que presentan la posibilidad de visualización y realización de análisis geográficos sobre estas. En lo adelante se tomará este criterio como definición.

Los SIG se aplican en la mayoría de los sectores y pueden ser utilizados como una herramienta de ayuda a la gestión y toma de decisiones, algunos de ellos son: gestión territorial, urbanismo, medio ambiente, transporte, y recursos mineros e hídricos.

Dispositivo Móvil: Es la palabra modernamente aplicada a los teléfonos móviles, que son dispositivos electrónicos inalámbricos; y cuando se refiere a la telefonía celular móvil se puede decir que es la que permite el funcionamiento de los celulares, para poder alcanzar el propósito de establecer comunicación, objetivo para los que fueron creados. Son aparatos de pequeño tamaño, con memoria limitada y algunas capacidades de procesamiento. Suelen tener una pantalla y botones pequeños, aunque algunos carecen totalmente de botones y se manejan con pantallas táctiles (Hilda, 2009).

Los dispositivos móviles son más potentes y livianos, permitiendo que la comunicación sea cada vez más eficaz. Tanto la cantidad como la diversidad en que aparecen, así como sus capacidades hacen muy interesante para los proveedores de servicios y contenidos el disponer de un entorno normalizado que permita ofrecer sus servicios a los usuarios de las redes móviles.

Un PDA (*Personal Digital Assistant* o Ayudante Personal Digital) es un tipo de dispositivo móvil de pequeño tamaño que combina un ordenador, teléfono/fax, Internet y conexiones de red. A los PDAs también se les puede encontrar con los siguientes nombres: palmtops, *hand held computers* (ordenadores de mano) y *pocket computers* (ordenadores de bolsillo).

Portal: El término de portal es amplio y cuando se utiliza puede referirse a varios elementos que en su espectro se encuentra. En la narrativa de ciencia ficción y fantasía es un dispositivo imaginario tecnológico que sirve como una vía de pasaje entre dos sitios distantes en el espacio y/o tiempo. Esta definición no es la más adecuada cuando se refiere a portal de un sitio en internet o una intranet corporativa. El término válido en el entorno que se enmarca este concepto para el resto de la investigación, es el que ofrece el diccionario informático quien deja claro que un portal es un sitio web que ofrece a los usuarios acceso a numerosos recursos y servicios tales como buscadores, chats y foros.

WAP: Acrónimo a (*Wireless Application Protocol*), en español (Protocolo de Aplicaciones Inalámbricas). Es la combinación de una serie de protocolos de red móvil, es un estándar abierto internacionalmente que permite a los dispositivos inalámbricos realizar comunicaciones en tiempo real y un uso fácil de los disímiles servicios.

Las definiciones anteriores conducen a **Portal WAP,** que es la combinación del término informático de portal, siendo este último la puerta de enlace para uso de dispositivos móviles, en función de utilizar los servicios ofrecidos por el Protocolo de Aplicaciones Inalámbricas para clientes del portal que hacen uso de las entradas y salidas de dichos servicios.

1.2 Objeto de estudio

1.2.1 Descripción general.

El proceso de diseño arquitectónico representa la estructura de los datos y los componentes del programa que se requieren para construir un sistema de computadora. Constituye el estilo de la arquitectura de software que tendrá el sistema, las estructuras de datos, las propiedades de los componentes y la interrelación que tiene con otros componentes arquitectónicos de la propia aplicación informática.

Este proceso tiene gran importancia porque proporciona los elementos del software que se va a desarrollar, apreciándose un nivel de detalle y relación muy amplio entre los distintos componentes que conformarán el sistema informático. Se asegura además que la construcción de la aplicación informática sea correcta, completa y centre a los desarrolladores en lo que se desea implementar.

Durante el proceso de diseño de la arquitectura de software se toman decisiones importantes orientadas a la naturaleza estructural del sistema y la abstracción de la representación de la información, aspectos que son necesarios para los desarrolladores de software. Dicho proceso hace representaciones coherentes y bien planificadas de los programas, centrándose en las interrelaciones de los componentes de mayor nivel y en las operaciones lógicas implicadas en los niveles inferiores.

El diseño de la arquitectura de software en todo su proceso debe realizarse basado en la política de la utilización y creación de software no privativo, aspecto que es esencial para la sociedad actual cubana ya que se obtendría soberanía tecnológica cubana sobre este tipo de sistema y un paso de avance en la comunidad de software libre internacional. Diseñar software libre conlleva a que se realice un estudio más detallado de las tendencias y tecnologías actuales teniendo cuidado en las licencias de software que sean privativos y eligiendo el software libre para desarrollar que sea más adecuado para la contrucción del sistema informático que se propone.

Para lograr un diseño arquitectónico correcto se debe especializar una persona en el rol de arquitecto de software, el cual debe conocer todas las actividades y artefactos que debe generar; esta persona debe además saber a cabalidad que es la arquitectura de software y como funciona su proceso de diseño. Un aspecto importante es que el o los encargados del diseño de la arquitectura de software debe interaccionar con el equipo de trabajo en su conjunto y fijar las pautas para que el proceso de desarrollo tenga muy en cuenta la arquitectura, el rol de arquitecto de software interviene y se nutre de las actividades que realizan los demás roles.

Resulta de vital importancia mencionar las principales características de la arquitectura de software como tal para esclarecer cuestiones que se han presentado del proceso de diseño de la misma, a continuación se describe.

1.2.1.1 Descripción de la Arquitectura de Software

La arquitectura de software es el diseño de más alto nivel de la estructura de un sistema que establece los fundamentos para que analistas, diseñadores y programadores, trabajen en una línea común que permita cubrir las necesidades y objetivos de dicho sistema.

La definición de la Arquitectura de Software (AS, en lo adelante) no es única, existen muchas definiciones y algunas son cuestionables. A continuación se presentan las más completas para el entendimiento de la actual investigación.

La Arquitectura de Software de un sistema de programa o computación, es la estructura de las estructuras del sistema, comprende los componentes del software, las propiedades de esos componentes visibles externamente y las relaciones entre ellos. (Pressman, Roger R, 2002)

Una definición reconocida es la de Clements cuando plantea que: La AS es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones.(Clements, Paul, 1996)

En una definición tal vez demasiado amplia, David Garlan establece que la AS constituye un puente entre el requerimiento y el código, ocupando el lugar que en los gráficos antiguos se reservaba para el diseño. La definición “oficial” de AS se ha acordado que sea la que manifiesta el documento de IEEE Std 1471-2000, adoptada también por Microsoft, que dice así: “La Arquitectura de Software es la organización fundamental de un sistema emarcada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución”. (Garlan, David, 2004)

Como se puede apreciar, cada autor expresa su criterio,una vez visto los conceptos anteriores se puede decir que tienen algunos puntos en común pero no convergen a una misma idea. Simplemente el objetivo principal de la Arquitectura del Software es aportar elementos que ayuden a la toma de decisiones y al

mismo tiempo proporcionar conceptos y un lenguaje común que permitan la comunicación entre los equipos que participen en el desarrollo de un sistema. Es fundamental en el desarrollo de las aplicaciones y define cuales son las principales características del software, con cierto nivel de abstracción.

A continuación se presentan los aportes de la AS según los expertos Clements y David Garlan:

Comunicación mutua:La AS representa un alto nivel de abstracción común que la mayoría de los participantes, pueden usar como base para crear entendimiento mutuo, formar concenso y comunicarse entre sí. En sus mejores expresiones, la descripción arquitectónica expone las restricciones de alto nivel sobre el diseño del sistema, así como la justificación de decisiones arquitectónicas fundamentales. (Garlan, David, 2004)

Decisiones tempranas de diseño: La AS representa la encarnación de las decisiones de diseño más tempranas sobre un sistema, y esos vínculos tempranos tienen un peso fuera de toda proporción en su gravedad individual con respecto al desarrollo restante del sistema, su servicio en el despliegue y su vida de mantenimiento. (Clements, Paul, 1996)

Restricciones constructivas: Una descripción arquitectónica proporciona planos parciales para el desarrollo, indicando los componentes y las dependencias entre ellos. Por ejemplo, una vista en capas de una arquitectura documenta típicamente los límites de abstracción entre las partes, identificando las principales interfaces y estableciendo las formas en que unas partes pueden interactuar con otras. (Clements, Paul, 1996)

Reutilización, o abstracción transferible de un sistema: La AS encarna un modelo relativamente pequeño, intelectualmente tratable, de la forma en que un sistema se estructura y sus componentes se entienden entre sí; este modelo es transferible a través de sistemas; en particular, se puede aplicar a otros sistemas que exhiben requerimientos parecidos y puede promover reutilización en gran escala. El diseño arquitectónico soporta reutilización de grandes componentes o incluso de frameworks en el que se pueden integrar componentes. (Garlan, David, 2004)

Evolución: La AS puede exponer las dimensiones a lo largo de las cuales puede esperarse que evolucione un sistema. Haciendo explícitas estas “paredes”perdurables, quienes mantienen un sistema pueden comprender mejor las ramificaciones de los cambios y estimar con mayor precisión los costos de

las modificaciones. Esas delimitaciones ayudan también a establecer mecanismos de conexión que permiten manejar requerimientos cambiantes de interoperabilidad,prototipado y reutilización.

Análisis: Las descripciones arquitectónicas aportan nuevas oportunidades para el análisis, incluyendo verificaciones de consistencia del sistema, conformidad con las restricciones impuestas por un estilo, conformidad con atributos de calidad, análisis de dependencias y análisis específicos de dominio y negocios.

Administración: La experiencia demuestra que los proyectos exitosos consideran una arquitectura viable como un logro clave del proceso de desarrollo industrial. La evaluación crítica de una arquitectura conduce típicamente a una comprensión más clara de los requerimientos, las estrategias de implementación y los riegos potenciales.

1.2.2 Descripción actual del dominio del problema.

El proyecto SIG-Móviles perteneciente al Centro (GEYSED), está orientado al desarrollo de Sistemas de Información Geográfica (SIG) para dispositivos móviles. Existen dos líneas de desarrollo en las cuales dicho proyecto está dividido, la primera es la que se centra en el desarrollo de aplicaciones informáticas basadas en el protocolo WAP denominada MóvilMap y la segunda desarrolla aplicaciones de escritorio que están orientadas al aprovechamiento de las funcionalidades que brindan los dispositivos mencionados.

La línea correspondiente al sistema MóvilMap está basada en el desarrollo de un portal de acceso inalámbrico que utiliza el protocolo WAP y su interacción con el servidor de mapas MapServer para constituir un Sistema de Información Geográfica para dispositivos móviles. La idea original de su concepción se centra en el desarrollo actual de la WEB y la creciente extensión de la misma a protocolo inalámbrico; los SIG que se desarrollan en el departamento Geoinformática cuentan con un producto denominado GENESIG el cual se pretende extender a protocolo WAP mediante la herramienta MóvilMap.

Una parte del equipo de desarrollo se encuentra estudiando diferentes herramientas de código abierto ya que existen numerosos software que implementan funcionalidades que no se tienen que hacer desde el inicio, solamente se deben de reutilizar y extender a las nuevas funcionalidades que necesite el proyecto,

las cuales serán desarrolladas en el ambiente de los dispositivos móviles, esta actividad complementa el trabajo de la línea de investigación basada en aplicaciones de escritorio para dispositivos móviles.

La línea MóvilMap basada en el uso del protocolo WAP ha identificado un conjunto de requisitos tanto funcionales como no funcionales mediante los cuales debe regirse el sistema. Para un diseño correcto del mismo se hace necesario identificar los componentes que conformarán la aplicación informática a desarrollarse y las relaciones existentes entre los mismos, aspectos que maneja, diseña y detalla la arquitectura de software como tal.

Obteniéndose un diseño arquitectónico basado en tecnologías no privativas se desarrollará una aplicación informática bien diseñada, con una correcta interacción entre los componentes que la conforman y una mejor representación de los requisitos funcionales y no funcionales identificados. Lo anteriormente planteado brindará mejoras a la sociedad ya que dicho sistema informático garantizará mayor eficiencia en la toma de desiciones en un tiempo relativamente rápido respecto a términos geográficos y de localización. Los usuarios finales obtendrán más portabilidad en las aplicaciones, esta vez desde dispositivos móviles lo cual posibilitará la consulta de la información de forma sencilla y rápida. Además este tipo de software representa un rublo exportable para el país una vez finalizado su desarrollo, lo cual con una correcta promoción y obtención de clientes extranjeros el pueblo cubano obtendría ganancias monetarias a mediano plazo.

1.2.3 Situación problemática

El proyecto SIG-Móviles no cuenta con una metodología que guíe a los involucrados en el desarrollo del proyecto. Tampoco cuenta con un arquitecto de software responsable de definir las vistas de la arquitectura de la aplicación, dar soporte tecnológico a desarrolladores, clientes y expertos en negocios, conceptualizar y experimentar con distintos enfoques arquitectónicos.

Se hace necesario analizar los estilos y patrones arquitectónicos alternos para derivar la estructura que corresponda con los requisitos del cliente. Además crear documentos de modelos y componentes, especificaciones de interfaces, validar la arquitectura contra requerimientos y suposiciones. El responsable de estas tareas debe tener claro conocimiento de la estrategia de negocio de la organización,

de los ciclos de planificación, proceso de toma de decisiones y conocimiento del contexto de la organización (competencia, productos y factores principales que afectan el éxito de la organización).

1.3 Análisis de propuestas arquitectónicas para dispositivos móviles sobre protocolo WAP.

A nivel mundial se han presentado definiciones de arquitecturas para entornos donde se utiliza el protocolo WAP para la comunicación inalámbrica con dispositivos móviles, a continuación se presentan las más utilizadas y de mayor importancia:

1.3.1 Arquitectura en capas para aplicaciones sobre protocolo WAP.

El principal objetivo de este tipo de arquitectura es extender las funcionalidades del protocolo WAP con servicios más sofisticados como son el soporte para efectuar transacciones y el mantenimiento de las aplicaciones luego que se efectúe una desconexión. Esta arquitectura en capas cuenta con cuatro capas en su totalidad, las cuales integran elementos de los componentes estándares del protocolo WAP y los dominios WWW mediante el protocolo HTTP, esto último permite realizar las siguientes funcionalidades:

- Establecer comunicaciones entre los navegadores de los dispositivos móviles y las aplicaciones servidoras con su lógica de negocio.
- Permitir el acceso remoto a base de datos.
- Dar soporte a los servicios de transacciones.
- Mejoras en la administración de la conexión y desconexión de los servicios de transacciones.

Las 4 capas mencionadas en esta arquitectura son las siguientes:

- Capa de Usuario: Representa la terminal inhalámbrica con su navegador wml u otro lenguaje de marcado que soporte el dispositivo móvil.
- Capa de Red: Representada por la puerta de enlace WAP que actúa como un proxy HTTP. Es puerta de enlace convierte el protocolo WAP proveniente de los dispositivos móviles al protocolo HTTP que soporta el Servidor WEB donde está la aplicación.

- ❖ Capa de Aplicación: Compuesta por el servidor de aplicaciones que implementa la lógica de negocio.
- ❖ Capa de Datos: Compuesta por los servidores de base de datos.

Las interfaces de comunicación entre los niveles adyacentes se muestran a continuación:

Interfaz	Protocolo	Dominio
Capas Usuario - Red	WAP	Red Inalámbrica. Comunicación mediante GSM.
Capas Red - Aplicación	HTTP	Red Fija (Internet)
Capas Aplicación - Datos	TCP/IP	Red Fija (Red de Área Local LAN)

Tabla 1. Interfaces de comunicación

1.3.2 Arquitectura Modelo-Vista-Controlador para aplicaciones sobre protocolo WAP.

Esta arquitectura es la más utilizada para el entorno de comunicación mediante protocolo WAP. La misma se centra en el patrón Modelo-Vista-Controlador explicado detalladamente en el Capítulo #2 del presente trabajo. En el nuevo entorno se aprovecha dicho patrón de forma distribuida tanto en el lado cliente (dispositivos móviles según sus potencialidades) y el lado servidor (Servidor WAP, Servidores de Base de Datos y Servidores de Servicios WEB), con la posibilitad de tener las capas modelo, vista y controlador en ambos lados previamente mencionados según las necesidades del sistema.

Esta arquitectura es muy adaptable al uso de bibliotecas modernas y repositorios de dispositivos móviles que mejoran la implementación del protocolo WAP por ejemplo el repositorio WURFL y las bibliotecas de clases WALL. La arquitectura es adaptable completamente al uso de servicios WEB, servicios basados en localización, transacciones entre el lado servidor y el lado cliente, la interacción con Sistemas Gestores de Base de Datos y la administración de conexiones seguras.

1.3.3 Conclusiones sobre arquitecturas analizadas.

Las Arquitecturas mencionadas anteriormente son las más generales ya que encapsulan todos los elementos basados en las funcionalidades de las aplicaciones desarrolladas sobre protocolo WAP. La presentada como arquitectura basada en el patrón arquitectónico Modelo-Vista-Controlador se adapta más a las funcionalidades que se quieren lograr ya que se puede distribuir fácilmente cada una de las capas tanto en el lado cliente como en el servidor, además posibilita una mejor interacción con el repositorio de dispositivos móviles WURFL y la biblioteca WALL las cuales mejoran la implementación sobre el protocolo WAP y de forma general se puede hacer uso de las funcionalidades que brinda la arquitectura en capas antes mencionada.

Conclusiones Parciales

En el presente capítulo se trataron diferentes conceptos que son de vital importancia para el entendimiento y solución del problema planteado. Se analizaron propuestas arquitectónicas para dispositivos móviles sobre protocolo WAP con características y funcionalidades similares al de SIG para Dispositivos Móviles.

Capítulo II: Propuesta de arquitectura y tecnologías a utilizar.

Introducción

Existen numerosas tecnologías en el mundo de la informática que con el transcurso del tiempo tienden a evolucionar de manera progresiva. Por lo que se hace necesario en este capítulo realizar un estudio profundo de las mismas, para posteriormente seleccionar las que más se ajusten al diseño arquitectónico del software que se pretende desarrollar. Para esta selección se tendrán en cuentan características, funcionalidades y ventajas que justifiquen la propuesta realizada. Una vez definidas las herramientas, estas permitirán que el equipo de desarrollo del sistema trabaje con rapidez y eficiencia para lograr un producto de buena calidad.

2.1 Estilos arquitectónicos.

Un estilo describe una categoría de sistema que abarca un conjunto de componentes que realizan una función requerida por la aplicación, un conjunto de conectores que posibilitan la comunicación, la coordinación y cooperación entre los componentes, las restricciones que definen como se integran los componentes para conformar el sistema, y los modelos semánticos que facilitan al diseñador el entendimiento de todas las partes del sistema. (Pressman, Roger R, 2002)

Un estilo es una familia de sistemas de software en términos de su organización estructural, que define una forma de articulación u organización arquitectónica. El conjunto de los estilos cataloga las formas básicas posibles de estructuras de software. Los estilos arquitectónicos sirven para sintetizar y tener un lenguaje que describa la estructura de las soluciones, definen los patrones posibles de las aplicaciones y permiten evaluar arquitecturas alternativas con ventajas y desventajas conocidas ante diferentes conjuntos de requisitos no funcionales. (S, Roger y Pressman, Roger S, 2002)

Como pudo apreciarse anteriormente existen diversos conceptos, algunos más abarcadores que otros, pero todos convergen a un mismo criterio. Existen además varios grupos de estilos que han sido escritos por diferentes autores durante el transcurso de los años, a continuación se detallan los más usados y representativos:

- **Estilos de Flujo de Datos:** Esta familia de estilos se enfatiza en la reutilización y los cambios a determinados sistemas. Es apropiada para aplicaciones que implementan transformaciones de datos en pasos sucesivos. (Reynoso, Carlos y Kicillof, Nicolás, 2004) Ejemplares de la misma serían las arquitecturas de tubería-filtros y las de proceso secuencial en lote.

Una tubería es una popular arquitectura que conecta componentes computacionales (filtros) a través de conectores, de modo que las computaciones se ejecutan a la manera de un flujo. Los datos se transportan a través de las tuberías entre los filtros, transformando gradualmente las entradas en salidas. Debido a su simplicidad y su facilidad para captar una funcionalidad, es una arquitectura configurable cada vez que se trata de demostrar ideas sobre la formalización del espacio de diseño arquitectónico, igual que el tipo de datos en lista o estructurados lo fue en las especificaciones algebraicas o en los tipos de datos abstractos. (Raúl, De Villa, 2009)

El estilo tubería-filtros se percibe como una serie de transformaciones sobre sucesivas piezas de los datos de entrada. Los datos entran al sistema y fluyen a través de los componentes. En el estilo secuencial por lotes los componentes son programas independientes; el supuesto es que cada paso se ejecuta hasta completarse antes que se inicie el paso siguiente.

- **Estilos Centrados en Datos:** Esta familia es apropiada para sistemas que se fundan en acceso y actualización de datos para estructuras de almacenamiento. Subestilos característicos de la familia serían los repositorios, las bases de datos, las arquitecturas basadas en hipertextos y las arquitecturas de pizarra. (Raúl, De Villa, 2009)

Estos sistemas se han usado en aplicaciones que requieren complejas interpretaciones de proceso de señales (reconocimiento de patrones, reconocimiento de habla, entre otros.), o en sistemas que involucran acceso compartido a datos con agentes débilmente acoplados. También se han implementado estilos de este tipo en procesos en lotes de base de datos y ambientes de programación organizados como colecciones de herramientas en torno a un repositorio común. Muchos más sistemas de los que se cree están organizados como repositorios: bibliotecas de componentes reutilizables, grandes bases de datos y motores de búsqueda. Los estilos de pizarra se utilizan principalmente en el campo de la robótica, en

programación evolutiva, inteligencia artificial, gramáticas complejas, entre otros. Es decir, un sistema de pizarra se implementa cuando existen problemas para los que no se encuentra una solución analítica.

- **Estilos de Llamada y Retorno:** Esta familia de estilos se enfatiza en los cambios a sistemas o aplicaciones. Son los estilos más generalizados en sistemas en gran escala. Los miembros de esta familia son las arquitecturas de programa principal y subrutina, los sistemas basados en llamadas a procedimientos remotos, los sistemas orientados a objeto y los sistemas jerárquicos en capas. Se mencionan a continuación las arquitecturas que están dentro de este tipo de estilo debido a su importancia en el desarrollo de sistemas. (Raúl, De Villa, 2009)

  - Modelo Vista Controlador
  - Arquitecturas en Capas
  - Arquitecturas Orientadas a Objetos
  - Arquitecturas Basadas en Componentes

- **Estilos de Código Móvil:** Esta familia de estilos enfatiza la portabilidad. Ejemplos de la misma son los intérpretes, los sistemas basados en reglas y los procesadores de lenguaje de comando. Fuera de las máquinas virtuales y los intérpretes, los otros miembros del conjunto han sido rara vez estudiados desde el punto de vista estilístico. Los sistemas basados en reglas, que a veces se agrupan como miembros de la familia de estilos basados en datos, han sido estudiados particularmente. Dentro de este se encuentra a su vez el estilo:

  - Arquitectura de máquinas virtuales.

- **Estilos Peer-to-Peer:** Esta familia, también llamada de componentes independientes, se enfatiza en las transformaciones por medio de la separación de las diversas partes que intervienen en la computación. Consiste por lo general en procesos independientes o entidades que se comunican a través de mensajes. Cada entidad puede enviar mensajes a otras entidades, pero no controlarlas directamente.

Los mensajes pueden ser enviados a componentes nominados o propalados mediante *broadcast*[1]. (Devadas, Srinivas, 2001). Dentro de esta familia de estilos se encuentran:

- ✓ Arquitecturas Basadas en Eventos.
- ✓ Arquitecturas Orientadas a Servicios (SOA).
- ✓ Arquitecturas Basadas en Recursos.

2.2 Patrones arquitectónicos

Los patrones son formas de describir las mejores prácticas, buenos diseños, y encapsulan la experiencia de forma tal que es posible para otros reutilizar dicho conocimiento. Constituyen mecanismos cuyo objetivo es la solución de problemas que ocurren repetidamente dentro de un contexto muy bien definido. Las soluciones que son propuestas a través de patrones involucran algunas clases de estructuras que permiten contemplar los requisitos no funcionales.

Los patrones son los que definen la estructura de un sistema software, los cuales a su vez se componen de subsistemas con sus responsabilidades, también tienen una serie de directivas para organizar los componentes del sistema, con el objetivo de facilitar la tarea del diseño de tal sistema. (Devadas, Srinivas, 2001).

Los patrones existen en diversas áreas de interés y tecnologías. Según la escala o nivel de abstracción suelen clasificarse en patrones arquitectónicos, de diseño, organizativos, de análisis y de programación o idiomas.

Cada uno de estos patrones describe soluciones desde el análisis hasta el diseño y desde la arquitectura hasta la implementación, es decir, se encuentran presentes en todas las fases de desarrollo de un sistema. En este caso se profundizará en los patrones arquitectónicos, dentro de estos se encuentran:

[1]Transmisión de un paquete que será recibido por todos los dispositivos en una red.

2.2.1 Modelo Vista Controlador o MVC: describe una forma, muy utilizada de organizar el código de una aplicación separando los datos, la interfaz de usuario, y la lógica de control en tres componentes distintos. (Devadas, Srinivas, 2001)

Modelo: Administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).
Vista: Presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
Controlador: Recibe eventos del usuario, invoca servicios ofrecidos por el modelo y selecciona la vista adecuada para presentar los resultados.

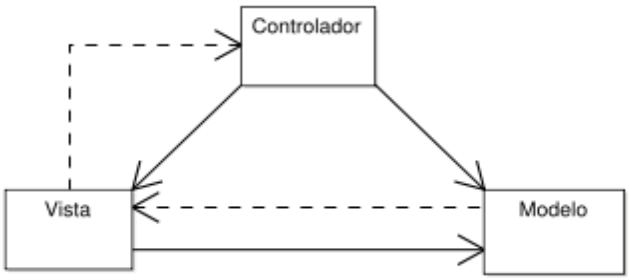

Figura 1. Patrón Modelo Vista Controlador (MVC).

2.2.2 Arquitectura en tres capas: Es un patrón en el que se divide el sistema en capas. La principal ventaja de este consiste en facilitar la gestión de los cambios en un sistema. Solo basta con trabajar con el nivel que se necesite realizar el cambio sin tener que tocar el resto de los niveles. Por lo general se divide el sistema en tres capas fundamentales, presentación, lógica de negocio y acceso a datos. Esto facilita que cualquier cambio que se realice sobre alguna capa sea transparente respecto a las otras capas que se implementan en el sistema. La abstracción de los desarrolladores de los otros niveles es además otra de las ventajas, estos solo necesitan conocer la interfaz que conecta cada una de las capas.

- Capa de presentación: Esta capa es la que permite la interacción con los usuarios de la aplicación. Su función principal es gestionar los datos con el usuario.

- Capa de Negocio: Aquí se establecen todas las reglas del negocio que deben ser cumplidas. Esta interactúa con la capa de presentación para recibir los datos provenientes del usuario, así como para enviar respuestas a dichas peticiones. Además se relaciona con la capa de Datos para gestionar la información que se almacena en los medios de almacenamiento.
- Capa de Datos: En esta reside toda la información necesaria para la aplicación. Además es la encargada de la gestión de los mismos.

2.2.3 Arquitectura Orientada a Servicios: La Arquitectura basada en Servicios (SOA, por sus siglas en inglés) está diseñada para establecer una integración con otras aplicaciones, independientemente del modo de acceso a las funcionalidades que esta brinde. Normalmente cuando se habla de SOA se trata el término de servicios WEB, lo cual no es necesario para implementar una arquitectura SOA. Los servicios para comunicarse entre sí se basan en una definición formal independiente de la plataforma y del lenguaje de programación, como el lenguaje de descripción de servicios Web. La definición de interfaces permite encapsular la implementación, lo que la hace independiente del desarrollador, del lenguaje de programación o de la tecnología de desarrollo. (Devadas, Srinivas, 2001)

SOA permite mejorar la toma de decisiones ya que facilita unificar la información con mejor calidad, no existen limitaciones con las tecnologías de la información y facilita una mayor capacidad de respuesta a los clientes ya que las aplicaciones son más flexibles, y seguras. El comportamiento de la red constituye uno de los problemas más relevantes que proporciona esta arquitectura. Ejemplo de ello es el tamaño de los mensajes respecto a estado de la red, mala configuración y la inseguridad en las comunicaciones.

2.3 Metodologías de desarrollo

El proceso de desarrollo de software en su inicio no solo se necesita de un personal altamente calificado, sino que se hace necesario seguir ciertas pautas predefinidas en el mismo y llevar un comportamiento metódico, es decir seguir una metodología. Esta sería la encargada de garantizar la calidad, eficacia y de reducir la pérdida de tiempo en un proyecto.

Existen dos tipos de metodologías de desarrollo, ágiles y tradicionales. Las metodologías tradicionales (formales) se focalizan en documentación, planificación y procesos (plantillas, técnicas de administración,

revisiones). Esta se ajusta a proyectos a largo plazo de gran envergadura, con equipos de desarrollo grandes donde la organización sea fundamental. Además es aconsejable cuando se requiera una documentación amplia que detalle cada elemento para lograr un entendimiento posterior del software u otras razones. En cambio las metodologías ágiles eliminan el burocratismo de las robustas, que en ocasiones resulta contraproducente emplearlas. Estas se centran en la capacidad de las personas implicadas en el proceso, evitando ir al detalle en cada paso, pero obteniendo el mayor fruto del trabajo de cada integrante. Es perfecta para equipos de desarrollo pequeños, con gran experiencia, y en proyectos que lo fundamental sea el producto final y la rapidez con que se concluya, sin que la documentación sea primordial.

Metodologías Ágiles	**Metodologías Tradicionales**
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios.
Impuestas internamente (por el equipo de desarrollo).	Impuestas externamente.
Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas/normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos artefactos.	Más artefactos.
Pocos roles.	Más roles.
Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

Tabla 2. Comparación entre metodologías ágiles y tradicionales.

Dentro de las metodologías ágiles se encuentra Programación Extrema (XP) y dentro de las metodologías tradicionales Rational Unified Process (RUP) y Microsoft Solution Framework (MSF). Estas serán descritas a continuación:

2.3.1 Metodología RUP (*Rational Unified Process*).

El Proceso Unificado de Rational es un proceso de ingeniería del software. Proporciona un acercamiento disciplinado a la asignación de tareas y responsabilidades en una organización de desarrollo. Su propósito es asegurar la producción de software de alta calidad que se ajuste a las necesidades de sus usuarios finales con unos costos y calendario predecibles. (Pressman, Roger R., 2002)

En definitiva RUP es una metodología de desarrollo de software que intenta integrar la mayoría de los aspectos a tener en cuenta durante el ciclo de vida del software, con el objetivo de hacer abarcables tanto pequeños como grandes proyectos software. Además Rational proporciona herramientas para todos los pasos del desarrollo así como documentación en línea para sus clientes. Las características principales de RUP son: Dirigido por casos de uso, Centrado en la arquitectura e Iterativo e incremental.

La metodología RUP divide el desarrollo en 4 fases que definen su ciclo de vida que se subdividen en iteraciones y nueve flujos de trabajo, seis flujos de ingeniería y tres de apoyo (Pressman, Roger R., 2002) (Ver anexo 1). Las **Fases** son: Inicio, Elaboración, Construcción y Transición. Los **Flujos de ingeniería** son: Modelación del negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba (Testeo) e Instalación. Los **Flujos de apoyo** son: Administración del proyecto, Administración de configuración y cambios y Ambiente.

RUP establece diferentes roles dentro del desarrollo del proyecto, sin duda los roles son una herramienta muy útil para designar los encargados del proceso de desarrollo de un sistema informático, uno de los roles más importantes es el rol del arquitecto de software.

¿Qué es un arquitecto de software? Un arquitecto es una persona equipo u organización responsable por la arquitectura del sistema. Es aquel que se esfuerza por mantener la visión del producto final, así como los procesos que se derivan de los requisitos y de la implementación de esa visión. Un arquitecto debe

entender el propósito del software, debe ser capaz de ver todos los usos y problemas para poder diseñar un software funcional. Las características de un arquitecto de software (Contreras, Domingo, 2010):

- Posee competencia técnicas y conocimiento tecnológico.
- Investiga nueva tecnologías.
- Comprende los frameworks arquitectónicos y las mejores prácticas.
- Desarrolla rápidamente un profundo conocimiento en una tecnología.
- Tiene liderazgo y autoridad.
- Sigue y dirige a la vez, siendo además un buen comunicador.
- Posee un amplio dominio del negocio y tiene una amplia visión.
- Identifica e interactúa con los interesados en el proyecto para asegurarse que sus necesidades son satisfechas.

De manera general el arquitecto de software entiende y tiene un amplio conocimiento o sea una visión global del proceso de desarrollo del software garantizando así que todos los miembros del equipo trabajen de forma coordinada.

El arquitecto de software tiene la responsabilidad global de dirigir las principales decisiones técnicas, expresadas como la AS. Esto habitualmente incluye la identificación y la documentación de los aspectos arquitectónicamente significativos del sistema, que incluye Línea Base de la Arquitectura, Documento de Descripción de Arquitectura de Software, Informe del Levantamiento de Información para la Arquitectura de Información, Arquitectura de Información, Modelo de análisis, Modelo de diseño, Modelo de despliegue, Modelo de implementación, Prototipos de arquitectura, visualizar el comportamiento del sistema, crear los planos del sistema, definir la forma en la cual los elementos del sistema trabajan en conjunto, proveer una guía técnica clara y consistente. Además, establecer la estructura global de cada vista de la arquitectura: la descomposición de las vistas, el agrupamiento de elementos y las interfaces de los mismos. (Raúl, De Villa Cano,2009)

La metodología RUP señala que los arquitectos moldean el sistema para darle una forma. Es esta la forma: la arquitectura, que debe diseñarse para permitir que el sistema evolucione, no sólo en su desarrollo inicial, sino también a lo largo de las futuras generaciones. Para encontrar esa forma, los arquitectos deben trabajar sobre la comprensión general de las funciones clave, es decir, sobre los casos de uso clave del sistema.

2.3.2 Programación Extrema o XP (*Extreme Programming*).

Es una metodología para el desarrollo de software y consiste básicamente en ajustarse estrictamente a una serie de reglas que se centran en las necesidades del cliente para lograr un producto de buena calidad en poco tiempo. La Programación Extrema es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software. Promueve el trabajo en equipo, enfocándose en todo momento en el aprendizaje de los desarrolladores y estableciendo un buen clima de trabajo. (Alarcón Almenare, Bertha, 2009).

Este tipo de método está basado en una realimentación continuada entre el cliente y el equipo de desarrollo con una comunicación fluida entre todos los participantes; simplificando las soluciones implementadas para los múltiples cambios. Este tipo de programación es la adecuada para los proyectos con requisitos imprecisos, muy cambiantes y con un riesgo técnico excesivo.

Para un desarrollo exitoso con esta metodología es necesario que exista (Alarcón Almenare, Bertha, 2009):

- La comunicación, entre los clientes y los desarrolladores.
- La simplicidad, al desarrollar y codificar los módulos del sistema.
- La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

2.3.3 *Microsoft Solution Framework (MSF)*

Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. Es adaptable, escalable y flexible. Más que una metodología rígida de administración de proyectos, MSF es una serie de modelos que pueden adaptarse a cualquier proyecto de tecnología de información y además están encargados de planificar las diferentes partes implicadas en el desarrollo de dicho proyecto.
Concretamente MSF se compone de principios, modelos y disciplinas (S, Roger y Pressman, Roger S, 2002):

Los principios son: Promover comunicaciones abiertas, trabajar para una visión compartida, fortalecer los miembros del equipo, establecer responsabilidades claras y compartidas, focalizarse en agregar valor al negocio, permanecer ágil y esperar los cambios e invertir en calidad.

Los modelos son: Equipo de trabajo y proceso.

Las disciplinas son: Gestión de proyectos, control de riesgos y control de cambios.

2.4 Lenguaje Unificado de Modelado (UML).

El *Lenguaje Unificado de Modelado* (UML) permite modelar sistemas y describirlos con cierto grado de formalismo que puedan ser entendidos por los clientes o usuarios de aquello que se modela. Es el procedimiento que emplean los ingenieros para el diseño de software antes de pasar a su construcción. (S, Roger y Pressman, Roger S, 2002)

Sin embargo, desde el punto de vista puramente tecnológico, UML tiene una gran cantidad de propiedades que han sido las que, realmente, han contribuido a hacer de él un estándar para la industria de software. Algunas de las propiedades de UML como lenguaje de modelado estándar son:

- Concurrencia, es un lenguaje distribuido y adecuado a las necesidades de conectividad actuales y futuras.

- Reemplaza a decenas de notaciones empleadas con otros lenguajes.
- Modela estructuras complejas.
- Las estructuras más importantes que soportan tienen su fundamento en las tecnologías orientadas a objetos, tales como objetos, clase, componentes y nodos.
- Emplea operaciones abstractas como guía para variaciones futuras, añadiendo variables si es necesario.

Comportamiento del sistema: casos de uso, diagramas de secuencia y de colaboraciones, que sirven para evaluar el estado de las máquinas.

Otra característica de UML es que es independiente del lenguaje de implementación, de tal forma que los diseños realizados se pueden implementar en cualquiera de ellos y que soporte dicho modelado visual (principalmente lenguajes orientados a objetos). Permitiendo así una mayor escalabilidad del software y reutilización del diseño.

2.5 Herramientas CASE

Las herramientas CASE (Ingeniería de Software Asistida por Computadora) fueron creadas para el desarrollo de la ingeniería de software y constituyen un conjunto de aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. (Valcárcel Pérez, Victoria, 2010). Estas se acoplan con las metodologías para dar una forma de representar sistemas y suponen una forma de abstracción del código fuente, a un nivel donde la arquitectura y el diseño se hacen más aparentes y fáciles de entender y modificar. Cuanto mayor es un proyecto, más importante es el uso de una tecnología CASE. Entre las herramientas CASE orientadas a UML están: Poseidon for UML, ArgoUML, MagicDraw, Borland Together, Rational Rose, Visual Paradigm y Enterprise Architect, siendo las tres últimas las más utilizadas. Las mismas se explican a continuación:

2.5.1 Rational Rose

Es la herramienta que permite construir y desarrollar a través del UML los casos de uso en diferentes diagramas, modelando los flujos de trabajo por los que transita el desarrollo de un software. Permite la realización de los diferentes diagramas y la posterior generación del código, todo orientado a objetos. Posee librerías que facilitan la obtención de una ingeniería inversa sobre diferentes lenguajes como Java,

C++ y Visual Basic. Esta herramienta posibilita gestionar la evolución del ciclo de vida de un proyecto de software. Racional Rose aligera la implementación al automatizar los modelos arquitectónicos. Además, permite visualizar, entender y refinar los requerimientos y arquitectura antes de introducirse en el código. (S, Roger y Pressman, Roger S, 2002)

2.5.2 Visual Paradigm

Es una herramienta multiplataforma, permitiendo su uso en cualquier sistema operativo. Utiliza UML como lenguaje de modelado, permitiendo una rápida construcción de las aplicaciones con alta calidad. Permite dibujar diagramas de clases, y generar script para diferentes Sistemas Gestores de Bases de Datos, (SGBD en lo adelante). Permite una integración con sistemas de control de versiones que almacenan centralmente los artefactos y realizan un seguimiento de los cambios realizados sobre un proyecto. Los desarrolladores lo utilizan para facilitar el modelado simultáneo, almacenar los archivos de proyectos y hacer un seguimiento de los cambios**.** (Pressman, Roger R, 2002)

Tiene como características principales las siguientes: Licencia gratuita y comercial, generación de código en diferentes lenguajes de programación, generación de código para distintos SGBD, se puede integrar con diferentes IDE, genera documentación de lo modelado, y es fácil de instalar y actualizar.

2.5.3 *Enterprise Architect*

Es una plataforma avanzada para el modelado y diseño de software, sistemas o negocios. El ambiente de trabajo está potenciado por UML 2.1 y BPMN (*Business Process Modeling Notation* por sus siglas en inglés, Notación para el Modelado de Procesos de Negocio, por sus siglas en español), con herramientas que pueden proveerle una estructura competitiva en modelado de negocio, diseño de software, ingeniería de sistemas, arquitectura de empresas y gestión de requisitos. (Devadas, Srinivas, 2001). Esta herramienta posibilita la realización de ingeniería inversa a partir de un determinado número de lenguajes de programación. Es posible importar una base de datos a partir de sus esquemas, establece seguridad basada en roles para el trabajo en equipo, así como el control de versiones con cualquier herramienta compatible e importar códigos binarios desde .NET o Java. Además de presentar corrector ortográfico y una documentación de alta calidad, otra de sus principales características es que tiene soporte para Java, C#, C++, Delphi, Visual Basic, Python y PHP.

2.6 Sistemas Gestores de Base de Datos (SGDB)

Los sistemas gestores de bases de datos son un tipo de software muy específico, dedicado a servir de interfaz entre la datos de datos, el usuario y las aplicaciones. Es necesario un Sistema Gestor de Base de Datos (SGDB) para el desarrollo del proyecto ya que este es el encargado de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante para la organización. Son muchos los SGDB existentes pero los más conocidos son MySQL, SQL Server, Oracle y PosgreSQL. (Devadas, Srinivas, 2001)

2.6.1 MySQL

Es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones, licenciado bajo la GPL[2]. Este gestor fue creado por la empresa sueca MySQL AB, que mantiene el *copyright*[3] del código fuente del servidor SQL, así como la marca.

Aunque MySQL es software libre distribuye una versión comercial, que no se diferencia de la versión libre más que en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de no ser así, no sería factible la GPL.(Valcárcel Pérez, Victoria, 2010)

Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración. Las principales características de este gestor de bases de datos son las siguientes:

- Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.
- Soporta gran cantidad de tipos de datos para las columnas.
- Dispone de API's [4] en gran cantidad de lenguajes (C, C++, Java, PHP).

[2]Licencia pública general.

[3] Derecho de copia.

- Gran portabilidad entre sistemas.
- Soporta hasta 32 índices por tabla.
- Gestión de usuarios y passwords, manteniendo un buen nivel de seguridad en los datos.

2.6.2 *Oracle*

Es una herramienta cliente/servidor para la gestión de Base de Datos relacionales. Esta herramienta hace uso de los recursos del sistema informático en todas las arquitecturas de hardware para garantizar su aprovechamiento al máximo en ambientes cargados de información. Con nuevas funciones de autogestión, Oracle elimina las tareas administrativas lentas y propensas a errores, lo que permite a los administradores de las BD concentrarse en los objetivos estratégicos de la empresa en lugar de prevenir problemas de rendimiento y disponibilidad. (Devadas, Srinivas, 2001)

Oracle como uno de los sistemas de bases de datos más completos se destaca por soporte de transacciones, estabilidad, escalabilidad y soporte multiplataforma. Su dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco, ya que recientemente sufre la competencia del Microsoft SQL Server de Microsoft y la propuesta de otros RDBMS con licencia libre como PostgreSQL, MySql o Firebird. Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo GNU/Linux.

Es gestor que se destaca por las siguientes características: Manejo de grandes BD (hasta de terabytes), usuarios concurrentes, sistemas distribuidos, y portabilidad. (Devadas, Srinivas, 2001)

Oracle Spatial: Una opción de *Oracle Database Enterprise Edition*, incluye servicios Web y 3D completos para administrar todos los datos geoespaciales, incluso los datos de vectores y cuadrículas, la topología y los modelos de red. Está diseñado para satisfacer las necesidades de los sistemas avanzados de

---

[4]Interfaz de programación de aplicaciones o (inglés Application Programming Interface), es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usados generalmente en las bibliotecas.

información geográfica para aplicaciones como la administración de espacios, instalaciones y seguridad nacional/defensa. El formato espacial nativo y abierto de Oracle elimina el coste de los sistemas específicos de propiedad exclusiva y es compatible con todos los principales productos GIS. Sólo Oracle proporciona seguridad, rendimiento, escalabilidad y capacidad de gestión líderes para los activos de información espacial críticos. (Devadas, Srinivas, 2001)

2.6.3 PostgreSQL

El SGBD relacional orientado a objetos conocido como PostgreSQL está derivado del paquete Postgres escrito en Berkeley, distribuida bajo licencia BSD. Este gestor de bases de datos de código abierto más ofrece control de concurrencia multi-versión, soportando casi toda la sintaxis SQL (incluyendo subconsultas, transacciones, tipos y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación como pueden ser C, C++, Java, y Python. Al mismo tiempo ofrece una potencia adicional sustancial al incorporar los siguientes cuatro conceptos adicionales básicos en una vía en la que los usuarios pueden extender fácilmente el sistema: clases, herencia, tipos y funciones. PostgreSQL corre en la mayoría de los Sistemas Operativos más utilizados incluyendo, Linux y Windows. Las siguientes sonuna algunas de sus características:

SGBD Objeto-Relacional: Aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas, herencia, y *arrays*.

Altamente Extensible: Soporta operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario.

Integridad Referencial: Soporta integridad referencial, la cual es utilizada para garantizar la validez de la información de la base de datos.

Cliente/Servidor: PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor. Esta es similar al método del Apache 1.3.x para manejar procesos. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que se intente conectar a PostgreSQL.

PostGIS es un módulo que añade soporte de objetos geográficos a la base de datos objeto-relacional PostgreSQL, convirtiéndola en una base de datos espacial para su utilización en Sistema de Información Geográfica. Se publica bajo la licencia pública general de GNU. Ha sido desarrollado por la empresa canadiense Refraction Research, especializada en productos "*Open Source*" entre los que habría que citar a Udig. PostGIS es un producto que demuestra eficiencia, demostrando 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 conocida Oracle. (Devadas, Srinivas, 2001).

2.7 Lenguajes de Programación

Para posibilitar la comunicación entre las computadoras y los humanos se hace necesario de la existencia de los lenguajes de programación para lograr tal interactividad. Un lenguaje de programación está conformado por una serie de reglas sintácticas y semánticas que serán utilizadas por el programador y a través de las cuales creará un programa o subprograma; las instrucciones que forman dicho programa son conocidas como código fuente. (Rena, 2008)

Desde el surgimiento de los lenguajes de programación, han sido muchos los que han aparecido, cada uno con sus características específicas. Cada lenguaje tiene ventajas y desventajas que lo hacen más o menos potente ante una necesidad de software determinada, por lo que no se puede afirmar que exista un lenguaje que cubra todas las necesidades requeridas por un software. De aquí la importancia de que a la hora de seleccionar el lenguaje de programación para un proyecto de software se haga un análisis de estos a partir de las insuficiencias del mismo. Son muchos los lenguajes que pudieran satisfacer las necesidades del proyecto, pero se decidió solo realizar un análisis de los más conocidos. Entre estos se pueden mencionar :

2.7.1 PHP

PHP es considerado por muchos uno de los lenguajes más usados para sistemas Web a través de la arquitectura LAMP (Linux + Apache + MySQL + PHP), aunque las cosas han cambiado desde que MySQL fue adquirida por Sun Mycrosystems. Es un lenguaje interpretado de alto nivel que puede ser embebido en páginas HTML y ejecutado en el servidor. El código PHP es interpretado en el servidor Web y genera código HTML y otro contenido que el visitante puede ver (De Luz, Elizabeth, 2010). Dentro de sus características más notorias se pueden ver:

Alto rendimiento: PHP es muy eficiente, mediante el uso de un único servidor, puede servir millones de accesos al día.

Interfaces para una gran cantidad de sistemas de base de datos diferentes: Dispone de una conexión propia a todos los sistemas de base de datos. Además de MySQL, puede conectarse directamente a las bases de datos de PostgreSQL, Oracle, InterBase y Sybase.

Bibliotecas incorporadas para muchas tareas Web habituales: Como se ha diseñado para su uso en la Web, PHP incorpora numerosas funciones integradas para realizar útiles tareas relacionadas con la Web. Puede establecer conexiones a otros servicios de red, enviar correos electrónicos, trabajar con *cookies* y generar documentos PDF, todo con unas pocas líneas de código.

Facilidad de aprendizaje y uso: La sintaxis de PHP se basa en otros lenguajes de programación, principalmente en C y Perl.

Portabilidad: PHP está disponible para una gran cantidad de sistemas operativos diferentes, se puede escribir código de este lenguaje en todos los sistemas operativos gratuitos del tipo Unix, como Linux y sus versiones comerciales, además de las diferentes versiones de *Microsoft Windows*. Su código funcionará sin necesidad de aplicar ninguna modificación a los diferentes sistemas que ejecute PHP. (De Luz, Elizabeth, 2010).

Acceso al código abierto: Dispone de acceso al código fuente de PHP. A diferencia de los productos comerciales y de código cerrado, si desea modificar algo o agregar un elemento al programa, puede hacerlo con total libertad.

2.7.2 Java

Fue diseñado por la compañía *Sun Microsystems Inc*., con el propósito de crear un lenguaje que pudiera funcionar en redes computacionales heterogéneas (redes de computadoras formadas por más de un tipo de computadora), y que fuera independiente de la plataforma en la que se vaya a ejecutar. Esto significa que un programa de Java puede ejecutarse en cualquier máquina o plataforma. La compañía Sun describe el lenguaje Java como "simple, multitarea, orientado a objetos, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico". (Peñalver, 2008)

2.7.3 Python

Se utiliza como lenguaje de programación interpretado, orientado a objetos y permite dividir el programa en módulos reutilizables desde otros programas escritos en Python. Permite varios estilos de programación como programación orientada a objetos, programación orientada a aspectos, programación estructurada y programación funcional. Usa tipo de dato dinámico, resolución dinámica de nombres, lo que enlaza un método y un nombre de variable durante la ejecución del programa. (Pilgrim, Mark, 2009)

Python fue diseñado para ser leído con facilidad, en vez de delimitar los bloques de código mediante el uso de llaves utiliza la indentación[5]. Es un lenguaje libre, de código abierto, sencillo y rápido de programar. Entre sus ventajas, además de las ya mencionadas se encuentra el hecho de que es multiplataforma y portable. En Python, todo es un objeto y además soporta herencia múltiple y polimorfismo.

[5] Significa mover un bloque de texto hacia la derecha insertando espacios o tabuladores para separarlo del texto adyacente.

2.8 Entorno de Desarrollo Integrado (IDE)

Un entorno de desarrollo integrado (en inglés *integrated development environment*) es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien poder utilizarse para varios. (Gómez, Francisco, 2009)

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. En los siguientes acápites se dará una breve descripción de algunos de los IDEs más utilizados en el desarrollo de aplicaciones o sistemas informáticos:

2.8.1 Eclipse

Es un entorno de desarrollo integrado de código abierto multiplataforma que permite el desarrollo en varios lenguajes por sus características modulares. Cuenta con una comunidad que desarrolla constantemente nuevas extensiones y mejoras las existentes. (Toledo, Álvarez, 2010)

El eclipse permite incorporar el módulo de PHP *Development Tools* (PDT) 2.0 el cual es una popular herramienta de código abierto para el desarrollo en PHP. Ofrece todas las capacidades básicas de edición de código que los desarrolladores necesitan. La versión 2.0 se enfoca en el apoyo a la programación orientada a objeto para PHP. Entre las características de esta versión se encuentran (Toledo, Álvarez, 2010):

- ✓ Rápida y fácil navegación por la jerárquica de objetos PHP.
- ✓ Método de navegación que permite una fácil búsqueda de código PHP.
- ✓ Editor sensible al contexto, provee resaltado de código, asistente de código y autocompletado de código.
- ✓ Soporte para el debug de código PHP.
- ✓ Resaltador de sintaxis.
- ✓ Compilación en tiempo real.
- ✓ Asistentes (wizards) para creación de proyectos, clases y pruebas.

2.8.2 Zend Studio

Es un editor de texto para páginas PHP que proporciona un número de ayudas desde la creación y gestión de proyectos hasta la depuración del código, esta última es una herramienta muy interesante que permite ejecutar páginas y conocer en todo momento el contenido de las variables de la aplicación y las variables del entorno como las *cookies*[6]. Existen dos ediciones: Standard y Professional. Zend Studio divide sus funcionalidades en dos partes: la del cliente y la del servidor las cuales se instalan por separado. La parte del cliente contiene la interfaz de edición y además permite hacer depuraciones simples de scripts. (Studio, Zend, 2010)

Entre sus principales características se destaca la de constituir un IDE líder para PHP, sin embargo ofrece soporte básico para otros lenguajes Web, como HTML, JavaScript y XML. Proporciona la visualización, edición y la capacidad de ejecución para bases de datos populares que poseen lenguaje SQL, incluyendo MySQL, Oracle, IBM DB2 y Cloudscape, Microsoft SQL Server, SQLite y PostgreSQL; es desarrollado el despliegue de código rápido y el control de calidad; soporte de servicios WEB (SOAP) y posee un API de rendimiento. (Studio, Zend, 2010). Además de la detección de errores de sintaxis en tiempo real tiene soporte para gestión de grandes proyectos de desarrollo.

2.8.3 NetBeans

Es una plataforma para el desarrollo de aplicaciones de escritorio usando el lenguaje Java y un entorno de desarrollo integrado (*IDE*) para desarrollar bajo esta plataforma, pero también admite otros lenguajes de programación como C, PHP y C++ mediante los cuales se pueden desarrollar aplicaciones. (NetBeans, 2008)

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de Java escritas para interactuar con las API de NetBeans y un archivo especial (*manifest file*) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las

[6]Es un fragmento de información que se almacena en el disco duro del visitante de una página web a través de su navegador, a petición del servidor de la página.

aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

Sirve a los programadores para escribir, compilar, depurar y ejecutar programas. Dado que cuenta con el mejor soporte a estándares industriales de la tecnología Java, el proyecto NetBeans ha hecho que el desarrollo de aplicaciones Java de tipo empresarial sea rápido y sencillo. (NetBeans, 2008)

Dispone de soporte para crear interfaces gráficas de forma visual sin importar donde se instale sea Linux, Windows u otro sistema operativo, el funcionamiento del programa creado será igual. Permite elaborar potentes aplicaciones de escritorio y para dispositivos portátiles como teléfonos móviles o Pocket PC sin cambiar la forma de programar. Se integra con control de versiones, así como se pueden desarrollar aplicaciones visuales con solo arrastrar y soltar objetos sobre la interfaz de un formulario.

2.9 Bibliotecas de clase Wall y Wurfl

WURFL (*Wireless Universal Resource File*), en español (Archivo de Recursos Universal para Móviles) y **WALL** (*The Wireless Abstraction Library*), en español (Biblioteca de Abstracción Móvil) son un conjunto de ficheros y librerías orientadas a facilitar el desarrollo de aplicaciones de cliente ligero, con una gran portabilidad entre distintos tipos de navegadores embebidos dentro de clientes móviles (WAP, navegadores cHTML, iMode, XHTML). (Debnath, Narayan, 2010)

Mediante su uso, se puede construir aplicaciones visualizables en diferentes entornos, adaptadas a las capacidades disponibles en cada dispositivo. Así, si un dispositivo puede mostrar gráficos en color, se usarán estos, y en caso de visualización en blanco y negro, otros gráficos diferentes, o ningún gráfico si el dispositivo no dispone de estas funcionalidades. Así, el mismo aplicativo puede emplearse en distintos dispositivos sin tener que poseer varias versiones o codificar usando muchos bloques condicionales en función del elemento. (Tilley, Scott, 2008)

Wurfl es el repositorio de terminales más extendido, y sus librerías son usadas de forma extensiva por gran parte de la comunidad de desarrolladores de aplicaciones para dispositivos móviles. Es un fichero de configuración XML el cual contiene información acerca de características y capacidades de los

dispositivos para una variedad de dispositivos móviles. La información de los dispositivos es contribución de desarrolladores de todo el mundo y el WURFL es actualizado de forma frecuente reflejando los nuevos dispositivos móviles que entran en el mercado. (Debnath, Narayan, 2010)

Wall es una biblioteca de tags JSP que permite al desarrollador escribir páginas móviles parecidas al HTML plano, mientras que entrega WML, C-HTML y XHTML Perfil Móvil al dispositivo el cual origina peticiones HTTP, dependiendo de las habilidades actuales del dispositivo en sí mismo. Las habilidades del dispositivo son consultadas dinámicamente usando la API de WURFL. (Tilley, Scott, 2008)

Las librerías vistas anteriormente están implementadas en los lenguajes de programación Java y PHP, se hará uso de estas porque el sistema a desarrollar se pretende implementar con el lenguaje PHP. Además se desea seguir con la línea de la plataforma GENESIG.

2.10 Servidor de Mapas

Los servidores de mapas permiten al usuario la máxima interacción con la información geográfica. Por un lado el usuario o cliente accede a la información en su formato original, de manera que es posible realizar consultas tan complejas como las que haría un SIG. Un servidor de mapas funciona enviando, a petición del cliente, desde su browser o navegador de internet, una serie de páginas HTML (normalmente de contenido dinámico DHTML), con una cartografía asociada en formato de imagen (por ejemplo, una imagen GIF o JPG sensitiva). Un servidor de mapas es, de hecho, un SIG a través de internet.

2.10.1 MapServer

Es un ambiente de desarrollo multiplataforma de código abierto para construir aplicaciones web que manejan información referenciada geográficamente. Su función es proveer fácilmente vía Web, datos que normalmente requieren de un software específico de geoprocesamiento. Posee una estructura cliente-servidor, donde el usuario solo necesita estar conectado a Internet y utilizando un navegador podrá tener acceso a las informaciones referenciadas geográficamente. (Pascuzzi, Domenico, 2001)

MapServer es el software libre para la realización de mapas más extendido y con una numerosa comunidad de usuarios. Presenta diferentes métodos de utilización según plataforma y lenguajes de programación. El procesamiento es realizado del lado del servidor. Programado en el lenguaje C, accede

a todos los formatos soportados por OGR y GDAL, y permite programación con PHP y otros lenguajes. (Pascuzzi, Domenico, 2001). Sus características principales son:

- Formatos vectoriales soportados: ESRI shapefiles, PostGIS, ESRI ArcSDE, GML.
- Formatos raster soportados: TIFF/GeoTIFF, GIF, PNG, ERDAS, JPEG y EPPL7.
- Soporta fuentes TrueType.
- Sencillez de configuración y administración.
- Plataformas sobre las que se puede operar.
- Velocidad de accesos a datos.

2.10.2 GeoServer

Es una aplicación de código abierto construida en Java que permite publicar datos geoespaciales usando el Web Map Server. Se distingue por tener instaladores sencillos y herramientas de configuración basadas en red. (Pascuzzi, Domenico, 2001) Entre sus características principales se encuentran:

- Multiplataforma: Corre bajo plataformas Linux/Apache, Mac y Windows.
- Salidas como GML, JPEG, GIF, PNG, SVG, KML y PDF.
- Soporte robusto para PostGIS, Oracle, ArcSDE, DB2 y Shapefiles.
- Soporte para transacciones atómicas en datos de respaldo con WFS-T.

2.11 Herramientas a utilizar

Patrón arquitectónico: Para desarrollar la base arquitectónica del proyecto SIG-Móviles se propone el patrón Modelo Vista Controlador (MVC) que forma parte del estilo arquitectónico de llamada y retorno. En dicho patrón cada una de las partes son independientes, la comunicación entre ellas es mediante interfaces, que abstraen sus estructuras internas. Esto permite desarrollar el modelo, la vista y el controlador de forma independiente, así como realizar modificaciones en sus partes, sin afectar a las demás. (Cross, Anna, 2010)

Los equipos de trabajos pueden ser divididos por roles de acuerdo a los tres componentes de esta arquitectura. Los encargados de desarrollar las funcionalidades relacionadas con la manipulación de los datos pueden especializarse en el Modelo. Los diseñadores de la interfaz se vinculan directamente con la

vista, siendo este el punto de interacción con el usuario, y donde se muestran los datos. Si la información no conlleva un procesamiento de datos profundo, se puede acceder directamente a las funciones de acceso a datos. Si las funcionalidades son complejas, que implique un procesamiento de datos relativamente grande se deben realizar en la capa Controladora. Para especializarse en este componente es necesario conocer cuáles serán las entradas desde la vista y la interfaz de comunicación con el modelo. Requiere de un mejor dominio de todo el entorno, pero no es necesario profundizar tanto en las particularidades de cada componente. El procesamiento es independiente de la forma en que se muestra o almacena la información. (Cross, Anna, 2010)

Al emplear esta arquitectura, en los equipos de desarrollo no existirá una sola persona que lleve la carga principal del producto y se convierte en indispensable para que funcione completamente, por lo que se fomenta el concepto de trabajo en equipo, que favorece la calidad y rapidez de desarrollo del sistema.

Metodología de desarrollo: De las metodologías anteriormente vistas la que más se adapta a las condiciones del sistema que se quiere desarrollar es RUP. La misma se define con el propósito de asegurar la producción de un software de alta calidad que se ajuste a las necesidades de los usuarios finales con unos costos y calendario predecibles.

Se utilizará como metodología de desarrollo debido a que es un proceso que provee un acercamiento disciplinado a la asignación de tareas y responsabilidades. RUP intenta integrar todos los aspectos a tener en cuenta durante todo el ciclo de vida del software, con el objetivo de hacer abarcables tanto pequeños como grandes proyectos software. También es conveniente pues utiliza el lenguaje unificado de modelado (UML) para preparar todos los diseños del sistema de software.Otra ventaja que se tuvo en cuenta es que debido a que se puede ver la evolución del software en cuatro fases (Inicio, Elaboración, Construcción, Transición), al final de las cuales, y tras una serie de iteraciones, establece objetivos a alcanzar bien definidos.

RUP desde sus inicios cuenta con una documentación profunda y detallada de todo el proceso en sentido general que permite tener un mayor control y seguimiento del ciclo de desarrollo. Propone varias iteraciones en el período de producción del software facilitando la corrección de errores y mejoras del

software a medida que avanza el proyecto, lo que permite a los desarrolladores lograr un producto con gran calidad.

Lenguaje unificado de modelado: La metodología de desarrollo RUP que se usará, utiliza UML en su versión 2.0 como lenguaje de modelado, éste es más eficiente y apropiado como medio potente y efectivo para describir, administrar y modelar el desarrollo de software. El proceso de RUP combinado con UML constituye la metodología más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Por lo que se define UML como lenguaje de modelado, además de las características que se presentaron anteriormente.

Herramienta CASE: Para el modelado del sistema se propone Visual Paradigm utilizando UML para la confección de diagramas y generación de documentación. Esta tecnología posibilita el desarrollo completo del software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de la aplicación proporcionando una alta calidad, mejoras y a un menor tiempo de desarrollo.

Visual Paradigm permite además, representar los diferentes tipos de diagramas de clases, generar código desde diagramas y viceversa; genera la documentación del mismo automáticamente en varios formatos como Web o Pdf (Portable Document Format), la cual puede ser utilizada para la conformación de los documentos del expediente de proyecto. La herramienta CASE también facilita demostraciones interactivas, abundantes tutoriales y proyectos UML. Soporta múltiples usuarios trabajando sobre el mismo proyecto, además de permitir el control de versiones. Otra ventaja que tiene con respecto al Rational y al Enterprise Architect es que es una herramienta multiplataforma, permitiendo su uso en cualquier sistema operativo y presenta una licencia comercial y gratuita.

Lenguaje de Programación: Se propone PHP porque fue con el cual se implementó la plataforma GENESIG y el proyecto SIG-Móviles será una extensión de la misma. Existe voluntad para continuar con dicho lenguaje debido a la experiencia por los desarrolladores anteriores y a la cantidad de funcionalidades hechas, ya que posteriormente pueden ser reutilizadas.

Específicamente PHP en su versión 5.0 posee características que lo tributan como la herramienta que posee la mejor mezcla entre rendimiento y flexibilidad a la hora de realizar páginas Web dinámicas. Aunque UNIX es el sistema operativo donde se pueden aprovechar mejor sus prestaciones puede ser utilizado en distintos sistemas operativos. Es un lenguaje que posee una amplia documentación por lo que resulta fácil de aprender, posee una biblioteca que aumenta a medida que sus nuevas versiones van surgiendo, además cuenta con disímiles funciones fáciles de utilizar siendo solo necesario hacer llamadas de forma apropiada y especificar los parámetros para realizar tareas como el acceso a BD, encriptación, creación de PDF y tratamiento de ficheros.

Entre sus principales características se destaca que es libre, multiplataforma y cuenta con muchas funcionalidades de manera nativa. Tiene comportamiento modular, brinda la posibilidad de adicionar nuevas funcionalidades a través de nuevos módulos. Permite conectarse a varios sistemas de bases de datos brindando múltiples funcionalidades, destacando en este aspecto la conectividad con PostgreSQL y MySQL que son gestores muy utilizados en el desarrollo de aplicaciones web, destacando que los desarrolladores están preparados en este lenguaje. Permite el empleo de Programación Orientada a Objetos, no requiere que se le especifique el tipo de datos de las variables y maneja excepciones a partir de la versión 5.0. Php se integrará con el modulo php5 MapScript, el cual tiene funcionalidades para la configuración y peticiones al servidor de mapas MapServer.

Sistema gestor de base de datos: Los gestores Oracle y SQLServer a pesar de ser herramientas potentes, son privativas y su adquisición es muy costosa, por lo que se dificulta su acceso. MySql es Open Source, muy rápido y robusto para aplicaciones pequeñas y medianas, el sistema que se quiere desarrollar es grande y complejo; por lo que no es una opción escoger este SGDB. Debido a las políticas de migración a software libre de la universidad se hace necesario utilizar PostgreSQL en su versión 8.3, ya que este posee características de código abierto, constituyendo uno de los SGBD libres más avanzado. Es el que más se adecua a las necesidades de la aplicación, para almacenar grandes volúmenes de datos geológicos y permitir el acceso de diferentes usuarios al mismo tiempo, acreditando permiso a cada uno de ellos. Es multiplataforma y orientado a objetos, esto permite que cada uno de los componentes de la estructura de la BD pueda ser tratado como un objeto, característica de suma importancia en el sistema que se implementará, facilitando enormemente la Importación de Bases de Datos Geográficas.

PostgreSQL permite la administración y la escalabilidad. Ayuda a lograr ahorros considerables en costos de operación conservando todas las características, estabilidad y rendimiento; es extensible, el código fuente está disponible para todos y es posible modificarlo en dependencia de las necesidades de la aplicación. Los requisitos para la instalación son mínimos y posibilita realizar copias de seguridad evitando la pérdida de información.

Entorno de Desarrollo Integrado: De los IDE descritos anteriormente se propone utilizar NetBeans 6.9, el cual posee herramientas sólidas para el trabajo con Interfaces Gráficas, es un producto libre, gratuito, sin restricciones de uso. Contiene todos los módulos necesarios para el desarrollo de aplicaciones escritas en lenguaje Java y PHP, permitiendo a los desarrolladores comenzar el trabajo inmediatamente.

NetBeans soporta cualquier tipo de sistema operativo y el asistente genera un programa que cuenta con un menú de opciones, una barra de estado, funciones necesarias para persistir y recuperar el estado de las ventanas, es posible además realizar eficientemente conexiones con las BD. Permite a los diseñadores la selección, facilitando la colocación y personalización de los componentes, así como el cambio en las conexiones. Existe un modo de previsualización de la interfaz de forma inmediata y simple que permite comprobar como será el aspecto final sin necesidad de compilar y ejecutar la aplicación. Otra característica importante es la reducción al mínimo del código, pues la conexión y consulta a BD se realiza mediante asistentes.

Las opciones para aplicaciones WEB son múltiples, es posible desarrollar y probar las aplicaciones Web directamente desde el entorno de desarrollo sin necesidad de instalar Apache u otro servidor Web. El asistente que genera la nueva aplicación permite elegir tanto el servidor que se utilizará, así como los frameworks de los que se quiera disponer. Permite la creación de aplicaciones Web con PHP y está preparado para trabajar con PHP en su versión 5.0 o superior. NetBeans es más adaptable que Eclipse y Zend Studio en cuanto al uso de las librerías para el desarrollo de aplicaciones para dispositivos móviles, es decir WURFL y WALL.

Servidor de mapas: Se seleccionó MapServer, pues de las tecnologías de código abierto que permiten la representación geoespacial, este es hasta el momento el más estable. Además, su funcionalidad aumenta

si se combina su instalación con PostgresSQL y PostGIS como gestor de BD y PHP como lenguaje de programación, por lo que se ajusta a las tecnologías seleccionadas anteriormente. Se tuvo en cuenta para seleccionar este servidor de mapas su uso en la plataforma GENESIG ya que el proyecto SIG-Móviles sigue la línea de la misma.

Conclusiones Parciales

En este capítulo se realizó un estudio detallado de los principales estilos y patrones arquitectónicos que permitirán lograr un diseño de alto nivel, que a su vez se ajuste a dicho sistema. Además se hizo una selección de las principales tendencias y tecnologías según sus ventajas, necesidades del cliente y facilidades que brindan para la implementación. Se tuvo en cuenta además que el producto debe ser desarrollo desde y para software libre.

Capítulo III: Descripción de la arquitectura de software.

Introducción

En el presente capítulo se hace una propuesta de solución al problema planteado en el diseño teórico de la investigación, tomando como base las descripciones de tecnologías y herramientas del Capítulo 2 de la investigación. Se describe la arquitectura del Proyecto SIG-Móviles mediante la representación de las vistas arquitectónicas definidas por RUP y se analizan otros aspectos importantes que de manera general proveen un mejor entendimiento de la organización y distribución del software que se quiere desarrollar.

3.1 Requisitos Funcionales

Los Requisitos Funcionales no son más que capacidades o condiciones que el sistema debe cumplir. A continuación se exponen los requisitos funcionales que la aplicación a desarrollar debe tener, teniendo en cuenta las necesidades de la misma.

- RF 1 El sistema debe permitir al usuario realizar diferentes acciones en el mapa, transformando el nivel de referencia con que este se visualiza.
- RF 1.1 El sistema debe permitir al usuario aumentar el nivel de referencia.
- RF 1.2 El sistema debe permitir al usuario disminuir el nivel de referencia.
- RF 1.3 El sistema debe permitir al usuario trasladar el mapa hacia la izquierda sin modificar la escala del mapa.
- RF 1.4 El sistema debe permitir al usuario trasladar el mapa hacia la derecha sin modificar la escala del mapa.
- RF 1.5 El sistema debe permitir al usuario trasladar el mapa hacia arriba sin alterar la escala del mapa.
- RF 1.6 El sistema debe permitir al usuario trasladar el mapa hacia abajo sin alterar la escala del mapa.
- RF 2 El sistema debe permitir que el usuario pueda seleccionar las capas que desee visualizar en el mapa.

- RF 3 El sistema debe permitir que el usuario ubique donde reside una persona en el mapa además de ofrecer información asociada a esa persona.
- RF 4 El sistema debe permitir que el usuario pueda buscar una edificación según el área temática al que pertenezcan.
- RF 5 El sistema debe permitir que el usuario pueda realizar la búsqueda de un edificio, además debe ofrecer información asociada al edificio.
- RF 6 El sistema debe permitir que el usuario pueda conocer el camino mínimo desde un punto a otro.
- RF 7 El sistema debe permitir que el usuario pueda realizar trazados formando una región determinada para poder visualizar el cálculo de área y perímetro de la misma.

3.2 Requisitos no funcionales del sistema

Los requisitos no funcionales son propiedades o cualidades que la aplicación debe tener. Es decir, son las características que hacen al producto atractivo, usable, rápido o confiable. El proyecto SIG-Móviles debe cumplir con los siguientes requerimientos no funcionales:

3.2.1 Requisitos de Usabilidad

- La aplicación puede ser utilizada por especialistas con conocimiento básicos en informática.
- El software tendrá siempre visible la opción de Ayuda, lo que posibilitará un mejor aprovechamiento por parte de los usuarios de sus funcionalidades.

3.2.2 Requisitos de Fiabilidad

- El sistema debe estar disponible todo el tiempo para sus usuarios, descontando el tiempo en que se encuentre en mantenimiento.
- La información será visible para el usuario pero no podrá ser descargada.

3.2.3 Requisitos de Eficiencia

- El sistema debe responder en un tiempo relativamente rápido a las peticiones del usuario (menos de 3 segundos).

3.2.4 Requisitos de Soporte

- El mantenimiento que recibirá la aplicación se efectuará por el equipo de desarrollo y los clientes una vez al año.

3.2.5 Restricciones de diseño

- Se deben emplear los estándares establecidos (lenguaje de programación PHP, servidor de base de datos PostgreSQL).
- El sistema debe proporcionar un diseño sencillo, fácil de usar y entendible, donde no sea necesario mucho entrenamiento para utilizar el sistema.
- El patrón arquitectónico que se debe emplear en el desarrollo es el modelo-vista-controlador.

3.2.6 Requisitos para la documentación de usuarios en línea y ayuda del sistema.

- El software tendrá siempre la posibilidad de ayuda disponible para cualquier usuario.

3.2.7 Requisitos de Interfaz

Interfaces de usuario

El sistema debe:

- Contar con un diseño gráfico sencillo orientado a las características de los dispositivos móviles y sin uso de múltiples imágenes.
- El tamaño de la página debe ser apropiado para las limitaciones de memoria y capacidad de visualización de los dispositivos móviles.
- Mostrar de forma organizada sus funcionalidades.

3.2.8 Requisitos de software

Para los dispositivos móviles clientes:

- Debe tener un navegador wap para poder visualizar el sitio wap.

Para los Servidores:

- Sistema operativo GNU/Linux.

- Servidor Web Apache 2.0 o superior, con el CGI de MapServer en su versión 5.2.2.
- Lenguaje PHP en su versión 5.0 o superior, configurado con los módulos PHP5-gd, PHP5-pgsql y PHP5-MapScript.PostgreSQL como Sistema Gestor de Base de Datos.
- PostGIS como extensión de PostgreSQL como soporte de datos espaciales.
- PgRouting como extensión de PostgreSQL para análisis de rutas.

3.2.9 Requisitos de Hardware

Para los dispositivos móviles clientes**:**

- Soportar wifi o bluetooth para sus conexiones mediante protocolo WAP.

Para las pc servidoras:

- Se requiere tarjeta de red.
- Se requiere que el Servidor de Mapas tenga como mínimo 512 MB de RAM y 40GB de disco duro.
- Se requiere que el Servidor de BD tenga como mínimo 1GB de RAM y 40GB de disco duro.
- Se requiere que el Procesador sea de 3 GHz como mínimo.

3.2.10 Requisitos de Licencia

- Las herramientas de desarrollo que se necesitan y utilizan son libres.

3.2.11 Requisitos Legales y de Derecho de Autor.

- La aplicación se distribuye bajo las reglas legales establecidas en el registro comercial de la Universidad de las Ciencias Informáticas formulado por las entidades jurídicas de la misma.

3.3 Vistas Arquitectónicas

Una vista es la presentación de un modelo, es una descripción completa de un sistema desde una particular perspectiva. Siguiendo la metodología RUP como se expuso en el Capítulo1, a continuación se desarrolla la descripción de la arquitectura a través de las 4+1 vistas propuestas por Kruchten.

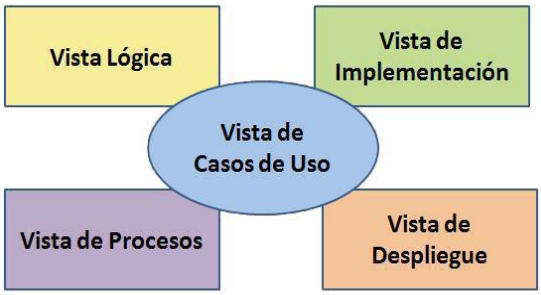

Figura 2. Vistas arquitectónicas de RUP.

3.3.1 Vista de Casos de Uso

La vista de casos de usos en el marco arquitectónico, representa los casos de usos arquitectónicamente significativos; seleccionados a partir de los casos de uso catalogados de críticos en cada uno de los módulos. Estos serían los casos de usos que modelan las principales funcionalidades del sistema de acuerdo con las exigencias del cliente.

Una vez presentados los requisitos funcionales en el acápite 3.1 se procede a realizar la vista de casos de uso del sistema, seleccionando los casos de uso críticos, es decir, los de mayor prioridad a la hora de comenzar la implementación del sistema.

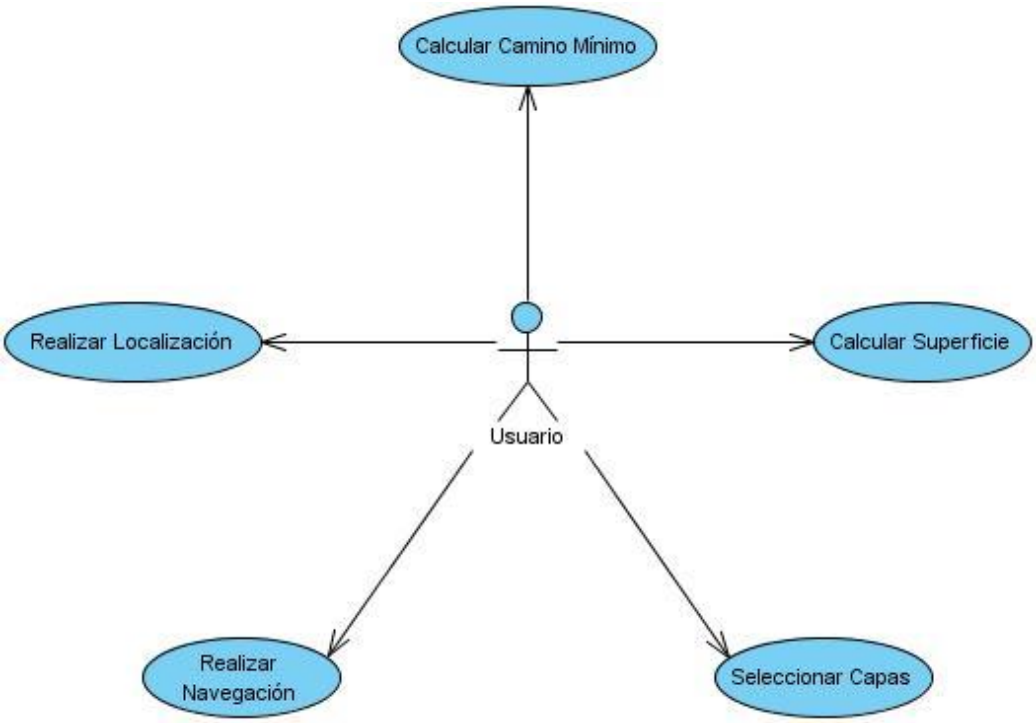

Figura 3. Vista de Casos de Uso

A continuación se realiza una breve descripción de los casos de uso críticos del SIG-Móviles:

- Calcular camino mínimo: Permite al usuario conocer el camino mínimo desde un punto a otro.
- Calcular superficies: Permite al usuario realizar trazados formando una región determinada para poder visualizar el cálculo del área y perímetro de la misma.
- Seleccionar capas: Permite al usuario seleccionar las capas en el mapa para que posteriormente el sistema las visualice.
- Realizar localizaciones: Permite al usuario localizar objetos geográficos y seleccionarlos sobre el mapa.
- Realizar navegación: Permite que el usuario pueda realizar las diferentes funciones de zoom sobre el mapa: zoom +, zoom-, extenso, previo, siguiente, además de realizar paneo sobre el mapa.

3.3.2 Vista Lógica

La vista lógica describe el diseño más importante de las clases y su organización en paquetes y subsistemas, y la organización de éstos en capas. También contiene algunas realizaciones de casos de uso. Ésta muestra como la funcionalidad es diseñada en el interior del sistema, en términos de la estructura estática y comportamiento dinámico del sistema. (S, Roger y Pressman, Roger S, 2002). Esta vista se realizó teniendo en cuenta los requisitos no funcionales planteados en el acápite 3.2.

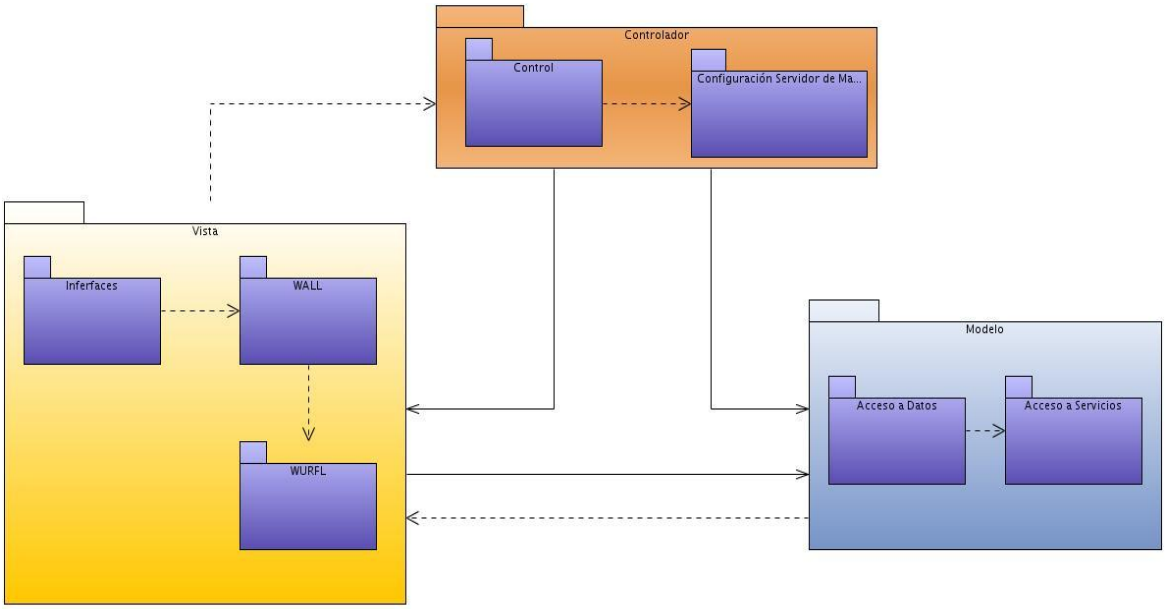

Figura 4. Vista Lógica del Sistema.

La vista está compuesta por tres paquetes principales:

En la capa **Vista** se encuentran los formularios y las clases interfaces, las cuales se encargan de la presentación del contenido de la aplicación e interactuar con el usuario. Estas se apoyan de las librerías Wall y Wurfl para construir aplicaciones visualizables en diferentes entornos, adaptadas a las capacidades disponibles en cada dispositivo móvil.

En la capa **Controlador** se encuentran las clases controladoras que encapsulan la lógica del negocio separado en los paquetes control y configuración de servidor de mapas. Las clases control están

orientadas a controlar las funcionalidades existentes, estas clases por sí solas no pueden realizar su función, pues para ello necesitan de una configuración de servidor de mapas.

En la capa **Modelo** se encuentran las clases de acceso a datos, las cuales necesitan de las clases acceso a servicios para posteriormente mostrar los datos.

3.3.3 Vista de Despliegue

La vista de despliegue muestra la distribución física de los procesos del sistema (ordenadores, dispositivos) y sus conexiones. Esta vista es un grafo de nodos unidos por conexiones de comunicación, la misma es perfeccionada durante las iteraciones. (S, Roger y Pressman, Roger S,2002). Mediante los requisitos de hardware y software determinados anteriormente se pudo realizar dicha vista.

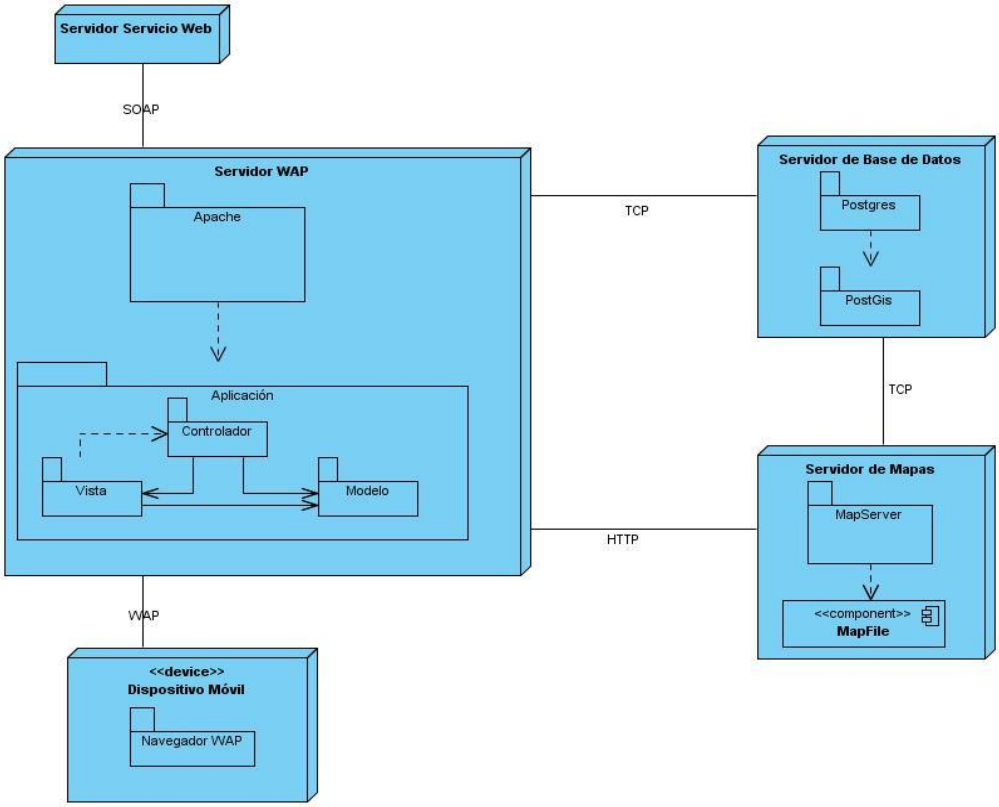

Figura 5. Vista de despliegue del Sistema.

Un diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema. Es un conjunto de nodos unidos por conexiones de comunicación. Un nodo puede contener instancias de componentes software, objetos y procesos.(S, Roger y Pressman, Roger S, 2002). La distribución física de los principales componentes del producto SIG-Móviles se puede adaptar a múltiples configuraciones dependiendo del presupuesto disponible. Puede estar separado en servidores independientes el servidor de WAP, el servidor de base de datos, el servidor de mapas y el servidor de servicios WEB.

A continuación se describirán los nodos físicos representados en el diagrama de despliegue anterior:

Servidor de Servicios WEB: Consiste en uno o varios servidores de servicios WEB con los cuales la aplicación pueda interactuar con el objetivo de mostrar más funcionalidades adaptadas al ambiente donde se tenga el sistema.

Servidor WAP: Contiene el servidor WEB apache donde se encuentra instalada la aplicación que contiene las funcionalidades a utilizar por los usuarios, configurado para protocolo WAP.

Servidor de Base de Datos: Se encuentra el SGDB PostgreSQL que hace uso del módulo PostGIS.

Servidor de Mapas: Contiene el servidor de mapas MapServer, el cual contiene el archivo de mapas Mapfile donde se encuentran las especificaciones y configuración acerca de cómo se va a presentar las capas y demás contenido por el servidor de mapas.

Los protocolos de comunicación entre los componentes serán descritos a continuación:

TCP: El servidor WAP se conecta a través de este protocolo al servidor de base de datos para acceder a la información almacenada en la misma. Este último servidor se conecta además por este protocolo al servidor de mapas.

CGI: Es el protocolo que se utiliza para conectarse al servidor de mapas.

SOAP: El servidor de servicios WEB se conecta a través de este protocolo al servidor WAP.

WAP: Mediante un navegador a través de este protocolo se conectan los dispositivos móviles a la aplicación.

3.3.4 Vista de Implementación

La vista de Implementación incluye la colección de componentes y subsistemas de implementación mediante el modelo de implementación (diagrama de componentes). Esta vista define además, ejecutables, bibliotecas, ficheros, subsistemas y dependencias entre ellos. Se presenta la dependencia existente entre las capas en el siguiente diagrama y se describe básicamente la relación que existe internamente entre los componentes de cada vista.

La vista de implementación proporciona una descripción de las principales capas y subsistemas de componentes de la aplicación. Los paquetes principales de la aplicación por cada uno de los módulos son:

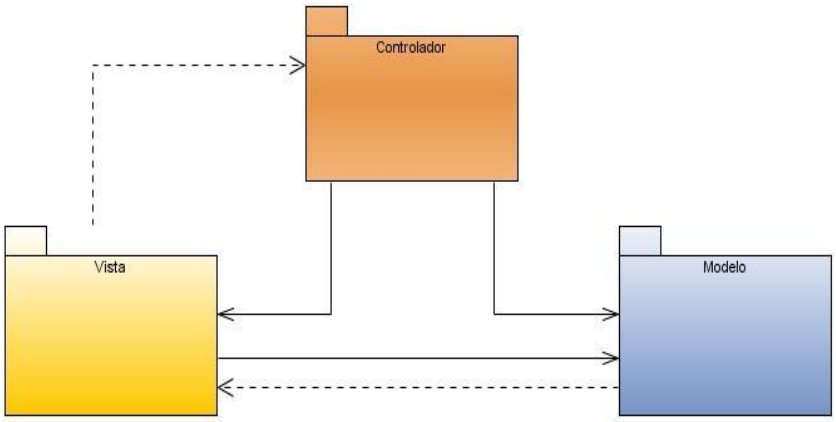

Figura 6. Paquetes principales de la aplicación.

Los paquetes Modelo, Vista y Controlador encapsulan uno o más componentes que se interrelacionan entre ellos para darle solución a la aplicación. La distribución de los componentes mediante capas y paquetes de la aplicación se realiza de la siguiente forma:

Vista: La capa Vista contiene los componentes relacionados a la presentación del contenido. En este caso los componentes inicio.php, principal.php y menú.php están relacionados con la librería WALL, la cual depende de la librería WURFL para posteriormente realizar las funcionalidades.

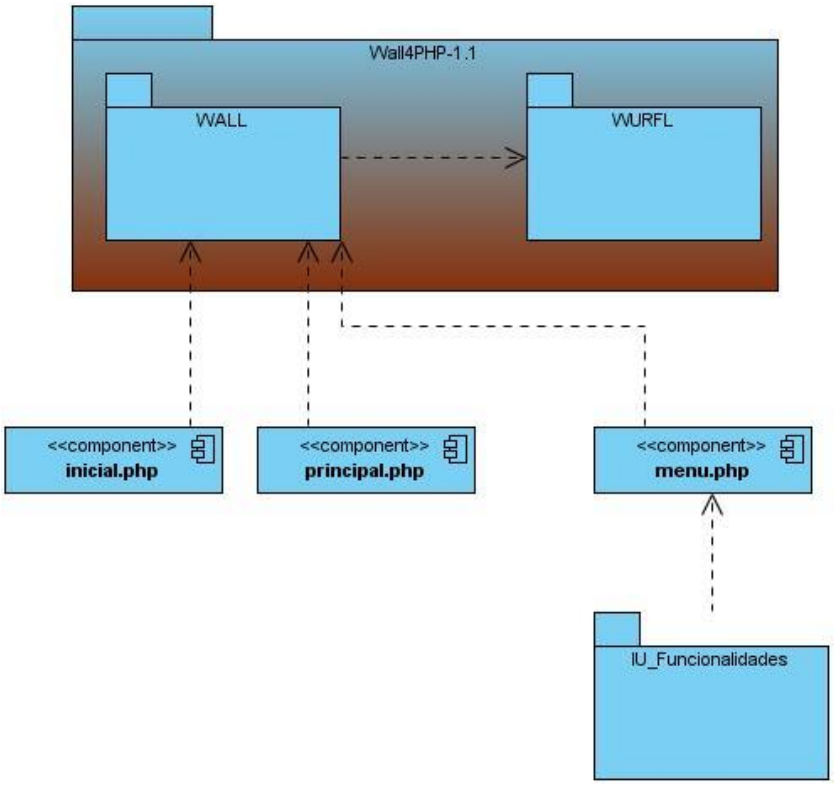

Figura 7. Componentes de la Capa Vista.

Controlador: La capa Controlador contiene los componentes correspondientes al procesamiento, es decir la clase controlador.php lleva el control de las funcionalidades. La clase construir página para realizar su función depende de las librerías WALL, la cual a su vez depende de la librería WURFL. De la clase negocio.php es necesario una clase *mapserver.class.php* para a través de PHP 5 MapScript conectarse al servidor de mapas y realizar las funciones requeridas.

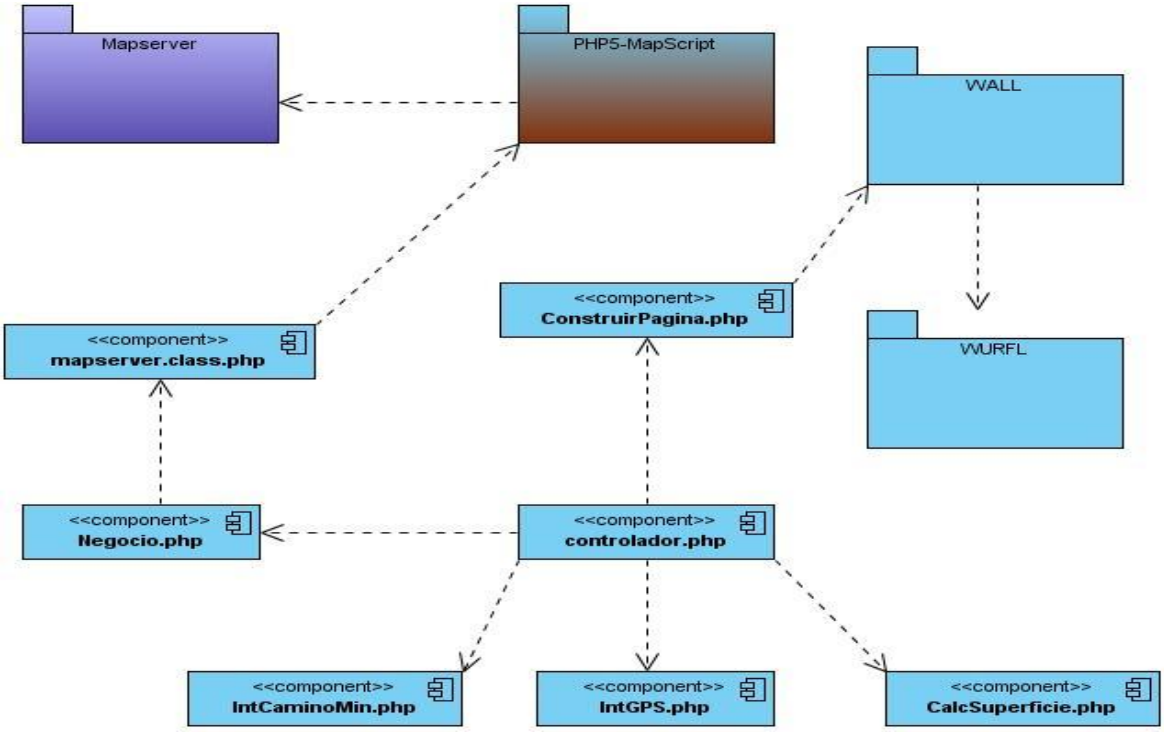

Figura 8. Componentes de la Controlador.

Modelo: La capa Modelo contiene los componentes correspondientes al acceso a datos, estos están relacionados con el componente Conexión.php que contiene las funcionalidades que posibilita la abstracción con la Base de Datos. LocObjetivos.php además de tener relación con conexión.php depende de WebService.php para realizar su función.

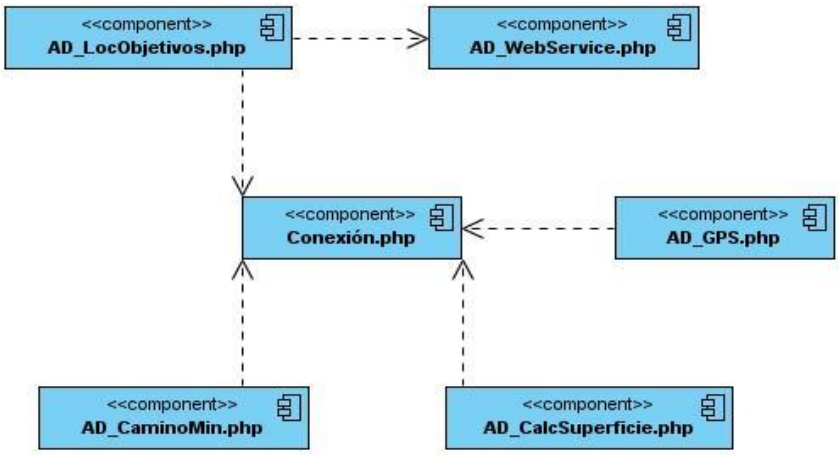

Figura 9. Componentes de la Capa Modelo**.**

3.3.5 Vista de Procesos

La vista de procesos proporciona una base para el entendimiento de la organización de los procesos de un sistema. La misma solo suele usarse cuando el sistema presenta procesos concurrentes o hilos. El acceso de los clientes al sistema de información geográfica para dispositivos móviles es independiente de la ejecución de otro cliente. La concurrencia de uso del servidor WAP y la base de datos es manejada por los propios servidores. No existen procesos de negocio concurrentes que requieran ser manejados por la aplicación. Por estas razones no es necesario una vista de procesos.

Conclusiones Parciales

En el capítulo se plantearon las bases tecnológicas que debía cumplir el sistema arquitectónicamente traducido en los requisitos no funcionales, elemento significativo en la arquitectura de software. Se presentaron las 4+1 vistas propuestas por la metodología utilizada, en este caso RUP, permitiendo a través de cada una de las vistas visualizar de manera concreta los componentes de la arquitectura. Mediante la vista de casos de uso se logró representar las principales funcionalidades que debe cumplir el sistema. La vista lógica permitió percibir la organización en paquetes y sus relaciones. La vista de despliegue posibilitó identificar la ubicación de los nodos físicamente donde residirán los diferentes componentes de la aplicación y mediante la vista de implementación como interactúan las clases y librerías del sistema específicamente dentro de la aplicación.

Capítulo IV: Propuesta de evaluación de la arquitectura.

Introducción

En el presente capítulo se pretende evaluar la arquitectura propuesta analizando los resultados obtenidos de acuerdo con las técnicas y métodos de evaluación de arquitecturas de software. La misma debe estar diseñada para desarrollar los casos de usos críticos y a medida que avance el proyecto permita incluir los restantes. Además se busca que la propuesta sea funcional y cumpla con los atributos de calidad definidos en los diferentes modelos de evaluación; por lo que debe ser sometida a prueba, con el fin de que el sistema que la soporte sea robusto y seguro. De esta forma permite conocer los problemas, debilidades y puntos de interés para posibilitar una corrección a tiempo de los mismos.

4.1 Aspectos a tener en cuenta para evaluar la arquitectura.

Cuando se trata de evaluar una arquitectura se pueden plantear las siguientes interrogantes: ¿por qué evaluar una arquitectura?, ¿quiénes intervienen en la evaluación? y ¿cuándo una arquitectura puede ser evaluada?

El propósito de realizar evaluaciones a la arquitectura, es para analizar e identificar riesgos potenciales en su estructura y sus propiedades, que puedan afectar al sistema de software resultante, verificar que los requerimientos no funcionales estén presentes en la arquitectura, así como determinar en qué grado se satisfacen los atributos de calidad. Cabe señalar que los requerimientos no funcionales también son llamados atributos de calidad.

Generalmente las evaluaciones a la arquitectura se hacen por miembros del equipo de desarrollo, como por ejemplo: arquitecto o diseñador. Sin embargo puede haber también situaciones en las que intervengan personas especialistas en el tema. El cliente es uno de los más interesados en los resultados de una evaluación, ya que en dependencia de los mismos puede tomar decisiones sobre continuar o no con el proyecto.

Es posible realizar la evaluación en cualquier momento del proceso de desarrollo del software en cuestión, pero se proponen dos variantes según las etapas de desarrollo: temprana y tarde.

- ✓ **La evaluación temprana:** Para realizar este tipo de evaluación no es necesario que la arquitectura se encuentre completamente realizada. Permite efectuar decisiones sobre la arquitectura en cualquier nivel, puesto que se pueden atribuir cambios arquitectónicos producto de una evaluación en función de los atributos de calidad esperados.
- ✓ **La evaluación tardía:** Para realizar este tipo de evaluación es necesario que la arquitectura del sistema se encuentre establecida y su implementación esté concluida, es decir, en el momento que el sistema ya esté terminado. Se considera que la evaluación en este punto es muy importante y útil, puesto que puede observarse el cumplimiento de los atributos de calidad asociados al sistema y su comportamiento general. La realización de la evaluación de la arquitectura de software debe realizarse cuando estén listos los elementos necesarios para justificarla. Puede ser determinada en el momento que el equipo de desarrollo necesite tomar decisiones que dependen de la arquitectura propuesta y que el costo de no tenerlas en cuenta conllevaría a un costo superior al de llevar a cabo una evaluación.

4.2 ¿Por cuáles cualidades puede ser evaluada la Arquitectura de Software?

No es posible evaluar una arquitectura si no se sabe qué cualidades de esta se desea evaluar realmente. A estas cualidades se le llaman atributos de calidad, aunque muchas veces son imprecisos. Estos se clasifican en observables en vía de ejecución y no observables en vía de ejecución. Como su nombre lo indica, los primeros determinan el comportamiento en tiempo de ejecución y los segundos se establecen durante el desarrollo del sistema.

Se considera que la calidad del software está determinada por el grado en el que esta posee una combinación deseada de atributos. Estos atributos son características que el sistema debe satisfacer. Estas características o atributos, según Barbacci, se conocen con el nombre de atributos de calidad, los cuales se definen como las propiedades de un servicio que presta el sistema a sus usuarios (Barbacci, 1995).

En las siguientes tablas se describen algunos de estos atributos de calidad, agrupados según la clasificación propuesta por Bass.

Atributo de Calidad	Descripción
Disponibilidad	Es la medida de disponibilidad del sistema para el uso **(Barbacci, 1995).**
Confidencialidad	Es la ausencia de acceso no autorizado a la información **(Barbacci, 1995).**
Funcionalidad	Habilidad del sistema para realizar el trabajo para el cual fue concebido **(Kazman, 2001)**.
Desempeño	Es el grado en el cual un sistema o componente cumple con sus funciones designadas, dentro de ciertas restricciones dadas, como velocidad, exactitud o uso de memoria **(IEEE 610.12).**
Confiabilidad	Es la medida de la habilidad de un sistema a mantenerse operativo a lo largo del tiempo **(Barbacci, 1995).**
Seguridad externa	Ausencia de consecuencias catastróficas en el ambiente. Es la medida de ausencia de errores que generan pérdidas de información **(Barbacci, 1995).**
Seguridad interna	Es la medida de la habilidad del sistema para resistir a intentos de uso no autorizados y negación del servicio, mientras se sirve a usuarios legítimos **(Kazman, 2001)**.

Tabla 3. Descripción de atributos de calidad observables vía ejecución.

Atributo de Calidad	Descripción
Configurabilidad	Posibilidad que se otorga a un usuario experto a realizar ciertos cambios al sistema **(Booch, 1999)**.
Integrabilidad	Es la medida en que trabajan correctamente componentes del sistema que fueron desarrollados separadamente al ser integrados. **(Bass, 1998)**.

Integridad *(Integrity)*	Es la ausencia de alteraciones inapropiadas de la información **(Barbacci, 1995).**
Interoperabilidad	Es la medida de la habilidad de que un grupo de partes del sistema trabajen con otro sistema. Es un tipo especial de integrabilidad **(Bass, 1998)**.
Modificabilidad	Es la habilidad de realizar cambios futuros al sistema **(Booch, 1999)**.
Portabilidad	Es la habilidad del sistema para ser ejecutado en diferentes ambientes de computación. Estos ambientes pueden ser hardware, software o una combinación de los dos **(Kazman, 2001)**.
Reusabilidad	Es la capacidad de diseñar un sistema de forma tal que su estructura o parte de sus componentes puedan ser reutilizados en futuras aplicaciones **(Bass, 1998)**.
Escalabilidad	Es el grado con el que se pueden ampliar el diseño arquitectónico, de datos o procedimental **(Pressman, 2002).**
Capacidad de Prueba	Es la medida de la facilidad con la que el software, al ser sometido a una serie de pruebas, puede demostrar sus fallas. Es la probabilidad de que, asumiendo que tiene al menos una falla, el software fallará en su próxima ejecución de prueba **(Bass, 1998)**.

Tabla 4. Descripción de atributos de calidad no observables vía ejecución.

Conociendo los atributos de calidad a medir en la arquitectura todavía no se está en condiciones de medir la calidad de una arquitectura si no se utiliza alguna técnica existente para estas evaluaciones.

4.3 Técnicas de evaluación de arquitecturas.

Las técnicas de evaluación se clasifican en cualitativas y cuantitativas, (ver Figura 11). En la primera clasificación se encuentran las técnicas que son usadas cuando la arquitectura se encuentra aún en construcción y de ellas se obtienen respuestas concisas. Entre las mismas se encuentran: escenarios,

cuestionarios y listas de chequeo. La segunda clasificación agrupa las técnicas que se utilizan cuando la arquitectura ya ha sido implantada. Estas son: métricas, normas, máximos y mínimos teóricos, simulaciones y prototipos.

En vista de que el interés es tomar decisiones de tipo arquitectónico en las fases tempranas del desarrollo, son necesarias técnicas que requieran poca información detallada y puedan conducir a resultados relativamente precisos (Bosch, 2000). Las técnicas que cumplen con lo antes planteado son las cualitativas, que son las más empleadas por los arquitectos debido al bajo costo que implica llevar a cabo estas técnicas en comparación con las cuantitativas.

Las técnicas de evaluación de arquitecturas existentes permiten hacer una evaluación cuantitativa de los atributos de calidad a nivel arquitectónico. A continuación se presentan algunas de las técnicas:

- ✓ Evaluación basada en escenarios: Esta consiste en crear escenarios donde exista un contexto determinado y una respuesta, descrita a través de la arquitectura que detalla cómo debería responder el sistema.
- ✓ Evaluación basada en simulación: Consiste en la implementación de componentes de la arquitectura y la implementación a cierto nivel de abstracción del contexto del sistema donde se supone va a ejecutarse. Una vez implementados estos se pueden usar un conjunto de escenarios para evaluar los atributos de calidad.
- ✓ Evaluación basada en modelos matemáticos: Esta técnica es usada para validar atributos de calidad operables. Este puede ser combinado con la técnica de simulación donde la entrada de uno es el resultado del otro.
- ✓ Evaluación basada en experiencia: Esta evaluación es propiciada por arquitectos del proyecto o externos a este. Está basada en la experiencia y la intuición de estos.

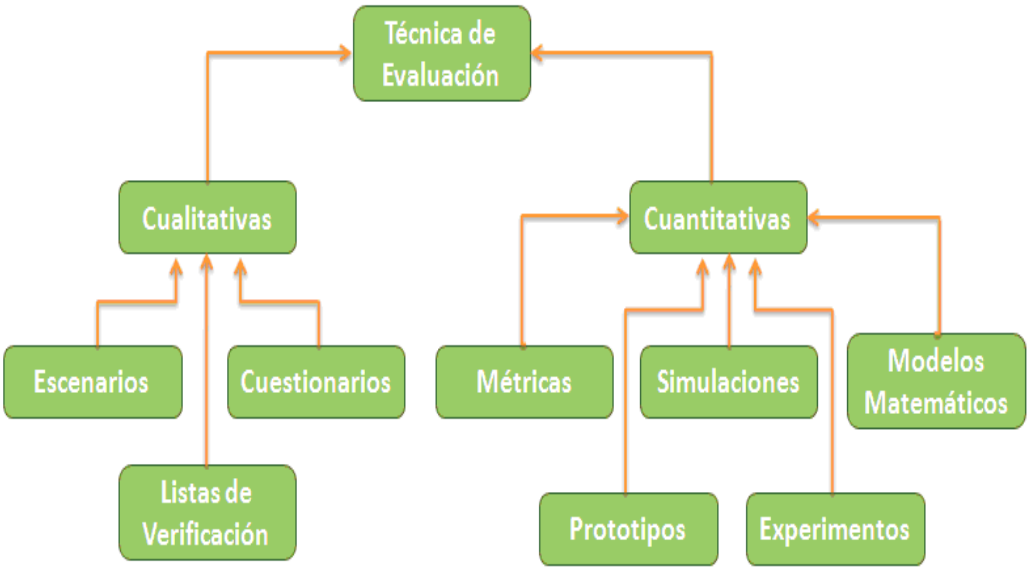

Figura 10. Técnicas de evaluación

4.4 Principales métodos de prueba de arquitectura de software.

Los métodos de evaluación de arquitecturas permiten identificar los conflictos entre las arquitecturas y las soluciones que se desarrollan. Existen varios métodos que ayudan a esta identificación, entre estos se encuentran Software Architecture Analysis Method o Método de Análisis de Arquitectura de Software(SAAM), Architecture Trade-off Analysis Method (ATAM) y Active Review for Intermediate Designs o Revisión Activa para Diseños Intermedios(ARID).

4.4.1 SAAM.

El método de análisis de arquitecturas de software (SAAM, por sus siglas en inglés) es uno de los primeros que fue ampliamente promulgado y documentado. Originalmente fue creado para el análisis de modificabilidad de la arquitectura, pero ha demostrado ser muy útil para evaluar ciertos atributos de calidad, tales como modificabilidad, portabilidad, escalabilidad e integrabilidad (R. Kazmar, 2001). Este consiste en la enumeración de escenarios que representarán los probables cambios a los que el sistema estará sometido en el futuro. Como entrada principal necesitará la descripción de la arquitectura de dicho sistema, la cual será evaluada.

La evaluación de este método está compuesta por seis pasos, los cuales son:

1. Desarrollo de escenarios: Un escenario es una breve descripción de usos anticipados o deseados del sistema. De igual forma, estos pueden incluir cambios a los que puede estar expuesto el sistema en el futuro.

2. Descripción de la arquitectura: La arquitectura propuesta debe ser descrita haciendo uso de alguna notación arquitectónica que sea común a todas las partes involucradas en el análisis. Deben incluirse los componentes de datos y conexiones relevantes, así como la descripción del comportamiento general del sistema. El desarrollo de escenarios y la descripción de la arquitectura son usualmente llevados a cabo de forma intercalada, o a través de varias iteraciones.

3. Clasificación y asignación de la prioridad de los escenarios: La clasificación de los escenarios puede hacerse en dos clases: directos e indirectos. Un escenario directo es el que puede satisfacerse sin la necesidad de modificaciones en la arquitectura. Un escenario indirecto es aquel que requiere modificaciones en la arquitectura para poder satisfacerse. Los escenarios indirectos son de especial interés para SAAM, pues son los que permiten medir el grado en el que una arquitectura puede ajustarse a los cambios de evolución que son importantes para los involucrados en el desarrollo.

4. Evaluación individual de los escenarios indirectos: Para cada escenario indirecto, se listan los cambios necesarios sobre la arquitectura, y se calcula su costo. Una modificación sobre la arquitectura significa que debe introducirse un nuevo componente o conector, o que alguno de los existentes requiere cambios en su especificación.

5. Evaluación de la interacción entre los escenarios: Cuando dos o más escenarios indirectos proponen cambios sobre un mismo componente, se dice que interactúan sobre ese componente. Es necesario evaluar este hecho, puesto que la interacción de componentes semánticamente no relacionados revela que los componentes de la arquitectura efectúan funciones semánticamente distintas. De forma similar, puede verificarse si la arquitectura se encuentra documentada a un nivel correcto de descomposición estructural.

6. Creación de la evaluación global: Debe asignársele un peso a cada escenario, en términos de su importancia relativa al éxito del sistema. Esta asignación de peso suele hacerse con base en las metas del

negocio que cada escenario soporta. En el caso de la evaluación de múltiples arquitecturas, la asignación de pesos puede ser utilizada para la determinación de una escala general.

Dependiendo del objetivo de la evaluación será el resultado de la misma. Por ejemplo, si el objetivo es evaluar una sola arquitectura, este método enumera los lugares más vulnerables a fallos dentro de la misma, en términos de los requerimientos de modificabilidad. También puede ser para el caso de que se deseen evaluar varias arquitecturas con el fin de seleccionar una que satisfaga mejor los requerimientos de calidad y con la menor cantidad de modificaciones posibles.

4.4.2 ATAM.

EL método de análisis de acuerdos de arquitectura (ATAM, por sus siglas en inglés) está centrado en tres aéreas distintas, el estilo arquitectónico, el análisis de los atributos de calidad y el método SAAM. Su nombre surge del hecho de que revela la forma específica en que una arquitectura cumple con algunos atributos de calidad, así como la interacción de estos con otros. (R. Kazmar, 2001) El método se centra en identificar el estilo arquitectónico o enfoque arquitectónico utilizado. Estos representan los métodos aplicados en la arquitectura para cumplir con los atributos de calidad.(R. Kazmar, 2001) La metodología de ATAM comprende nueve pasos divididos en cuatro fases.

Fase 1: Presentación.

1. Presentación del ATAM: El líder de evaluación describe el método a los participantes, trata de establecer las expectativas y responde las preguntas propuestas.
2. Presentación de las metas del negocio: Se realiza la descripción de las metas del negocio que motivan el esfuerzo, y aclara que se persiguen objetivos de tipo arquitectónico.
3. Presentación del negocio: El arquitecto describe la arquitectura, enfocándose en cómo ésta cumple con los objetivos del negocio.

Fase 2: Investigación y análisis.

4. Identificación de los enfoques arquitectónicos: Estos elementos son detectados, pero no analizados.
5. Generación del Utility Tree12: Se priorizan los atributos de calidad que engloban la “utilidad” del sistema (desempeño, disponibilidad, seguridad, modificabilidad, usabilidad), especificados en forma de escenarios. Se anotan los estímulos y respuestas, así como se establece la prioridad entre ellos.

6. Análisis de los enfoques arquitectónicos: Con base en los resultados del establecimiento de prioridades del paso anterior, se analizan los elementos del paso 4. En este paso se identifican riesgos arquitectónicos, puntos de sensibilidad y puntos de balance.

Fase 3: Pruebas.

7. Tormenta o lluvia de ideas y establecimientos de la prioridad de los escenarios: Con la colaboración de todos los involucrados, se complementa el conjunto de escenarios.

8. Análisis de los enfoques arquitectónicos: Este paso repite las actividades del paso 6, haciendo uso de los resultados del paso 7. Los escenarios son considerados como casos de prueba para confirmar el análisis realizado hasta el momento.

Fase 4: Reportes.

9. Presentación de los resultados: Basado en la información recolectada a lo largo de la evaluación del ATAM, se presentan los hallazgos a los participantes.

4.4.3 ARID.

El método de Análisis de diseños intermedios es conveniente para realizar revisiones parciales en etapas tempranas del desarrollo. Es conveniente saber si el diseño propuesto es conveniente, desde el punto de vista de otras partes de la arquitectura. ARID es un híbrido entre Active Design Review (ADR) y ATAM. ADR es utilizado para la evaluación de diseños detallados de unidades de software como los componentes o módulos. En el caso de ADR proporciona una documentación detallada del diseño y completan cuestionarios. En el caso de ATAM se realiza una evaluación de la arquitectura en su conjunto. Por esta combinación de filosofías surge ARID para la evaluación temprana de arquitecturas de software. El método de evaluación comprende nueve pasos divididos en dos fases.

Fase 1: Actividades previas.

1. Identificación de los encargos de la revisión: Los encargados de la revisión son los ingenieros de software que se espera que usen el diseño, y todos los involucrados en el diseño. En este punto, converge el concepto de encargado de revisión de ADR e involucrado del ATAM.

2. Preparar el informe de diseño: El diseñador prepara un informe que explica el diseño. Se incluyen ejemplos del uso del mismo para la resolución de problemas reales. Esto permite al facilitador anticipar el tipo de preguntas posibles, así como identificar áreas en las que la presentación puede ser mejorada.
3. Preparar los escenarios bases: El diseñador y el facilitador preparan un conjunto de escenarios base. De forma similar a los escenarios del ATAM y el SAAM, se diseñan para ilustrar el concepto de escenario, que pueden o no ser utilizados para efectos de la evaluación.
4. Preparar los materiales: Se reproducen los materiales preparados para ser presentados en la segunda fase. Se establece la reunión, y los involucrados son invitados.

Fase 2: Revisión.

5. Presentación del ARID: Se explica los pasos del ARID a los participantes.
6. Presentación del diseño: El líder del equipo de diseño realiza una presentación, con ejemplos incluidos. Se propone evitar preguntas que conciernen a la implementación o argumentación, así como alternativas de diseño. El objetivo es verificar que el diseño es conveniente.
7. Lluvia de ideas y establecimiento de prioridad de escenarios: Se establece una sesión para la lluvia de ideas sobre los escenarios y el establecimiento de prioridad de escenarios. Los involucrados proponen escenarios a ser usados en el diseño para resolver problemas que esperan encontrar. Luego, los escenarios son sometidos a votación, y se utilizan los que resultan ganadores para hacer pruebas sobre el diseño.
8. Aplicación de los escenarios: Comenzando con el escenario que contó con más votos, el facilitador solicita pseudo-código que utiliza el diseño para proveer el servicio, y el diseñador no debe ayudar en esta tarea. Este paso continúa hasta que ocurra alguno de los siguientes eventos:

- Se agota el tiempo destinado a la revisión
- Se han estudiado los escenarios de más alta prioridad
- El grupo se siente satisfecho con la conclusión alcanzada.

Puede suceder que el diseño presentado sea conveniente, con la exitosa aplicación de los escenarios, o por el contrario, no conveniente, cuando el grupo encuentra problemas o deficiencias.

9. Resumen: Al final, el facilitador recuenta la lista de puntos tratados, pide opiniones de los participantes sobre la eficiencia del ejercicio de revisión, y agradece por su participación.

No se puede decir que un método es mejor que otro, en cambio si se puede afirmar que en determinadas condiciones y para determinados atributos de calidad existen métodos que evalúan la arquitectura de manera más eficiente que otros.

4.5 Propuesta de estrategia de evaluación de la arquitectura.

Para la evaluación de la Arquitectura de Software de SIG-Móviles se proponen dos estrategias de evaluación. Se realizará una primera evaluación en etapas tempranas del desarrollo con el objetivo de medir los principales atributos de calidad presentados y tomar medidas de acuerdo a los resultados arrojados por la misma. La segunda evaluación se efectuará cuando la arquitectura del sistema se encuentre establecida y su implementación esté finalizada, lo cual podrá lanzar resultados satisfactorios para el conocimiento del grado de cumplimiento de los atributos de calidad propuestos, cuando el sistema esté en funcionamiento.

4.5.1. Estrategia de evaluación Temprana (ARID).

En la estrategia de evaluación temprana de la arquitectura se analizarán los atributos de calidad de Modificabilidad, Interoperabilidad, Seguridad, Funcionalidad y Disponibilidad. Se empleará el método de evaluación ARID que posibilita la evaluación en etapas tempranas del desarrollo de software evaluando el grado en que los atributos de calidad satisfacen en cada uno de los escenarios definidos. Este método comprende nueve pasos agrupados en dos fases:

Fase 1 (Actividades Previas)

- ✓ Identificación de los encargados de la revisión.
- ✓ Preparar el informe de diseño.
- ✓ Preparar los escenarios base.
- ✓ Preparar los materiales.

Fase 2 (Evaluación)

- ✓ Presentación del ARID.
- ✓ Presentación del diseño.
- ✓ Lluvia de ideas y establecimiento de prioridad de escenarios.
- ✓ Aplicación de los escenarios.
- ✓ Conclusiones.

A continuación se detallarán puntos claves a tener en cuenta durante la realización de la estrategia de evaluación según los atributos de calidad:

Evaluación del atributo Modificabilidad:

Debido a que la Modificabilidad es la habilidad de realizar cambios futuros al sistema, se propone:
Valorar las consecuencias y el costo de:

- ✓ Cambiar la interfaz de la aplicación.
- ✓ Cambiar el servidor WAP.
- ✓ Cambiar el Sistema Gestor de Base de Datos.
- ✓ Cambiar el Servidor WEB.

Analizar los requerimientos que puedan cambiar para futuras versiones de la aplicación y valorar la factibilidad de la arquitectura para adaptarse a los mismos.

Evaluación del atributo Interoperabilidad:

Teniendo en cuenta que la Interoperabilidad es la medida de la habilidad de que un grupo de partes del sistema trabajen con otro sistema, se propone que:
El equipo de evaluación escogerá las funcionalidades que puedan ser de interés para otras aplicaciones.
Valorará el costo de realizar las funcionalidades necesarias para posibilitar la interrelación con otras aplicaciones (preferiblemente usando servicios WEB).
Se deben tener en cuenta las siguientes funcionalidades:

- ✓ Mostrar artículos de contenidos publicados.
- ✓ Mostrar las nuevas publicaciones de artículos de contenido.

El equipo de desarrollo deberá incluir otras funcionalidades.

Evaluación del atributo Seguridad:

Debido a que la Seguridad es la medida de ausencia de errores que generan pérdidas de información y la habilidad del sistema para resistir a intentos de usos no autorizados y negación del servicio, mientras se sirve a usuarios legítimos, se propone:

El equipo de evaluación seleccionará diferentes escenarios donde el acceso a los recursos esté limitado a un rol de usuario, al menos dos escenarios por cada rol del sistema. Se probará en cada escenario definido el acceso de usuario con diferente roles.
Realizar intentos de acceso a la interfaz WEB de administración por usuarios con privilegios y usuarios sin privilegios.
Introducir cadenas que contengan códigos que puedan dañar el funcionamiento de la aplicación en el formulario de registro.

Evaluación del atributo Funcionalidad:

Teniendo en cuenta que la Funcionalidad es la habilidad del sistema para realizar el trabajo para el cual fue concebido, se propone:
El equipo de evaluación medirá el grado de cumplimiento de cada uno de los requerimientos funcionales definidos en el sistema.

Evaluación del atributo Disponibilidad:

Teniendo en cuenta que la Disponibilidad es la medida de disponibilidad del sistema para el uso, se propone:
Valorar las consecuencias sobre los servicios cuando se efectúan las acciones siguientes:

- ✓ Realizar modificaciones en el Sistema Gestor de Bases de Datos sin afectar la conexión con el resto de la aplicación.
- ✓ Realizar modificaciones en el Servidor de WAP sin afectar la conexión con el resto de la aplicación.
- ✓ Realizar modificaciones en el Servidor WEB sin afectar la conexión con el resto de la aplicación.
- ✓ Detener los servicios del Sistema Gestor de Base de Datos.
- ✓ Detener los servicios del Servidor WEB.
- ✓ Detener los servicios del Servidor WAP.
- ✓ Solicitar recursos independientes por múltiples usuarios.
- ✓ Incorporar a estas pruebas otras acciones de mantenimiento que el equipo de evaluación estime conveniente.

Los resultados obtenidos de la evaluación se expondrán ante todos los involucrados en el desarrollo de la aplicación. Si se encuentran dificultades se procederá a valorar los cambios que se imponen en la

arquitectura, aplicar los mismos de forma inmediata y comenzar un nuevo proceso evaluativo empleando nuevamente el método ARID, enfatizando en los errores detectados en la anterior evaluación.

4.5.2. Estrategia de evaluación Tardía (ATAM).

En la estrategia de evaluación tardía de la arquitectura se realizará una vez que el sistema esté implementado completamente y esté en funcionamiento. Se analizarán los atributos de calidad definidos en la evaluación temprana Modificabilidad, Interoperabilidad, Seguridad, Funcionalidad y Disponibilidad conjuntamente con los atributos de Mantenibilidad, Reusabilidad y Flexibilidad. Se empleará el método de evaluación ATAM que revela la forma en que la arquitectura satisface los atributos de calidad definidos y provee una visión de la forma que estos atributos interactúan entre sí.

ATAM define tres tipos de escenarios: Escenarios de casos de Usos (*Use Case Scenarios*) son los que describen las interacciones de los usuarios con el sistema, Escenarios de crecimiento (*Growth Scenarios*) son los que representa la anticipación de los cambios y los Escenarios exploratorios (*Exploratory Scenarios*) son los que tienen como objetivo fundamental exponer al sistema a condiciones extremas, llevar el diseño a casos extremos de condiciones. ATAM consta de nueve pasos, divididos en cuatro fases.

Fase #1: Presentación	
Presentación del ATAM	El líder de evaluación describe el método al resto del equipo y se establece el alcance.
Presentación de las metas de negocio	Se realiza la descripción de las metas del negocio que motivan el esfuerzo y aclara que se persiguen objetivos de tipo arquitectónico.
Presentación de la Arquitectura	El arquitecto describe la arquitectura, enfocándose en cómo esta cumple con los objetivos del negocio.
Fase #2: Investigación y análisis	
Identificación de los enfoques arquitectónicos	Se detectan sin entrar en detalles.
Generación del Utility Tree	Se priorizan los atributos de calidad que engloban

	las funcionalidades del sistema, especificados en forma de escenarios. Se tienen en cuenta los estímulos y respuestas, así como se establece las prioridades entre ellos.
Análisis de los enfoques arquitectónicos.	Con base en los resultados del establecimiento de prioridades del paso anterior, se analizan los elementos identificados. Se identifican riesgos arquitectónicos, puntos de sensibilidad y puntos de balance.
Fase #3: Pruebas	
Lluvia de ideas y establecimiento de prioridad de escenarios.	Con la colaboración de todos los involucrados, se complementa el conjunto de escenarios.
Análisis de los enfoques arquitectónicos.	Este paso repite las actividades del análisis de los enfoques arquitectónicos, haciendo uso de los resultados de la lluvia del paso anterior. Los escenarios son considerados como caso de prueba para confirmar el análisis realizado hasta el momento.
Fase #4: Reporte	
Presentación de los resultados	Basado en la información recolectada a lo largo de la evaluación del ATAM, se presentan los hallazgos a los participantes.

Figura 11. Fases del ATAM

De forma similar a la anterior estrategia los resultados obtenidos de la evaluación se expondrán ante todos los involucrados en la realización del sistema. Si se detectan dificultades se procederá a valorar la factibilidad de realizar cambios en la arquitectura u otras variantes que contribuyan con la mitigación de estos problemas. De ser necesario se realizarán los cambios y al concluir el mismo se proponen realizar nuevamente la evaluación de la arquitectura empleando el método ATAM.

Conclusiones Parciales

En este capítulo se realizó la evaluación de la arquitectura propuesta para el SIG-Móviles. Se analizaron brevemente los atributos de calidad de la arquitectura propuesta, teniendo en cuenta el método ARID, arrojando como resultado una serie de riesgos y puntos sensibles, así como los escenarios identificados. La arquitectura brinda al sistema seguridad, integridad en los datos que gestiona, flexibilidad y adaptabilidad. Por lo anteriormente descrito se considera que esta arquitectura puede ser usada para SIG-Móviles. Estos resultados serán tomados por el arquitecto del proyecto facilitándole la toma de decisiones.

Conclusiones Generales

Teniendo en cuenta la importancia que tiene la arquitectura de software en el proceso de desarrollo de un software se ha propuesto un diseño arquitectónico para SIG-Móviles. Para poder cumplir con el objetivo trazado se definió la línea base de la arquitectura, para la misma se realizó un estudio de estilos y patrones que proporcionarán al sistema calidad, claridad y sencillez en la estructura y diseño de la solución. A través del diseño de las 4 + 1 vistas propuestas por RUP se logró desarrollar la propuesta de solución de la arquitectura del sistema. Se hizo además un estudio de los sistemas existentes actualmente que pudieran resolver las necesidades de SIG-Móviles, determinando que ninguna de las analizadas cumplían a cabalidad las exigencias del mismo.

Con vista a lograr un diseño arquitectónico que cumpliese con los requerimientos de SIG-Móviles se decidió utilizar una arquitectura por capas, implementando el patrón modelo-vista-controlador. Con el objetivo de obtener una mayor reutilización de los diferentes subsistemas se propuso que la estructura de los mismos contuvieran todas las capas. Esta estructura comprende una organización de los diferentes componentes. Para la propuesta de validación del diseño planteado se utilizó el método ARID y ATAM. Estos métodos permitieron proponer la evaluación de determinados atributos de calidad, requeridos por SIG-Móviles, como modificabilidad, integrabilidad y reusabilidad. Se llegó a la conclusión mediante la propuesta de evaluación que el diseño arquitectónico propuesto permite el cumplimiento de los requerimientos de SIG-Móviles.

Recomendaciones

Después de culminar el presente trabajo se plantean las siguientes recomendaciones:

- Aplicar otros métodos de evaluación de la arquitectura para aumentar la fortaleza del sistema.
- Mantener esta propuesta en constante refinamiento.
- Realizar un estudio de factibilidad y una evaluación de los costes de la tecnología a utilizar, por los recursos de hardware que demanda.

Referencias Bibliográficas

1. **Bosque Sendra, Joaquín.***Sistemas de Información Geográfica.* Alcalá : s.n., 1997.

2. **Martínez Marin, Rubén.***Topografía y Sistemas de Información.* Madrid : Ediciones Técnicas y Científicas, BELLISCO, 1995.

3. **Hilda.** deConceptos. [En línea] Febrero de 2009. http://deconceptos.com/informatica/celular.

4. **Pressman, Roger R.***Un enfoque práctico.* Madrid : 5ta Edición, 2002.

5. **Clements, Paul.***A Survey of Architecture Description Languages.* Alemania : s.n., 1996.

6. **Avenia Delgado, Harold Enrique.** Sistemas de Información Geográfica. [En línea] 2010. http://es.bsherpas.com/index.php/proyectos/sistemas-de-informacion- geografica/productos.html.

7. **S, Roger y Pressman, Roger S.***Ingeniería de Software . Un enfoque práctico. .* Madrid : 5ta edición,, 2002.

8. **GeoMation Keitai.** TecnoMaps. [En línea] 2006. http://www.tecnomaps.com/geomation-keitai.

9. **Valcárcel Pérez, Victoria.** ABC DIGITAL. *LOS MÉTODOS DE ENSEÑANZA.* [En línea] 2010. http://archivo.abc.com.py/2003-08-29/articulos/63595/preparacion-y-desarrollo-de-la-clase-magistral-.

10. **Reynoso, Carlos y Kicillof, Nicolás.***Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Buenos Aires : s.n.2004

11. **Raul, De Villa Cano.***El rol del arquitecto.* 2009.

12. **Cross, Anna.** LenguaWeb. *LenguaWeb.* [En línea] 12 de 2010. http://www.lenguaweb.info/metodologia/100-la-clase-magistral.

13. **Contreras, Domingo.** Educacion.idoneos. *Educacion.idoneos.* [En línea] 2010. http://educacion.idoneos.com/index.php/118466.

14. **Alarcón Almenare, Bertha.** Moddle Universidad Holguin. [En línea] 2009. http://moodle.uho.edu.cu/mod/resource/view.php?id=4206.

15. **Devadas, Srinivas**. Patrones de diseño. 2001. [En línea] 2001 http://mit.ocw.universia.net/6.170/6.170/f01/pdf/lecture-12.pdf

16. Lenguajes de Programación. 2009.

17. **Alvarez Taboada, Flor**. Localización mediante SIG de zonas potencialmente truferas en la provincia de León [En línea] 2009 http://e-spacio.uned.es/fez/eserv.php?pid=bibliuned:ETFSerieVI-3D6894BC-139A-B7FF-6730-3C67B12FD113&dsID=localizacion_mediante.pdf

18. **Lantada Zarzosa, Nieves**. Sistemas de Información Geográfica Prácticas Arc View CD. [En línea] 2004. http://www.etp.com.py/fichaLibro?bookId=49009

19. **Garlan, David**. Introducción a la Arquitectura de Software. [En línea] 2004. http://cic.puj.edu.co/wiki/lib/exe/fetch.php?media=materias:introarq.pdf

20. **Rena**, Lenguajes de Programación. [En línea] 2008 .http://www.rena.edu.ve/cuartaEtapa/Informatica/Tema13.html

21. **De Luz, Elizabeth**. Manual de PHP. [En línea] 2010. http://www.librosdeluz.net/manual-de-php-elizabeth-libro-gratis/

22. **Peñalver, Jesús**. Arquitectura Orientada a Servicios en Java. [En línea] 2008 https://oficinavirtual.ugr.es/apli/posgrado/detalle_cep.jsp?cod=10/CA/091

23. **Pilgrim, Mark**. Inmersion en Python . [En línea] 2009 . http://www.biblioteca-digital.net.ve/wordpress/2010/05/06/libro-inmersion-en-python-3-dive-into-python-3/

24 **Gómez, Francisco**. Entorno Integrado de Desarrollo para Ruby [En línea] 2009. http://inza.wordpress.com/2006/07/31/entorno-integrado-de-desarrollo-para-ruby-on-rails/

25 **Toledo, Álvarez**. Eclipse SDK. [En línea] 2010. http://eclipse-sdk.uptodown.com/

26 **Studio, Zend**. Enciclopedia de Zend. [En línea] 2010 http://e-ciencia.com/recursos/enciclopedia/Zend_Studio

27.**NetBeans**. El Entorno de Programación de Fuente Abierto NetBeans. [En línea] 2010. http://es.scribd.com/doc/967380/Tutorial-Netbeans#

28. **Zhang, Meiyu**. Design and Implementation for Mobile-based GIS Urban Information System. 2008.

29. **Pascuzzi, Domenico**. A Component-Based Architecture for the Development and Deployment of WAP-Compliant Transactional Services.2001.

30. **Debnath, Narayan**. Rigorous Definition/Specification in RAISE Specification Language of a Framework for Web Services about Geographic Information Systems. 2010

31. **Tilley, Scott** . Issues in Accessing Web Sites from Mobile Devices. 2008