UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 6

Título: Plataforma para el desarrollo de Sistemas de Información Geográfica para dispositivos móviles en la Universidad de las Ciencias Informáticas

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Marbelis Rojas Rodríguez

Ariel Martínez Montiel

Tutor: Ing. Liester Cruz Castro

Co-Tutor: Ing. Eusebio Andrés García Más

La Habana, Junio 2012

"Año 54 de la Revolución"

"Poseemos sin embargo invencibles armas. La principal es la educación....Todo lo transformará y seremos pronto el pueblo más educado y culto del mundo. Ya nadie lo duda dentro y fuera de Cuba".

Declaración de Autoría

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 6 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Marbelis Rojas Rodríguez

_____

Firma del Autor

Ariel Martínez Montiel

_____

Firma del Autor

Ing. Liester Cruz Castro

_____

Firma del Tutor

Datos de Contacto

Autor(es): Marbelis Rojas Rodríguez
Correo Electrónico: mrrodriguez@estudiantes.uci.cu

Autor(es): Ariel Martínez Montiel
Correo Electrónico: amontiel@estudiantes.uci.cu

Tutor: Ing. Liester Cruz Castro
Formación Académica: Ingeniero en Ciencias Informáticas (Julio/2008)
Centro Laboral: Universidad de las Ciencias Informáticas (UCI)
Correo Electrónico: lcruz@uci.cu

Co-tutor: Eusebio Andrés García Más
Formación Académica: Ingeniero en Ciencias Informáticas (Julio/2011)
Centro Laboral: Datys, Tecnología y Sistemas
Correo Electrónico: eusebio.garcia@datys.cu

Agradecimientos

Marbelis

- ✓ A nuestra Revolución que sin ella hubiera sido imposible mi formación como profesional.
- ✓ A mi mamá y mi papá por apoyarme siempre y luchar tanto para que nunca me faltara nada.
- ✓ A toda mi familia a mi hermana, abuelas, tíos, tías y primos que estuvieron siempre pendientes de mí.
- ✓ A la memoria de mi “aguelo” Juan.
- ✓ A mi co-tutor y novio Eusebio Andrés García Más por sobre todas las cosas que supo guiarme por el camino correcto, dándome su apoyo ilimitado para que yo saliera adelante, la verdad que todo este trabajo sin su ayuda no hubiese sido posible.
- ✓ A los profesores que en mi primer año me apoyaron incondicionalmente, sin su apoyo no estuviese hoy en esta universidad.
- ✓ A todos muchísimas gracias.

Ariel

- ✓ A mis padres por siempre confiar en mí y ayudarme en todo lo que pudieron.
- ✓ A mi hermana.
- ✓ A todos mis familiares que me sirvieron de apoyo.
- ✓ A todas las personas que de una forma u otra se preocupó por mis estudios.
- ✓ A la Revolución que me dio la posibilidad de mis estudios.
- ✓ A todos los maestros y profesores que ayudaron en mi formación.

Dedicatoria

Marbelis

- ✓ A mi mamá y mi papá por apoyarme siempre y luchar tanto para que nunca me faltara nada.
- ✓ A toda mi familia a mi hermana, abuelas, tíos, tías y primos que estuvieron siempre pendientes de mí.
- ✓ A mi novio por siempre estar pendiente de mi y apoyarme en todo lo que yo necesitaba.
- ✓ A la memoria de mi “aguelo” Juan, nunca te voy a olvidar.

Ariel

- ✓ A mis padres por siempre confiar en mí y ayudarme en todo lo que pudieron.
- ✓ A mi hermana.

Resumen

Desde el surgimiento de los dispositivos móviles estos han sido muy utilizados por las personas debido a su portabilidad y funcionalidad, por lo que la industria del *software* comenzó a crear diversas aplicaciones para éstos. Tal es el caso de las plataformas gvSIG Mobile y gvSIG Mini, las cuales permiten el desarrollo de Sistemas de Información Geográfica (SIG), pero estas plataformas necesitan estar conectadas todo el tiempo a *Internet* para consumir servicios de mapas, por lo que no satisfacen las necesidades que posee la línea gvMap. En dicha línea se han realizado varias aplicaciones, pero las mismas no permiten reutilizar sus funcionalidades, por lo que no se ahorra tiempo, no se facilita el trabajo del equipo de desarrollo y estos no pueden adaptarse a las necesidades de los clientes. Se plantea entonces una problemática ¿Cómo agilizar el desarrollo de SIG para dispositivos móviles en la Universidad de las Ciencias Informáticas (UCI)?, donde el objetivo general de la investigación es desarrollar una plataforma para agilizar el desarrollo de SIG para dispositivos móviles en la UCI. Para alcanzar la solución deseada se utilizaron los métodos Analítico - Sintético, Histórico - Lógico, Modelación y la Entrevista no Estructurada. Como resultado fue creada una plataforma con un conjunto de funcionalidades básicas de los SIG, la cual permite visualizar cartografías en los móviles desde cualquier servidor de mapas que soporte Servicios Web de Mapas (WMS), sobre dicha plataforma se puede construir un nuevo SIG para dispositivos móviles programando poco o nada.

Palabras Clave: Dispositivo Móvil, Plataforma, Sistemas de Información Geográfica

Abstract

Since the emergence of such mobile devices have been widely used by people because of their portability and functionality, so the software industry began to create various applications for these devices. Such is the case of platforms gvSIG Mobile and gvSIG Mini, which allow the development of Geographic Information Systems (GIS), but these platforms need to be connected all the time to Internet to consume mapping services, so they don't meet the needs that gvMap line has. In this line there have been developed several applications, but them don't allow to reuse their functionality, so don't saves time, don't facilitates the work of the development team and they can't adapt to customer needs. This raises a problem, How to speed up the development of GIS for mobile devices in the University of Informatics Sciences (UIS)?, where the overall goal of the research is to develop a platform to accelerate the development of GIS for mobile devices in the UIS. To achieve the desired solution Analytical - Synthetic, Historic – Logical, Modeling and Unstructured Interview methods were used. As result was created a platform with a set of basic functions of GIS, which allow to display maps in mobiles from any map server that supports Web Map Services (WMS), on that platform you can build a new mobile GIS doing little or no programming.

Keywords: Mobile Device, Platform, Geographic Information Systems.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Desde la antigüedad, el hombre tenía la necesidad de conocer acerca de la geografía que lo rodeaba, por ejemplo, en las paredes de las cuevas de Lascaux, Francia, los hombres del Cro-Magnon pintaban los animales que cazaban y líneas asociados a los mismos, que se cree que sean las rutas de migración de esas especies (Curtis, 2006). Este ejemplo, aunque rudimentario comparado con las tecnologías actuales, se puede considerar como uno de los primeros indicios de información geográfica.

Con el paso del tiempo, el hombre fue perfeccionando sus técnicas y creó los mapas, donde el primero de ellos data de 500 años antes de Cristo, el cual fue creado en Babilonia (García, 2010). Los mapas fueron transformándose según evolucionaba el conocimiento del hombre acerca de la geografía, conteniendo cada vez más información, adaptándose de esta forma a determinados contextos.

Después de más dos milenios de utilizar los mapas, éstos no contenían información suficiente para algunas áreas del conocimiento, por lo que en 1854 el Dr. Jhon Snow mostró las incidencias de casos de cólera en un mapa de Soho, Londres. Este hecho fue la primera cartografía conocida, que utilizando métodos cartográficos, no solo representaba la realidad, sino que por primera vez analizaba un conjunto de fenómenos geográficos dependientes (York University, 2011).

A pesar del gran avance que existía en la creación y utilización de mapas, éstos eran inexactos y las técnicas de obtención de datos se realizaban de forma manual, por lo que no lograban alcanzar un alto grado de perfección. En la segunda mitad del siglo XX con la creación de las computadoras y los satélites artificiales, surge la oportunidad de perfeccionarlos, por lo que en el año 1962 en *Ottawa*, Ontario, Canadá, Roger Tomlinson creó el Sistema de Información Geográfica de Canadá (**CGIS** por sus siglas en inglés, ***C****anadian* ***G****eographic* ***I****nformation* ***S****ystem*), utilizado para la gestión de los recursos naturales de ese país (Olaya, 2011). Este *software* fue el primer **S**istema de **I**nformación **G**eográfica (en lo adelante **SIG**) en el mundo. Desde ese entonces se han creado varios SIG como son: *ArcView*, *MapInfo*, *PolyMap*, *JT Maps* y *gvSIG Desktop.*

A finales del siglo XX surgen los primeros dispositivos móviles, el primero de estos fue desarrollado por la compañía Apple y fue nombrado Newton. Este implementaba un sistema de reconocimiento de escritura, podía sincronizarse con un ordenador de sobremesa y es considerado como el primer Ayudante Personal Digital (Tardáguila Moro, 2009). Los dispositivos móviles son muy utilizados por las personas debido a su portabilidad y funcionalidad, por lo que la industria del *software* comenzó a crear diversas aplicaciones para éstos. Dentro de las aplicaciones desarrolladas se encuentran los SIG para móviles como por ejemplo: *gvSIG Mini* y *gvSIG Mobile*.

A partir de 1987 se comienzan a utilizar en Cuba los SIG, dando surgimientos a varios productos entre los que se encuentran: el SIG cubano llamado por muchos SIGC y *TeleMap,* desarrollados a finales de 1980 y a principio de 1990 respectivamente (Batista Silva, 2005). En el 2002 surge la **U**niversidad de las **C**iencias **I**nformáticas (en lo adelante UCI), la cual tiene entre sus principales tareas la creación de *software* para la exportación y el consumo nacional. La UCI actualmente está dividida en centros de desarrollo de *software* que se especializan en determinadas ramas de la informática. El centro de desarrollo de Geoinformática y Señales Digitales (en lo adelante GEYSED), perteneciente a la facultad 6, tiene entre sus principales objetivos el desarrollo de SIG; en dicho centro se ha desarrollado una plataforma para la creación de estos, la cual es nombrada GeneSIG. Mediante el uso de la plataforma antes mencionada se han realizado varias personalizaciones como son SIG-UCI, SIG-Rutas, SIG-Salud, SIG-MIC, SIG-Energía, IARASIG, SIG-ONRM, SIG-ONEI, SIG-Verano, entre otros.

En el centro GEYSED se encuentra el proyecto Control de Flotas, el cual posee varias líneas de investigación y desarrollo y una de ellas está orientada al desarrollo de SIG para dispositivos móviles. Esta línea se subdivide a su vez en dos líneas, Movilmap que se encarga de que los dispositivos móviles se conecten de forma inalámbrica a determinados servicios y gvMap que está destinado a la realización de aplicaciones nativas para dispositivos móviles. En gvMap se han analizado diferentes herramientas existentes en el mundo para el desarrollo de los SIG para móviles y estas no se adaptan a las necesidades del proyecto, debido a que las mismas están concebidas para conectarse todo el tiempo a *Internet* consumiendo servicios de mapas previamente definidos en servidores internacionales y en Cuba es difícil que los dispositivos móviles logren usar estos servicios todo el tiempo, por el poco ancho de banda y la infraestructura actual que posee el país. En la línea se han desarrollado varias aplicaciones, pero las mismas no son reutilizables, ya que son soluciones a la medida para clientes y entornos específicos y en su mayoría han sido adaptaciones de herramientas internacionales, arrastrando los problemas de las mismas. Además estas son soluciones para dispositivos de alta capacidad de procesamiento y en el proyecto no se ha desarrollado una solución para dispositivos de pocas prestaciones. Todos estos problemas traen consigo que no se ahorre tiempo en el proyecto, no se facilite el trabajo del equipo de desarrollo y que los sistemas existentes no permitan adaptarse a las necesidades de la mayoría de los clientes, de una forma rápida y eficiente, al no poder reutilizar las soluciones que ya se han desarrollado. Por lo que se define **como problema a resolver:**

¿Cómo agilizar el desarrollo de Sistemas de Información Geográfica para dispositivos móviles en la Universidad de las Ciencias Informáticas?

La investigación tiene como **objeto de estudio** el proceso de desarrollo de Sistemas de Información Geográfica, constituyendo el **campo de acción** el proceso de desarrollo de Sistemas de Información Geográfica para dispositivos móviles.

El **objetivo general** de la investigación es desarrollar una plataforma para agilizar el desarrollo de Sistemas de Información Geográfica para dispositivos móviles en la Universidad de las Ciencias Informáticas.

Como **idea a defender** se plantea que con el desarrollo de una plataforma de Sistemas de Información Geográfica para dispositivos móviles en la Universidad de las Ciencias Informáticas se logrará desarrollar un Sistema de Información Geográfica con mayor facilidad.

Para darle solución al objetivo planteado se definen las siguientes **tareas investigativas**:

- ✓ Fundamentar las tendencias actuales, tecnologías y conceptos más importantes relacionados con los Sistemas de Información Geográfica.
- ✓ Analizar y seleccionar la metodología de desarrollo de *software* a utilizar.
- ✓ Analizar y seleccionar las herramientas para el desarrollo de la aplicación informática.
- ✓ Especificar los requisitos funcionales y no funcionales del sistema.
- ✓ Desarrollar el diseño de la aplicación informática.
- ✓ Implementar la Plataforma de Sistemas de Información Geográfica para dispositivos móviles.
- ✓ Realizar pruebas funcionales a la Plataforma de Sistemas de Información Geográfica para verificar su buen funcionamiento.

Es por ello que se considera como **posibles resultados**:

- ✓ Los principales documentos y artefactos asociados al proceso de desarrollo, definidos por la metodología escogida.
- ✓ Plataforma para el desarrollo de Sistemas de Información Geográfica para dispositivos móviles.

Con el objetivo de alcanzar la solución deseada y de facilitar el desarrollo de la investigación, se utilizan los siguientes **métodos de investigación**:

Métodos teóricos:

- ✓ El método Analítico - Sintético se utiliza con el fin de analizar libros, páginas web y otras bibliografías para buscar los elementos más importantes que se relacionan con esta investigación.

- ✓ El método Histórico-Lógico se utiliza con el objetivo de verificar teóricamente cómo ha evolucionado el tema tratado en esta investigación.
- ✓ El método Modelación se utiliza con el objetivo de reproducir la interacción de los objetos en la vida real mediante la creación de modelos.

Métodos empíricos:

- ✓ La Entrevista no Estructurada se utiliza con el objetivo de obtener información valiosa que permita conocer las principales funcionalidades a implementar en la plataforma.

El presente trabajo se conforma de la siguiente manera: introducción, tres capítulos, conclusiones, recomendaciones y bibliografía.

Los capítulos abordan los siguientes temas:

CAPÍTULO 1: Fundamentación teórica. Incluye los principales conceptos asociados al dominio del problema, un estudio de los SIG existentes para dispositivos móviles más importantes para esta investigación, de las metodologías de desarrollo de s*oftware*, los lenguajes de modelado, los lenguajes de programación para dispositivos móviles, J2ME como edición reducida de *java*, los entornos de desarrollo integrado, las herramientas de ingeniería de *software* asistida por computadora y el servidor de mapas a utilizar en el desarrollo de la aplicación.

CAPÍTULO 2: Características de la Plataforma para el desarrollo de SIG. Define las principales características que debe cumplir el sistema a desarrollar en términos de requisitos funcionales y no funcionales. Se enumeran los requisitos funcionales y no funcionales del sistema, mediante diagramas de casos de uso se describen las relaciones existentes de los actores que se involucran con la plataforma y mediante la descripción textual de casos de uso se explica la propuesta de funcionamiento de la plataforma.

CAPÍTULO 3: Construcción y prueba de la solución propuesta. Se hace referencia a las bibliotecas de clases, los patrones arquitectónicos y de diseño que se utilizan para la construcción de la plataforma, se implementan las funcionalidades que requiere la misma, se describen las pruebas funcionales y se mencionan las relevancias sociales, económicas y técnicas de la plataforma.

CAPÍTULO 1: Fundamentación teórica

Introducción

El presente capítulo incluye los principales conceptos asociados al dominio del problema, un estudio de los SIG existentes para dispositivos móviles más importantes para esta investigación, de las metodologías de desarrollo de s*oftware*, los lenguajes de modelado, los lenguajes de programación para dispositivos móviles, la edición reducida de java, los entornos de desarrollo integrado, las herramientas de ingeniería de *software* asistida por computadora y el servidor de mapas a utilizar en el desarrollo de la aplicación.

1.1. Conceptos asociados al dominio del problema

Para entender mejor los temas que serán abordados en la investigación, se hace necesario relacionar a continuación un conjunto de conceptos asociados al dominio del problema.

1.1.1 Plataforma de desarrollo de *software*

Para comprender que es un plataforma de desarrollo de *software* es necesario analizar los conceptos por separado de plataforma informática y plataforma para el desarrollo de *software* que se presentan a continuación.

Plataforma Informática:

Es un término de carácter genérico que designa normalmente una arquitectura de *hardware*, aunque también se usa a veces para sistemas operativos o para el conjunto de ambos. Los ordenadores VAX de la firma Digital, por ejemplo, serían una plataforma en la que se pueden soportar aplicaciones que, a su vez, corren (ver correr) en otras plataformas (Glosario de Informática e Internet, 2006).

Es un sistema que sirve como base para hacer funcionar determinados módulos de *hardware* o de *software* con los que es compatible. Dicho sistema está definido por un estándar alrededor del cual se determina una arquitectura de *hardware* y una plataforma de *software* (incluyendo entornos de aplicaciones). Al definir plataformas se establecen los tipos de arquitectura, sistema operativo, lenguaje de programación o interfaz de usuario compatibles (RED DE APRENDIZAJE, 2012).

Es la tecnología básica de *software* de un sistema informático que define cómo se maneja una computadora y determina qué otros tipos de *software* se pueden utilizar (Answers, 2012).

Plataforma de Desarrollo de *Software:*

Es una solución completa para desarrollar *software* y sistemas basados en *software*. Permite al equipo de desarrollo que utilice una plataforma de desarrollo de *software* construir, integrar, extender, modernizar e implantar *software* y sistemas basados en *software* (GSInnova, 2007)

De forma general y de acuerdo a los conceptos planteados anteriormente una plataforma para el desarrollo de *software* es un sistema informático base, sobre el cual se ejecutan o realizan un conjunto de servicios que este sistema puede ofrecer.

1.1.2 Cartografía

La cartografía es la ciencia que se encarga de expresar gráficamente a través de los mapas las características naturales y sociales de la tierra (GIS Dictionary).

Por otra parte la **R**eal **A**cademia **E**spañola (en lo adelante **RAE**) plantea que la cartografía es el arte de trazar mapas geográficos (RAE, 2001).

Por lo que se puede decir que la cartografía estudia la representación gráfica de determinados datos geográficos en un mapa.

1.1.3 Sistemas de Información Geográfica

Los Sistemas de Información Geográfica son un sistema de *hardware*, *software* y procedimientos, diseñados para soportar la captura, el manejo, la manipulación, el análisis, el modelado y el despliegue de datos espacialmente referenciados, para la solución de los problemas complejos del manejo y planeamiento territorial (Candeaux Duffatt, et al., 1994).

Se define a un Sistema de Información Geográfica como aquel que integra el *hardware*, el *software* y los datos geográficos, con el fin de resolver problemas complejos de planificación, análisis, manipulación, gestión, modelado, y exposición de datos espacialmente referenciados (NCGIA, 2008).

Con los conceptos planteados anteriormente se puede llegar a la conclusión de que un SIG es un sistema que integra tecnologías informáticas, personas e información geográfica, cuya principal función consiste en capturar, analizar, manejar, editar, almacenar y representar datos geográficos.

1.1.4 Dispositivo Móvil

Un dispositivo móvil es un micro-ordenador lo suficientemente ligero como para ser transportado por una persona, y el mismo dispone de una capacidad de batería suficiente como para poder funcionar de forma autónoma, normalmente son versiones limitadas en prestaciones y por tanto en funcionalidades (Tardáguila Moro, 2009).

Para comprender mejor que es un dispositivo móvil es necesario conocer los términos que a continuación se mencionan.

Teléfono:

El teléfono es un conjunto de aparatos e hilos conductores con los cuales se transmite a distancia la palabra y toda clase de sonidos por la acción de la electricidad, además a cada teléfono se le asigna un número (RAE, 2001).

Dispositivo:

Un dispositivo es un mecanismo o artificio dispuesto para producir una acción prevista (RAE, 2001).

Móvil:

Un móvil es un aparato portátil de un sistema de telefonía móvil (RAE, 2001).

Por lo que un dispositivo móvil es un objeto (generalmente pequeño) que se caracteriza por su gran portabilidad, a través del cual se puede tener acceso o no a determinados servicios telefónicos. Generalemente estos dispositivos poseen una memoria limitada.

1.1.5 Sistemas de Información Geográfica para Dispositivos Móviles

Los SIG para móviles son una combinación de las tecnologías SIG con los dispositivos móviles, el acceso inalámbrico a *Internet* y los sistemas de posicionamiento. Ofrecen la ventaja de poder acceder a la información desde cualquier lugar, de acuerdo a las posibilidades que brindan los dispositivos móviles, en cuanto a su portabilidad y funcionalidad. Existen dos tipos de SIG para móviles los cuales son:

SIG convencional: en donde el sistema se ejecuta en el dispositivo, centrándose más en los trabajos propios del SIG y en la recolección y edición de datos, el usuario es el operario del SIG (Olaya, 2011).

Servicios basados en localización: son servicios ofrecidos por terceros en función de la posición del dispositivo y del usuario, este pasa a ser el consumidor de un servicio (Olaya, 2011).

La plataforma se desarrollará utilizando el SIG convencional porque solo realizará las funcionalidades básicas de un SIG.

1.2. Sistemas de Información Geográfica como objeto de estudio

A continuación se explican los SIG como objeto de estudio y se realiza una descripción de los principales problemas que existen en el proyecto gvMap, para comprender mejor como se desarrollará la investigación.

1.2.1 Descripción General

Los SIG son muy utilizados ya que permiten brindar soluciones a muchos problemas de la vida cotidiana, como por ejemplo en servicios informativos de transporte urbano, correo postal, turismo de ciudad, entre otros, los cuales requieren que se accedan a varios tipos de información, que sólo pueden ser visualizadas geográficamente. Los SIG permiten almacenar y manipular información usando la geografía para analizar patrones, relaciones y tendencias en la información, todo encaminado a tomar mejores decisiones.

Estos generalmente están compuestos por programas geográficos, datos, recursos humanos y métodos. Estos sistemas informáticos contribuyen al estudio de la distribución y monitoreo de recursos naturales, humanos, tecnológicos, de infraestructura y sociales, así como en la evaluación del impacto de las actividades humanas sobre el medio ambiente. Los SIG son aplicados a diferentes sectores de la sociedad, los cuales son utilizados como una herramienta de ayuda a la toma de decisiones.

Los SIG funcionan con dos tipos de información geográfica: el modelo vectorial y el modelo ráster como se plantea en (The University of Melbourne, 1999):

- ✓ Modelo ráster: es un método para el almacenamiento, el procesamiento y la visualización de datos geográficos. Cada superficie a representar se divide en filas y columnas, formando una malla o rejilla regular. Cada celda de la rejilla guarda tanto las coordenadas de la localización como el valor temático. Los datos ráster son una abstracción de la realidad.
- ✓ Modelo vectorial: es una estructura de datos utilizada para almacenar datos geográficos. Los datos vectoriales constan de líneas o arcos, definidos por sus puntos de inicio y fin, y puntos donde se cruzan varios arcos, los nodos. La localización de los nodos y la estructura topológica se almacena de forma explícita. Las entidades quedan definidas por sus límites solamente y los segmentos curvos se representan como una serie de arcos conectados. En el modelo de datos vectorial, los datos geográficos se representan en forma de coordenadas.

1.2.2 Descripción actual del dominio del problema

El proyecto Control de Flotas perteneciente al Centro GEYSED, posee varias líneas de investigación y una de ellas está orientada al desarrollo de SIG para dispositivos móviles. La misma está subdividida a

su vez en dos líneas de desarrollo, la primera es Movilmap que se encarga de que los dispositivos se conecten de forma inalámbrica a determinados servicios y la segunda es gvMap, que está destinada a la realización de aplicaciones nativas que permitan aprovechar al máximo las funcionalidades que brindan estos dispositivos.

La línea correspondiente a gvMap está basada en el desarrollo de SIG para dispositivos móviles. Una parte del equipo de desarrollo se encuentra estudiando diferentes herramientas de código abierto, para poder reutilizar algunas funcionalidades de estas y adaptarlas a las necesidades de la línea. En gvMap solamente se han desarrollado aplicaciones basadas en la plataforma gvSIG Mobile, la cual está destinada a los dispositivos de gran capacidad de procesamiento.

1.2.3 Situación Problemática

La línea de investigación y desarrollo gvMap tiene pocos años de creada, en la misma se analizan diferentes herramientas existentes en el mundo para el desarrollo de los SIG, ya que estas son muy cambiantes de acuerdo a la tecnologías actuales. En esta línea se han desarrollado varias aplicaciones entre las que se encuentran el SIG UCI y la aplicación de Mapas para las Rutas de Transporte de La Habana. Estas soluciones no son reutilizables ya que son soluciones a la medida para clientes y entornos específicos y las mismas han sido adaptaciones de herramientas internacionales, arrastrando los problemas de estas herramientas. Además estas aplicaciones están destinadas a dispositivos de gran capacidad de procesamiento. Hasta el momento no se ha desarrollado una aplicación en gvMap destinada a los dispositivos móviles de poca capacidad de procesamiento, que sea capaz de implementar funcionalidades comunes para los SIG en la mayoría de los dispositivos móviles. Todos estos problemas traen consigo que no se ahorre tiempo en el proyecto, no se facilite el trabajo del equipo de desarrollo y que los sistemas existentes no permitan adaptarse a las necesidades de la mayoría de los clientes, de una forma rápida y eficiente, al no poder reutilizar las soluciones que ya se han desarrollado.

1.3. Análisis de otras soluciones existentes

Actualmente existen una gran cantidad de soluciones de SIG destinadas a dispositivos móviles. Con el objetivo de identificar las funcionalidades necesarias para la plataforma que se desea desarrollar, a continuación se hace un resumen de las principales plataformas para el desarrollo de SIG para móviles, así como algunas personalizaciones de las mismas realizadas en la línea gvMap.

1.3.1. gvSIG Mobile

Es una plataforma creada en la Consejería de Infraestructuras y Transporte (CIT) de España, permite el desarrollo abierto y de libre distribución con todas las capacidades de un SIG para móviles. La misma, basa su desarrollo en la plataforma gvSIG Desktop. Surge como una necesidad para ampliar los entornos de ejecución de gvSIG Desktop, a una gama de dispositivos móviles, basados en la edición de *java* para móviles, utilizando la Configuración de Dispositivos Conectados (ver Configuración de Dispositivos Conectados) para dar respuesta a las necesidades de un creciente número de usuarios. La plataforma aprovecha muchas de las funcionalidades ya desarrolladas en gvSIG Desktop, agregando nuevas funcionalidades como son: la gestión de capas, la gestión de proyectos, la visualización de información local y remota usando el Servicio Web de Mapa (en lo adelante WMS, por sus siglas en inglés ***W****eb* ***M****ap* ***S****ervice*), la consulta de información alfanumérica de elementos y la edición de datos mediante formularios personalizados (Montesinos, et al., 2009)**.**

1.3.2. gvSIG Mini

gvSIG Mini es una plataforma de código abierto, la cual basa su desarrollo en gvSIG Mobile. Surge como una necesidad para ampliar las entornos de ejecución de gvSIG Mobile, a una gama de dispositivos móviles basados en la edición de *java* para móviles, utilizando la Configuración Limitada de Dispositivos Conectados (ver Configuración Limitada de Dispositivos Conectados) para dar respuesta a las necesidades de un creciente número de usuarios. gvSIG Mini posee una arquitectura basada en multihilos (del inglés *multithreading*), para la creación de la interfaz gráfica de usuario utiliza la biblioteca de clases *Kit* de Herramientas de Interfaz de Usuario Ligero (en lo adelante **LWUIT**, por sus siglas en inglés ***L****ight* ***W****eight* ***U****ser* ***I****nterface* ***T****oolkit),* permitiendo que la aplicación se comporte igual en todos los dispositivos, de forma análoga a *Swing* en la edición estándar de *java.* En cuanto a la visualización de los mapas utiliza algunos servicios de mapas de *Internet como son Yahoo Maps, Name Finder, Google Maps y Open Street Map* (Montesinos, et al., 2009).

A continuación se mencionan las aplicaciones desarrollada en la línea gvMap.

1.3.3. SIG-UCI para Móviles

El SIG-UCI para Móviles es una personalización de gvSIG Mobile y del SIG-UCI para computadoras. Entre sus principales funcionalidades se encuentran la visualización y el análisis de la información geográfica y socioeconómica de la UCI; a partir de las capas que contiene el sistema y de los servicios web que brinda el repositorio de la Universidad. De esa forma se logró facilitar la toma de decisiones en la comunidad universitaria (Vázquez Manso, 2011).

1.3.4. Aplicación de Mapas para las Rutas de Transporte de La Habana

La Aplicación de Mapas para las Rutas de Transporte de La Habana es otra personalización realizada de gvSIG Mobile, heredando la mayoría de las características de este sistema. Se realizó para conocer las rutas del sistema de transporte público de La Habana. La misma utiliza la cartografía del Transporte Urbano de La Habana e información socio-económico asociada a esta cartografía. Esta personalización permite localizar paradas de intercambio, rutas y la visualización de la información socio-económica asociada a estas rutas de transporte (Comas Pérez, 2011).

1.3.5. Análisis de las soluciones existentes

Entre las principales deficiencias que poseen las aplicaciones analizadas anteriormente, se encuentran que las mismas se conectan a servidores de mapas predefinidos y estas deben estar conectadas en todo momento a *Internet* para obtener diferentes servicios de mapas. De acuerdo a las limitantes mencionadas anteriormente, la plataforma a desarrollar debe darle solución a estos problemas, funcionando para la mayoría de los móviles, permitiendo la conexión con varios servidores de mapas que soporten el servicio WMS y el trabajo sin conexión a *Internet* utilizando imágenes. La principal desventaja en cuanto al trabajo sin conexión de la plataforma a desarrollar radica en que se trabajará con la imagen de un mapa previamente guardada en el móvil y sobre esta no se podrá realizar ciertas acciones.

1.4. Metodologías de Desarrollo de *Software*

Antes de definir lo que es una metodología de desarrollo de *software*, es necesario preguntarse qué es un proceso de desarrollo de *software*, que no es más que un proceso el cual define “quién” está haciendo “qué”, “cuándo” y “cómo” alcanzar un determinado objetivo (Jacobson, et al., 2000).

Una metodología de desarrollo de *software* es un conjunto de pasos y procedimientos que deben seguirse para desarrollar *software*. Una metodología define cómo dividir un proyecto en etapas, qué tareas se llevan a cabo en cada etapa, qué restricciones deben aplicarse, qué técnicas y herramientas se emplean y cómo se controla y gestiona el proceso de desarrollo de *software* (Hernando, 2009).

Otros autores plantean que una metodología de *software* es una colección de métodos aplicados a lo largo del proceso de desarrollo de *software* y unificados por alguna aproximación general o filosófica, donde los métodos aplicados son un proceso disciplinado para generar un conjunto de modelos que

describen varios aspectos de un sistema de *software* en desarrollo, utilizando alguna notación bien definida (Cueva Lovelle, 1999).[1]

Las metodologías de desarrollo de *software* se dividen en dos grandes grupos, las tradicionales o pesadas que se basan en la idea de que el éxito del producto se puede lograr si se tiene todo correctamente documentado y las ágiles que se centran en el proceso de desarrollo de *software* y no en la documentación alrededor de este, no obviando por completo la documentación (Tinoco Gómez, et al., 2010). A continuación en la *Tabla 1: Comparación entre metodologías ágiles y tradicionales,* se muestran los principales elementos entre estos tipos de metodologías.

Metodología Ágil	Metodología Tradicional
Pocos artefactos.	Más artefactos.
Pocos roles.	Más roles.
No existe un contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo (además *in-situ*).	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (< 10 integrantes) y trabajando en el mismo sitio.	Grupos grandes.
Menos énfasis en la arquitectura.	La arquitectura es esencial.

Tabla 1: *Comparación entre metodologías ágiles y tradicionales.*[2]

1.4.1. Proceso Unificado de Rational

El Proceso Unificado de *Rational* (en lo adelante RUP, por sus siglas en inglés ***R****ational* ***U****nified* ***P****rocess*), es una metodología pesada, donde el proceso de desarrollo de *software* ofrece un conjunto de actividades para transformar los requisitos de un usuario en un sistema informático.

RUP es una metodología adaptable al contexto y a las necesidades de cada organización, que en conjunto con el Lenguaje Unificado de Modelado, constituye una metodología muy utilizada para la construcción y documentación de sistemas informáticos.

[1] Para la investigación este es el concepto que más se adapta a la misma.

[2] *Extracto de la tabla presente en* (Tinoco Gómez, et al., 2010).

El ciclo de vida de RUP, según (Jacobson, et al., 2000), está caracterizado por ser:

- ✓ Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requisitos. A partir de aquí los casos de uso guían el proceso de desarrollo.
- ✓ Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los casos de uso relevantes desde el punto de vista de la arquitectura.
- ✓ Iterativo e Incremental: RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros.

Al aplicar en un proyecto de desarrollo de *software* la metodología RUP, este estará compuesto por cuatro fases y nueve flujos de trabajo (de los cuales seis son ingenieriles y tres de soporte), tal como se muestra en la *Figura 1: Proceso de desarrollo según RUP.*

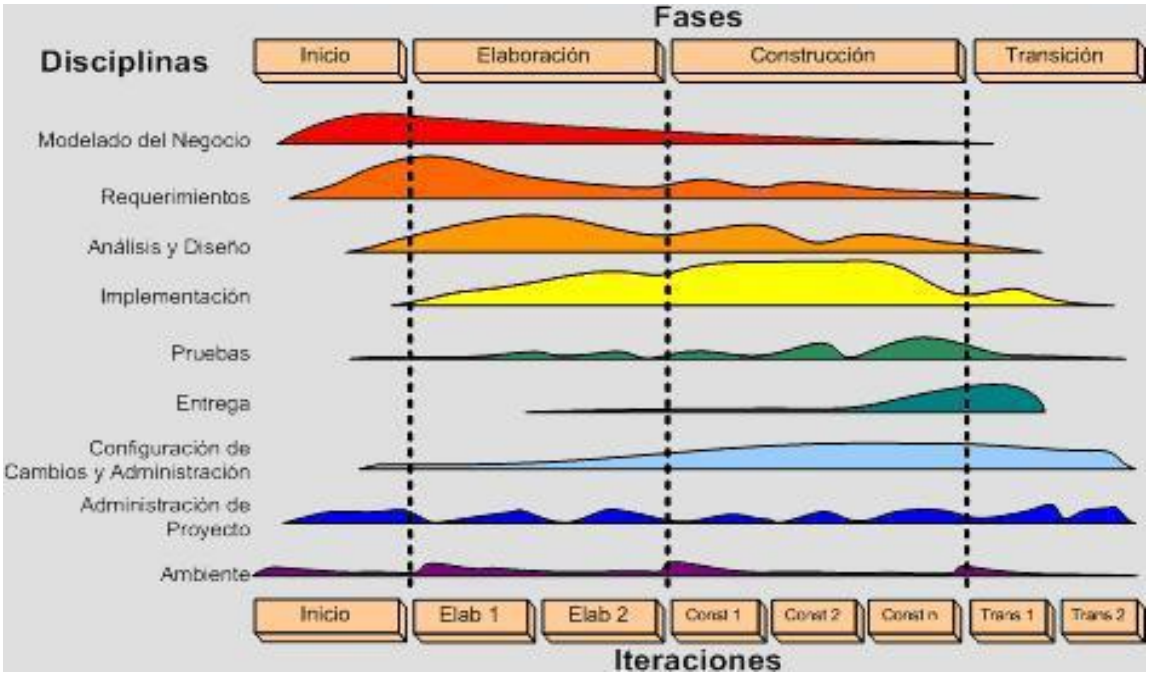

Figura 1: *Proceso de desarrollo según RUP.*[3]

Las fases en las que se dividirá el proyecto son:

1. Inicio: permite desarrollar el análisis del negocio hasta un determinado punto para justificar de esta forma la puesta en marcha del proyecto. Al finalizar la fase todos los involucrados en el proyecto deben tener una comprensión bastante buena de la idea del mismo y de que este es factible llevarlo a cabo.
2. Elaboración: permite recopilar la mayor parte de los requisitos que aún quedan pendientes y establece la línea base de la arquitectura. Al finalizar la fase se obtiene generalmente un modelo completo de negocio que describe el contexto del sistema y una arquitectura robusta que da cumplimiento a todos los requisitos.
3. Construcción: desarrolla un producto de *software* inicial listo para ser usado por los usuarios, llamado comúnmente versión beta. Al finalizar la fase se obtiene un producto de *software* "completo" que al usuario interactuar con él, el mismo puede develar algunos errores y estos pueden ser modificados.
4. Transición: permite implantar el producto en el entorno del cliente. Al finalizar esta fase se obtiene el producto de *software* final libre de errores y con toda su documentación asociada.

Los flujos de trabajo por los cuales transitará el proyecto son (Jacobson, et al., 2000):

[3] *Tomado de* (Prieto Alvarez, 2009)

1. Modelado del Negocio: describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
2. Requerimientos: define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
3. Análisis y diseño: describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requisitos), por lo que indica con precisión lo que se debe programar.
4. Implementación: define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
5. Pruebas: busca los defectos a los largo del ciclo de vida.
6. Instalación o despliegue: produce una versión ejecutable (del término *release* en inglés) del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el *software* a los usuarios finales.
7. Administración del proyecto: involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
8. Administración de configuración y cambios: describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, entre otros.
9. Ambiente: contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

Debido a que RUP es una metodología de desarrollo de *software* tradicional, que plantea que debe existir un contrato prefijado, que el grupo de desarrollo de *software* debe ser de más de diez personas y que se deben crear artefactos por cada fase y flujo de trabajo, no se decide utilizar RUP ya que el grupo de desarrolladores de la línea gvMap es de solo dos personas y además la creación de los artefactos que define RUP por cada fase y flujo de trabajo hacen engorroso el proceso de desarrollo de la plataforma, atentando contra el tiempo de entrega de la misma.

1.4.2. Programación Extrema

La Programación Extrema (en lo adelante XP, por sus siglas en inglés *e**X**treme **P**rogramming*), es una metodología ágil, la cual se centra en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de *software*, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. Le da prioridad a las tareas que dan un resultado directo y que reducen la burocracia que hay alrededor de la programación (Beck, 1999).

Dentro de sus principales características se destacan:

- ✓ **Pruebas Unitarias:** Se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándose en algo hacia el futuro, puedan hacerse pruebas de las fallas que pudieran ocurrir.
- ✓ **Re fabricación:** Se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- ✓ **Programación en pares:** Una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en la misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.

XP plantea que todo proyecto transita por cuatro fases como se muestra en la *Figura 2: Metodología XP.*

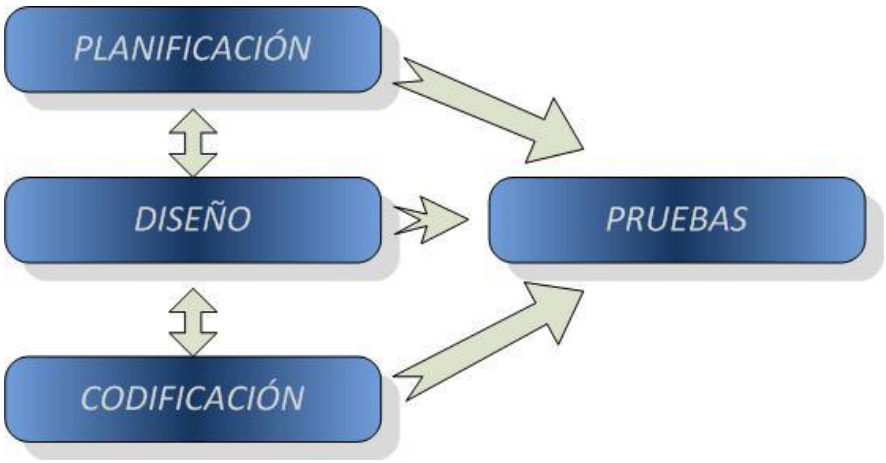

Figura 2: *Metodología XP.*[4]

Las cuatro fases según (Castillo, et al., 2011) consisten en:

[4] Tomada de (PROGRAMACION WEB, 2011)

- ✓ Planificación: permite definir historias de usuario (la cual tiene la misma finalidad que los casos de uso pero con algunas diferencias), se crea un plan de publicaciones (*release planning*) donde los desarrolladores y clientes establecen los tiempos de implementación ideales de las historias de usuario, la prioridad con la que serán implementadas y las historias que serán implementadas en cada versión del programa, esta fase se divide en iteraciones las cuales no pueden exceder de tres semanas de duración, se mide la velocidad del proyecto la cual representa la rapidez con la que se desarrolla el *software*, permite la programación en parejas ya que incrementa la productividad y la calidad del *software* desarrollado y además es necesario que los desarrolladores se reúnan diariamente y expongan sus problemas, soluciones e ideas de forma conjunta.
- ✓ Diseño: permite la realización de diseños simples, los cuales serán fácilmente entendibles, que a la larga costará menos tiempo y esfuerzo desarrollar, se crea un glosario de términos el cual ayudará a comprender el diseño y facilitará sus posteriores ampliaciones y la reutilización del código, se definen riesgos, no permite añadir funcionalidades extras que no se necesiten en ese momento, ya que es un desperdicio de tiempo y recursos, permite refactorizar lo cual no es más que mejorar y modificar la estructura y codificación de códigos ya creados sin alterar su funcionalidad, en esta fase se utilizan las tarjetas C.R.C (Clase, Responsabilidad y Colaboración, del inglés ***C****lass,* ***R****esponsabilities and* ***C****ollaboration*) que permiten al programador centrarse y apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de la programación procedural clásica.
- ✓ Codificación: debe hacerse ateniendo a estándares de codificación previamente creados ya que al programar bajo estándares, mantiene el código consistente y facilita su comprensión y escalabilidad. Se deben crear pruebas para verificar el correcto funcionamiento de los distintos códigos implementados. XP sugiere un modelo de trabajo el cual usa repositorios de código dónde las parejas de programadores publican cada pocas horas sus códigos implementados y corregidos junto a las pruebas que deben aplicar.
- ✓ Prueba: permite comprobar el funcionamiento de los códigos que se vayan implementando.

El sistema a desarrollar cuenta con requisitos bien definidos y estables, necesita ser documentado de forma eficiente para los usuarios finales y para su posterior uso y el equipo de desarrollo no está familiarizado con esta metodología, motivos por los cuales XP no es la metodología óptima para el desarrollo de la aplicación.

1.4.3. Proceso Unificado Ágil

El Proceso Unificado Ágil (en lo adelante AUP, por sus siglas en inglés ***A**gile **U**nified **P**rocess*), es una versión simplificada de RUP, fácil de entender y aplicar, debido a que utiliza varios conceptos vigentes en RUP, adaptados al contexto de una metodología ágil. En AUP, se puede apreciar que los flujos de trabajo con respecto a RUP cambian, dado que el flujo de modelo, engloba lo que en RUP es modelado del negocio, requerimientos y análisis y diseño (Adempiere, 2011), como se muestra en la *Figura 3: Metodología AUP.*

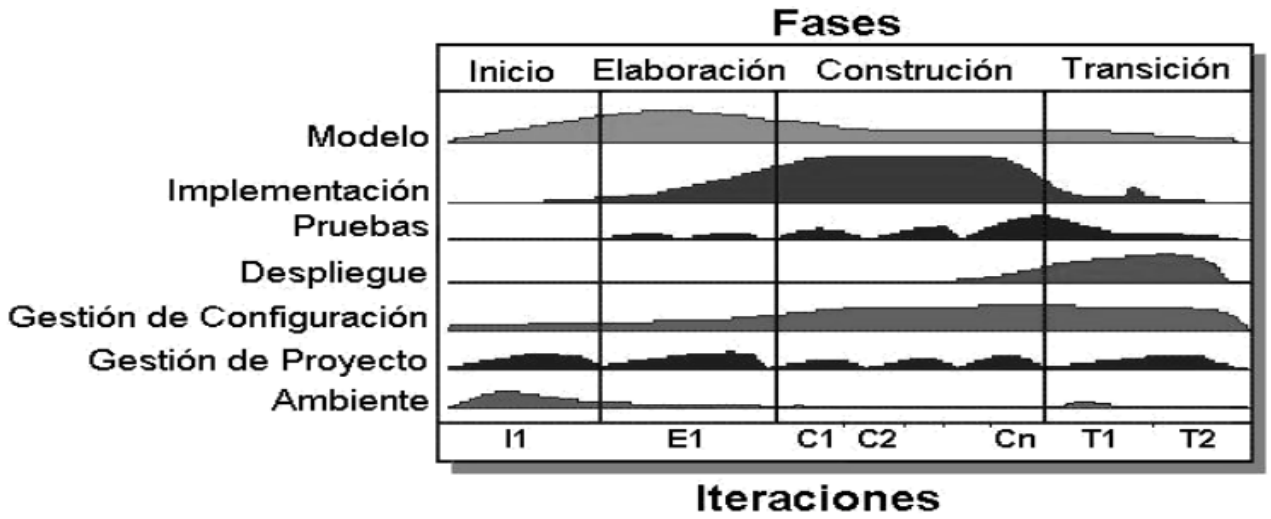

Figura 3: *Metodología AUP.*[5]

Un producto desarrollado bajo la metodología AUP sigue los siguientes principios básicos:

- ✓ El personal sabe lo que se está haciendo: Los integrantes del equipo de desarrollo no necesitan leer documentación de procesos detalladamente, aunque se pueden definir capacitaciones.
- ✓ Simplicidad: Todo el proceso de desarrollo se documenta con pocos documentos, de pocas páginas cada uno de ellos.
- ✓ Enfocado a las actividades de alto nivel: El proceso se centra en las actividades que realmente cuentan, no en cada detalle posible que podría pasar en un proyecto.

Debido a que la plataforma debe ser correctamente documentada para que se continúen desarrollando versiones de la misma y que el proceso de desarrollo de *software* con el uso de una metodología ágil es más cómodo que con el uso de una metodología tradicional por las características del equipo de

[5] Tomado de (Adempiere, 2011)

desarrollo, se decide utilizar AUP, como metodología de desarrollo ya que la misma es una variante de RUP pero adaptada a una metodología ágil.

1.5. Lenguajes de Modelado

Un lenguaje de modelado no es más que la notación (en su mayoría gráfica) que utilizan los métodos para expresar los diseños (Cueva Lovelle, 1999). A continuación se analizará el Lenguaje Unificado de Modelado el cual es utilizado por la metodología escogida anteriormente.

1.5.1. Lenguaje Unificado de Modelado

El Lenguaje Unificado de Modelado (en lo adelante **UML**, por sus siglas en inglés ***U****nified* ***M****odeling* ***L****anguage*) es un lenguaje estándar de modelado para *software*, que permite visualizar, especificar, construir y documentar los artefactos de un sistema, UML le da la posibilidad a los desarrolladores de visualizar el resultado de su trabajo en esquemas o diagramas estandarizados. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida y dominios de aplicación (Jacobson, et al., 2000).

Debido a las características antes expuestas, estas hacen propicia la elección de UML en su versión 2.0 como el lenguaje de modelado para la documentación de la aplicación.

1.6. Lenguajes de Programación para dispositivos móviles

Para comprender que es un lenguaje de programación es necesario consultar el concepto de programa que plantea el diccionario técnico (ctisa, 2009) que es: *“Un conjunto de órdenes para un ordenador. Un programa puede estar formado por apenas unas pocas órdenes (por ejemplo, uno que sume dos números) o por varios miles de órdenes (como un programa de gestión completo para una empresa). Cuando se trata de un programa ya terminado que se compra, se suele hablar de una Aplicación Informática. Los programas se deben escribir en un cierto lenguaje de programación. Los lenguajes de programación que se acercan más al lenguaje humano que al del ordenador reciben el nombre de "lenguajes de alto nivel" (como Pascal); los que se acercan más al ordenador son los de "bajo nivel" (como el ensamblador). Lo más habitual es crear los programas en un lenguaje de alto nivel (llamado " fuente ") y después convertirlos al lenguaje propio del ordenador ("compilarlos" para obtener un "ejecutable").”*

De esta forma se puede decir que un lenguaje de programación permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que pone a disposición del programador, para que éste pueda comunicarse con los dispositivos de *hardware* y *software* existentes.

Los lenguajes de programación para dispositivos móviles deben tener en cuenta la capacidad de procesamiento y de memoria de estos dispositivos, que no es igual al de las computadoras.

1.6.1. C/C++

C fue creado en 1972 por Kenneth L. Thompson y Dennis M. Ritchie en los Laboratorios Bell y a mediados de 1980 se crea C++ por Bjarne Stroustrup con el objetivo de extender el exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. C++ es a la vez un lenguaje procedural (orientado a algoritmos) y orientado a objetos. Como lenguaje procedural se asemeja a C y son compatible, aunque presenta ciertas ventajas con respecto a él. Como lenguaje orientado a objetos se basa en una filosofía completamente diferente, que exige del programador un completo cambio de mentalidad.

Las diversas razones por la cual se ha convertido en el tercer lenguaje más utilizado a nivel mundial según (TIOBE Software, 2011), son:

- ✓ El uso de constructores de alto nivel.
- ✓ El poder manejar actividades de bajo nivel.
- ✓ El generar programas eficientes.
- ✓ La posibilidad de poder ser compilado en una variedad de computadoras, con pocos cambios (portabilidad).
- ✓ Un punto en contra es que tiene una detección pobre de errores, lo cual en ocasiones es problemático para los programadores novatos.

C/C++ es un lenguaje no gestionado donde el programador es el que debe gestionar la memoria y el ciclo de vida de los objetos, lo que hace que sea muy propenso a errores, motivo por el cual no se decide utilizar este lenguaje para el desarrollo de la aplicación.

1.6.2. Java

La compañía *Sun Microsystems* creadora de *java* en 1991 describe el lenguaje como *"simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico"* (García de Jalón, et al., 2000).

Entre las características más notables de *java* se encuentran:

- ✓ Robusto.
- ✓ Gestiona la memoria automáticamente.

- ✓ No permite el uso de técnicas de programación inadecuadas.
- ✓ Soporta multihilos de ejecución.
- ✓ Mecanismos de seguridad incorporados.
- ✓ Herramientas de documentación incorporadas.
- ✓ Lenguaje independiente de la plataforma.
- ✓ El código funciona sobre cualquier plataforma de *software* y *hardware*.
- ✓ Es un lenguaje Orientado a Objetos.

La clave para que se lograra la portabilidad de *java* consistió en desarrollar un código "neutro" el cual estuviera preparado para ser ejecutado sobre una "máquina hipotética o virtual", denominada Máquina Virtual de *Java* (en lo adelante **JVM**, por sus siglas en inglés ***J**ava **V**irtual **M**achine*). La JVM interpreta código neutro convirtiéndolo a código particular de la CPU utilizada, lo cual evita tener que realizar un programa diferente para cada CPU o plataforma. La JVM es el intérprete de *java*. Ejecuta los "*bytecodes*" (ficheros compilados con extensión *.*class*) creados por el compilador de *java* (javac.exe). Tiene numerosas opciones, entre las que destaca la posibilidad de utilizar el denominado JIT (*Just-In-Time Compiler*), que puede mejorar entre 10 y 20 veces la velocidad de ejecución de un programa (García de Jalón, et al., 2000).

Para la implementación de la aplicación se ha seleccionado el lenguaje de programación *java* por su eficiencia, por la portabilidad de su plataforma y por la seguridad que aporta, además de que se utiliza en los principales sectores de la industria de desarrollo de *software* para móviles de todo el mundo y está presente en un gran número de dispositivos móviles (ORACLE, 2011).

1.7. Ediciones de *java*

El lenguaje de programación *java*, la máquina virtual y las API en su conjunto, forman la plataforma *j*ava (ver *Figura 4: Plataforma java.*), la cual se puede encontrar en tres ediciones (Nájera, 2005):

1. *Java 2 Platform, Enterprise Edition* (en lo adelante, J2EE): permite cubrir las necesidades que puedan tener las empresas para ofrecer servicios a sus clientes, proveedores y empleados, está diseñada además para computadoras de escritorio, puede trabajar en sistemas operativos como: Windows, Linux, MacOS, Solaris, OS x y otros.
2. *Java 2 Platform, Satandard Edition* (en lo adelante, J2SE): permite satisfacer las necesidades de los usuarios y programadores en sus equipos personales y estaciones de trabajo, es una

plataforma para aplicaciones multiusuario o empresariales, se basa en J2EE y agrega API para el trabajo en el servidor.

3. *Java 2 Micro Edition* (en lo adelante, J2ME): está enfocada tanto para productores de dispositivos portátiles de consumo, como para quienes proporcionan servicios de información aplicables a estos dispositivos, es un conjunto de tecnologías y especificaciones desarrolladas para dispositivos pequeños como los teléfonos celulares y PDA, utiliza derivados de componentes J2SE, como son una máquina virtual más pequeña y un conjunto de API menos potentes.

1.7.1. J2ME como edición reducida de *java*

J2ME es la versión reducida de java orientada a los dispositivos móviles. Debido a que estos dispositivos tienen una potencia de cálculo baja e interfaces de usuario pobres, es necesaria una versión específica de java destinada a los mismos, ya que el resto de las versiones de java (J2SE o J2EE) no cumplen con las características de estos. Por lo que J2ME es una versión "reducida" de J2SE, ver *Figura 4*: *Plataforma java.* (Prieto Alvarez, 2009).

En el mundo de los dispositivos móviles J2ME es muy utilizada, ya que es la única opción real que permite un desarrollo multiplataforma para estos dispositivos, además tiene una estructura altamente modular para adaptarse a las características de cada dispositivo.

Figura 4: *Plataforma java.*[6]

[6] Tomado de (Nájera, 2005)

Para poder tener un entorno de ejecución java en J2ME es necesario que se componga de:

- ✓ Una configuración, que estará compuesta por:
  - Un perfil.
  - Una máquina virtual.
  - Un conjunto de paquetes opcionales.

El perfil y la máquina virtual depende de la configuración escogida y esta última depende de la capacidad de memoria y procesamiento de cada dispositivo móvil. En la *Figura 5: Capas en un dispositivo móvil con soporte J2ME.,* se muestra como se comunican estos elementos.

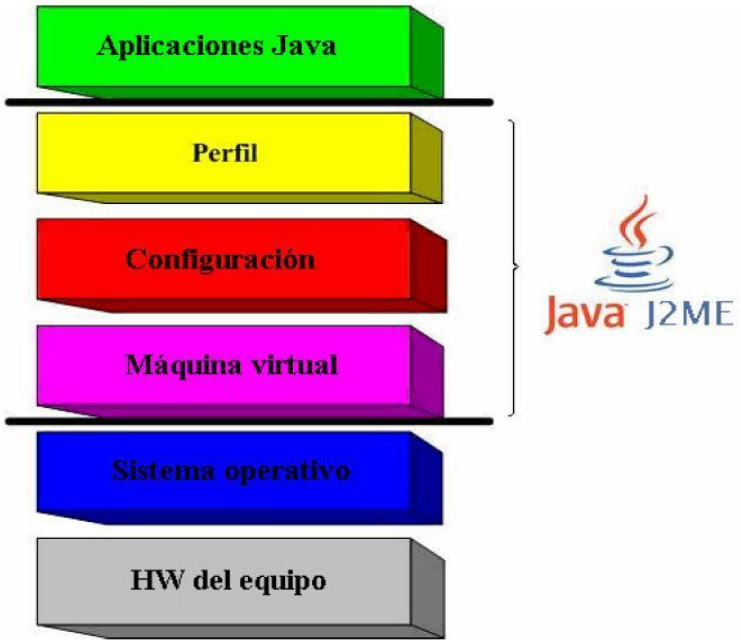

Figura 5: *Capas en un dispositivo móvil con soporte J2ME.*[7]

1.7.1.1. Configuraciones y perfiles para Dispositivos Móviles

Las Configuraciones para Dispositivos Móviles están compuestas por una máquina virtual y un grupo mínimo de API, las cuales proporcionan un conjunto de funcionalidades básicas para determinados dispositivos que compartan características similares, como gestión de memoria o conectividad en la red. Actualmente existen dos configuraciones las cuales son la Configuración de Dispositivos Conectados y la Configuración Limitada de Dispositivos Conectados.

[7] Tomado de (Prieto Donate, 2007)

El Perfil es un grupo más específico de API que proporcionan funcionalidades adicionales a las de la configuración que reside debajo de él, es decir, la configuración se ajusta a una familia de dispositivos y el perfil se orienta hacia un grupo determinado de dispositivos dentro de dicha familia.

1.7.1.1.1. Configuración de Dispositivos Conectados

La Configuración de Dispositivos Conectados (en lo adelante CDC, por sus siglas en inglés ***C**onnected **D**evice **C**onfiguration*), se caracteriza por tener un gran número de interfaces de usuario, procesadores más rápidos, conexiones permanentes a *Internet* de banda ancha de tipo TCP/IP y una capacidad de memoria de entre 2 y 16 *megabytes*. Incluye además una máquina virtual *java* completa y un subconjunto de API de la arquitectura J2SE, se orienta a dispositivos con una CPU de 32 *bits*. Entre los dispositivos que utilizan esta configuración se encuentran las *pocket* PC, los teléfonos inteligentes (en inglés *smart phones*), televisores y sistemas de navegación y entretenimiento para automóviles.

El Perfil Personal (en lo adelante PP, por sus siglas en inglés ***P**ersonal **P**rofile*), es utilizado en la configuración CDC, orientado a dispositivos que requieren una Interfaz Gráfica de Usuario (del inglés *Grafic User Interface*) completa y capacidad de ejecutar *applets* de *Internet*, como por ejemplo PDA de gama alta, consolas de juegos, entre otros. Incluye todas las bibliotecas de funciones de java ***A**bstract **W**indow **T**oolkit* (en lo adelante AWT) y ofrece fidelidad en la web, permitiendo la ejecución de *applets* diseñados para la utilización en entornos de sobremesa.

La configuración CDC utiliza la JVM para la versión de *java* J2ME.

1.7.1.1.2. Configuración Limitada de Dispositivos Conectados

La Configuración Limitada de Dispositivos Conectados (en lo adelante CLDC, por sus siglas en inglés ***C**onnected **L**imited **D**evice **C**onfiguration*), se caracteriza por disponer de interfaces simples, conexión no permanente a *Internet*, por tener 16 o 32 *bits* de CPU, menos ancho de banda y una capacidad de memoria muy reducidos entre 128 y 512 *kilobytes*. Incluye además una Máquina Virtual K (en lo delante KVM, por sus siglas en inglés ***K V**irtual **M**achine*) e implementa una serie de bibliotecas de clases y API que no se encuentran en J2SE y que son específicas de la configuración CLDC (Prieto Donate, 2007).

Parte de CLDC es un subconjunto de CDC, por lo que la portabilidad de aplicaciones se puede conseguir cuando se traslada de un entorno más restringido a otro más rico. De la misma manera, y siguiendo en el hilo de la portabilidad, una aplicación en J2ME podrá ejecutarse en J2SE normalmente, siempre y cuando no utilice las bibliotecas específicas de J2ME ver *Figura 6: Relación J2SE-J2ME.*

Los dispositivos más representativos en CLDC son los teléfonos móviles y las agendas personales (conocidas como PDA) (Prieto Donate, 2007).

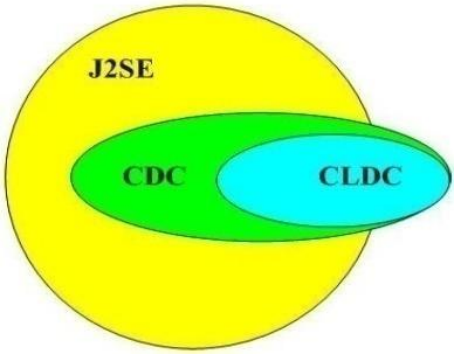

Figura 6: *Relación J2SE-J2ME.*[8]

El Perfil para Dispositivos Móviles de Información (en lo adelante MIDP, por sus siglas en inglés ***M****obile* ***I****nformation* ***D****evice* ***P****rofile*), es utilizado en la configuración CLDC, orientado a dispositivos móviles de recursos limitados y con conexión no permanente a *Internet*. Las aplicaciones J2ME desarrolladas bajo la configuración CLDC y con la especificación del perfil MIDP, se denominan MIDLets. Estos ofrecen funcionalidades básicas para las aplicaciones móviles, incluyendo la interfaz de usuario, conectividad a redes, almacenamiento local de datos y gestión del ciclo de vida de las aplicaciones.

La configuración CLDC utiliza la máquina virtual KVM, la cual está creada para dispositivos de poca capacidad de almacenamiento y con una capacidad de memoria de tan solo unos cuantos centenares de *kilobytes*, de ahí el nombre de KVM toma la K de *kilobyte*, haciendo referencia al poco tamaño que ocupa la plataforma.

1.7.1.2. Paquetes opcionales

La plataforma J2ME se puede ampliar combinando varios paquetes opcionales en CLDC o CDC junto con sus perfiles. Estos paquetes se han creado para responder a requisitos concretos de mercado y ofrecen un conjunto de API estándares para utilizar tanto tecnologías existentes como emergentes; entre estas se incluyen *Bluetooth*, servicios web, mensajería *wireless*, capacidades multimedia o conectividad a bases de datos. Dado que son modulares, los fabricantes de dispositivos pueden incorporarlos según los vayan necesitando para mejorar las características soportadas (Prieto Donate, 2007).

[8] Tomado de (Prieto Donate, 2007)

1.8. Herramientas de Ingeniería de *Software* Asistidas por Computadora

Las Herramientas de Ingeniería de *Software* Asistida por Computadora (en lo adelante CASE, por sus siglas en ingles ***C****omputer* ***A****ided* ***S****oftware* ***E****ngineering*) incluyen una serie de herramientas, lenguajes y técnicas de programación que permiten la generación de aplicaciones de manera semiautomática. Las herramientas CASE liberan al programador de parte de su trabajo y aumentan la calidad del programa, a la vez que disminuyen sus posibles errores (glosario.net, 2006).

1.8.1. Rational Rose

Permite la generación de códigos en Ada, ANSI, C++, CORBA, Java, Visual C++ y *Visual Basic*. Proporciona un lenguaje común de modelado que facilita la rápida creación de un producto. Posibilita además realizar ingeniería inversa para java y da la facilidad de generar documentación a partir de un modelo dado.

Rational Rose es una herramienta que se encarga de llevar a cabo la automatización de los sistemas para la posterior generación de código. Entre las diferentes ventajas que ofrece *Rational Rose,* la más importante es que utiliza la notación estándar en la arquitectura de *software* UML, permitiendo a los desarrolladores y arquitectos de *software* utilizar un lenguaje común para poder visualizar el sistema completo. También los diseñadores pueden modelar sus componentes e interfaces en forma individual y luego unirlos con otros componentes del proyecto.

*Rational Rose e*s una herramienta propietaria de la empresa estadounidense IBM, la cual por causa del bloqueo hacia Cuba limita la compra de una licencia para su uso y además para trabajar con *Rational Rose* se hace imprescindible utilizar máquinas con Sistemas Operativos propietarios como *Windows* o *Macintosh,* por lo que no se decide utilizar esta herramienta.

1.8.2. Visual Paradigm

Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de *software*. Este *software* ayuda a construir aplicaciones con calidad, mejores y a un menor costo. La herramienta UML *CASE* también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos (modelado10, 2011).

Brinda la posibilidad de generar código a partir de los diagramas, para plataformas como .Net, Java y PHP, así como obtener diagramas a partir de código. Esta es precisamente una gran ventaja puesto que el sistema será desarrollado en *java*. El análisis textual es una técnica útil y práctica para la captura de los requisitos del sistema y la identificación de las clases candidatas, *Visual Paradigm* es una de las pocas herramientas *CASE* que soporta el análisis textual.

Tiene disponibilidad en múltiples plataformas y en múltiples versiones. Esta característica es muy importante ya que *Visual Paradigm* puede ser usado en varios sistemas operativos como *Windows, Linux y Mac.*

Por todo lo dicho anteriormente la herramienta a utilizar para el modelado de la aplicación es el *Visual Paradigm* para UML versión 6.4 *Enterprise Edition*; *Suite Update* 3.4, debido fundamentalmente a que es una herramienta multiplataforma, que ayuda a una rápida construcción de aplicaciones de mejor calidad y a un menor costo y que la UCI posee una licencia para su uso. Permite desarrollar todos los tipos de diagramas de clases, soporta la realización de ingeniería tanto directa como inversa, presenta una interfaz de uso intuitiva y con muchas facilidades a la hora de modelar los diagramas, además de su robustez, usabilidad y portabilidad.

1.9. Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado (en lo adelante IDE, por sus siglas en inglés ***I****ntegrated* ***D****evelopment* ***E****nvironment*) es un programa informático compuesto por un conjunto de herramientas de programación. Consiste en un editor de código, un compilador, un depurador y en ocasiones un constructor de interfaz gráfica de usuario. Los IDE pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. Es un editor de código que además puede servir para depurar y facilitar las tareas necesarias en el desarrollo de cualquier tipo de aplicación.

Como el lenguaje que se va a utilizar para el desarrollo de la plataforma es *java* se analizan los IDE Eclipse y *NetBeans*.

1.9.1. Eclipse

Eclipse fue desarrollado originalmente por IBM y en la actualidad por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y lo promueve como una plataforma compartida para herramientas de desarrollo de *software*. Esta herramienta es una de las tantas existentes para desarrollar aplicaciones utilizando el lenguaje *java*. Contiene una arquitectura basada en *plugin* o extensiones lo que permite que sea fácilmente adaptable. Eclipse se distribuye bajo una licencia propia de los proyectos de la Fundación Eclipse, la Licencia Pública de Eclipse (EPL) (González Barahona, 2007). Este entorno de desarrollo incluye una serie de características únicas como la refactorización de código, el código de actualizaciones automáticas e instalaciones y permite a los desarrolladores crear aplicaciones de escritorio para diversos Sistemas Operativos.

Debido a que para obtener una aplicación para móviles en Eclipse es necesario instalar un conjunto de bibliotecas de clases y el mismo no trae emuladores propios, no se decide utilizarlo.

1.9.2. NetBeans

NetBeans IDE es una aplicación de código abierto ("*Open Source*") diseñada para el desarrollo de aplicaciones fácilmente portables entre distintas plataformas, haciendo uso de la tecnología Java. *NetBeans* IDE dispone además de soportes para crear interfaces gráficas de forma visual, desarrollo de aplicaciones web, control de versiones, colaboración entre varias personas, creación de aplicaciones compatibles con teléfonos móviles, resaltado de sintaxis y sus funcionalidades son extendibles mediante la instalación de paquetes (Ramírez, 2011).

Dispone de soportes para crear interfaces gráficas de forma visual sin importar en que sistema operativo se instale, 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.

Se decidió utilizar la versión del *NetBeans* IDE 7.0.1 ya que posee soporte nativo para las tecnologías móviles, además de que da la opción de escoger cualquiera de las dos configuraciones existentes para estos dispositivos, así como sus perfiles y paquetes opcionales.

1.10. Servidor de mapas

Los servidores de mapas permiten a los usuarios interactuar con la información geográfica. Las primeras versiones de los servidores de mapas sólo permitían realizar funciones básicas de visualización y consultas alfanuméricas simples. En las versiones recientes es posible realizar funciones mucho más avanzadas. El servidor de mapas es personalizable, es decir, se pueden preparar o programar las herramientas de manera que sean intuitivas para el usuario no experto en SIG (Cerda, 2010).

1.10.1 GeoServer

GeoServer es un servidor de código abierto (del inglés *Open Source*) certificado por el Consorcio Geoespacial Abierto (en lo adelante OGC, por sus siglas en inglés ***O**pen **G**eospacial **C**onsortium*) en tres estándares diferentes los cuales son:

- ✓ WCS 1.0: Servicio de cobertura en la Web (del inglés ***W**eb **C**overage **S**ervice*).
- ✓ WMS 1.1.1: Servicio de Mapas en la Web (del inglés ***W**eb **M**ap **S**ervice*).
- ✓ WFS 1.0: Servicio de Reportes en la Web (del inglés ***W**eb **F**eature **S**ervice*).

GeoServer puede publicar y editar datos usando estándares abiertos. Además de que la información está disponible en una gran variedad de formatos de mapas.

Entre las ventajas que permite GeoServer se encuentran:

- ✓ No utiliza bloc de notas.
- ✓ Interfaz de fácil interacción, utiliza *MapBuilder* (Un cliente que soporta *JavaScript* OGC WMS y WFS solicitudes, de modo que puede ver y editar datos espaciales a través del navegador web).
- ✓ Compatibilidad con ASP[9] para el desarrollo de *WebSite*.
- ✓ Visualización de la aplicación de GeoServer con *Google Earth* (Pozo, 2002).

Existen varios servidores de mapas, pero en el proyecto Control de Flotas se utiliza el servidor de mapas GeoServer en su versión 2.0.2, por lo que se decide utilizar este como servidor de mapas.

Conclusiones Parciales

Después de analizar las principales tendencias en el desarrollo de SIG para dispositivos móviles y haber realizado un estudio del estado del arte de las principales Metodologías de Desarrollo de *Software*, de los Lenguajes de Programación, los Entornos de Desarrollo Integrado, Versiones de *java*, Herramientas de Ingeniería de *Software* Asistida por Computadora y Servidores de mapas; quedaron definidos para el desarrollo de la Plataforma para Sistemas de Información Geográfica en dispositivos móviles: el uso de la metodología de desarrollo de *software* AUP, el lenguaje de programación *java*, *NetBeans* IDE en su versión 7.0.1 como entorno de desarrollo, J2ME como edición reducida de *Java*, UML v2.0 como lenguaje de modelado, Visual *Paradigm* para UML 6.4 *Enterprise Edition* como herramienta *CASE y* como servidor de mapas GeoServer en su versión 2.0.2 para probar el sistema.

[9] Servidor Activo de Páginas (ASP, por sus siglas en inglés ***A**ctive **S**erver **P**ages*) es una tecnología de *Microsoft* del tipo "lado del servidor" para páginas *web* generadas dinámicamente.

CAPÍTULO 2: Características de la Plataforma para el desarrollo de SIG.

Introducción

El presente capítulo define las principales características que debe cumplir el sistema a desarrollar en cuanto a los requisitos funcionales y no funcionales. Se enumeran los requisitos funcionales y no funcionales del sistema, mediante Diagramas de Casos de Uso se describen las relaciones existentes de los actores que se involucran con el sistema y la descripción textual de casos de uso explica la propuesta de funcionamiento del sistema.

2.1. Levantamiento de requisitos.

Los requisitos del *software* son las características y cualidades que el sistema debe tener. Estos se dividen en dos grupos, los funcionales y los no funcionales. Una vez definidos e identificados los mismos se tiene la visión general de lo que se quiere hacer en el sistema (Jacobson, et al., 2000).

2.1.1. Requisitos funcionales del sistema.

Los requisitos funcionales se definen como las condiciones o capacidades que el sistema debe cumplir. Responden a ¿Qué debe hacer el sistema? A continuación se listan los requisitos funcionales identificados para el desarrollo de la aplicación.

RF1 Cargar mapa desde el móvil.

Permite mostrar la imagen de un mapa previamente guardada en el dispositivo móvil, esta imagen no contiene ningún tipo de información georeferenciada.

RF2 Cargar mapa de servicio WMS.

Permite mostrar un mapa conectándose a un servidor que soporte el servicio WMS, este mapa si contiene información georeferenciada.

RF3 Guardar mapa en el móvil desde WMS.

Permite guardar la imagen de un mapa en un dispositivo móvil desde el servidor de mapas, esta imagen del mapa a guardar no contiene ningún tipo de información georeferenciada.

RF4 Realizar z*um*[10] más a un mapa.

Permite aumentar la escala de un mapa, realizando una petición al servidor de mapas que soporta el servicio WMS.

[10] Zum (del inglés *zoom*): Teleobjetivo especial cuyo avance o retroceso permite acercar o alejar la imagen (RAE, 2001).

RF5 Realizar z*um* menos a un mapa.

Permite disminuir la escala de un mapa, realizando una petición al servidor de mapas que soporta el servicio WMS.

RF6 Realizar paneo hacia la derecha.

Permite desplazarse hacia la derecha en el mapa, visualizando así la región deseada, sin modificar el tamaño del mapa.

RF7 Realizar paneo hacia la izquierda.

Permite desplazarse hacia la izquierda en el mapa, visualizando así la región deseada, sin modificar el tamaño del mapa.

RF8 Realizar paneo hacia arriba.

Permite desplazarse hacia arriba en el mapa, visualizando así la región deseada, sin modificar el tamaño del mapa.

RF9 Realizar paneo hacia abajo.

Permite desplazarse hacia abajo en el mapa, visualizando así la región deseada, sin modificar el tamaño del mapa.

RF10 Habilitar capas del mapa.

Permite habilitar varias capas en el mapa, solo si se está trabajando con el servidor de mapas.

RF11 Deshabilitar capas del mapa.

Permite deshabilitar varias capas en el mapa, solo si se está trabajando con el servidor de mapas.

RF12 Recentrar el mapa.

Permite calcular de acuerdo a la posición del cursor en el mapa la coordenada sobre la que este se encuentra y posiciona dicha coordenada en el centro de la pantalla del móvil.

RF13 Adicionar servidor.

Permite adicionar un servidor de mapas en el sistema.

RF14 Eliminar servidor.

Permite eliminar un servidor de mapas en el sistema.

RF15 Editar servidor.

Permite editar un servidor de mapas en el sistema.

RF16 Listar servidor.

Permite listar un servidor de mapas en el sistema.

RF17 Seleccionar lenguaje.

Permite seleccionar un lenguaje para utilizarlo en el sistema.

RF18 Seleccionar tema.

Permite seleccionar un tema para utilizarlo en la interfaz visual del sistema.

2.1.2. Requisitos no funcionales del sistema.

Los requisitos no funcionales se definen como las cualidades o propiedades que el *software* debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, funcional, rápido o confiable. Normalmente están vinculados a los requisitos funcionales, es decir, una vez que se conozca lo que el sistema debe hacer, podemos determinar cómo ha de comportarse. A continuación se listan los requisitos no funcionales identificados para el desarrollo de la aplicación.

Usabilidad

RnF1. El sistema debe poder ser utilizado por todos los usuarios que tengan en su poder un dispositivo móvil que cuente con los requisitos de *hardware* para soportarlo.

RnF2. El sistema debe soportar la conexión a múltiples servidores que soporten el servicio WMS.

RnF3. El sistema debe soportar tanto dispositivos táctiles como no táctiles.

RnF4. El sistema debe trabajar con imágenes cargadas desde el móvil.

Interfaz

El sistema debe:

RnF5. Contra con un botón que permita regresar al formulario anterior.

RnF6. Contra con un menú el cual permita ir de un formulario a otro.

RnF7. Contar con un botón que permita salir de la aplicación.

RnF8. Soportar múltiples temas (del término en inglés *skin*).

Requisitos de *software*

Para los dispositivos móviles es necesario:

RnF9. Contar con la máquina virtual KVM.

RnF10. Soportar la configuración CLDC 1.0 o superior y MIDP 2.0 o superior.

Para los servidores:

RnF11. Tener instalado un servidor de mapas que soporte el servicio WMS.

Requisitos de *hardware*

Para los dispositivos móviles clientes:

RnF12. La cantidad mínima de espacio en disco debe ser de 1.5 MB para la instalación del sistema.

Para los servidores:

RnF13. Se requiere que los mismos posean una tarjeta de red.

Soporte

RnF14. El equipo de desarrollo de la línea gvMap es el encargado de corregir los errores que puedan surgir en el sistema.

2.2. Casos de Uso del Sistema

Un caso de uso es un fragmento de funcionalidad que le brinda al usuario una información importante para él (Jacobson, et al., 2000), llámese usuario una persona u otro sistema externo a la aplicación. Cuando se unen todos los casos de uso, se conforma el diagrama de casos de uso, que contiene todas las funcionalidades del sistema. Es necesario tener en cuenta que para cada requisito funcional se debe identificar un caso de uso, aunque se pueden agrupar los casos de uso con funcionalidades comunes o separar los casos de uso complejos, y también se pueden aplicar patrones de casos de uso para el mejor entendimiento del diagrama. Un patrón muy usado en todo sistema es el de crear, obtener, actualizar y eliminar algún elemento, al cual se le llama **CRUD** (por sus siglas en ingles ***C**reate, **R**etrieve, **U**pdate and **D**elete*). En este trabajo, para mejorar el entendimiento del diagrama, en vez de situar “CRUD Elemento”, se decidió el uso de la palabra gestionar, para darle el nombre de “Gestionar Elemento”. Los casos de uso identificados en la plataforma son:

- ✓ Cargar mapa desde el móvil.
- ✓ Cargar mapa de servicio WMS.
- ✓ Guardar mapa en el móvil desde WMS.

- ✓ Controlar visualización de capas.
- ✓ Realizar zum.
- ✓ Realizar paneo.
- ✓ Recentrar el mapa.
- ✓ Gestionar servidor.
- ✓ Seleccionar lenguaje.
- ✓ Seleccionar tema.

2.2.1 Descripción de los Actores del Sistema

Un actor no es parte del sistema en desarrollo, es un agente externo que interactúa con el mismo en pos de obtener un resultado esperado (Jacobson, et al., 2000). El sistema cuenta con el actor que se especifica a continuación:

Actor	Descripción
Usuario	Es la persona que interactúa con el sistema.

Tabla 2: *Descripción de actores.*

2.2.2 Diagrama de Casos de Uso del Sistema

A continuación se muestra el diagrama de casos de uso del sistema.

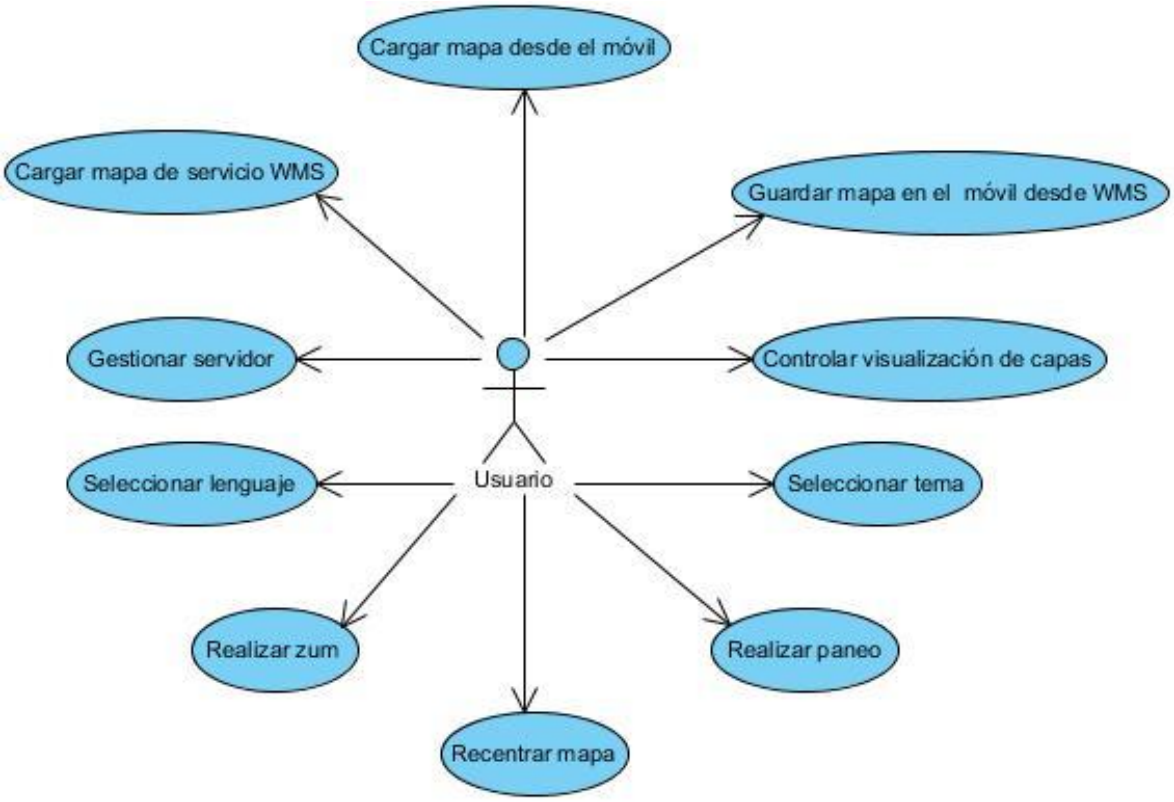

Figura 7: *Diagrama de casos de uso del sistema.*

2.2.3 Descripción de los Casos de Uso del Sistema

La descripción textual de los casos de uso es lo que explica el funcionamiento del sistema completo. A continuación se presentan algunas de las descripciones textuales de los casos de uso críticos identificados en la plataforma.

2.2.3.1. Descripción textual del caso de uso Gestionar Servidor

Objetivo	Este caso de uso permite gestionar un servidor.
Actores	Usuario (Inicia)
Resumen	Este caso de uso se inicia cuando el usuario decide adicionar, eliminar o editar un determinado servidor.
Complejidad	Alta
Prioridad	Alta
Precondiciones	
Postcondiciones	
Flujo de eventos	

Flujo básico: Gestionar Servidor		
	Actor	**Sistema**
1.	Escoge del menú principal la opción de *"Gestionar Servidores".*	
2.		Se listan los servidores existentes y se muestran las siguientes opciones: ✓ Adicionar servidor. ✓ Eliminar servidor. ✓ Editar servidor.
3.	Escoge una de las 3 opciones.	
4.		Si es seleccionada la opción: ✓ Adicionar servidor, ver sección 1: "Adicionar servidor". ✓ Eliminar servidor, ver sección 2: "Eliminar servidor". ✓ Editar servidor, ver sección 3: "Editar servidor".
Sección 1: "Adicionar servidor"		
Flujo básico: "Adicionar servidor"		
	Actor	**Sistema**
1.	Escoge la opción: "Adicionar servidor".	
2.		Muestra un formulario con tres campos de texto para entrar el nombre del servidor, la dirección URL y la versión del servicio WMS.
3.	Entra la los datos y selecciona guardar.	
4.		Se valida que los campos entrados no estén vacios.

5.		Guarda el nuevo servidor.
6.		Termina el caso de uso.
Flujos Alternos		
Nº 4 Datos incorrectos.		
	Actor	**Sistema**
1.		Muestran un mensaje indicando que los datos entrados son incorrectos.
2.		Comienza por el Nº 3 del Flujo básico de eventos.
Sección 2: "Eliminar servidor"		
Flujo básico: "Eliminar servidor"		
	Actor	**Sistema**
1.	Escoge la opción: "Eliminar servidor".	
2.		Muestra un listado con los servidores existentes.
3.	Selecciona el servidor a eliminar.	
4.		Comprueba si se seleccionó un servidor.
5.		Elimina el servidor.
6.		Termina el caso de uso.
Flujos Alternos		
Nº 4 No se seleccionó ningún servidor.		
	Actor	**Sistema**
1.		Muestra un mensaje indicando que debe seleccionar un servidor.
2.		Comienza por el Nº 3 del Flujo básico de eventos.
Sección 3: "Editar servidor"		
Flujo básico: "Editar servidor"		

	Actor	Sistema
1.	Escoge la opción: “Editar servidor“.	
2.		Muestra un listado con los servidores existentes.
3.	Selecciona el servidor a editar.	
4.		Muestra un formulario con tres campos de texto para entrar el nombre del servidor, la dirección URL y la versión servicio WMS a editar.
5.	Entra los nuevos datos y selecciona guardar.	
6.		Se valida que los campos entrados no estén vacios.
7.		Guarda los datos.
8.		Termina el caso de uso.
Flujos Alternos		
Nº 6 Datos incorrectos.		
	Actor	**Sistema**
1.		Muestran un mensaje indicando que los datos entrados son incorrectos.
2.		Comienza por el Nº 5 del Flujo básico de eventos.
Relaciones	**CU Incluidos**	
	CU Extendidos	
Requisitos no funcionales		
Asuntos pendientes		

Tabla 3: *Descripción textual del caso de uso Gestionar Servidor.*

2.2.3.2. Descripción textual del caso de uso Cargar mapa de servicio WMS

Objetivo	Este caso de uso permite cargar un mapa de servicio WMS en un

	dispositivo móvil.	
Actores	Usuario (Inicia)	
Resumen	Este caso de uso se inicia cuando el usuario decide cargar un mapa de servicio WMS en un dispositivo móvil.	
Complejidad	Alta	
Prioridad	Alta	
Precondiciones		
Postcondiciones	Queda mostrado un mapa en el móvil.	
Flujo de eventos		
Flujo básico: Cargar mapa de servicio WMS		
	Actor	**Sistema**
1.	Selecciona la opción *"Cargar mapa de servicio WMS".*	
2.		Comprueba si existe conexión con el servidor del mapas.
3.		Realiza una consulta al servidor, el servidor retorna un *array* de *bytes*.
4.		Los *bytes* se trasforman en una imagen.
5.		Se muestra el mapa.
6.		Termina el caso de uso.
Flujos alternos		
Nº 2 No existe conexión con el servidor de mapas.		
	Actor	**Sistema**
1.		Muestra un mensaje indicando que no existe conexión con el servidor de mapas.
2.		Comienza por el Nº 1 del Flujo básico de eventos.
Relaciones	**CU Incluidos**	
	CU Extendidos	
Requisitos no funcionales		

Asuntos pendientes	

Tabla 4: *Descripción textual del caso de uso Cargar mapa de servicio WMS.*

Conclusiones Parciales

En este capítulo se listaron los requisitos funcionales y no funcionales de la plataforma, los cuales permitieron especificar las funcionalidades y cualidades que esta debe cumplir. De acuerdo a los requisitos funcionales se identificaron los casos de uso, los cuales permitieron documentar y describir las funcionalidades que la plataforma debe tener, por lo que se puede comenzar con la implementación del sistema.

CAPÍTULO 3: Construcción y prueba de la solución propuesta

Introducción

En este capítulo se hace referencia a las bibliotecas de clases que se utilizan para la construcción de la plataforma, a los patrones arquitectónicos y de diseño, se implementan las funcionalidades, se describen las pruebas funcionales y se mencionan las relevancias sociales, económicas y técnicas.

3.1 Librerías de clases utilizadas

Para el desarrollo de la plataforma para SIG se utilizaron varias librerías de clases de *java* las cuales son: kXML para parsear el XML que devuelven las peticiones que se le realizan al servidor de mapas, TinyLineGZIP se utiliza para realizar peticiones mediante el formato de compresión gzip, el cual permite ahorrar ancho de banda y LWUIT para desarrollar la interfaz gráfica de la aplicación, ya que permite que la plataforma se vea de igual forma en la mayoría de los móviles, simplifica el trabajo con el sistema de archivos, entre otras funcionalidades.

3.2. Arquitectura de la solución propuesta

A continuación se define la arquitectura de la plataforma y se presentan los patrones arquitectónicos y de diseño que se utilizan para el desarrollo de la misma.

3.2.1. Patrones Arquitectónicos

Para comprender mejor qué es un patrón arquitectónico, hay que tener en cuenta que (Pressman, 2001) plantea que:

Cada patrón describe una categoría del sistema que contiene:

- ✓ Un conjunto de componentes (por ejemplo, una base de datos, módulos computacionales) que realizan una función requerida por el sistema.
- ✓ Un conjunto de conectores que posibilitan la comunicación, la coordinación y la cooperación entre los componentes.
- ✓ Restricciones que definen como se pueden integrar los componentes que forman el sistema.
- ✓ Modelos semánticos que permiten al diseñador entender las propiedades globales de un sistema para analizar las propiedades conocidas de sus partes constituyentes.

A continuación se describen los patrones arquitectónicos utilizados para el desarrollo de la plataforma informática, los cuales son:

3.2.1.1. Cliente-Servidor

La arquitectura consiste básicamente en que un programa cliente realiza peticiones a otro programa servidor que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras. En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debido a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

IBM define al modelo Cliente-Servidor como: *"La tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo y/o, a través de la organización, en múltiples plataformas. El modelo soporta un medio ambiente distribuido en el cual los requerimientos de servicio hechos por estaciones de trabajo inteligentes o "clientes", resultan en un trabajo realizado por otros computadores llamados servidores"* (INEI, 2012).

3.2.1.2. Modelo – Vista – Controlador

Actualmente las aplicaciones de *software* permiten un alto nivel de interacción con el usuario, donde la meta principal es mantener la lógica del negocio de la aplicación completamente independiente de la interfaz de usuario, por lo que el patrón Modelo – Vista – Controlador (en lo adelante MVC, por sus siglas en inglés ***M****odel* – ***V****iew* – ***C****ontroller*) divide una aplicación interactiva en tres componentes:

- ✓ El Modelo que contiene las funcionalidades principales y los datos.
- ✓ La Vista que muestra la información al usuario.
- ✓ El Controlador que maneja la interacción con el usuario y la entrada de datos.

La Vista y el Controlador componen la interfaz de usuario.

Cuando el usuario realiza una acción, ésta es recibida por el Controlador, el cual decide cuál Modelo utilizar, qué datos necesita del Modelo y también decide qué Vista utilizar para mostrar los resultados (Buschmann, et al., 1996).

3.2.1.3. Document – View

Document – View (en Español Documento – Vista) es una variante del patrón MVC, donde su principal diferencia radica en que no existe una separación entre el controlador y la vista, sino que se implementa como un solo componente. De forma general el componente *Document* corresponde al

modelo en MVC y el componente *View* combina las responsabilidades del control y la vista en MVC e implementa la interfaz de usuario del sistema (Buschmann, et al., 1996).

Para el desarrollo de la plataforma se aplicará la arquitectura Cliente – Servidor donde en el Servidor se encontrará un servidor WMS y el Cliente será la plataforma la cual estará contenida en un dispositivo móvil y estará construida utilizando la arquitectura *Document – View,* la cual es utilizada para no tener que generar nuevas clases que manejen las acciones de los comandos, lográndose con esto disminuir el tamaño de la plataforma y la complejidad. A continuación se explican otros patrones utilizados para la construcción de la plataforma.

3.2.1.4. Wrapper Facade

Wrapper Facade (en Español Fachada o Envoltorio) encapsula funciones que están en API que no son orientadas a objetos y a estas se les hace una fachada la cual es más robusta, portable, flexible y orientada a objetos (Buschmann, et al., 1996). Se utiliza este patrón para determinar algunas funcionalidades que soporta el móvil, como por ejemplo si soporta el sistema de archivos o no.

3.2.1.5. Half sync / Half async

Half-Sync / Half-Async (en Español Mitad Sincrónico / Mitad Asincrónico) separa los procesamientos sincrónicos y asincrónicos del sistema, para simplificar la programación y mejorar el rendimiento de la aplicación. Este patrón tiene dos capas que se comunican entre sí, una es para el procesamiento sincrónico y otra es para el procesamiento asincrónico (Buschmann, et al., 1996). *Half-Sync / Half-Async* se utiliza en la aplicación para manejar la cola de peticiones al servidor de mapas.

3.2.2. Patrones de Diseño

Un patrón de diseño es una solución bien conocida a un problema de diseño recurrente en un contexto determinado. Un patrón de diseño no es un diseño final que puede ser transformado directamente en el código, es una descripción o una plantilla para la forma de resolver un problema que se puede usar en diferentes situaciones. Los patrones de diseño pueden acelerar el proceso de desarrollo, la reutilización de estos permite evitar grandes complicaciones a la hora de programar y mejora la legibilidad del código (Gamma, et al., 1997).

Debido a que los patrones de diseño son ampliamente difundidos en el mundo y que los creadores de estos son de habla inglesa, se presenta en la investigación los nombres de los patrones en inglés ya que traducidos al español traerían consigo desconocimiento o confusión.

Los patrones de diseño utilizados para la creación de la plataforma son los siguientes:

3.2.2.1. Singleton

Singleton garantiza que una clase sólo tiene una única instancia, proporcionando un punto de acceso global a la misma (Gamma, et al., 1997). Este patrón es utilizado en varias clases sobre todo en las que se administran determinados recursos.

3.2.2.2. Abstract Factory

Abstract Factory proporciona una interfaz para la creación de familias de objetos interdependientes o interrelacionados, sin especificar sus clases concretas (Gamma, et al., 1997). Es utilizado para la construcción de las peticiones al servicio WMS.

3.2.2.3. Composite

Composite permite a los clientes tratar uniformemente a los objetos simples y compuestos de una estructura jerárquica recursiva (Gamma, et al., 1997). Este patrón es utilizado por la biblioteca de clases LWUIT, por tanto, todos los componentes visuales desarrollados en la plataforma hacen uso del mismo.

3.2.2.4. Facade

Facade proporciona una interfaz unificada de alto nivel que, representando a todo un subsistema, facilita su uso. *Facade* satisface a la mayoría de los clientes, sin ocultar las funciones de menor nivel a aquellos que necesiten acceder a ellas (Gamma, et al., 1997). Se utiliza dicho patrón para simplificar el acceso al servicio WMS, ya que se proporciona una fachada al subsistema WMS, para realizar las peticiones y analizar las respuestas del servidor de mapas.

3.2.2.5. Flyweight

Flyweight permite mejorar la eficiencia a la hora de mantener una gran cantidad de objetos “de grano fino”, usando la idea de la compartición. Un *flyweight* es un objeto compartido que puede ser usado en diferentes contextos simultáneamente, de forma transparente, indistinguible. El concepto clave es la distinción entre el estado intrínseco (información independiente del contexto del objeto, almacenada en el mismo y que favorece su compartición) y el estado extrínseco (información que depende y varía con el contexto donde se use el objeto, no puede ser compartida y se la proporciona el cliente que lo usa, a través de la fábrica) del objeto compartido (Gamma, et al., 1997). *Flyweight* se utiliza en la aplicación para guardar las instancias de los comandos utilizados, ya que en casi todos los formularios van a existir los mismos comandos, por lo que se evita que cada formulario tenga que crear los comandos una y otra vez.

3.2.2.6. Observer

Observer es útil para informar eventos o notificaciones de algún tipo entre clases relacionadas (Gamma, et al., 1997). Este patrón de diseño es utilizado para la gestión de eventos de la plataforma y por la biblioteca de clases LWUIT.

3.2.2.7. Template Method

Template Method define el esqueleto de un algoritmo para una operación, dejando para sus subclases la capacidad de redefinir el funcionamiento de los pasos de este algoritmo, siempre y cuando la estructura del mismo permanezca intacta (Gamma, et al., 1997). Este patrón es utilizado para la construcción de las peticiones al servicio WMS y para inicializar los formularios.

3.2.2.8. Builder

Builder tiene como propósito principal separar la construcción y la representación de un objeto complejo, para así permitir que el mismo proceso de construcción sirva para crear diferentes representaciones (Gamma, et al., 1997). *Builder* se utiliza para la construcción de la interfaz gráfica.

3.2.2.9. Iterator

Iterator permite proporcionar una forma de acceder secuencialmente a los elementos de un objeto compuesto por agregación sin necesidad de desvelar su representación interna (Gamma, et al., 1997). Es utilizado para el manejo de las colecciones de objetos de la plataforma.

3.3. Modelo del diseño

El modelo de diseño según plantea (Jacobson, et al., 2000):

- ✓ Se adapta al entorno de implementación elegido y a la reutilización de sistemas heredados u otros marcos de trabajo desarrollados para el proyecto, por lo que el modelo de diseño funciona como esquema para la implementación.
- ✓ Este modelo define clasificadores (clases, subsistemas e interfaces), relaciones entre esos clasificadores, y colaboraciones que llevan a cabo los casos de uso (las realizaciones de casos de uso).

3.3.1. Diagrama de Subsistemas de Diseño

A continuación se presenta el diagrama de subsistemas de diseño en el cual se encuentran contenidos los principales subsistemas desarrollados en la plataforma.

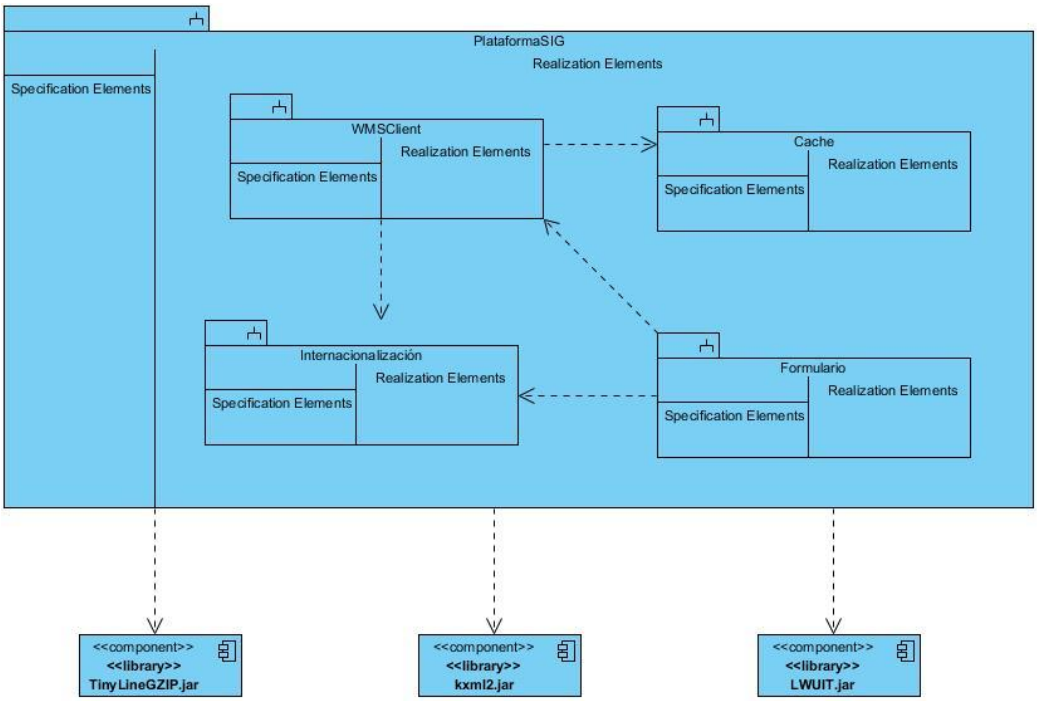

Figura 8: *Diagrama de Subsistemas de Diseño.*

3.3.2. Diagrama de Clases del Diseño

Los diagramas de clases del diseño son la base para entender cómo funciona el sistema. Para el diseño de estos diagramas de clases se tiene en cuenta tanto los requisitos funcionales como no funcionales que el sistema debe tener. A continuación se muestran los diagramas de clases del diseño correspondientes a los casos de uso Gestionar Servidor y Cargar mapa de servicio WMS.

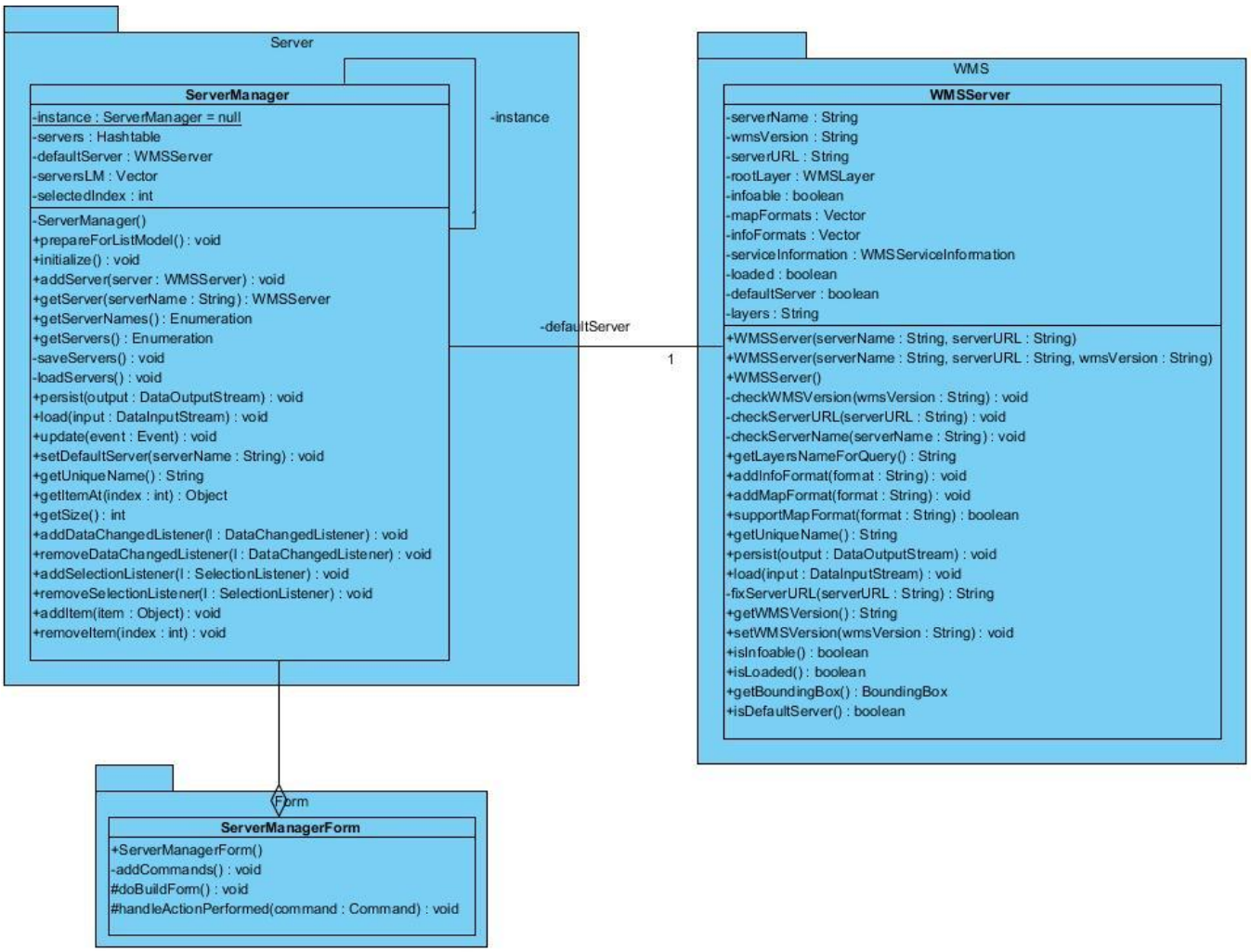

Figura 9: *Diagrama de Clases de Diseño del casos de uso Gestionar Servidor.*

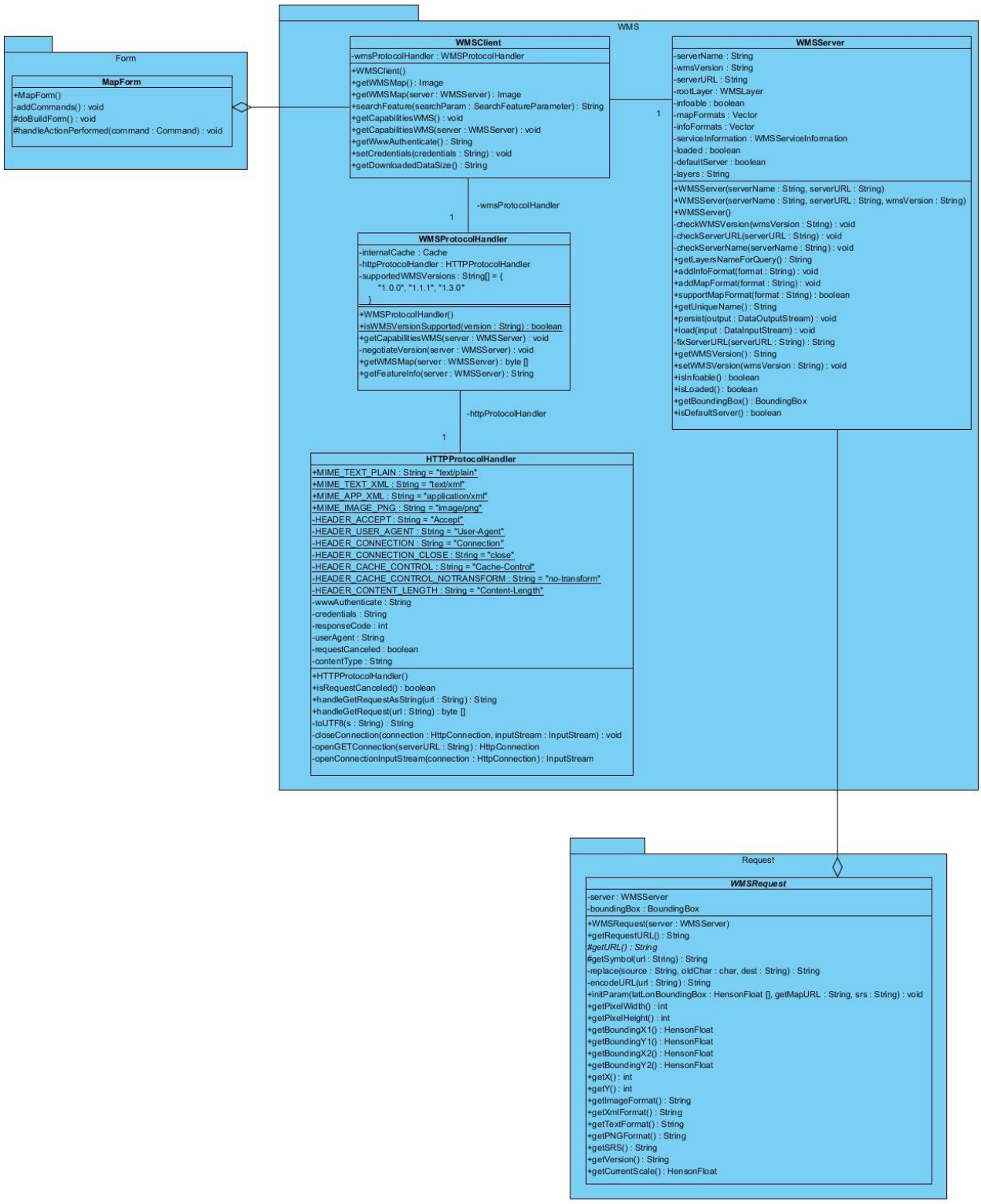

Figura 10: *Diagrama de Clases de Diseño del caso de uso Cargar mapa de servicio WMS.*

3.4. Implementación

La implementación es la parte del proceso de desarrollo de *software* que se ocupa de la creación del sistema en términos de componentes, es decir ficheros de código fuente, *scripts*, ejecutables y similares (Jacobson, et al., 2000)

Para la implementación de la plataforma se tuvieron en cuenta los patrones arquitectónicos y de diseño definidos anteriormente logrando una arquitectura flexible, configurable y robusta.

3.4.1. Diagrama de Componentes

Los diagramas de componentes describen los elementos físicos del sistema ya sean ejecutables, librerías, clases y las relaciones entre ellos. A continuación se presenta el diagrama de componentes general de la Plataforma SIG para dispositivos móviles:

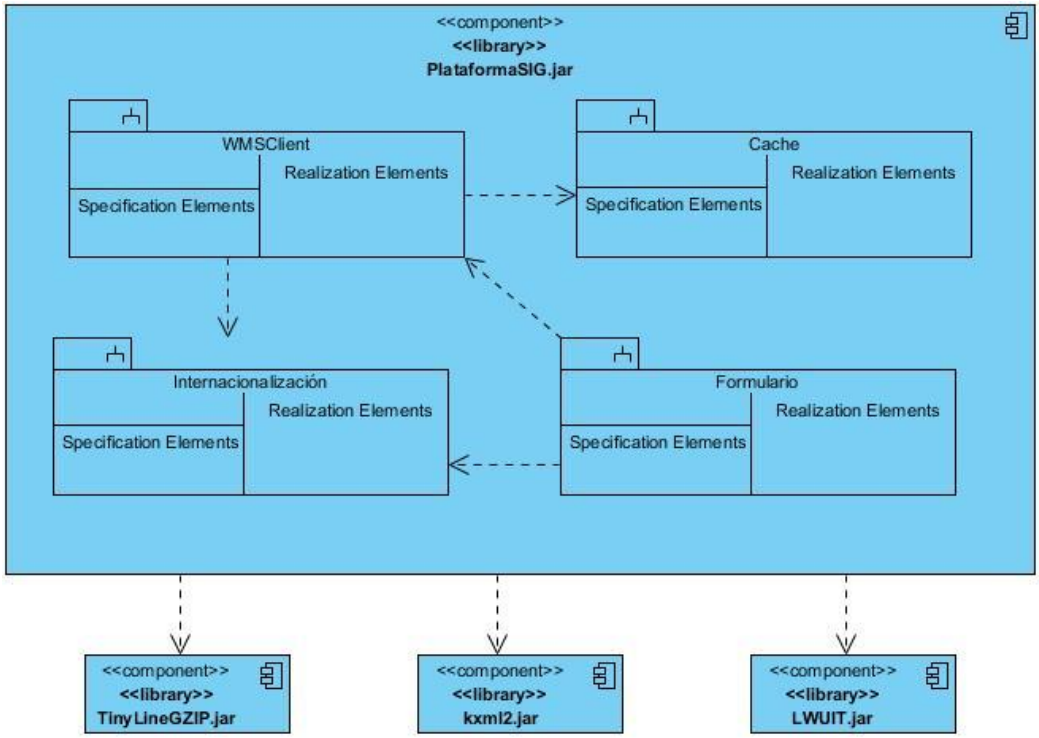

Figura 11: *Diagrama de Componentes.*

En la *Figura 11: Diagrama de Componentes.* se expone una vista general de los subsistemas con los que cuenta la plataforma, sus relaciones y las bibliotecas de clases utilizadas en el desarrollo de esta.

3.4.2. Modelo de Despliegue

El modelo de despliegue es un modelo conformado por objetos, donde se describe la distribución física que tiene el sistema y la comunicación que existe entre estos mediante protocolos. A continuación se muestra el modelado del diagrama de despliegue para la aplicación a desarrollar.

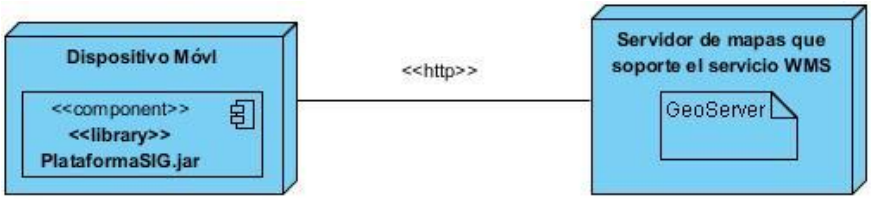

Figura 12: *Diagrama de despliegue.*

En la *Figura 12: Diagrama de despliegue****.*** se muestra la relación físicas entre el dispositivo móvil y el servidor de mapas mediante el protocolo http, donde en el móvil estará contenida la PlataformaSIG.jar y en el servidor estará un servidor de mapas WMS (el servidor de mapas utilizado para probar el sistema fue GeoServer, aunque la aplicación estará desarrollada para que funcione en cualquier servidor WMS).

3.5. Pruebas

Con las pruebas se verifica el resultado de la implementación, probando cada componente por separado y luego en su conjunto como un sistema listo a entregar (Jacobson, et al., 2000).

Existen dos tipos de métodos de prueba los cuales son:

3.5.1. Pruebas de caja blanca

Plantea (Pressman, 2001) que mediante los métodos de prueba de caja blanca, el ingeniero de *software* puede obtener casos de prueba que:

- ✓ Garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo.
- ✓ Ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa.
- ✓ Ejecuten todos los bucles en sus límites y con sus límites operacionales.
- ✓ Ejerciten las estructuras internas de datos para asegurar su validez.

Por lo que las pruebas de caja blanca están dirigidas principalmente a las funciones internas del sistema. Se realizan probando los caminos lógicos del sistema y examinando el estado del programa en varios puntos para determinar si el estado real coincide con el esperado. Como las pruebas de caja

blanca están hechas para probar cada uno de los caminos que puede tomar el código fuente, en la plataforma desarrollada no resulta práctico el uso de estas pruebas debido a la presión del tiempo.

3.5.2. Pruebas de caja negra

Plantea (Pressman, 2001) que la prueba de caja negra intenta encontrar errores de las siguientes categorías:

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

Por lo que las pruebas de caja negra se desarrollan sobre la interfaz del *software*. Estas pruebas se encargan de verificar que las funciones que debe desempeñar el sistema son operativas. Se centran en los requisitos funcionales del s*oftware*, sin interesarse por el funcionamiento interno del mismo.

Para el desarrollo de las pruebas de la plataforma informática se utiliza el método de pruebas de caja negra y se eligieron las técnicas de **Partición Equivalente** y **Análisis de valores límite** que se explican a continuación.

Partición Equivalente: es una técnica que divide el campo de entrada de un programa en clases de datos, de los que se pueden derivar casos de prueba. Se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar (Pressman, 2001).

Análisis de valores límite: es una técnica que completa a la Partición Equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el Árbol Binario de Búsqueda (en lo adelante AVL, por las iniciales de sus inventores *Adelson-Velskii y Landis*) lleva a la elección de casos de prueba en los extremos de la clase. En lugar de centrarse solamente en las condiciones de entrada, el AVL obtiene casos de prueba también para el campo de salida (Pressman, 2001).

3.5.3. Pruebas aplicadas a los componentes desarrollados

Un caso de prueba se diseña según las funcionalidades descritas en los casos de usos. Este diseño se elabora previamente a realizar las pruebas exploratorias, o de interfaz como también pueden llamarse, se parte de la descripción de los casos de usos del sistema, como apoyo para las revisiones. Cada

planilla de caso de prueba recoge la especificación de un caso de uso, dividido en secciones y escenarios, detallando las funcionalidades descritas en él y describiendo cada variable que recoge el caso de uso en cuestión, además quedan plasmadas las revisiones realizadas al caso de prueba; así como un registro de todo aquello que no satisface y corresponde a la calidad del *software*.

Existen casos de pruebas para los diferentes métodos: Caja Negra y Caja Blanca. El proceso de prueba que se utiliza en la aplicación hace uso del método Caja Negra y las técnicas de Partición Equivalente y Análisis de valores límite. Para detallar el caso de uso a probar se utiliza una tabla, donde se desglosa esta funcionalidad en secciones y estas a su vez en escenarios, para hacer más fructífera la ejecución de las pruebas. Esta tabla contiene los campos:

- ✓ **Nombre de la sección:** Se especifica el nombre de la sección [SC 1: Nombre de la sección].
- ✓ **Escenarios de la sección:** Se especifican los escenarios de cada sección [EC 1.1: Nombre del Escenario].
- ✓ **Descripción de la funcionalidad:** Se describe brevemente la funcionalidad del escenario.

A partir de la descripción realizada se detallan las variables que se encuentran en toda la interfaz asociadas al caso de uso que se le realiza el diseño de caso de prueba. Esta descripción está compuesta por campos tales como:

- ✓ **No:** Se enumera todos los campos o variable, descrito en el caso de uso.
- ✓ **Nombre de campo:** Se especifica el nombre del campo de entrada.
- ✓ **Clasificación:** Se especifica la clasificación según el componente de diseño utilizado [ejemplo: campo de texto, lista desplegable o lista de selección, entre otros].
- ✓ **Puede ser nulo:** Se especifica si el campo puede ser nulo o no, para ello solo se puso Sí o No.
- ✓ **Descripción:** Se describen brevemente los datos que debían introducirse.

La descripción anterior posibilita que se ejecute una matriz de datos, donde se evalúa y se prueba la validez de cada uno de los datos introducidos en el sistema, específicamente en la sección que se estará probando. Utilizando un juego de datos válidos e inválidos se identificará el empleo de la técnica de Partición de Equivalencia y Análisis de Valores Límites.

Esta matriz de datos contiene los siguientes aspectos:

- ✓ **Escenario:** Se especifica el nombre del escenario.

- ✓ Variables [1, 2,…, n]: Se especifica el nombre de la variable, o el número según la tabla de descripción de variables y en su celda correspondiente se indica el valor del dato [V (Válido), I (Inválido), N/A (No Aplica)].
- ✓ **Respuesta del sistema:** Se escribe el resultado que se esperaba al realizar la prueba.
- ✓ **Resultado de la prueba:** Se escribe el resultado que se obtuvo al realizar la prueba. (Satisfactorio o No Satisfactorio).
- ✓ **Flujo central:** Se expone el flujo central del caso de uso al que se le estaba diseñando el caso de prueba.

A continuación se presentan las pruebas realizadas a dos de los casos de uso desarrollados.

3.5.3.1. Descripción del caso de prueba Gestionar servidor

Nombre del CU: Gestionar Servidor.

Descripción General: Este caso de uso se inicia cuando el usuario decide adicionar, eliminar o editar un determinado servidor.

Secciones a probar en el Caso de Uso:

- ✓ Adicionar servidor.
- ✓ Editar servidor.
- ✓ Eliminar Servidor.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central

SC1: Adicionar servidor.	EC1.1: Servidor adicionado con éxito.	Una vez que el usuario selecciona la opción Gestionar servidores y selecciona “Adicionar Servidor”, el sistema muestra un formulario para entrar el nombre del servidor, la dirección URL y la versión WMS. Luego el usuario llena los datos correspondientes y pulsa en “Guardar servidor”. El sistema realiza las verificaciones pertinentes, y de esta forma queda insertado el nuevo servidor.	✓ Clic en el botón “Menú”. ✓ Clic en la opción “Gestionar servidores“. ✓ Clic en la opción “Adicionar servidor”. ✓ Llenar los campos. ✓ Clic en el botón “Guardar servidor”.
	EC1.2: Nuevo servidor adicionado con campos vacíos.	Si el usuario no llena los campos correspondientes, el sistema muestra un mensaje de error indicando que existen campos vacíos.	✓ Clic en el botón “Menú”. ✓ Clic en la opción “Gestionar servidores“. ✓ Clic en la opción “Adicionar servidor”. ✓ Dejar los campos vacios. ✓ Clic en el botón “Guardar servidor”.
SC2: Editar servidor.	EC2.1: Editar servidor con éxito.	Una vez que el usuario selecciona la opción “Gestionar servidores” y selecciona “Editar Servidor”, el sistema muestra un formulario para entrar el nombre del servidor, la dirección URL y la versión WMS. Luego el usuario llena los datos correspondientes y pulsa en “Guardar servidor”. El sistema realiza las verificaciones	✓ Clic en el botón “Menú”. ✓ Clic en la opción “Gestionar servidores“. ✓ Clic en la opción “Editar servidor”. ✓ Llenar los campos. ✓ Clic en el botón “Guardar servidor”.

		pertinentes, y de esta forma quedan cambiados los datos del servidor.	
	EC2.2: Servidor con campos vacíos.	Si el usuario no llena los campos correspondientes, el sistema muestra un mensaje de error diciendo que existen campos vacíos.	✓ Clic en el botón "Menú". ✓ Clic en la opción "Gestionar servidores". ✓ Clic en la opción "Editar servidor". ✓ Dejar los campos vacios. ✓ Clic en el botón "Guardar servidor".
SC 3: Eliminar Servidor.	EC 3.1: Eliminar Servidor con éxito.	Una vez que el usuario selecciona la opción "Gestionar servidores" y selecciona "Eliminar Servidor", el sistema muestra un listado de servidores, luego el usuario selecciona el servidor a eliminar seleccionando la opción "Eliminar servidor". El sistema elimina el servidor.	✓ Clic en el botón "Menú". ✓ Clic en la opción "Gestionar servidores". ✓ Clic en la opción "Eliminar servidor". ✓ Seleccionar un servidor. ✓ Clic en el botón "Guardar servidor".
	EC 3.1: Eliminar servidor sin seleccionar ninguno.	El usuario no selecciona ningún servidor para eliminar, el sistema muestra un mensaje de error señalándole al usuario que seleccione un servidor.	✓ Clic en el botón "Menú". ✓ Clic en la opción "Gestionar servidores". ✓ Clic en la opción "Eliminar servidor". ✓ No seleccionar ningún servidor. ✓ Clic en el botón

			“Guardar servidor”.

Tabla 5: *Diseño de Caso de Prueba de Caja Negra del Caso de Uso: Gestionar servidor.*

Descripción de Variables:

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1.	Nombre	Campo de texto.	no	Campo para la entrada del nombre.
2.	URL	Campo de texto.	no	Campo para la entrada de la dirección del servidor.
3.	Versión WMS	Campo de texto.	si	Campo para la entrada de la versión WMS.
4.	Servidores	Lista de selección.	no	Para marcar un servidor.

Tabla 6: *Descripción de Variables del Caso de Uso: Gestionar servidor.*

Matriz de Datos

SC1 Adicionar servidor

ID del escenario	Escenario	Variable 1 Nombre	Variable 2 URL	Variable 3 Lista de servidores	Respuesta del Sistema	Resultado de la Prueba
EC1.1	Servidor adicionado con éxito.	V/Webmap	V/http://10.54.10.13:5901/geoserver/	NA	Se adiciona nuevo servidor.	Satisfactorio
EC1.2	Nuevo servidor adicionado con campos vacíos.	I/ “vacío”	I/ “vacío”	NA	Muestra el mensaje “Existen campos sin llenar”.	Satisfactorio

Tabla 7: *Matriz de Datos. Adicionar servidor.*

SC2 Editar servidor

ID del escenario	Escenario	Variable 1 Nombre	Variable 2 URL	Variable 3 Lista de servidores	Respuesta del Sistema	Resultado de la Prueba
EC2.1	Editar servidor con éxito.	V/Webmap 1	V/http://10.54.10.14:5901/geoserver/	NA	Se cambian los valores del servidor.	Satisfactorio
EC2.2	Servidor con campos vacíos.	I/ “vacío”	I/ “vacío”	NA	Muestra el mensaje “Existen campos sin llenar”.	Satisfactorio

Tabla 8: *Matriz de Datos. Editar servidor*

SC3 Eliminar servidor

ID del escenario	Escenario	Variable 1 Nombre	Variable 2 URL	Variable 3 Lista de servidores	Respuesta del Sistema	Resultado de la Prueba
EC 3.1	Eliminar Servidor con éxito.	NA	NA	V/Se marca un servidor	Se elimina el servidor.	Satisfactorio
EC 3.2	Eliminar servidor sin seleccionar ninguno.	NA	NA	I/ “vacío”	El sistema muestra un mensaje de error señalándole al usuario que seleccione un servidor.	Satisfactorio

Tabla 9: *Matriz de Datos. Eliminar servidor.*

3.5.3.2. Caso de Prueba Cargar mapa de servicio WMS

Nombre del CU: Cargar mapa de servicio WMS.

Descripción General: Este caso de uso permite cargar un mapa de servicio WMS en un dispositivo móvil.

Secciones a probar en el Caso de Uso:

- ✓ Cargar mapa desde Servicio.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
SC1 Cargar mapa desde servicio WMS.	EC1.1: Mapa cargado con éxito.	Permite cargar un mapa desde servicio WMS.	✓ Clic en el botón “Menú”. ✓ Clic en la opción “Cargar mapa desde servicio WMS “
	EC 1.2: No hay conexión con el servidor.	Se intenta cargar un mapa pero no hay conexión con el servidor.	✓ Clic en el botón “Menú”. ✓ Clic en la opción “Cargar mapa desde servicio WMS “

Tabla 10: *Diseño de Caso de Prueba de Caja Negra del Caso de Uso: Cargar mapa de servicio WMS.*

Descripción de Variables: No existen variables para este Caso de Prueba.

Matriz de Datos:

SC1 Cargar mapa desde Servicio WMS.

ID del escenario	Escenario	Respuesta del Sistema	Resultado de la Prueba
EC1.1	Mapa cargado con éxito.	Se carga el mapa.	Satisfactorio
EC1.2	No hay conexión con servidor.	Muestra mensaje de error señalando que no hay conexión con el servidor.	Satisfactorio

Tabla 11: *Matriz de Datos. Cargar mapa desde servicio WMS.*

3.5.4 Análisis de las pruebas realizadas

En la *Tabla 12: Resultado de las pruebas realizadas en la primera iteración.* se muestra un resumen de la cantidad y el nivel de importancia de los errores detectados en esa iteración.

Nivel de importancia	Defectos	Justificación
Alto	5	Funcionalidades críticas.
Medio	8	Procesos de validación.
Bajo	2	Errores Ortográficos.
Total	15	

Tabla 12: *Resultado de las pruebas realizadas en la primera iteración.*

En la segunda iteración se encontraron errores como se muestra en la *Tabla 13: Resultado de las pruebas realizadas en la segunda iteración.*, pero los mismos fueron corregidos.

Nivel de importancia	Defectos	Justificación
Alto	1	Funcionalidades críticas.
Medio	2	Procesos de validación.
Bajo	0	Errores Ortográficos.
Total	3	

Tabla 13: *Resultado de las pruebas realizadas en la segunda iteración.*

Las pruebas realizadas en la tercera iteración no detectaron ningún error, por lo que de forma general se verificó el correcto funcionamiento de la plataforma. Aunque en las pruebas aplicadas en la tercera iteración no se hayan encontrado más errores, no significa que la plataforma esté libre de estos, pero los autores realizaron un gran esfuerzo por detectar la mayor cantidad de errores durante el proceso de pruebas.

3.6. Relevancia de la plataforma informática

Bajo un enfoque de Ciencia-Tecnología-Sociedad (CTS) definidos por el arquitecto, el analista y la líder del proyecto Control de Flotas, en conjunto con los autores del trabajo, se identificaron las principales relevancias sociales, económicas y técnicas que puede tener la plataforma en su utilización.

Relevancias sociales:

- ✓ Disminución del tiempo de desarrollo en los proyectos de gvMap, debido a que los desarrolladores ya no tienen que programar las funcionalidades básicas de un SIG, ni enfrentarse a las complejidades que lleva realizarlas, sino que solamente deben centrarse en lo que desee el cliente.
- ✓ Permite generar clímax de satisfacción en los desarrolladores de la línea gvMap que utilizan el sistema, ya que anteriormente no existía una plataforma de SIG para móviles destinada a dispositivos de bajas prestaciones, que respondiera a las necesidades de esta línea.

Relevancias económicas:

- ✓ Permite disminuir los costos en el desarrollo de personalizaciones, ya que los desarrolladores solo se centran en las funcionalidades específicas que solicite cada cliente.
- ✓ El uso de tecnologías libres permitió disminuir los costos, debido a que no fue necesario comprar ninguna licencia de *software*.

Relevancias técnicas:

- ✓ La plataforma está basada en el desarrollo de tecnologías libres, por lo que la comunidad científica puede utilizarla para construir, integrar, extender, modernizar e implantar nuevos sistemas basados en la misma.
- ✓ En la línea gvMap, anteriormente no existía una plataforma que permitiera el desarrollo de aplicaciones de SIG para móviles de bajas prestaciones y la plataforma desarrollada es la primera de su tipo en la línea.

Conclusiones Parciales

En este capítulo se mencionaron las bibliotecas de clases utilizadas, las cuales permitieron resolver parcialmente algunas funcionalidades de la plataforma. Se describieron los patrones arquitectónicos y de diseño, los que permitieron crear una arquitectura flexible, robusta y configurable. Además se implementó el sistema y se le realizaron las pruebas, las cuales verificaron su correcto funcionamiento.

Después de concluida la plataforma se analizó la relevancia de la misma para la línea gvMap. El sistema entra en su capacidad operacional inicial.

CONCLUSIONES

Una vez concluida la investigación, se ha determinado que el objetivo planteado se cumplió satisfactoriamente y se arribó a las siguientes conclusiones:

- ✓ La aplicación posee una arquitectura configurable, robusta y flexible, lo cual permite manejar determinados datos y satisfacer las necesidades de los clientes.
- ✓ El sistema fue desarrollado utilizando tecnologías libres, logrando disminuir de esta forma en costos y ganar en soberanía tecnológica.
- ✓ La plataforma permite ser utilizada en la mayoría de los Sistemas Operativos para móviles que contengan la Máquina Virtual K (KVM).
- ✓ Sobre la plataforma se pueden realizar diferentes personalizaciones, las cuales se pueden conectar a cualquier servidor de mapas que soporte el servicio WMS y soporta además el trabajo sin conexión cargando imágenes desde el móvil.
- ✓ Al desarrollar personalizaciones utilizando la plataforma se logrará ahorrar tiempo en la línea gvMap, facilitar el trabajo del equipo de desarrollo y disminuir los costos, resolviendo los problemas existentes en la línea de una forma rápida y eficiente, al poder reutilizar el código base.
- ✓ La plataforma desarrollada en la línea gvMap permite abrir un nuevo horizonte en el desarrollo de SIG para dispositivos móviles, sobre aplicaciones nativas de *java*, utilizando la configuración CLDC, ya que es la primera de su tipo creada en esta línea.

RECOMENDACIONES

Una vez vencido el objetivo de la investigación, y teniendo en cuenta las experiencias obtenidas, los autores recomiendan que el sistema:

- ✓ Se pruebe en dispositivos móviles reales, no en emuladores.
- ✓ Posibilite la gestión de rutas.
- ✓ Trabaje con diferentes medios de comunicación inalámbrica.
- ✓ Permita el trabajo con servicios basados en localización.
- ✓ Muestre el mapa utilizando teselas.
- ✓ Trabaje con WMS-*Cache*.
- ✓ Al estar trabajando sin conexión el sistema debe permitir que la imagen que se encuentra contenida dentro del móvil contenga información asociada a la misma.
- ✓ Permita la utilización de extensiones.
- ✓ Debe ser probado con otros servidores de mapas que soporten el servicio WMS, aunque el sistema se implementó para que el mismo se conecte a cualquier servidor de mapas, solo ha sido probado utilizando el servidor GeoServer.

BIBLIOGRAFÍA

Adempiere. 2011. ...Implementación Adempiere en Ubuntu. [En línea] Septiembre de 2011. [Citado el: 15 de 2012 de Enero.] http://ubuntu-adempiere.blogspot.com/2011/09/metodologia-aup-agile-unified-process.html.

Adobe Systems Incorporated. 2012. Adobe. [En línea] 2012. [Citado el: 20 de marzo de 2012.] http://www.adobe.com/svg/.

Answers. 2012. Answers. [En línea] 2012. [Citado el: 30 de Mayo de 2012.] http://www.answers.com/topic/platform.

babylon. 1997. babylon. *babylon.* [En línea] 1997. [Citado el: 13 de Marzo de 2012.] http://diccionario.babylon.com/bytecode/.

Batista Silva, Jose Luis. 2005. APLICACIÓN DE SISTEMAS DE INFORMACIÓN GEOGRÁFICA EN CUBA. [En línea] 2005. [Citado el: 27 de Noviembre de 2011.] http://www.mappinginteractivo.com/plantilla-ante.asp?id_articulo=1051.

Beck, Kent. 1999. *Extreme Programming Explained.* 1999. [Publicado por primera vez en Septiembre 29, 1999]. 0201616416.

Brisaboa, Nieves R., y otros. 2009. Laboratorio de Bases de Datos Universidad de Coruña. [En línea] 2009. [Citado el: 03 de febrero de 2012.] lbd.udc.es/Repository/Publications/Drafts/Defeimpde.pdfublications%2FDrafts%2FDefeimpde.pdf&ei=NeGFT8GmA4bDgAfW49hr&usg=.

Buschmann, Frank, y otros. 1996. *Pattern-Oriented Software Arquitecture.* West Sussex : John Wiley & Sons Ltd, 1996. 0-471-95889-7.

—. 1996. *Pattern-Oriented Software Arquitecture.* West Sussex : John Wiley & Sons Ltd, 1996. 0-471-95889-7.

Candeaux Duffatt, Rafael y Díaz Cisnerosd, Luís R. 1994. MAPPING Interactivo. *MAPPING Interactivo.* [En línea] Julio de 1994. [Citado el: 20 de Octubre de 2011.] http://www.mappinginteractivo.com/plantilla-ante.asp?id_articulo=1184. 1.131-9.100.

Castillo, Oswaldo, Figueroa, Daniel y Sevilla, Hector. 2011. Programación Extrema. *Programación Extrema.* [En línea] 2011. [Citado el: 20 de Marzo de 2012.] http://programacionextrema.tripod.com/fases.htm#primeraFase.

CCM-Benchmark Network . 2012. Kioskea.net. [En línea] 2012. [Citado el: abril de 2 de 2012.] http://es.kioskea.net/faq/2938-los-tablet-pc.

Cerda, Felipe. 2010. aeap. [En línea] 2010. [Citado el: 30 de Enero de 2012.] www.aeap.es/ficheros/be6241eaf9a61dcf6f169d49c79808e4.pdf.

clubdesarrolladores. 2009. clubdesarrolladores. [En línea] 13 de Diciembre de 2009. [Citado el: 23 de Marzo de 2012.] http://www.clubdesarrolladores.com/articulos/mostrar/115-observer-pattern-patron-observador.

Comas Pérez, Yassel. 2011. *Desarrollo de una aplicación de mapas de rutas de transporte de La Habana sobre gvSIG Mobile.* UCI. La Habana : s.n., 2011.

ctisa. 2009. ctisa. *ctisa.* [En línea] 2009. [Citado el: 9 de Febrero de 2012.] http://www.ctisa.com/diccionario.asp.

Cueva Lovelle, Juan Manuel. 1999. *Introducción a UML.Lenguaje para modelar objetos.* 1999.

Curtis, Gregory. 2006. *Cave Painters: Probing the Mysteries of the World's First Artists.* 2006. ISBN 1-4000-4348-4.

Definición.de. 2008. Definición.de. [En línea] 2008. [Citado el: 29 de Noviembre de 2011.] http://definicion.de/bluetooth/.

—. 2008. Definición.de. [En línea] 2008. [Citado el: 29 de Noviembre de 2011.] http://definicion.de/wireless/.

EcuRed. 2012. EcuRed. *EcuRed.* [En línea] 2012. [Citado el: 28 de Febrero de 2012.] http://www.ecured.cu/index.php/Proceso_Unificado_de_Desarrollo.

—. 2012. EcuRed. *EcuRed.* [En línea] 2012. [Citado el: 25 de Enero de 2012.] http://www.ecured.cu/index.php/Sun_Microsystems.

ESRI. 2009. Geosistec. *Geosistec.* [En línea] Enero de 2009. [Citado el: 11 de Noviembre de 2011.] http://www.geosistec.com/productos_arcpad.shtml.

Fernández, Tomas. 2008. Universidad Politécnica de Madrid. [En línea] 2008. [Citado el: 1 de febrero de 2012.] http://www.google.com.cu/url?sa=t&rct=j&q=GML:+El+lenguaje+de+marcado+extendido+(XML)+para+la+Ingenier%C3%ADa+Geogr%C3%A1fica.&source=web&cd=1&ved=0CCUQFjAA&url=http%3A%2F%2Fmapas.topografia.upm.es%2FgeoserviciosOGC%2Fdocumentacion%2FGML%2FGML_en_la_IngGe.

ferrovial. 2010. ferrovial. [En línea] 2010. [Citado el: 30 de Mayo de 2012.] http://memoria2010.ferrovial.com/es/index.asp?MP=107&MS=0&MN=1.

Fundación ONCE. 2011. *Libro blanco para el diseño de Tecnología Móvil accesible y fácil de usar.* España : s.n., 2011.

Gamma, Erich, y otros. 1997. *Design Patterns : Elements of Reusable Object-Oriented Software.* 1997.

García de Jalón, Javier, y otros. 2000. *Aprenda Java como si estuviera en primero.* 2000.

García, Miguel. 2010. Recuerdos de Pandora. *Recuerdos de Pandora.* [En línea] 19 de Febrero de 2010. [Citado el: 20 de Octubre de 2011.] http://recuerdosdepandora.com/historia/inventos/mapa-babilonico-del-mundo/.

Geología. 2010. Geología en Hermosillo. [En línea] 2010. [Citado el: 25 de Febrero de 2012.] http://geologia-uson.jimdo.com/materias/semestre-i/geograf%C3%ADa/.

ginga. 2011. Introducción al software libre. [En línea] 2011. [Citado el: 2 de abril de 2012.] http://ginga.softwarelibre.org.bo/doku.php?id=eclipse.

GIS Dictionary. Esri. [En línea] [Citado el: 31 de Octubre de 2011.] http://support.esri.com/en/knowledgebase/GISDictionary/search..

Glosario de Informática e Internet. 2012. Glosario de Informática e Internet. *Glosario de Informática e Internet.* [En línea] 2012. [Citado el: 10 de Enero de 2012.] http://www.internetglosario.com/916/API.html.

—. 2006. Glosario de Informática e Internet. *Glosario de Informática e Internet.* [En línea] 27 de Octubre de 2006. [Citado el: 20 de Octubre de 2011.] http://tecnologia.glosario.net/terminos-tecnicos-internet/plataforma-1335.html.

—. 2012. Glosario de Informática e Internet. *Glosario de Informática e Internet.* [En línea] 2012. [Citado el: 20 de Enero de 2012.] http://www.internetglosario.com/454/TCPIP.html.

—. 2012. Glosario de Informática e Internet. *Glosario de Informática e Internet.* [En línea] 2012. [Citado el: 20 de Enero de 2012.] http://www.internetglosario.com/796/CPU.html.

glosario.net. 2006. glosario.net. *glosario.net.* [En línea] 26 de Junio de 2006. [Citado el: 25 de Febrero de 2012.] http://tecnologia.glosario.net/terminos-tecnicos-internet/case-265.html.

González Barahona, Jesús M. 2007. Introducción al software libre. *Introducción al software libre.* [En línea] 2007. [Citado el: 2 de Marzo de 2012.] http://www.atenas.cult.cu/rl/informatica/manuales/sl/introduccion_al_SL/eclipse.html.

GSInnova. 2007. GSInnova. [En línea] 2007. [Citado el: 30 de Mayo de 2012.] http://www.rational.com.ar/herramientas/softwaredevelopmentplatform.htm.

GSMspain. 2012. GSMspain. [En línea] 2012. [Citado el: abril de 2 de 2012.] http://www.gsmspain.com/glosario/?palabra=SMARTPHONE.

Hernando, Roberto. 2009. rhernando.net. [En línea] 2009. [Citado el: 2 de Junio de 2011.] Blog personal de Roberto Hernando. http://www.rhernando.net/modules/tutorials/doc/ing/met_soft.html.

INEI. 2012. Instituto Nacional de Estadítias e Informática. [En línea] 2012. [Citado el: 28 de Febrero de 2012.] http://www.inei.gob.pe/biblioineipub/bancopub/inf/Lib5038/defi.HTM.

International Center for Tropical Agriculture. 2009. CIAT . [En línea] 2009. [Citado el: 21 de marzo de 2012.] http://webapp.ciat.cgiar.org/dtmradar/glosario.htm.

itiner. 2009. itiner. [En línea] 2009. [Citado el: 24 de Marzo de 2012.] http://www.itiner.pl/en/welcome/other?from=botMenu&p=technologies .

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo de Software.* España : Addison Wesley, 2000. ISBN:0-201-57169-2.

—. 2000. *El Proceso Unificado de Desarrollo de Software.* España : Addison Wesley, 2000. 0-201-57169-2.

Keogh, Jim. 2004. Java Programming Tutorials & Information. [En línea] 2004. [Citado el: 28 de Noviembre de 2011.] http://www.devarticles.com/c/a/Java/Multithreading-in-Java/.

Kuchana, Partha. 2004. *Software architecture design patterns in Java.* s.l. : CRC Press LLC, 2004. 0-8493-2142-5.

kXML. 2005. kXML. [En línea] 2005. [Citado el: 25 de Marzo de 2012.] http://kxml.sourceforge.net/ .

Larman, Craig. 1999. *UML y Patrones Introducción al análisis y diseño orientado a objetos.* Mexico : Prentice Hall, 1999. 970-17-0261-1.

masadelante. 1999. masadelante.com. *masadelante.com.* [En línea] 1999. [Citado el: 20 de Enero de 2012.] masadelante.com.

Meza, Rommel. 2005. elguille. *elguille.* [En línea] 12 de Agosto de 2005. [Citado el: 20 de Marzo de 2012.] http://www.elguille.info/colabora/NET2005/rommelmeza_SingletonNet.htm.

modelado10. 2011. modelado10. *modelado10.* [En línea] 30 de Noviembre de 2011. [Citado el: 20 de Enero de 2012.] http://modelado10.wordpress.com/.

Montesinos, M y Carrasco, J. 2009. *gvSIG Mobile y gvSIG Mini clientes móviles de una IDE.* España : s.n., 2009.

Nájera, Gilberto. 2005. mailxmail.com. *mailxmail.com.* [En línea] 14 de Julio de 2005. [Citado el: 28 de Enero de 2012.] http://www.mailxmail.com/curso-desarrollo-aplicaciones-dispositivos-inalambricos-j2me/introduccion.

NCGIA. 2008. COPADE. [En línea] 2008. [Citado el: 2 de Noviembre de 2011.] NCGIA (National Centre of Geographic Information and Analysis) concepto realizado en 1990. http://www2.neuquen.gov.ar/copade/contenido/contenido.aspx?PagNombre=ide-1-5.

Olaya, Víctor. 2011. Sistemas de Información Geografíca. [En línea] 24 de Marzo de 2011. [Citado el: 24 de Octubre de 2011.] http://sextante.googlecode.com/files/Libro_SIG.pdf.

Open Geospatial Consortium. 2012. OGC. [En línea] 2012. [Citado el: 19 de marzo de 2012.] http://www.opengeospatial.org/standards/kml.

ORACLE. 2011. Java. [En línea] 2011. [Citado el: 28 de Noviembre de 2011.] http://www.java.com/es/about/.

Oracle. 2011. java.net. [En línea] 2011. [Citado el: abril de 2 de 2012.] http://lwuit.java.net/.

Pcmag. 2010. Pcmag. [En línea] 2010. [Citado el: 22 de febrero de 2012.] http://www.pcmag.com/encyclopedia_term/0,2542,t=Pocket+PC&i=49428,00.asp.

Pozo, Pau Serra. 2002. mappinginteractivo. [En línea] Octubre de 2002. [Citado el: 25 de Enero de 2012.] http://www.mappinginteractivo.com/plantilla-ante.asp?id_articulo=179.

Pressman, Roger S. 2001. *Ingeniería del Software. Un enfoque práctico.* Quinta Edición. Madrid : s.n., 2001.

Prieto Alvarez, Julio Cesar. 2009. *Propuesta de arquitectura para el proyecto SIGESPRO.* 2009. pág. 32.

Prieto Donate, Francisco. 2007. e-REdING. *e-REdING.* [En línea] 2 de Octubre de 2007. [Citado el: 20 de Febrero de 2012.] http://bibing.us.es/proyectos/abreproy/11372/direccion/Memoria%252F.

PROGRAMACION WEB. 2011. PROGRAMACION WEB. *PROGRAMACION WEB.* [En línea] 18 de Enero de 2011. [Citado el: 14 de Noviembre de 2011.] http://programacionweb2011.blogspot.com/2011/01/metodologias-para-el-desarrollo-de.html.

RAE. 2001. Sitio web de la Real Academia Española. *Sitio web de la Real Academia Española.* [En línea] 2001. [Citado el: 18 de Octubre de 2011.] http://buscon.rae.es/draeI/.

Ramírez, Iván. 2011. softonic. [En línea] Agosto de 2011. [Citado el: 29 de Noviembre de 2011.] http://netbeans-ide.softonic.com/.

RED DE APRENDIZAJE. 2012. RED DE APRENDIZAJE. [En línea] 2012. [Citado el: 30 de Mayo de 2012.] http://www.reddeaprendizaje.com/inicio/item/47-plataforma-informatica.

Ríos González, Omar. 2010. *Generador de mapas croquis en formato SVG-Tiny para dispositivos móviles aplicando perfiles.* Cuernavaca, Morelos, México : s.n., 2010. Maestría en Ciencias en Ciencias de la Computación.

Romeu Carrasco, Alberto. 2009. *SIGATEX Móvil.* España : s.n., 2009.

Tardáguila Moro, César. 2009. Mosaic. [En línea] 2009. [Citado el: 27 de Octubre de 2011.] http://mosaic.uoc.edu/.

TechTarget . 2008. TechTarget . [En línea] 2008. [Citado el: 2 de abril de 2010.] http://whatis.techtarget.com/file-extension/ECW-2-FileFormat.html.

The University of Melbourne. 1999. Introducción a los SIG. [En línea] 1999. [Citado el: 30 de Noviembre de 2011.] http://www.geogra.uah.es/gisweb/1modulosespanyol/IntroduccionSIG/GISModule/GIST_Vector.htm.

Tinoco Gómez, Oscar, Rosales López, Pedro Pablo y Salas Bacalla, Julio. 2010. Criterios de selección de metodologías de desarrollo de software. [En línea] 2010. [Citado el: 2 de diciembre de 2011.] http://www.scielo.org.pe/scielo.php?pid=S1810-99932010000200009&script=sci_arttext. ISSN: 1810-9993.

TIOBE Software. 2011. TIOBE Software. *TIOBE Software.* [En línea] 2 de Septiembre de 2011. [Citado el: 19 de Enero de 2012.] http://testingbaires.com/tag/tiobe-software/.

Topografix. 2009. Topografix. [En línea] 2009. [Citado el: 6 de abril de 2012.] http://www.topografix.com/gpx.asp.

Universidad Politécnica de Madrid. 2006. Patrones del “Gang of Four”. [En línea] 2006. http://sunshine.prod.uci.cu/gridfs/sunshine/books/Patrones_del_Gang_of_Four.pdf.

Vázquez Manso, Ruben. 2011. *Desarrollo de un Sistema de Información Geográfica para la Universidad de las Ciencias Informáticas basado en la Plataforma gvSIG Mobile.* UCI. La Habana : s.n., 2011.

World Wide Web Consortium. 2008. W3C. [En línea] 2008. [Citado el: 21 de febrero de 2012.] http://www.w3.org/TR/SVGTiny12/.

Yahoo! . 2012. Yahoo! . [En línea] 2012. [Citado el: 2 de abril de 2012.] http://info.yahoo.com/privacy/espanol/yahoo/maps/.

York University. 2011. John Snow's Cholera Map Maps. *John Snow's Cholera Map Maps.* [En línea] 15 de Octubre de 2011. [Citado el: 20 de Octubre de 2011.] http://www.york.ac.uk/depts/maths/histstat/snow_map.htm.