UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 6

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

Herramienta para la creación de Sistemas de Información Geográfica.

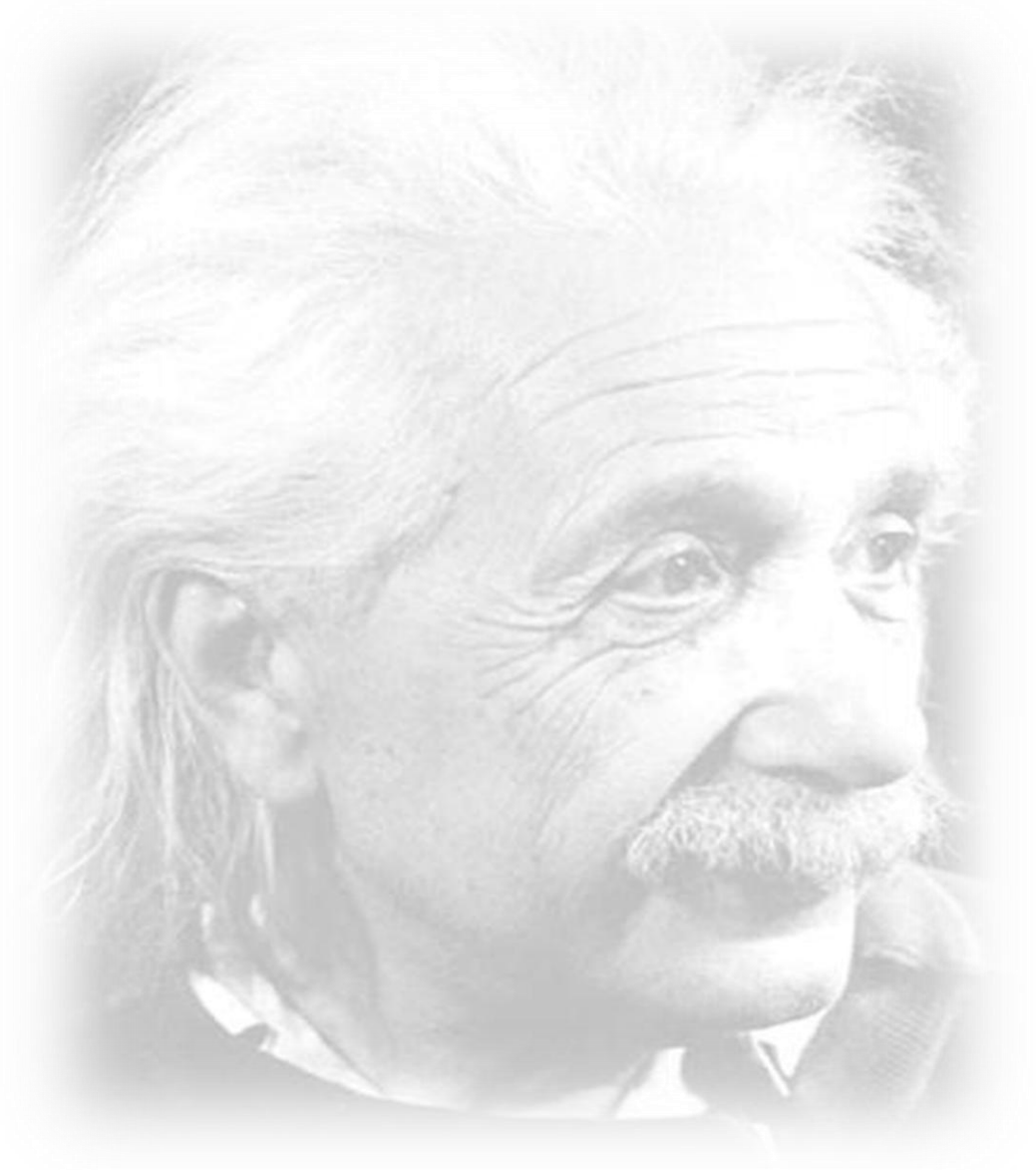
AUTORES: Raidel Páez LLopiz.

Ariel Labrada Delgado.

TUTOR: Ing. Alain León Companioni.

La Habana, junio del 2012.

Año 54 de la Revolución



La vida es muy peligrosa. No por las personas que hacen el mal, sino por las que se sientan a ver lo que pasa.

Declaración de Autoría.

Declaramos que somos los únicos autores de este trabajo y autorizamos a 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 _____.

Ariel Labrada Delgado

_____

Firma del Autor

Raidel Paez LLopiz

_____

Firma del Autor

Alain Léon Companioni

_____

Firma del Tutor

Datos de Contacto

TUTOR:

Nombre: Ing. Alain León Companioni.

Correo Electrónico: acompanioni@uci.cu

Universidad de las Ciencias Informáticas.

Agradecimientos

De Ariel

Agradezco especialmente:

… A mi madre por ser siempre la luz que guía mis pasos, y a mi padre por darme todo su apoyo.

… A mis hermanos presentes en los momentos más difíciles.

… A toda mi familia en general y a todas las personas que de una forma u otra me ayudaron a llegar aquí y confiaron en mí, sobre todo a aquellos que me comprometieron a esforzarme para ser quien soy.

De Raidel

… Agradezco a mi familia, en especial a mi madre Bertina que ha dedicado toda su vida a servir de cobijo a mi hermano y a mi, como un árbol que brinda su sombra a quien la necesita, a mi padre Rodolfo, a mi hermano Rodolfo alias el RO2 que ha sido mi ejemplo a seguir y a todos mis tíos y primos en general.

… A todas las amistades que he conocido en el transcurso de mis estudios que de una forma u otra me han apoyado en los momentos difíciles.

Agradecimientos

De Ambos

Agradecemos a nuestro tutor Alain León Companioni y a todos los profesores del proyecto Aplicativos SIG por todo el apoyo que nos han brindado durante esta importante y difícil etapa.

Resumen

En la actualidad los Sistemas de Información Geográfica (SIG) son una tecnología eficiente para analizar, manipular y generar información geográficamente referenciada, los cuales tienen gran impacto fundamentalmente en la toma de decisiones y planificación en los sectores económicos, políticos y sociales. En la Universidad de las Ciencias Informáticas (UCI) se ha desarrollado una plataforma con el fin de crear SIG como productos competentes a nivel internacional. Dicha plataforma, llamada GeneSIG, es un producto soberano desarrollado con herramientas y tecnologías libres que toma papel protagónico en la política de Cuba en cuanto a tener independencia tecnológica. Los procesos y tareas llevadas a cabo en el proyecto Aplicativos SIG para realizar personalizaciones a dicha plataforma, son engorrosas, pues se requiere un amplio dominio de la misma en cuanto a su estructura y funcionamiento, lo que conlleva a pérdida de tiempo y problemas de reutilización de los archivos y componentes involucrados. La presente investigación es el resultado del ciclo completo de desarrollo de *software* que propone la metodología XP, teniendo como resultado una aplicación de escritorio que permite agilizar y facilitar los procesos de personalización de la plataforma GeneSIG.

Palabras Claves:

Plataforma, SIG, Aplicación de escritorio.

Abstract

Currently the Geographic Information Systems (GIS) are an efficient technology to analyze, manipulate and generate geographically referenced information, which have great impact primarily on decision making and planning in the economic, political and social. At the University of Informatics Sciences (UCI) has developed a platform to create GIS as competing products internationally. This platform, called GeneSIG, is a sovereign product developed with free tools and technologies that take leading role in policy regarding Cuba have technological independence. The processes and tasks carried out in the project Aplicativos SIG to make customizations to the platform are troublesome because it requires a strong command of it in terms of structure and function, leading to loss of time and trouble of re-files and components involved. The present investigation is the result of the full cycle of *software* development methodology proposed by the XP, resulting in a desktop application that helps streamline processes and facilitate customization of the platform GeneSIG.

Keywords:

Platform, GIS, Desktop Application.

Índice de Contenido

Índice de Contenido

Índice de Contenido

Introducción

Las primeras manifestaciones cartográficas no pueden datarse a ciencia cierta. Aunque parezca incuestionable su existencia, los dibujos encontrados en cuevas y petroglifos fueron los primeros indicios que antecedieron a los mapas actuales. Estos representaban aspectos relacionados con la subsistencia y territorios (donde vivían y cazaban).Además, eran utilizados como un instrumento dirigido a un fin concreto, sirviendo de símbolo e ilustración. Con el transcurso del tiempo fue creciendo la evolución de los mapas lo cual contribuyó a análisis más profundos de la conceptualización del entorno geográfico.

Desde la mitad del siglo XX, la ciencia geográfica busca nuevas opciones para llevar a cabo sus estudios. El avance científico y el desarrollo tecnológico alcanzado modificaron la forma tradicional de abordar y realizar las actividades humanas.

La revolución tecnológica que dio paso a la era de la computación, trajo consigo la rápida evolución de la informática. Con ello se lograron reducir los tiempos para procesar, archivar y recuperar grandes volúmenes de datos. Se hizo posible confeccionar una amplia gama de combinaciones en el manejo de diversas variables, así como, el estudio y manipulación de situaciones hipotéticas que, sin el uso de las computadoras, serían muy difíciles de efectuar.

Así, gradualmente, se comenzaron a utilizar nuevas tecnologías para generar información geográfica. Entre estas tecnologías se encuentran: la percepción remota (imágenes de satélite), la moderna fotografía aérea, la fotogrametría digital, el sistema de posicionamiento global (GPS) y los Sistemas de Información Geográfica (SIG).

En los últimos tiempos se ha incrementado exponencialmente el uso de los SIG para la toma de decisiones, relacionado íntimamente con el incremento de la efectividad y la disminución de los costos que se obtienen con la puesta en marcha de un sistema de este tipo.

Estos programas permiten georeferenciar la información espacial y almacenarla en bases de datos. Además posibilitan que los mapas sean más dinámicos e interactivos permitiendo que sean manipulados de manera digital. En el mundo existe un creciente auge del uso e implementación de los SIG, cada vez más eficientes y con un gran número de funcionalidades. Existen numerosas y grandes compañías dedicadas a desarrollar este tipo de sistemas que han contribuido a que sean más sofisticados y novedosos.

Introducción

Cuba progresivamente ha alcanzado un desarrollo en la informática, lo que ha conllevado a procesos de transformaciones desde el punto de vista tecnológico. El país se encuentra en una fase de estudio y desarrollo de los SIG, así como, su vinculación y adaptación con las alternativas libres existentes en el mundo.

La Universidad de las Ciencias Informáticas (en lo adelante UCI) es uno de los máximos exponentes en el país en la creación de Sistemas de Información Geográfica. En el Centro de Desarrollo de Geoinformática y Señales Digitales (GEySED) de la UCI se desarrollan SIG tomando como soporte la plataforma GeneSIG.

La plataforma GeneSIG desarrollada en la UCI surge como necesidad de contar con un producto soberano que sirva como soporte al desarrollo de aplicaciones de SIG. Está implementada con herramientas y tecnologías libres, cumpliendo además con la política de migración al *software* libre y de soberanía tecnológica que impulsa el país. La plataforma tiene como objetivo fundamental realizar la representación geoespacial de la información asociada a negocios específicos, permitiendo además realizar análisis sobre dicha información. Presenta una arquitectura distribuida, que cuenta con una base cartográfica, y sobre ella un conjunto de objetos representados geoespacialmente, que contienen información asociada, ya que se identifican a partir de las necesidades particulares de los clientes finales. Constituye un producto de gran actualidad y con grandes expectativas en el mercado internacional.

Para la creación de una personalización sobre la plataforma GeneSIG es necesario tener un amplio dominio de la misma, lo que conlleva que, para la obtención de un nuevo producto, sean engorrosas las tareas realizadas, en cuanto al tiempo, organización y reutilización de los componentes y archivos utilizados en este proceso. Con lo planteado anteriormente surge el siguiente **problema a resolver**: *¿Cómo facilitar el proceso de personalización de la plataforma GeneSIG, para la obtención de nuevos productos?* Definiendo como **objeto de estudio** *el proceso de desarrollo de Herramientas de personalización para el desarrollo de aplicativos SIG*, donde el **campo de acción** lo constituye, *la personalización de la plataforma GeneSIG*, *en el desarrollo de aplicativos SIG*, teniéndose como **objetivo general,** d*esarrollar una herramienta para la creación de Sistemas de Información Geográfica a partir de la plataforma GeneSIG que facilite el proceso de personalización*.

Se defiende la siguiente idea: *el desarrollo de una herramienta para la creación de Sistemas de Información Geográfica a partir de la plataforma GeneSIG, facilitará el proceso de personalización de la misma.*

Para ello se identificaron las siguientes tareas:

1. Caracterización del proceso de desarrollo de los Sistemas de Información Geográfica a partir de la plataforma GeneSIG.
2. Selección y fundamentación de las tendencias y tecnologías actuales a utilizar en el proceso.
3. Identificación de los conceptos asociados al entorno donde está enmarcado el negocio.
4. Identificación de las principales funcionalidades del sistema.
5. Diseño del sistema propuesto.
6. Implementación de las funcionalidades del sistema propuesto.
7. Validación del resultado obtenido certificando la veracidad de los algoritmos empleados.

Los **métodos científicos** que se utilizan en el desarrollo de la investigación son los siguientes:

Métodos Teóricos:

- **Análisis y Síntesis:** Se realizó un análisis detallado de documentos, herramientas y bibliografías, permitiendo la extracción de los elementos esenciales que componen el proceso de personalización de la plataforma GeneSIG.

Métodos Empíricos:

- **Observación:** Se utiliza para identificar los principales procesos que se desarrollan en la personalización de la plataforma GeneSIG y para definir los requisitos tanto funcionales como no funcionales.

Al concluir la investigación se esperan los siguientes resultados:

1. La herramienta de creación de Sistemas de Información Geográfica.
2. La documentación técnica asociada al desarrollo de la herramienta de creación de Sistemas de Información Geográfica.

El siguiente trabajo de diploma estará dividido en 4 capítulos fundamentales:

Capítulo 1: Fundamentación Teórica. En este capítulo se expone la descripción del entorno donde se

encuentra el negocio y su organización, se describe detalladamente la situación problemática y el análisis de otras soluciones que puedan brindar respuesta al problema de la investigación planteado en el presente trabajo.

Capítulo 2: Tendencias y tecnologías actuales a desarrollar. En este capítulo se explican las principales tecnologías, lenguajes de programación y herramientas que se utilizarán para la construcción de la solución propuesta, así como las ventajas de utilizarla.

Capítulo 3. Presentación de la solución propuesta: Este capítulo aborda las dos primeras fases de la Programación Extrema (XP). En la fase de exploración se define las historias de usuario, para un mejor entendimiento del *software* y en la fase de planificación se realiza la estimación de esfuerzos por historia de usuario.

Capítulo 4. Construcción y validación de la solución propuesta: Este capítulo aborda las restantes fases de la Programación Extrema (XP), construcción y prueba. Se describen las tarjetas CRC(Contenido, Responsabilidad y Colaboración) y se detallan las iteraciones llevadas a cabo durante la etapa de construcción de la aplicación, así como las tareas generadas por cada historia de usuario y las pruebas de aceptación efectuadas sobre el sistema.

Capítulo 1

Fundamentación Teórica

1 Introducción

En el presente capítulo se abordan los temas relacionados con la fundamentación teórica de la investigación. Se describen los conceptos asociados al dominio del problema, así como la descripción del objeto de estudio, referente a la personalización de la plataforma GeneSIG, en el Centro de Desarrollo de Geoinformática y Señales Digitales (GEySED) de la Universidad de las Ciencias Informáticas. Además, se analiza el conjunto de soluciones que son utilizadas en la actualidad para creación de SIG y se identifican los principales problemas que motivaron el desarrollo de la investigación.

1.1 Conceptos asociados al dominio del problema

Dato geográfico

Los datos geográficos son la información que permite conocer lo que se encuentra en una determinada posición en el espacio, de qué manera y en qué tiempo. Los datos geográficos no son más que variables que almacenan una determinada información geoespacial de un evento, ubicándolo en tiempo y espacio. (1)

Sistema de Información Geográfica

Un Sistema de Información Geográfica (SIG o GIS) consiste en información de naturaleza diversa sobre un determinado territorio, almacenada en un conjunto de bases de datos tanto gráficas como alfanuméricas, cuya relación con el territorio se realiza a través de un sistema de referencia geográfico y se gestiona a través de uno o varios programas informáticos específico; el conjunto es soportado por un sistema de computadores y por un personal especializado. (2)

Plataforma Informática

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. (3)

Aplicación Informática

Las aplicaciones informáticas forman parte del *software* de la computadora ya que son una serie de programas que han sido desarrollados para facilitarle al usuario la utilización de la máquina para una acción o fin determinados. Una aplicación está diseñada y optimizada como una herramienta para un propósito específico, como respuesta ante una necesidad del usuario.

Existen aplicaciones de distintos tipos, que dependen de la finalidad que tenga. De esta manera, se puede encontrar *software* de aplicaciones de negocios, de utilería, personales o de entretenimiento. Algunos ejemplos de aplicaciones son los procesadores de texto, las planillas de cálculo, los editores de gráficos, los administradores de bases de datos, etc. Además, las aplicaciones tienen que ser independientes del *hardware* de manera tal que puedan ser utilizadas por cualquier computadora. (4)

1.2 Objeto de Estudio

1.2.1 Descripción General

El mundo actual se debe, en gran medida, al surgimiento continuo de nuevas herramientas que contribuyen a este proceso de evolución constante. Tal es el caso de los Sistemas de Información Geográfica (SIG).Un SIG es una integración muy organizada de *software*, *hardware* y datos geográficos, diseñado para capturar, analizar, almacenar, manipular y desplegar la información geográficamente referenciada. (5)

Puede definirse también como un modelo de una parte de la realidad referido a un sistema de coordenadas terrestres, construido principalmente para satisfacer la necesidad de información y ubicación geográfica del mundo. (6) Estos deben ser capaces de ubicar un objeto determinado en el espacio; de encontrar donde está un cuerpo con respecto a otro; de brindar información sobre su perímetro, área y volumen; de encontrar el camino mínimo de un punto a otro, así como la generación de modelos a partir de fenómenos o actuaciones simuladas.

Durante el proceso de creación y manejo de mapas, y la posterior salida de los mismos se suele utilizar gran cantidad de programas, puesto que ningún *software* SIG es el mejor de los programas y no se llegan a cubrir todas las posibles expectativas. Dependiendo de los intereses existen programas más competitivos que otros y con mejores resultados.

El manejo de estas herramientas es llevado a cabo generalmente por profesionales de diversos campos del conocimiento con experiencia en Sistemas de Información Geográfica (cartografía, geografía, topografía, etc.), ya que el uso de estas herramientas requiere un aprendizaje previo en un área específica.

La UCI es una universidad productiva, cuya misión es la formación académica de futuros profesionales, además de producir *software* y servicios informáticos a partir de la vinculación estudio-trabajo como modelo de formación y la de ejecutar las soluciones planificadas para incrementar el impacto en la informatización del país y las exportaciones. En la universidad se promueve el desarrollo de productos y servicios informáticos, logrando disímiles soluciones para hacer más eficiente y eficaz el trabajo en los proyectos productivos y así estar a la vanguardia en lo que a soluciones informáticas se refiere.

El Centro de Desarrollo de Geoinformática y Señales Digitales (GeySED) perteneciente a la universidad tiene como uno de sus principales productos la plataforma GeneSIG. El surgimiento de esta plataforma está ligado al incremento exponencial del uso de los Sistemas de Información Geográficos (SIG) para la toma de decisiones, lo cual está íntimamente relacionado con el incremento de la efectividad y la disminución de los costos que se obtienen con la puesta en marcha de un sistema de este tipo funcionando sobre tecnologías libres.

GeneSIG tiene gran relación con el desarrollo incremental que han tenido los SIG a nivel mundial. Además se encuentra estrechamente vinculado con nuevos modelos de análisis y representación para sistemas de este tipo, así como de *frameworks* de desarrollo y servidores de mapas.

Tiene como principal objetivo realizar la representación geoespacial de la información asociada a negocios específicos, permitiendo además realizar análisis sobre dicha información. Son muchas las empresas en el mundo que están inmersas en la búsqueda de mecanismos y soluciones que permitan mejorar los sistemas y las técnicas existentes, lo que propicia el nacimiento de instrumentos de este tipo.

El proceso de personalización de GeneSIG consiste en adecuar las funcionalidades de dicha herramienta a las condiciones necesarias para la creación de un SIG enmarcado a cualquier negocio. Por tanto, lograr la representación y análisis de objetivos geográficos mediante esta plataforma, permitirá al usuario intercambiar con un *software* diseñado de acuerdo a sus necesidades.

Mediante esta personalización, el usuario tendrá la oportunidad de interactuar con el sistema en dependencia de los datos específicos que desee manipular, sin necesidad alguna de contar con un sistema generalizado que le dificulte el trabajo y facilitándole el manejo de los datos reales y específicos. Sobre esta base el proyecto Aplicativos SIG, el cual pertenece al centro GeySED tiene el objetivo de desarrollar SIG, proyectándose de tal forma que su propósito de existencia o su forma de funcionar sea semejante a una Línea de Productos de *Software* (LPS).

Aplicativos SIG propone un sistema de producción basado en la reutilización de componentes en un determinado dominio, tomando como idea básica el ensamblaje de partes de *software* previamente elaboradas y como antecedentes el desarrollo de *Software* Basado en Componentes, asumiendo la existencia de una industria de partes.

El proyecto para su funcionamiento está dividido en varios equipos de trabajos. Dentro de estos se encuentra el equipo de Integración y Componentes, el cual se encarga de preparar los componentes principales de GeneSIG de manera que sean fáciles de integrar por los demás desarrolladores. Además se encarga de seleccionar los componentes que serán reutilizados y de escogerlas tecnologías más livianas y sólidas para desarrollar las aplicaciones, incluyendo las arquitecturas orientadas a servicio.

1.2.2 Situación problemática

Los principales problemas que afectan actualmente al proyecto Aplicativos SIG del Centro de Desarrollo GeySED se centran en el proceso de personalización de la plataforma GeneSIG para la obtención de nuevos productos. Esto se debe a que la mayoría de los procesos y tareas llevadas a cabo se realizan manualmente, como son, tareas de gestión de proyectos, plugins, mapas, capas, conexiones y otras acciones como limpiar cache negra, borrado de mapas de usuario y la edición de ficheros de configuración. Obligando a los especialistas que laboran en el proyecto a requerir un amplio dominio sobre dicha plataforma para entender en profundidad el funcionamiento interno de la misma. Esto conlleva a una pérdida valiosa de tiempo, trayendo consigo riesgos de retrasos en los convenios de trabajo, en el establecimiento de negocios y el desaprovechamiento de clientes potenciales.

El Centro GeySED requiere de una solución a los problemas antes mencionados para evitar de esta forma retrasos en el proceso de desarrollo, logrando centralizar y elevar la eficiencia del trabajo en el proyecto. Como consecuencia surge la necesidad de crear un sistema que cumpla con las normas

establecidas para la personalización de la plataforma GeneSIG en el proyecto Aplicativos SIG perteneciente al centro.

1.3 Soluciones Existentes

En el mundo existe un gran número de aplicaciones para el trabajo con Sistemas de Información Geográfica. Hasta el momento el módulo de configuración de la plataforma GeneSIG es la única herramienta existente capaz de personalizar dicha plataforma. Se seleccionan además herramientas como: NetBeans y ZendStudio que utilizan *frameworks* como base para llevar a cabo sus soluciones. Lo cual es un proceso importante a tener en cuenta a la hora de realizar tareas como Limpiar Caché Negra y Borrar mapas de usuarios sobre la plataforma GeneSIG.

1.3.1 NetBeans

El IDE NetBeans es un entorno de desarrollo integrado escrito en Java pero puede servir para cualquier otro lenguaje de programación. En él se pueden desarrollar aplicaciones Web utilizando *frameworks* como: *Symfony* y *ZendFramework*. Permite crear aplicaciones Web con PHP 5, con un potente *debugger* integrado. Gracias a NetBeans ya es posible dejar de lado la consola de comandos de *Symfony* y centrarse en desarrollar en el IDE, además se encuentra cargadas todas las clases, ayuda en línea, etc.

1.3.2 ZendEstudio (7)

ZendStudio o *Zend Development Environment* es un completo entorno de desarrollo integrado para el lenguaje de programación PHP. Está escrito en Java, y está disponible para las plataformas *Microsoft Windows*, Mac OS X y GNU/Linux. En él se pueden desarrollar aplicaciones Web utilizando como *framework* de desarrollo *ZendFramework*, además de servir de editor de texto para páginas PHP proporcionando una serie de ayudas que pasan desde la creación y gestión de proyectos hasta la depuración de código.

1.3.3 Módulo de configuración de GeneSIG

La plataforma GeneSIG cuenta actualmente con un módulo de configuración desarrollado como un componente de Cartoweb, con interfaz de usuario en extJS. Este módulo presenta las siguientes funcionalidades:

Gestionar Usuario: Consiste en crear un registro de los usuarios de la plataforma GeneSIG y realizar operaciones fundamentales sobre los mismos como adicionar, eliminar, modificar y asignarle funcionalidades.

Gestionar Plugin: Consiste en crear un registro de los plugins de la plataforma GeneSIG y realizar operaciones fundamentales sobre los mismos como adicionar, eliminar y modificar.
Gestionar Funcionalidades: Consiste en crear un registro de las funcionalidades de la plataforma GeneSIG y realizar operaciones fundamentales sobre las mismos como adicionar, eliminar, modificar y establecer dependencias entre los mismos.
Gestionar Proyectos: Consiste en crear nuevos proyectos de GeneSIG brindar las opciones de configurar sus archivos de configuración tanto dentro como fuera del proyecto y los de los plugins que los integran.

1.3.4 Comparación de las Soluciones Existentes

Luego de realizar un estudio de las soluciones existentes expuestas anteriormente la más cercana a la solución del problema planteado es el módulo de configuración de la plataforma GeneSIG. Dicho módulo ayuda a realizar personalizaciones sobre la plataforma permitiendo una configuración personalizada por plugins y funcionalidades para poder dar un tratamiento personalizado sobre los mismos, efectuando operaciones básicas de gestión tales como: adicionar, modificar y eliminar.

A pesar de las ventajas que presenta el módulo de configuración para el proceso de personalización, es necesario constar con una aplicación que desarrolle estos procesos sin que sea necesario ejecutar la plataforma GeneSIG junto con los servidores *MapServer* y Apache. Además se necesita de funcionalidades para gestionar los mapas, capas y conexiones a los datos de los proyectos. También es necesario borrar los mapas de usuarios, limpiar caché negra del sistema y exportar e importar plugins con las configuraciones pertinentes a los mismos.

1.4 Conclusiones del capítulo

En el actual capítulo se ha visto la definición de los principales términos propios de la presente investigación, para así lograr un mejor entendimiento del problema a resolver. La solución más cercana a resolver el problema planteado es el módulo de configuración de la plataforma GeneSIG. Este realiza personalizaciones sobre dicha plataforma, pero no posee las funcionalidades necesarias para optimizar las tareas que se desarrollan en esta. Además, se necesita de una aplicación que pueda trabajar sobre la plataforma sin necesidad de que esta esté en ejecución, para así reducir esfuerzos y lograr mejores resultados. Se determinó que se debía desarrollar una herramienta para disminuir en tiempo las actividades realizadas y facilitar a su vez el proceso de personalización de la plataforma GeneSIG.

Capítulo 2

Tendencias y tecnologías actuales a desarrollar.

2.1 Introducción

Este capítulo está dirigido al análisis de las diferentes tecnologías y herramientas que se emplean en la construcción de la solución a través de un estudio detallado de las mismas. Mediante este razonamiento se logra determinar la necesidad e importancia de su manejo y utilización.

2.2 Arquitectura de Software

La Arquitectura del *Software* es la organización fundamental de un sistema formado por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución. (8)

En sí, la Arquitectura de *Software* da una visión de lo que el sistema debe hacer y se apoya para ello en el arquitecto que es quien define las principales herramientas y tecnologías, dígase patrones de diseño, estilos arquitectónicos, lenguajes de implementación, *framework*, etc. Por consiguiente esta sienta las bases para el desarrollo y la construcción de un sistema de *software* de manera económico y fiable con la realización de la mayor cantidad de requerimientos no funcionales, obteniendo una alta satisfacción del cliente.

Por lo antes mencionado, el diseño de una arquitectura robusta es muy importante para que el *software* salga con la calidad requerida, obteniendo como resultado el éxito del proyecto. Además de eso, la arquitectura permite una muy buena comunicación entre las personas involucradas en el desarrollo del sistema y una temprana documentación de las decisiones de diseño tomadas.

2.3 Arquitectura Modelo-Vista-Controlador

Modelo Vista Controlador (MVC) es un estilo de arquitectura de *software* que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Aunque el patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el mismo no está estrechamente vinculado con este tipo de aplicaciones, pues al ser un estilo arquitectónico, propone una vista estructural de alto nivel. El modelo lo constituyen los datos con los que trabaja la aplicación y la lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista. (9)

El modelo: Conjunto de clases que representan la información del mundo real que el sistema debe procesar, sin tomar en cuenta ni la forma en la que esa información va a ser mostrada ni los mecanismos que hacen que esos datos estén dentro del modelo. El modelo es el responsable de definir las reglas de negocio y llevar un registro de las vistas y controladores del sistema. (10)

Vista: Una vista obtiene del modelo solamente la información que necesita para desplegar y se actualiza cada vez que el modelo del dominio cambia por medio de notificaciones generadas por el modelo de la aplicación. Las vistas son responsables de recibir los datos del modelo y mostrarlos al usuario además de tener un registro de su controlador asociado. (11)

El controlador: Objeto que se encarga de dirigir el flujo del control de la aplicación debido a mensajes externos, como datos introducidos por el usuario u opciones del menú seleccionadas por él. A partir de estos mensajes, el controlador se encarga de modificar el modelo o de abrir y cerrar vistas. Es el responsable de recibir los eventos de entrada y contener las reglas de gestión de eventos.

Ventajas

MVC aporta una construcción de *software* sostenible, en la que se pueden localizar de forma ágil los errores. Supone un diseño modular, y muy poco acoplado, favoreciendo la reutilización. Facilita la labor de todo el equipo: diseñadores gráficos, programadores, diseñadores de base de datos.

Al separar la presentación de la programación (o lógica de negocio), la aplicación es más fácil de modificar en el futuro.

El resultado es más claro, y el reparto de tareas dentro del equipo de trabajo es más fácil; la depuración de la aplicación es más sencilla y, finalmente, puede utilizarse un marco de trabajo (o *framework*) bien testeado.

2.4 Metodología de desarrollo a utilizar

Las metodologías imponen un proceso disciplinado sobre el desarrollo de *software* con el fin de hacerlo más predecible y eficiente. Lo hacen desarrollando un proceso detallado con un fuerte énfasis en planificar inspirado por otras disciplinas de la ingeniería. Las metodologías guían todo el proceso de desarrollo de *software*, determinando pautas a seguir y haciendo hincapié en la planificación, con el objetivo de hacer más eficiente el proceso. (12)

En el desarrollo de aplicaciones, es sin dudas de vital importancia la selección de una metodología de desarrollo. Actualmente existe un sin número de éstas, siendo imposible determinar una que satisfaga en su totalidad las necesidades de un proyecto de desarrollo en específico; de esta forma, la selección

de una metodología de desarrollo debe llevarse a cabo teniendo en cuenta diferentes puntos que influyen en el desarrollo del sistema, entre estos: recursos técnicos, humanos y tiempo de desarrollo esperado o disponible. Básicamente consiste en trabajar estrechamente con el cliente, haciendo pequeñas iteraciones (mini-entregas), cada dos semanas, donde no existe más documentación que el código en sí cada versión contiene las modificaciones necesarias según el cliente vaya retroalimentando el sistema (por eso es necesaria la disponibilidad del cliente durante todo el desarrollo).

Existen varias propuestas metodológicas entre las que se encuentran las metodologías pesadas o tradicionales y las ágiles o ligeras. Las metodologías pesadas o tradicionales resultan convenientes para proyectos que requieran mucho tiempo y recursos, sin embrago para proyectos donde el entorno del sistema es muy cambiante y el tiempo se encuentra restringido es necesario utilizar una metodología ágil.

Las metodologías ágiles están orientadas especialmente a proyectos pequeños y permiten que los programadores se concentren solamente en aquellas funciones que se necesitan inmediatamente. Se realizan entregas al cliente frecuentemente, de las cuales se obtiene retroalimentación constante, posibilitando respuestas rápidas a los cambios en el negocio. (13)

2.4.1 RUP

RUP (Proceso Unificado de *Rational*) provee un acercamiento disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de *software* de alta calidad que satisfaga los requisitos de los usuarios finales (respetando cronograma y presupuesto). Fue desarrollado por *Rational Software*, y está integrado con toda la *suite Rational* de herramientas. Puede ser adaptado y extendido para satisfacer las necesidades de la organización que lo adopte. (14)

RUP es un proceso para el desarrollo de un proyecto de *software* que define quién, cómo, cuándo y qué debe hacerse en el proyecto. Está dirigido por los casos de usos: que orientan al proyecto a la importancia para los usuarios y lo que este quiere; está centrado en la arquitectura: que relaciona la toma de decisiones que indican cómo tiene que ser construido el sistema y en qué orden; y es iterativo e incremental: dividiéndose el proyecto en mini proyectos donde los casos de uso y la arquitectura cumplen su uso de manera más depurada.

RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en un número variable según el proyecto y en las que se hace un mayor o menor hincapiés en las distintas

actividades.

Ventajas:

- Evaluación en cada fase que permite cambios de objetivos.
- Funciona bien en proyectos de innovación.
- Es sencillo, ya que sigue los pasos intuitivos necesarios a la hora de desarrollar el *software*.
- Seguimiento detallado en cada una de las fases.

2.4.2 XP

XP (Programación Extrema), es la metodología más destacada de los procesos ágiles de desarrollo de *software* formulada por Kent Beck. La programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. (14)

Las características fundamentales de la programación extrema son:

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación.
- Programación por parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto.
- Frecuente interacción del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- Corrección de todos los errores antes de añadir nueva funcionalidad y hacer entregas frecuentes.
- Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenimiento pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.

- Simplicidad en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo. (14)

El ciclo de vida ideal consta de 4 fases: (15)

1. Exploración: En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo.

2. Planificación: En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas.

3. Construcción: La fase de construcción requiere de las tarjetas CRC (Contenido, Responsabilidad, Colaboración), estas permiten desprenderse del método de trabajo basado en procedimientos y trabajar con una metodología basada en objetos. En esta fase se realiza el desarrollo de las iteraciones, donde las historias de usuario se descomponen en tareas de programación o en tareas de ingeniería.

4. Prueba: Esta fase garantiza el correcto funcionamiento del código que se va implementando. Las pruebas se dividen en dos: en pruebas unitarias y pruebas de aceptación. Las unitarias verifican que el código correspondiente a un módulo específico se comporte de manera esperada.

Ventajas:

- Apropiado para entornos volátiles.
- Estar preparados para el cambio, significa reducir su coste.
- Planificación más transparente para nuestros clientes, conocen las fechas de entrega de funcionalidades.
- Permitirá definir en cada iteración cuales son los objetivos de la siguiente.
- Permite tener realimentación de los usuarios muy útil.

- La presión está a lo largo de todo el proyecto y no en una entrega final. (14)

2.4.3 Selección de la metodología a utilizar

Luego del análisis realizado entre dos de las metodologías más usadas en la universidad se selecciona XP para la realización del trabajo, debido a que se adapta en gran medida tanto al tipo de proyecto a desarrollar en este caso pequeño ya que consta de dos personas. Además, la Programación Extrema es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo del *software*. Este tipo de método se basa en una realimentación continua entre el cliente y el equipo de desarrollo, con una comunicación fluida entre todos los participantes. También busca la simplicidad en las soluciones implementadas y el coraje para enfrentar los cambios.

A continuación se exponen varias de las razones que llevaron al uso de esta metodología. El proyecto es pequeño y XP está concebida para ser utilizada dentro de proyectos pequeños. No existe un contrato previo especificando tiempo, recursos y alcance. Para el desarrollo del sistema no se dispone de un contrato con un presupuesto ni un alcance previamente definidos, puesto que es un proyecto para el uso interno de la UCI y será llevado a cabo por programadores pertenecientes a la misma. Los requisitos del sistema cambian frecuentemente. Con la aceptación de nuevos tipos de proyectos, con estructura y requerimientos disímiles, el sistema debe cambiar y ampliar sus funcionalidades de forma que sea capaz de adaptarse a cada nueva situación. Uno de los principios básicos de XP es que el cambio frecuente de los requisitos es algo normal en el proceso de desarrollo. Esta metodología se adapta perfectamente a los proyectos cuyos requisitos cambian a menudo.

2.5 Lenguajes de programación

Un lenguaje de programación es un conjunto de sintaxis y reglas semánticas que definen los programas del computador. Es una técnica estándar de comunicación para entregarle instrucciones al computador. Un lenguaje le da la capacidad al programador de especificarle al computador los tipos de datos y las acciones a ejecutar bajo una variada gama de circunstancias, utilizando un lenguaje relativamente próximo al lenguaje humano. Permiten crear programas y *software*. Los lenguajes de programación facilitan la tarea de programación, ya que disponen de formas adecuadas que permiten ser leídas y escritas por personas, a su vez resultan independientes del modelo de computador a utilizar. (16)

2.5.1 Java

Java es un lenguaje que ha sido diseñado para producir *software*, confiable y multiplataforma. La compatibilidad es total (a nivel de fuentes, a nivel de bibliotecas, a nivel del código compilado), flexible y robusto. (17)

Características del lenguaje Java (18)

Lenguaje simple.

Java posee una curva de aprendizaje muy rápida. Además elimina la complejidad de los lenguajes de programación como el C y el C++, incorporando un conjunto de características ya utilizadas en la historia de los lenguajes de programación que proporcionan facilidad de uso, como son:

- Herencia Múltiple y Simple.
- Control de Excepciones.
- La concurrencia y la multitarea.
- Vínculos dinámicos y la recogida automática de elementos no utilizados.

Debido a su semejanza con C y C++, y dado que la mayoría de la gente los conoce aunque sea de forma elemental, resulta muy fácil aprender Java. Los programadores experimentados en C++ pueden migrar muy rápidamente a Java y ser productivos en poco tiempo.

Orientado a objetos.

Java fue diseñado como un lenguaje orientado a objetos desde el principio. Los objetos agrupan en estructuras encapsuladas tanto sus datos como los métodos (o funciones) que manipulan esos datos. La tendencia del futuro, a la que Java se suma, apunta hacia la programación orientada a objetos, especialmente en entornos cada vez más complejos y basados en red.

Distribuido.

Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.

Interpretado y compilado a la vez.

Java es compilado, en la medida en que su código fuente se transforma en una especie de código máquina, los *bytecodes*, semejantes a las instrucciones de ensamblador. Por otra parte, es interpretado, ya que los *bytecodes* se pueden ejecutar directamente sobre cualquier máquina a la cual

se hayan portado el intérprete y el sistema de ejecución en tiempo real (*run-time*).

Robusto.

Java fue diseñado para crear *software* altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. El compilador y cargador de clases aseguran la corrección de todas las llamadas de métodos, lo que evita las diferencias implícitas entre tipos y las incompatibilidades entre versiones. Sus características de memoria liberan a los programadores de una familia entera de errores (la aritmética de punteros), ya que se ha prescindido por completo los punteros, y la recolección de basura elimina la necesidad de liberación explícita de memoria.

Seguro.

El sistema de Java tiene ciertas políticas que evitan se puedan codificar virus con este lenguaje. Java no permite el manejo de apuntadores, y evita que un programa malicioso corrompa los espacios de memoria. En cuanto a la ejecución de programas Java integra funciones de seguridad: el verificador de código de bit, el cargador de clases y el gestor de seguridad. Esto garantiza que el código no seguro realice operaciones seguras, como leer el disco duro. Otro aspecto de la seguridad recientemente incorporado a Java tiene que ver con la autentificación, autorización y encriptación para proteger la privacidad y asegurar la integridad de los datos.

2.5.2 C++

Es uno de los lenguajes más potentes que existentes ya que permite programar a alto y a bajo nivel, es complicado porque el programador debe hacerlo casi todo. Tiene una enorme compatibilidad con el C principalmente por dos razones: Por la gran cantidad de código C que comparten, y para facilitar el paso de los programadores de C al nuevo lenguaje C++. Este lenguaje no es un lenguaje orientado a objetos puros, porque nace como una evolución de otro anterior. (19)

Características del Lenguaje C++ (20)

- Tiene un conjunto completo de instrucciones de control.
- Permite la agrupación de instrucciones.
- Incluye el concepto de puntero (variable que contiene la dirección de otra variable).
- Los argumentos de las funciones se transfieren por su valor.

- E/S no forma parte del lenguaje, sino que se proporciona a través de una biblioteca de funciones.

Desde luego, C++ es un lenguaje de programación extremadamente largo y complejo. También se puede decir que prácticamente no hay una regla sin su correspondiente excepción. Cuando se aprende que algo no se puede hacer, hay siempre algún truco escondido para hacerlo. (19)

Características del Lenguaje C++ (20)

- Tiene un conjunto completo de instrucciones de control.
- Permite la agrupación de instrucciones.
- Incluye el concepto de puntero (variable que contiene la dirección de otra variable).
- Los argumentos de las funciones se transfieren por su valor.
- E/S no forma parte del lenguaje, sino que se proporciona a través de una biblioteca de funciones.

Desde luego, C++ es un lenguaje de programación extremadamente largo y complejo. También se puede decir que prácticamente no hay una regla sin su correspondiente excepción. Cuando se aprende que algo no se puede hacer, hay siempre algún truco escondido para hacerlo.

A pesar de todo, ha experimentado un extraordinario éxito desde su creación. De hecho, muchos sistemas operativos, compiladores e intérpretes han sido escritos en C++ (el propio Windows y Java). Una de las razones de su éxito es ser un lenguaje de propósito general que se adapta a múltiples situaciones. (21)

2.5.3 C#

Es un lenguaje de programación orientado a objetos que quiso mejorar con respecto de los dos lenguajes anteriores de los que deriva el C, y el C++.Con el C# se pretendió que incorporase las ventajas o mejoras que tiene el lenguaje JAVA. Así se consiguió que tuviese las ventajas del C, del C++, pero además la productividad que posee el lenguaje JAVA y se le denomino C#.
Se puede utilizar este lenguaje para crear aplicaciones cliente para Windows tradicionales, servicios Web XML, componentes distribuidos, aplicaciones cliente-servidor, aplicaciones de bases de datos, y muchas tareas más. La sintaxis de C# es muy expresiva, aunque cuenta con menos de 90 palabras claves; también es sencilla y fácil de aprender. Está basada en signos de llave, podrá ser reconocida inmediatamente por cualquier persona familiarizada con C, C++ o Java.

Algunas de las características del lenguaje de programación C# son:

- Su código se puede tratar íntegramente como un objeto. Su sintaxis es muy similar a la del JAVA. Es un lenguaje orientado a objetos y a componentes.
- Armoniza la productividad del *Visual Basic* con el poder y la flexibilidad del C++.
- Ahorramos tiempo en la programación ya que tiene una librería de clases muy completa y bien diseñada.

El proceso de generación de C# es simple en comparación con el de C y C++, y es más flexible que en Java. No hay archivos de encabezado independientes, ni se requiere que los métodos y los tipos se declaren en un orden determinado. Un archivo de código fuente de C# puede definir cualquier número de clases, estructuras, interfaces y eventos. (22)

2.5.4 Selección del lenguaje de programación a utilizar

Teniendo en cuenta el estudio realizado sobre los lenguajes de programación expuestos anteriormente, se llega a la conclusión de que el lenguaje a utilizar para la implementación del sistema es Java. Esto se debe fundamentalmente a que el lenguaje se ajusta perfectamente a los requisitos funcionales que se desean implementar y el equipo de desarrollo posee un amplio dominio del mismo. Además, es un lenguaje libre, gratuito y multiplataforma lo bastante potente para desarrollar aplicaciones en cualquier ámbito. Prácticamente todo aquello que se puede hacer en cualquier lenguaje se puede hacer también en Java y muchas veces con grandes ventajas.
Es un lenguaje dinámico donde la máquina virtual de java, enlaza los programas java en tiempo de ejecución, eliminando la necesidad de enlazarlos con las librerías en tiempo de compilación. Todo esto, unido a la portabilidad, potencia, seguridad y estabilidad que presenta lo ha hecho convertirse en un lenguaje utilizado por millones de programadores de todo el mundo.

2.6 Entornos de desarrollo

En la actualidad existen distintos programas que permiten desarrollar el código Java, dada a la distribución gratuita del *Java(tm) Development Kit* (JDK), el cual lleva implícito un conjunto de programas y librerías que permiten desarrollar, compilar y ejecutar programas en Java. Incorpora además la posibilidad de ejecutar parcialmente el programa, deteniendo la ejecución en el punto deseado y estudiando en cada momento el valor de cada una de las variables (con el denominado *Debugger*). Cuenta además con una versión reducida del JDK, denominada JRE (*Java Runtime Environment*) destinada únicamente a ejecutar código Java. (23)

Un IDE (Entorno de Desarrollo Integrado), tal y como su nombre indica, es un entorno de desarrollo

integrado. En un mismo programa es posible escribir el código Java, compilarlo y ejecutarlo sin tener que cambiar de aplicación. Estos entornos integrados permiten desarrollar las aplicaciones de forma mucho más rápida, incorporando en muchos casos librerías con componentes ya desarrollados, los cuales se incorporan al proyecto o programa, proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación como C++, Java, C#, *Delphi*, *Visual Basic*, *Object* Pascal, entre otros. (23)

2.6.1 NetBeans

NetBeans IDE es un entorno de desarrollo visual para aplicaciones programadas a partir del uso del lenguaje de programación Java, de modo que puede ejecutarse en cualquier ambiente que ejecute Java, es uno de los lenguajes de programación más poderosos del momento. Las amplias posibilidades de desarrollo multiplataforma de NetBeans atraen a muchos desarrolladores que han trabajado con otras herramientas. Posee numerosas características que hacen que el IDE sea atractivo para cualquier desarrollador, incluyendo la amplia integración de las características específicas de la tecnología Java que no se encuentran disponibles en otros conjuntos de herramientas de aplicaciones multiplataforma.

Es un producto de código abierto, con todos los beneficios del *software* disponible en forma gratuita, el cual ha sido examinado por una comunidad de desarrolladores. Es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. (24)

2.6.2 Eclipse

Eclipse es un entorno de desarrollo integrado multiplataforma (utilizado para los lenguajes C, C++, *Python* y Java entre otros) de código abierto, utilizado en su mayoría para desarrollar otros entornos de desarrollo (como el JDT).Fue originalmente desarrollado por IBM para pasar a la familia de herramientas *VisualAge* que la marca poseía. Sin embargo en la actualidad esta herramienta está siendo desarrollada por la Fundación Eclipse, una organización independiente sin ánimo de lucro que intenta fomentar una comunidad de código abierto así como un conjunto de productos complementarios, capacidades y servicios.

La versión existente en la actualidad de esta herramienta ofrece las siguientes características: editor de texto, resaltado de sintaxis, compilación en tiempo real, pruebas unitarias con JUnit, control de versiones con CVS, integración con Ant y asistentes para el inicio en algunos de los elementos soportados (como proyectos, clases y test). Además, a través del uso de plugins se pueden utilizar tanto *Subversion* para realizar el control de versiones como *Hibernate* para desarrollar las funciones

de un motor de persistencia. (25)

La Plataforma Eclipse está diseñada para afrontar las siguientes necesidades:

- Soportar la construcción de gran variedad de herramientas de desarrollo.
- Soportar las herramientas proporcionadas por diferentes fabricantes de *software* independientes.
- Soportar herramientas que permitan manipular diferentes contenidos (HTML, Java, C, JSP, EJB, XML, y GIF).
- Facilitar una integración transparente entre todas las herramientas y tipos de contenidos sin tener en cuenta al proveedor.
- Proporcionar entornos de desarrollo gráfico (GUI) o no gráficos.
- Ejecutarse en una gran variedad de sistemas operativos, incluyendo Windows® y Linux™.
- Hacer hincapié en que el lenguaje de programación sea Java para la construcción de nuevos plugins.

2.6.3 Selección del entorno de desarrollo a utilizar

Para el desarrollo del sistema se utilizará NetBeans IDE versión 7.0 ya que es un producto sin restricciones de uso, completo y modular. Facilita la construcción de aplicaciones completas para el cliente, además de permitir la creación de ventanas, menús, barras de herramientas y acciones de una forma más amena. Aunque Eclipse es un IDE que cumple con lo expuesto anteriormente se puede decir que NetBeans completa el código de forma más inteligente y resaltado; al igual que brinda mayores facilidades para el diseño de aplicaciones de escritorio.

2.7 Framework a utilizar: Swing

Swing es un *framework* de interfaz gráfica para Java. Incluye componentes para interfaz gráfica de usuario tales como cajas de texto, botones, desplegables y tablas. Swing introdujo un mecanismo que permitía que el aspecto de cada componente de una aplicación pudiese cambiar sin introducir cambios sustanciales en el código de la aplicación. La introducción de soporte ensamblable para el aspecto permitió a Swing emular la apariencia de los componentes nativos manteniendo las ventajas de la independencia de la plataforma. También contiene un conjunto de herramientas que permiten crear un interfaz atractivo para los usuarios.

Ventajas

- El diseño en Java puro posee menos limitaciones de plataforma.

- El desarrollo de componentes Swing es más activo.
- Los componentes de Swing soportan más características.

2.8 Conclusiones del Capítulo

En este capítulo se realizó un estudio profundo de las herramientas, la arquitectura, el lenguaje y la metodología que se utilizaron para darle cumplimiento al desarrollo de la aplicación. Se adopta el uso del lenguaje de programación Java para desarrollar las funcionalidades de la herramienta, seleccionando NetBeans como Entorno de Desarrollo. Como Arquitectura se utiliza **MVC** y no **MVC (1)** ya que se puede acceder directamente a los datos tanto desde la vista como desde las clases controladoras. Es más flexible utilizar como metodología XP por los diferentes cambios que pueden ocurrir en el desarrollo de las funcionalidades del sistema, además de tener buena interacción y proximidad con el cliente.

Capítulo 3

Presentación de la Solución Propuesta

3.1 Introducción

En el presente capítulo se realiza una caracterización de las dos primeras fases del ciclo de vida de la metodología XP (Programación Extrema), exploración y planificación. Se plasman las historias escritas por los usuarios para lograr una mejor comprensión y entendimiento acerca del *software*. Las historias de usuarios son escritas por el cliente reflejando las necesidades del sistema.

3.2 Fase de Exploración

La primera fase de la metodología de desarrollo XP es exploración, en esta fase los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Asimismo el equipo de trabajo se familiariza con la tecnología, las herramientas y practicas a utilizar en el proyecto. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología. En esta fase, el cliente define lo que necesita mediante la redacción de sencillas “historias de usuarios”. Los programadores estiman los tiempos de desarrollo en base a esta información. Debe quedar claro que las estimaciones realizadas en esta fase son primarias, ya que estarán basadas en datos de muy alto nivel, y podrían variar cuando se analicen más en detalle en cada iteración. (26)

3.2.1 Historias de Usuarios

Las Historias de Usuarios son unos de los artefactos más importantes que genera la metodología de desarrollo ágil XP. Estas tienen el mismo propósito que los casos de uso y son escritas por el propio cliente, tal y como ven ellos las necesidades del sistema, por tanto son descripciones cortas y escritas en el lenguaje del usuario, sin terminología técnica, se realiza una por cada funcionalidad del sistema, se emplean para hacer estimaciones de tiempo y para el plan de lanzamientos, reemplazan un gran documento de requisitos y presiden la creación de las pruebas de aceptación. Cuando llega la hora de implementar una historia de usuario, el cliente y los desarrolladores se reúnen para concretar y detallar lo que tiene que hacer dicha historia. El tiempo de desarrollo ideal para una historia de usuario es entre 1 y 3 semanas, estas permiten responder rápidamente a los requerimientos cambiantes. (26)

3.2.2 Relación de las Historias de Usuarios

Como parte del proceso de trabajo dentro de la fase de exploración se muestran a continuación las HU “Cargar Plataforma” y “Crear Proyecto”. Las demás HU se detallan en los anexos.

Tabla # 1 HU Cargar Plataforma.

Historia de Usuario	
Número: 1.	**Nombre de historia:** Cargar Plataforma.
Usuario: Cliente.	
Prioridad en el negocio: Alta.	**Riesgo en Desarrollo:** Medio.
Puntos estimados: 1.	**Iteración asignada:** 1.
Programador responsable: Ariel Labrada Delgado**.**	
Descripción: Permite al usuario cargar la plataforma GeneSIG que sirve como base para el desarrollo de las tareas que se llevan a cabo en la aplicación. Para esto se da la posibilidad de eliminar y adicionar direcciones a cargar de la Plataforma.	
Observaciones:	

Tabla # 2 HU Crear Proyecto.

Historia de Usuario

Número: 2.	**Nombre de historia:** Crear Proyecto.
Usuario: Cliente**.**	
Prioridad en el negocio: Alta.	**Riesgo en Desarrollo:** Alto.
Puntos estimados: 1**.**	**Iteración asignada:** 1.
Programador responsable: Ariel Labrada Delgado y Raidel Páez LLopiz.	
Descripción: Crear Proyecto permite al usuario crear un proyecto nuevo. El usuario tiene la opción de crear un proyecto en blanco, un proyecto copia o un proyecto personalizado a partir de un proyecto existente en la Plataforma.	
Observaciones: La aplicación debe tener cargada la plataforma GeneSIG para crear proyectos a partir de los proyectos que esta contenga.	

3.3 Fase de Planificación

La fase de planificación lleva a cabo una estimación del esfuerzo que costará implementar cada una de las Historias de Usuarios. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Como unidad de medida se utilizará el punto de estimación, considerando que una semana de trabajo sin interrupciones es equivalente a un punto.

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses.

Esta fase dura unos pocos días. Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la “velocidad” de desarrollo, establecida en puntos por

iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración.

La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación. (26)

3.3.1 Estimación de esfuerzos

Para lograr un desarrollo eficiente y satisfactorio, se realizó una estimación de esfuerzos para cada una de las Historias de Usuarios identificadas en el proceso de planificación, llegando a los resultados que se muestran a continuación.

Tabla # 28 Estimación de Esfuerzos.

Historias de Usuarios	Puntos de Estimación
Cargar Plataforma.	1 semana.
Crear Proyecto.	1 semana.
Eliminar Proyecto	½ semana.
Modificar Proyecto.	½ semana.
Listar Proyectos.	½ semana.
Exportar Proyecto.	½ semana.
Importar Proyecto	½ semana.

Listar Conexiones.	½ semana.
Crear Conexión.	½ semana.
Eliminar Conexión.	½ semana.
Modificar Conexión.	½ semana.
Crear Plugin.	½ semana.
Modificar Plugin.	½ semana.
Eliminar Plugin.	½ semana.
Listar Plugin.	½ semana.
Exportar Plugin	½ semana.
Importar Plugin	½ semana.
Crear Capa.	½ semana.
Listar Capa.	½ semana.
Eliminar Capa.	½ semana.
Modificar Capa.	½ semana.
Exportar Capa.	½ semana.
Eliminar Capa.	½ semana.

Reordenar Capa.	½ semana.
Modificar Mapa.	1 semana.
Limpiar Caché.	1 semana.
Borrar Mapa.	½ semana.

3.3.2 Plan de Iteraciones

Esta fase incluye varias iteraciones sobre el sistema donde el Plan de entrega de las iteraciones no excederá de tres semanas. Una vez descritas e identificadas las Historias de Usuarios y estimado el esfuerzo propuesto para la realización de cada una de ellas, se procede a la planificación de la etapa de implementación del sistema. En éste se especifica cuáles son las HU que van a ser implementadas para cada iteración y la determinación de las posibles fechas de entrega. Después de haber definido las HU y estimado el esfuerzo propuesto para la realización de cada una de ellas, se tomó la decisión de realizar el sistema en 6 iteraciones, las cuales se detallan a continuación:

Iteración # 1:

La primera iteración tiene como objetivo la implementación de las HU #1, #2, #3 y #4 con prioridad para el cliente de Alta, Alta, Baja y Alta respectivamente. En esta fase la HU Cargar Plataforma es la base para implementar las demás iteraciones. Se dispone de 2 semanas para implementar todas las tareas. Se obtiene como resultado una primera versión del sistema propuesto la cual será mostrada al cliente para desarrollar la retroalimentación con el equipo de desarrollo, para luego pasar a la siguiente iteración.

Iteración # 2:

En esta iteración se darán cumplimiento a las HU #5, #6 y #7 las cuales hacen alusión a todo lo referente con importar, exportar y listar los proyectos existentes. Se cuenta con 2 semanas para llevar a cabo la implementación de esta iteración. Al finalizar se obtendrá una segunda versión del sistema propuesto y se le hará llegar al cliente la iteración anterior junto con la presente para la aprobación o cambios pertinentes con el cliente.

Iteración # 3:

La tercera iteración hace alusión a las HU #8, #9, #10, #11 las cuales representan todas las tareas asociadas con el origen de datos que posee cada proyecto. Se realizarán las tareas de crear, listar, modificar y eliminar conexiones. Finalizada la implementación se obtendrá una versión con un 40% de la implementación del sistema, los cuales serán aprobados por el cliente para comenzar con la próxima iteración.

Iteración # 4:

La presente iteración corresponde a las HU #12, #13, #14, #15, #16 y #17 las cuales representan las tareas asociadas a la gestión de plugins (Crear, Modificar, Eliminar, Listar, Exportar e Importar).En la HU Modificar Plugins se realizará la configuración de los ficheros de configuración de los plugins de un proyecto, correspondiente a unas de las tareas más importantes en el negocio de la aplicación. El resultado obtenido será entregado al cliente para su verificación y aprobación. Luego de ser aprobado se pasará a la próxima iteración.

Iteración # 5:

La quinta iteración corresponde a las HU #18, #19, #20, #21, #22, y #23 en esta iteración se desarrollan las tareas Crear, Listar, Modificar, Exportar, Importar y Eliminar las capas de un proyecto seleccionado. Como resultado se podrá mostrar todas las capas que posee un SIG, así como poder añadir o eliminar algunas de las existentes. También se dará la posibilidad de exportar e importar capas existentes a un directorio específico o a otro proyecto de la plataforma. Luego de ser aprobada por el cliente esta iteración se pasará a la última iteración.

Iteración # 6:

La iteración # 6 y última corresponde a las HU #24, # 25, #26 y #27. Las cuales tienen como objetivo Modificar y Reordenar las capas de un SIG. Además se incluyen funcionalidades como Limpiar Caché y Borrar Mapa, en el caso de Limpiar Caché se encarga de limpiar la caché negra del sistema para el trabajo activo con las aplicaciones a desarrollar. En esta iteración se completará el 100% de las funcionalidades del sistema propuesto para ser probada por el cliente en su totalidad.

Las iteraciones a desarrollar presentan un nivel medio y alto de complejidad para el desarrollo con alta prioridad para el cliente. Se dispondrán de 16 semanas para desarrollar todas las iteraciones propuestas.

3.3.3 Plan de duración de las iteraciones

Para lograr una mayor organización del trabajo se crea un plan de duración de las iteraciones; el mismo tiene como objetivo mostrar la duración de cada iteración así como el orden en que serán implementadas las historias de usuarios en cada una de ellas. Este plan tiene como finalidad mostrar la duración de cada iteración, así como el orden en que serán implementadas las HU en cada una de ellas.

Tabla # 29 Plan de duración de Iteraciones.

No Iteración	Historias de Usuario	Duración Total de Iteraciones
Iteración # 1	Cargar Plataforma.	3 semanas.
	Crear Proyecto.	
	Eliminar Proyecto.	
	Modificar Proyecto.	
Iteración # 2	Importar proyecto.	2 semanas.
	Listar Proyectos.	
	Exportar Proyecto.	
Iteración # 3	Listar Conexiones.	2 semanas.
	Crear Conexión.	
	Eliminar Conexión.	
	Modificar Conexión.	
Iteración # 4	Crear Plugin.	3 semanas.
	Modificar Plugin.	
	Eliminar Plugin.	

	Listar Plugin.	
	Exportar Plugin	
	Importar Plugin	
Iteración # 5	Crear Capa.	3 semanas.
	Listar Capa.	
	Eliminar Capa.	
	Modificar Capa.	
	Exportar Capa.	
	Importar Capa.	
Iteración # 6	Modificar Mapa	3 semanas.
	Reordenar Capa.	
	Limpiar Caché.	
	Borrar Mapa.	

3.3.4 Plan de Entrega

A continuación se muestra el plan de entregas desarrollado para dar solución al problema planteado. Para desarrollar el mismo se tuvo en cuenta los puntos de estimación para obtener un resultado final.

Tabla # 30 Plan de Entrega.

Nro. Iteración	Duración (Iter.)	Fecha Inicio.	Fecha Final.
Iteración # 1	3 semanas.	1-2-2012.	21-2-2012.

Iteración # 2	2 semanas.	22-2-2012.	6-3-2012.
Iteración # 3	2 semanas.	7-3-2012.	20-3-2012.
Iteración # 4	3 semanas.	21-3-2012	13-4-2012.
Iteración # 5	3 semanas.	14-4-2012	5-5-2012
Iteración # 6	3 semanas.	6-5-2012	27-5-2012.

3.4 Conclusiones Parciales

En el presente capítulo se describió la propuesta de solución para la creación de Sistemas de Información Geográfica a partir de la plataforma GeneSIG. Se abordó la información correspondiente a la fase de exploración y planificación generando así la mayor cantidad de artefactos posibles. La fase de exploración permitió representar brevemente el comportamiento del sistema, donde se llegó a una correcta comunicación entre el cliente y el equipo de desarrollo. En la fase de Planificación se determinó que se realizarán 6 iteraciones con un total de duración de 16 semanas. Todo esto se desarrolla para lograr el sistema deseado y poder satisfacer la necesidad del cliente.

Capítulo 4

Construcción y validación de la solución propuesta

4.1 Introducción

El contenido de este capítulo abarca una valoración de las fases de construcción y prueba de la metodología XP. Se describen las tarjetas CRC (Contenido, Responsabilidad y Colaboración) para un mejor entendimiento del sistema y se detallan las seis iteraciones llevadas a cabo durante la etapa de construcción de la aplicación, exponiéndose las tareas de programación o ingeniería generadas por cada historia de usuario y las pruebas de aceptación efectuadas sobre el sistema.

4.2 Diseño de la solución propuesta

La Metodología XP plantea que la implementación de un *software* debe realizarse de forma iterativa, obteniendo al culminar cada iteración un producto funcional que debe ser probado y mostrado al cliente para incrementar la visión de los desarrolladores con la opinión de éste. Esta es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo del *software*, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo.

En el diseño de las aplicaciones realizadas bajo las reglas de la metodología XP, no es requerida la representación del sistema mediante diagramas de clase utilizando notación UML, sino que se utilizan otras técnicas, como las llamadas tarjetas CRC. Sin embargo la utilización de estos diagramas puede aplicarse siempre y cuando influya en el mejoramiento de la comunicación entre el equipo de desarrollo y se enfoquen en la información importante. (26)

4.2.1 Tarjetas CRC

Las tarjetas CRC permiten desprenderse del método de trabajo basado en procedimientos y trabajar con una metodología basada en objetos. Además permiten que el equipo completo contribuya en la tarea del diseño. (26)

El factor decisivo para utilizar esta técnica para diseñar la aplicación que se desea desarrollar fueron las características que esta brinda en cuanto a facilidad de su uso y entendimiento. A continuación se

muestran las tarjetas CRC correspondientes a las entidades “Plataforma” y “CrearProyecto”. Las demás tarjetas CRC se detallan en los anexos.

Tabla #31: Plantilla para Tarjetas CRC.

Tarjeta CRC	
Clase: Nombre de la clase que se está modelando	
Responsabilidades: Es una descripción de alto nivel del propósito de la clase.	**Colaboraciones:** Indica con cuales otras clases se requiere relación para cumplir la responsabilidad

Tabla #32: Plataforma.

Tarjeta CRC	
Clase: Plataforma	
Responsabilidades:	**Colaboraciones:**
getConfiguracionServer	FicheroInI
llenarProyectos	Configuracion
getListaPlugin	Error
crearProyecto	Proyecto
getListaNomProyectos	Plugin
proyectosValidosExistentes	CrearProyecto
proyectoRepetido	
getProyectoXNombre	
cargarErroresGenesig	
actualizarMapas	

Tabla #33: CrearProyecto.

Tarjeta CRC	
Clase: CrearProyecto	
Responsabilidades: proyectoEnBlanco proyectoCopia proyectoPersonalizado	**Colaboraciones:**

4.3 Desarrollo de las iteraciones

En la segunda fase de la metodología XP, Planificación, se describieron las HU correspondientes a cada iteración a desarrollar, teniendo en cuenta las necesidades requeridas por el cliente. Durante el transcurso de las iteraciones se lleva a cabo una revisión del plan de iteraciones y se modifica en caso de ser necesario. Como parte de este plan, se descomponen las HU en tareas de programación o ingeniería, las mismas pueden ser escritas en lenguaje técnico debido a que son para uso estricto de los programadores, donde el usuario no tiene que necesariamente comprenderlas. (26)

De acuerdo a la planificación realizada se llevó a cabo el desarrollo del sistema en seis iteraciones, con el fin de obtener una aplicación con todas las condiciones y características propuestas por el cliente. A continuación se definen cada una de las iteraciones.

4.3.1 Iteración 1:

Esta iteración tiene como objetivo darle cumplimiento a las HU que se consideraron de mayor importancia para el desarrollo del *software*. Al concluir dicha iteración se contará con todas las funcionalidades descritas en las HU #1, #2, #3 y #4.

Tabla #53: HU abordadas en la primera iteración.

Historias de Usuario	Tiempo de implementación (semanas)	
	Estimación	**Real**

Cargar Plataforma	1	1
Crear Proyecto	1	1
Eliminar Proyecto	1 2	1 2
Modificar Proyecto	1 2	1 2

A continuación mediante tablas se evidencian las tareas de programación o ingeniería en las que las Historias de Usuario mencionada anteriormente fue desglosada, para un mejor funcionamiento de la aplicación.

Tabla #54: Tarea #1 de la HU #1.

Tareas de la Ingeniería	
No. de la tarea: 1.	**No. de la HU:** 1.
Nombre de la tarea: Adicionar dirección.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1 2
Fecha inicio: 1-2-2012.	**Fecha fin:** 4-2-2012.
Programador responsable: Ariel Labrada Delgado.	
Descripción: Adiciona una dirección que corresponde a una plataforma diferente.	

Las demás tareas de Ingeniería correspondientes a la primera iteración se detallan en los anexos.

4.3.2 Iteración 2:

Esta iteración tiene como finalidad desarrollar las HU #5, #6 y #7.

Tabla #63: HU abordadas en la segunda iteración.

Historias de Usuario	Tiempo de implementación (semanas)	
	Estimación	**Real**
Listar Proyectos	1	1
Exportar proyecto	$1\frac{1}{2}$	$1\frac{1}{2}$
Importar Proyecto	$1\frac{1}{2}$	$1\frac{1}{2}$

Tabla #64: Tarea #1 de la HU #5.

Tareas de la Ingeniería	
No. de la tarea: 1.	**No. de la HU:** 5.
Nombre de la tarea: Listar proyectos en vista física.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** $1\frac{1}{2}$
Fecha inicio: 22-2-2012.	**Fecha fin:** 24-2-2012.
Programador responsable: Ariel Labrada Delgado.	
Descripción: Muestra en la interfaz principal la lista de todos los proyectos pertenecientes a la plataforma cargada de forma física.	

Las demás tareas de Ingeniería correspondientes a la segunda iteración se detallan en los anexos.

4.3.3 Iteración 3:

Esta iteración tiene como finalidad desarrollar las HU #8, #9, #10 y #11.

Tabla #68: HU abordadas en la tercera iteración.

Historias de Usuario	Tiempo de implementación (semanas)	
	Estimación	**Real**
Listar Conexiones	1 2	1 2
Crear Conexión	1 2	1 2
Eliminar Conexión	1 2	1 2
Modificar Conexión	1 2	1 2

Tabla #69: Tarea #1 de la HU #8.

Tareas de la Ingeniería	
No. de la tarea: 1.	**No. de la HU:** 8.
Nombre de la tarea: Listar Conexiones.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1 2
Fecha inicio: 7-3-2012.	**Fecha fin:** 9-3-2012.
Programador responsable: Ariel Labrada Delgado.	
Descripción: Muestra las conexiones que posee el proyecto seleccionado al origen de los datos.	

Las demás tareas de Ingeniería correspondientes a la tercera iteración se detallan en los anexos.

4.3.4 Iteración 4:

Esta iteración tiene como finalidad desarrollar las HU #12, #13, #14, #15, #16 y #17.

Tabla #73: HU abordadas en la cuarta iteración.

Historias de Usuario	Tiempo de implementación (semanas)	
	Estimación	**Real**
Crear Plugin	1 2	1 2
Modificar Plugin	1 2	1 2
Eliminar Plugin	1 2	1 2
Listar Plugin	1 2	1 2
Exportar Plugin	1 2	1 2
Importar Plugin	1 2	1 2

Tabla #74: Tarea #1 de la HU #12.

Tareas de la Ingeniería	
No. de la tarea: 1.	**No. de la HU:** 12.
Nombre de la tarea: Crear Plugin.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1 2
Fecha inicio: 21-3-2012.	**Fecha fin:** 23-3-2012.

Programador responsable: Ariel Labrada Delgado.
Descripción: Permite al usuario crear un plugin a un proyecto en específico. Al mismo se le define si tendrá servidor o no, se le definen las clases *Request* y *Result* así como los atributos que contendrá cada una en particular.

Las demás tareas de Ingeniería correspondientes a la cuarta iteración se detallan en los anexos.

4.3.5 Iteración 5:

Esta iteración tiene como finalidad desarrollar las HU #18, #19, #20, #21, #22 y #23.

Tabla #80: HU abordadas en la quinta iteración.

Historias de Usuario	Tiempo de implementación (semanas)	
	Estimación	**Real**
Crear Capa	1 2	1 2
Listar Capa	1 2	1 2
Eliminar Capa	1 2	1 2
Modificar Capa	1 2	1 2
Exportar Capa	1 2	1 2
Importar Capa	1 2	1 2

Tabla #81: Tarea #1 de la HU #18.

Tareas de la Ingeniería	
No. de la tarea: 1.	**No. de la HU:** 18.
Nombre de la tarea: Crear Capa.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1 2
Fecha inicio: 11-4-2012.	**Fecha fin:** 13-4-2012.
Programador responsable: Raidel Paez LLopiz.	
Descripción: Permite al usuario adicionar una capa a un proyecto seleccionado.	

Las demás tareas de Ingeniería correspondientes a la quinta iteración se detallan en los anexos.

4.3.6 Iteración 6:

Esta iteración tiene como finalidad desarrollar las HU #18, #19, #20, #21, #22 y #23.

Tabla #87: HU abordadas en la sexta iteración.

Historias de Usuario	Tiempo de implementación (semanas)	
	Estimación	**Real**
Reordenar Capa	1 2	1 2
Modificar Mapa	1 2	1 2
Limpiar Caché	1 2	1 2
Borrar Mapa	1 2	1 2

Tabla #88: Tarea #1 de la HU #24.

Tareas de la Ingeniería	
No. de la tarea: 1.	**No. de la HU:** 24.
Nombre de la tarea: Reordenar Capa.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1 2
Fecha inicio: 2-5-2012.	**Fecha fin:** 4-5-2012.
Programador responsable: Raidel Paez LLopiz.	
Descripción: Permite reordenar las posiciones de las capas para su visibilidad.	

Las demás tareas de Ingeniería correspondientes a la sexta iteración se detallan en los anexos.

4.4 Requisitos no funcionales del sistema

Los requisitos no funcionales son propiedades o cualidades que el producto debe cumplir. Estas propiedades son las características que hacen al producto atractivo, usable, fiable, rápido y confiable.

Usabilidad

RNF 1. El sistema tiene como usuario final los especialistas del proyecto Aplicativos SIG que desarrollen sobre la plataforma GeneSIG.

Fiabilidad

RNF 2. El sistema realiza íntegramente las funcionalidades especificadas por el cliente.

Soporte

RNF 3. Cualquier cambio ocurrido en cuanto a la estructura de la plataforma GeneSIG, es responsabilidad del administrador del sistema actualizar el sistema orientado a los cambios realizados en la plataforma.

Restricciones de diseño

RNF 4. El producto de *software* final debe diseñarse sobre una arquitectura modelo-vista-controlador.

Interfaz

RNF 5. El sistema debe mostrar de forma organizada las funcionalidades del sistema.

RNF 6. El sistema debe tener un diseño sencillo, donde los colores seleccionados sean de tonalidades claras siguiendo una misma línea.

RNF 7. Los botones utilizados en la aplicación deben tener un nombre o descripción entendible para el usuario.

Requisitos de hardware

RNF 8. La PC de trabajo debe tener como mínimo 256 de RAM y 10 GB de disco duro.

Requisitos de software

RNF 9. Sistemas operativos Ubuntu versión 9.4 en adelante.

RNF 10. Tener instalada la máquina virtual de Java JRE 1.6.

RNF 11. Tener al menos una plataforma GeneSIG.

Requisitos de Licencia

RNF 12. La herramienta, de acuerdo a los tipos de licencias de las herramientas que se utilizaron, es legalmente de modelo libre, permitiendo la utilización, modificación y distribución de las mismas por terceros sin necesidad de obtener la autorización de sus respectivos titulares.

Requisitos Legales, de Derecho de Autor y otros

RNF 13. La mayoría de las herramientas de desarrollo son libres y las licencias están avaladas.

Estándares Aplicables

RNF 14. El sistema será desarrollado bajo estándar internacional de codificación Camel.

4.5 Pruebas

La metodología XP propone como una de sus prácticas es el uso de las pruebas para garantizar el funcionamiento de los códigos que se vayan implementando. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección.

La metodología ágil XP divide las pruebas en dos grupos: pruebas unitarias y pruebas de aceptación. Las pruebas unitarias son desarrolladas por los programadores y se encargan de verificar el código automáticamente y las pruebas de aceptación están destinadas a verificar que al final de cada iteración las Historias de Usuario cumplen con la funcionalidad asignada y satisfagan las necesidades del cliente. (15)

4.5.1 Pruebas unitarias

De acuerdo con lo que plantea la metodología XP, las pruebas unitarias o pruebas de unidad consisten en comprobaciones (manuales o automatizadas) desarrolladas por los programadores. Las cuales se realizan para verificar que el código correspondiente a un módulo concreto se comporta de manera esperada. Las pruebas unitarias proporcionan beneficios tales como: (15)

- Brindan al programador una inmediata retroalimentación de cómo está realizando su trabajo.
- El programador puede realizar cambios de forma segura respaldada por efectivos casos de prueba.
- Permite saber si una determinada funcionalidad se puede agregar al sistema existente sin alterar el funcionamiento actual del mismo.

Las pruebas unitarias no se le aplicarán al *software* debido a que según los expertos en la metodología de desarrollo XP recomiendan utilizar las pruebas de aceptación. Las pruebas de aceptación son consideradas las más adecuadas, pues significa el grado de satisfacción que tenga el cliente con el producto. En estas pruebas el cliente puede comprobar si todas las historias de usuario se implementaron de acuerdo a lo concebido.

4.6 Pruebas de aceptación

Las pruebas de aceptación se enfocan en un objetivo diferente que las pruebas unitarias dado que significan la satisfacción del cliente con el producto desarrollado y el final de una iteración y el comienzo de la siguiente. Las pruebas de aceptación se elaboran a lo largo de la iteración, en paralelo con el desarrollo del sistema, y adaptándose a los cambios que el sistema sufra. Las pruebas de

aceptación son consideradas como “pruebas de caja negra”. Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. Así mismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución. Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación. (15)

A continuación se muestra el caso de prueba para la HU “Cargar Plataforma”. Los demás casos de prueba se especifican en los anexos.

Tabla #92: Prueba de aceptación para la HU “Cargar Plataforma.”

Caso de prueba de aceptación	
Código: HU1_PA1	**No. De la HU:** 1.
Nombre: Cargar Plataforma.	
Descripción: Permite al usuario cargar la plataforma GeneSIG que sirve como base para el desarrollo de las tareas que se llevan a cabo en la aplicación. Para esto se da la posibilidad de eliminar y adicionar direcciones a cargar de la Plataforma.	
Condiciones de Ejecución: El cliente debe probar que la dirección seleccionada es una plataforma válida.	
Entrada/ Pasos de ejecución: Se solicita la opción Cargar Plataforma para poder probar que la plataforma seleccionada es válida.	
Resultado Esperado: Es cargada la plataforma seleccionada, mostrando los proyectos contenidos en caso de que contenga.	
Evaluación de la Prueba: Prueba satisfactoria.	

Se realizó una prueba piloto donde se tomó una muestra de 10 personas para ver el correcto funcionamiento del sistema. Los parámetros a seguir fueron:

1) Funcionalidad Cargar Plataforma.
2) Funcionalidad Crear Proyecto.

3) Funcionalidad Eliminar Proyecto.
4) Funcionalidad Listar Proyecto.
5) Funcionalidad Exportar Proyecto.
6) Funcionalidad Importar Proyecto.
7) Funcionalidad Listar Conexiones.
8) Funcionalidad Crear Conexión.
9) Funcionalidad Eliminar Conexión.
10) Funcionalidad Modificar Conexión.
11) Funcionalidad Crear Plugin.
12) Funcionalidad Modificar Plugin.
13) Funcionalidad Eliminar Plugin.
14) Funcionalidad Listar Plugin.
15) Funcionalidad Exportar Plugin.
16) Funcionalidad Importar Plugin.
17) Funcionalidad Crear Capa.
18) Funcionalidad Listar Capa.
19) Funcionalidad Eliminar Capa.
20) Funcionalidad Modificar Capa.
21) Funcionalidad Exportar Capa.
22) Funcionalidad Importar Capa.
23) Funcionalidad Reordenar Capa.
24) Funcionalidad Modificar Mapa.
25) Funcionalidad Limpiar Cache.
26) Funcionalidad Borrar Mapas de Usuario.

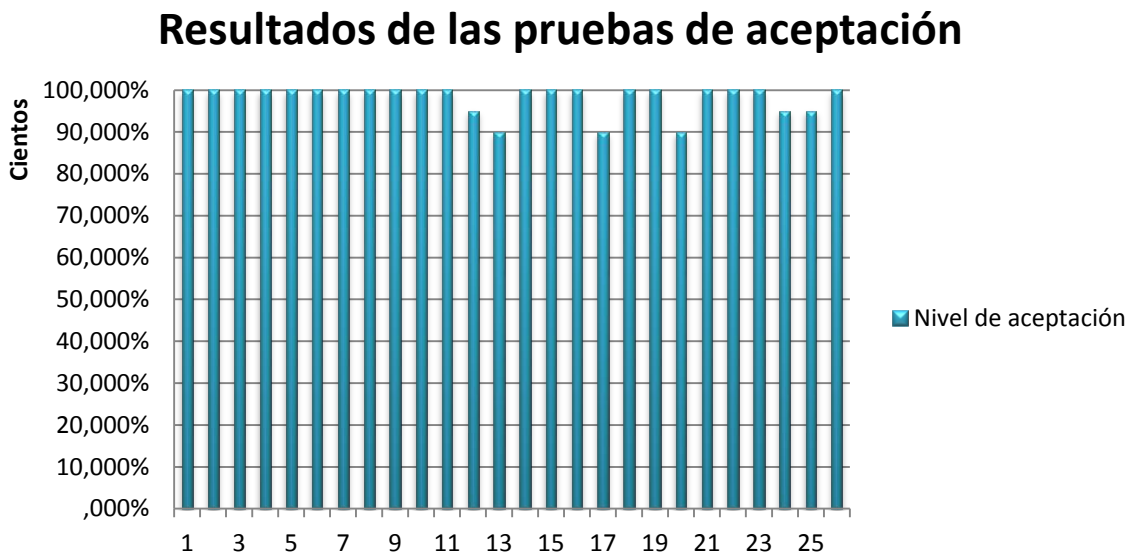


Figura #1: Resultado de las pruebas de aceptación.

Estos resultados arrojaron un nivel promedio de satisfacción del cliente, de un 98,3% evidenciando que los resultados esperados de cada funcionalidad son completamente fiables.

4.7 Conclusiones Parciales

En este capítulo se realizó la construcción y validación de una versión de la solución propuesta que corresponde a un 100% de la aplicación final. Se abordaron las fases de construcción y prueba de la metodología de desarrollo del *software* XP. Se describieron las tarjetas CRC para lograr mejor entendimiento del sistema. Se detallaron las seis iteraciones llevadas a cabo en la etapa de construcción de la aplicación, así como las tareas de programación o ingeniería generadas por cada historia de usuario. Se realizó un profundo estudio sobre las pruebas unitarias y de aceptación, seleccionando estas últimas como las más indicadas para comprobar el funcionamiento del *software*, debido a que demuestran la satisfacción del cliente con el producto.

Conclusiones Generales

En este punto se consideran cumplidos los objetivos trazados al tener desarrollado una herramienta para la creación de Sistemas de Información Geográfica. La misma cumple todos los requisitos planteados para su desarrollo por lo que se convierte en una herramienta cuyas funcionalidades básicas muestran los resultados esperados.

Se creó además una documentación técnica relacionada con la herramienta, en la misma se detallan todos los artefactos generados a partir de la aplicación de la Metodología de Desarrollo de *Software* XP.

Este trabajo representa un aporte importante al Centro de Desarrollo de Geoinformática y Señales Digitales (GEySED) de la UCI y en especial al proyecto Aplicativos SIG, ya que permite crear y personalizar Sistemas de Información Geográfica de una forma más amena. Logrando así elevar la eficiencia y calidad del trabajo en el proyecto.

Recomendaciones

Una vez concluida la investigación y basándose en las experiencias acumuladas a lo largo del desarrollo de la misma, se proponen las siguientes recomendaciones:

- Implementar un módulo que permita diseñar gráficamente la interfaz de un Sistema de Información Geográfica.
- Implementar un módulo para visualizar los mapas de los proyectos.

Referencias Bibliográficas

1. Ariza López , Pinilla Ruiz. [En línea] Abril de 2000. http://www.mappinginteractivo.com/plantilla-ante.asp?id_articulo=341.

2. **Fallas, Jorge.** *SISTEMAS DE INFORMACIÓN GEOGRÁFICA.* 2011.

3. **Eduin Cruz Mosquera, Yeirson santos Sandoval.** [En línea] 2011. http://www.slideshare.net/edwcrumo/plataformas-medios-de-difucin.

4. **Marcos Guglietmetti , Analia Lanzillotta.** [En línea] Septiembre de 2004. http://www.mastermagazine.info/termino/3874.php.

5. ACIMED. [En línea] 2009. http://scielo.sld.cu/scielo.php?pid=S1024-94352009001100007&script=sci_arttext.

6. GeoInfo. [En línea] 3 de 4 de 2012. http://www.geoinfo-int.com/htmls/sig.html.

7. [En línea] 2010. http://www.zend.com/en/.

8. **Josep Casanovas.** [En línea] 6 de 2 de 2009. http://www.desarrolloweb.com/articulos/1622.php.

9. UNADECODIGO.El Paradigma Modelo Vista Controlador. [En línea] http://www.unadecodigo.com/2007/05/30/el-paradigma-modelo-vistacontrolador-tutorial-ror-ii.

10. cencomed.sld.cu. [En línea] http://cencomed.sld.cu/socbio2007/trabajos/pdf/t072.pdf.

11. Patrón Modelo , Vista,Controlador. [En línea] http://www.proactiva-calidad.com/java/patrones/mvc.html.

12. **Erly Delgado Expósito.** [En línea] http://www.monografias.com/trabajos60/metodologias-desarrollo-software/metodologias-desarrollo-software.shtml.

13. **José H. Canós , Patricio Letelier, Ma. Carmen Penadés.** Metodologías ágiles en el desarrollo del software. [En línea] http://www.willydev.net/descargas/prev/TodoAgil.Pdf.

14. Metodologías tradicionales vs. Metodologías ágiles. [En línea] http://www.mygnet.net/manuales/software/metodologias_tradicionales_vs_dot_metodologias_agiles..

15. **J. J. Gutiérrez,M. J. Escalona, M. Mejías,J. Torres.** SISTEMA EN PROGRAMACIÓN EXTREMA. [En línea] http://www.lsi.us.es/~javierj/investigacion_ficheros/PSISEXTREMA.pdf.

16. Lenguaje de Programación. [En línea] http://es.kioskea.net/contents/langages/langages.php.

17. Que es Java. [En línea] http://java.ciberaula.com/articulo/que_es_java.

18. Curso Introductorio de Java. [En línea] http://www.mailxmail.com/curso-java/caracteristicas-lenguaje-java-3.

19. Introducción al Lenguaje C. [En línea] http://www.articulandia.com/premium/article.php/12-03-2007El-Lenguaje-C.htm.

20. Yordan Delgado. [En línea] 28 de Junio de 2011. http://teoria-de-programacion.globered.com/categoria.asp?idcat=34.

21. Lenguaje C++. [En línea] http://www.zator.com/Cpp/E1_2.htm.

22. Introducción al Lenguaje C# y al Framework.NET. [En línea] http://msdn.microsoft.com/es-es/library.

23. **Javier García de Jalón.** Aprenda Java como si estuviera en primero,Ciudad de la Habana. [En línea] 2000. http://www.tecnun.es/asignaturas/Informat1/AyudaInf/aprendainf/Java/Java2.pdf.

24. Bienvenido a NetBeans. [En línea] http://netbeans.org/index_es.html.

25. Entornos de Desarrollo Inrtegrados. [En línea] http://petra.euitio.uniovi.es/~i1667065/HD/documentos/Entornos%20de%20Desarrollo%20Integrado.pdf.

26. **Ing.José Joskowicz.** [En línea] 2008. http://iie.fing.edu.uy/~josej/docs/XP%20-0Jose%20Joskowicz.pdf.

Bibliografía

1. Ariza López , Pinilla Ruiz. [En línea] Abril de 2000. http://www.mappinginteractivo.com/plantilla-ante.asp?id_articulo=341.

2. **Fallas, Jorge.** *SISTEMAS DE INFORMACIÓN GEOGRÁFICA.* 2011.

3. **Eduin Cruz Mosquera, Yeirson santos Sandoval.** [En línea] 2011. http://www.slideshare.net/edwcrumo/plataformas-medios-de-difucin.

4. **Marcos Guglietmetti , Analia Lanzillotta.** [En línea] Septiembre de 2004. http://www.mastermagazine.info/termino/3874.php.

5. ACIMED. [En línea] 2009. http://scielo.sld.cu/scielo.php?pid=S1024-94352009001100007&script=sci_arttext.

6. GeoInfo. [En línea] 3 de 4 de 2012. http://www.geoinfo-int.com/htmls/sig.html.

7. [En línea] 2010. http://www.zend.com/en/.

8. **Josep Casanovas.** [En línea] 6 de 2 de 2009. http://www.desarrolloweb.com/articulos/1622.php.

9. UNADECODIGO.El Paradigma Modelo Vista Controlador. [En línea] http://www.unadecodigo.com/2007/05/30/el-paradigma-modelo-vistacontrolador-tutorial-ror-ii.

10. cencomed.sld.cu. [En línea] http://cencomed.sld.cu/socbio2007/trabajos/pdf/t072.pdf.

11. Patrón Modelo , Vista,Controlador. [En línea] http://www.proactiva-calidad.com/java/patrones/mvc.html.

12. **Erly Delgado Expósito.** [En línea] http://www.monografias.com/trabajos60/metodologias-desarrollo-software/metodologias-desarrollo-software.shtml.

13. **José H. Canós , Patricio Letelier, Ma. Carmen Penadés.** Metodologías ágiles en el desarrollo del software. [En línea] http://www.willydev.net/descargas/prev/TodoAgil.Pdf.

14. Metodologías tradicionales vs. Metodologías ágiles. [En línea] http://www.mygnet.net/manuales/software/metodologias_tradicionales_vs_dot_metodologias_agiles..

15. **J. J. Gutiérrez,M. J. Escalona, M. Mejías,J. Torres.** SISTEMA EN PROGRAMACIÓN EXTREMA. [En línea] http://www.lsi.us.es/~javierj/investigacion_ficheros/PSISEXTREMA.pdf.

16. Lenguaje de Programación. [En línea] http://es.kioskea.net/contents/langages/langages.php.

17. Que es Java. [En línea] http://java.ciberaula.com/articulo/que_es_java.

18. Curso Introductorio de Java. [En línea] http://www.mailxmail.com/curso-java/caracteristicas-lenguaje-java-3.

19. Introducción al Lenguaje C. [En línea] http://www.articulandia.com/premium/article.php/12-03-2007El-Lenguaje-C.htm.

20. Yordan Delgado. [En línea] 28 de Junio de 2011. http://teoria-de-programacion.globered.com/categoria.asp?idcat=34.

21. Lenguaje C++. [En línea] http://www.zator.com/Cpp/E1_2.htm.

22. Introducción al Lenguaje C# y al Framework.NET. [En línea] http://msdn.microsoft.com/es-es/library.

23. **Javier García de Jalón.** Aprenda Java como si estuviera en primero,Ciudad de la Habana. [En línea] 2000. http://www.tecnun.es/asignaturas/Informat1/AyudaInf/aprendainf/Java/Java2.pdf.

24. Bienvenido a NetBeans. [En línea] http://netbeans.org/index_es.html.

25. Entornos de Desarrollo Inrtegrados. [En línea] http://petra.euitio.uniovi.es/~i1667065/HD/documentos/Entornos%20de%20Desarrollo%20Integrado.pdf.

26. **Ing.José Joskowicz.** [En línea] 2008. http://iie.fing.edu.uy/~josej/docs/XP%20-0Jose%20Joskowicz.pdf.

27. **Denise Pumain , Lauren Chapelon.** [En línea] 2004. http://www.hypergeo.eu/spip.php?page=auteur-list.

28. Software Architecture. [En línea] 15 de 12 de 2009. http://www.sei.cmu.edu/architecture/start/community.cfm.

29. **Ron Jeffries.** [En línea] www.xprogramming.com ,www.extremeprogramming.org.

30. **Rocha,Jorge Luis Aréchiga.** Un Enfoque al Lenguaje de Programación Java. [En línea] http://www.nexos-software.com.co/Articulo_25.htm.

31. Java. [En línea] http://arechiga.50megs.com/tpoo2/javah1.html#Ventajas.

32. arechiga.50megs.com Java. [En línea] http://arechiga.50megs.com/tpoo2/2dept/informacionjava2.html. 10. —. arechiga.50megs.com.

33. [En línea] http://macromedia-dreamweaver.programas-gratis.net/.

34. Programación en Java. [En línea] casidiablo.net/java.

35. The Java Tutorial. [En línea] http://docs.oracle.com./javase/tutorial.

36. **Bustacara, Ing.Cesar Julio.** XML Bajo la paltaforma Java. [En línea] 2006.

37. **Ing.Yoenis Pantoja, Ing.Eliani Varen.** Estrategia para implementar Sistemas de Información Geográficos petroleros. [En línea] http://vinculado.org./educacion/sistemas_de_informacion_geograficos_petroleros.html.

38. **J. J. Gutiérrez, M. J. Escalona, M. Mejías, J. Torres.** *PRUEBAS DEL SISTEMA EN PROGRAMACIÓN EXTREMA.*

39. MapInfo. Guía de Usuario MapInfo Professional. [En línea] www.scribd.com/doc/.../Map-Info-75..

40. CartoWeb. [En línea] http://www.cartoweb.org/doc_head/docbook/xhtml/dev.newplugin.html ..

41. Camptocamp SA. *CartoWeb.* [En línea] 2008. http://cartoweb.org/..

42. **Yoenis Pantoja Zaldivar, Lidisy HernÃ¡ndez Montero.** *Documento Vision v1.0.* 2010.

Glosario de términos

Bytecodes: Es un código intermedio más abstracto que el código máquina. Habitualmente es tratado como un fichero binario que contiene un programa ejecutable similar a un módulo objeto.

JRE: *Java Runtime Environment*, es un conjunto de utilidades que permite la ejecución de programas Java.

Framework: Es un esquema, esqueleto o patrón para el desarrollo y/o la implementación de una aplicación.

HTML: Es un lenguaje de programación que se utiliza para el desarrollo de páginas de Internet.

Multiplataforma: Sistema informático que corre sobre varios sistemas operativos, sin prescindir de ninguna de sus funcionalidades

Java Development Kit o (JDK): Es un *software* que provee herramientas de desarrollo para la creación de programas en Java. Puede instalarse en una computadora local o en una unidad de red.

***VisualAge*:** Familia de entornos informáticos de desarrollo integrado de IBM, que incluye soporte para múltiples lenguajes de programación.

***JUnit*:** Conjunto de clases (*framework*) que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera.

***Subversion*:** Sistema de control de versiones diseñado para acceder a repositorios a través de redes, permite ser usado por personas que se encuentran en distintas computadoras para modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones.

Apache Ant: Biblioteca de Java y una herramienta de línea de comandos cuya misión es impulsar los procesos descritos en la construcción de archivos como los objetivos y puntos de extensión que dependen uno del otro. Su principal uso conocido es la construcción de aplicaciones Java.

***Hibernate*:** Herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java, que proporciona un marco para la asignación de un modelo de dominio orientado a objetos a una base de datos relacional tradicional.

Glosario de términos

***Zend Plataform*:** Entorno de ejecución para PHP que entrega la confiabilidad, escalabilidad, e interoperabilidad necesaria para aplicaciones de clase corporativa. Provee nuevas herramientas de depuración y análisis de código, administración de la configuración, e integración con Java.

***Socket*:** Método de comunicación entre el servidor y el cliente, o bien entre programas dentro el mismo ordenador.

***Pascal*:** Pascal es un lenguaje de programación, creado con el objetivo de facilitar el aprendizaje de programación en estudiantes, utilizando la programación estructurada y estructuración de datos.