**UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS**

**Facultad 6**


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

## Módulo para la visualización de interfaz de usuario para la herramienta ATOM.

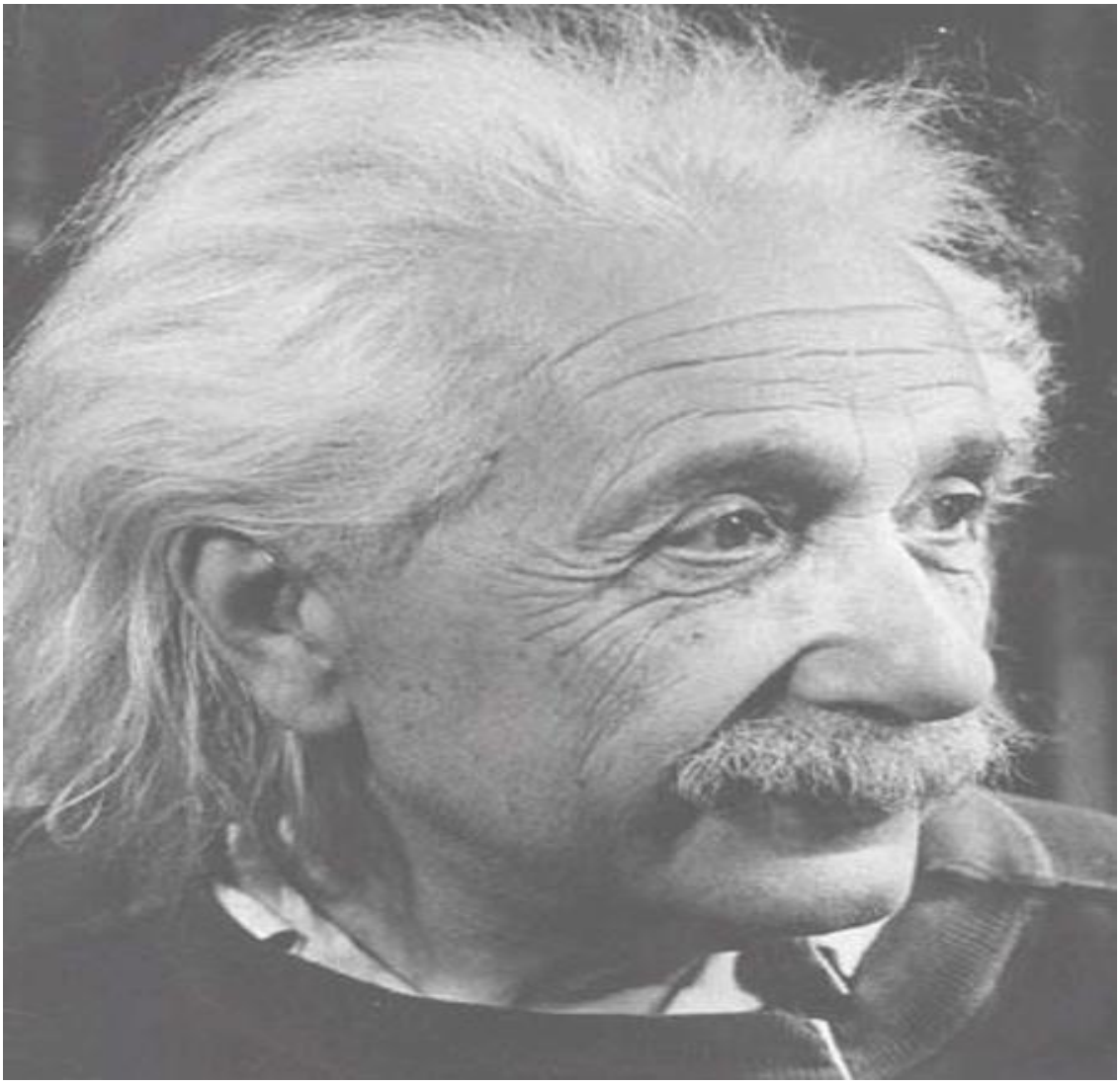
**AUTOR:** Emir Mutajar Salej Hernández.

**TUTOR:** Ing. Alain Leon Companioni.

**La Habana, Junio /2013**

**¨Año 55 de la Revolución¨**

# Pensamiento



"La imaginación es más importante que el conocimiento. El conocimiento es limitado, mientras que la imaginación no"

*Albert Einstein.*

# Dedicación de Autoría

## Declaración de Autoría.

Declaro que soy el único autor de este trabajo y autorizo 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 _____.

| | |
|---|---|
| _____ | _____ |
| Emir Mutajar Salej Hernández | Ing. Alain Leon Companioni |
| Autor | Tutor |

# Dedicación y Agradecimientos

## *Dedicado a:*

*Mi abuela Angela que aunque no estuvo conmigo en gran parte de mi vida sé que desde algún lugar está apoyándome…. GRACIAS por todo….*

*En especial… a mi abuelo RICARDO que aunque para muchos no estés, para mi sigues viviendo, exactamente en el corazón, sé que me quisiste mucho y debes saber que te quiero…GRACIAS por todo….*

*A mi tío Ricardito… que más que tío fue padre y que hasta hoy ha estado siempre ahí, a mi lado, haciéndome saber cuánto me quiere…. GRACIAS por todo….*

*A ti… mamá, que fuiste, eres y serás lo más grande del mundo, que viviste tu vida para mí, que eres mi ejemplo, que vivirás siempre, que te QUIERO CON EL ALMA… GRACIAS por todo….*

*Y a ti… hermanito mío, que aunque todavía no lo sepas has sido muy importante en mi vida, que desde que llegaste cambiaste la vida de todos… trataré de darte lo mejor del mundo y espero que sean muchas cosas…. GRACIAS por todo….*

## *Agradecimientos:*

*Agradecer a alguien muy especial en mi vida…. mi novia Arianna que pase lo que pase está a mi lado, que no importa lo que diga…. que yo sé que está muerta de amor conmigo…. Que me quiere tanto como yo la quiero a ella….*

*A mi hermano UCI… Renier, que más que yo agradecerle debería agradecerme el a mí por todo lo que le he enseñado…. que hemos pasado los mejores momentos de la universidad y muchos momentos buenos de nuestras vidas….*

*A David G. que es uno de mis mayores fans… a Maricet que sin ella a lo mejor no estuviera escribiendo estas líneas, gracias por tanta ayuda en los primeros año… al piquete de la RedBull que no los puedo mencionar a todos porque son muchos, pero ellos saben quiénes son…. A todas las niñas del edif 95 que sé que me tienen en lo más alto… a mi tutor Alain por confiar en mí y ayudarme en este trabajo….*

*A esa gran familia de amigos que tengo en mi casa, que aunque me fui por mucho tiempo siempre se mantuvieron ahí y me dieron el apoyo que en algún momento necesité… gracias a Yan, David, Laziel, Alfredo, Yohander…..*

*GRACIAS A TODOS……*

# Datos de Contacto

**Autor:**

Emir Mutajar Salej Hernández

Universidad de las Ciencias Informáticas

Ciudad de la Habana, Cuba

E-mail: emsalej@estudiantes.uci.cu

**Tutor:**

Nombre: Alain León Companioni.

Ingeniero en Ciencias Informáticas, graduado en 2010 en la Universidad de las Ciencias Informáticas (UCI).

Correo Electrónico: acompanioni@uci.cu

## Resumen:

Los Sistemas de Información Geográfica (SIG) son sistemas para el análisis, manipulación y generación de información geográfica, utilizados fundamentalmente en los sectores económicos, políticos y sociales para la toma de decisiones en negocios específicos. La Universidad de las Ciencias Informáticas (UCI) ha desarrollado una plataforma con el propósito de crear SIG que sean capaces de competir y establecerse en un estándar de calidad de nivel internacional. Esta plataforma de nombre Genesig es una aplicación SIG Web, capaz de soportar una amplia gama de funcionalidades relacionadas con la gestión de datos espaciales, por parte de usuarios especializados y nuevos consumidores de servicios de datos geográficos, también fue creada la herramienta ATOM v1.0 capaz de personalizar esta plataforma de forma rápida e intuitiva, todos estos productos fueron desarrollados con tecnología y herramientas libres estando acorde con la política de independencia tecnológica que enfrenta Cuba. A pesar de las facilidades que trae consigo el uso de la herramienta ATOM, no es posible visualizar la construcción de la interfaz de SIG instantáneamente, lo que conlleva a que se tenga que contar con servidores y estaciones de trabajo para lograr este objetivo. La presente investigación es el resultado del ciclo completo de desarrollo de software que propone la metodología Programación Extrema (XP), teniendo como resultado un módulo para ATOM capaz de mostrar los cambios realizados en la plataforma y ejercer algún tipo de configuración a la interfaz de usuario desde la misma.



**Palabras clave:**

Datos Espaciales, Metodología XP, Plataforma, Sistemas de Información geográfica.

# Índice

# Índice

# Índice

# Introducción

La historia de la cartografía data desde los primeros trazos en la arena o nieve, la aparición de los mapas se produjo antes de la historia, es decir, con anterioridad a la aparición del relato escrito y se utilizaron para establecer distancias, recorridos, localizaciones y así los seres humanos pudieran desplazarse de un lugar a otro. Con la evolución de las tecnologías, el desarrollo de los mapas aumentó considerablemente fomentando el análisis y conocimiento exacto del terreno, además de prestar innumerables servicios al estudio de la geografía.

Las Nuevas Tecnologías de la Información y las Comunicaciones (en lo adelante TICs) son un conjunto de tecnologías desarrolladas para gestionar información y enviarla de un lugar a otro. Incluyen las tecnologías para almacenar información y recuperarla después, enviar y recibir información de un sitio a otro, o procesar información para poder calcular resultados y elaborar informes. La informática ha extendido su uso a través de sistemas de diseño y control de los procesos de información y ha dado solución a problemas que se presentaban en distintos sectores, precisamente en el sector de las nuevas tecnologías de la información, para generar información geográfica. De estas, algunas de las más usadas actualmente son el Sistema de Posicionamiento Global (en lo adelante GPS) y los Sistemas de Información Geográfica (en lo adelante SIG).

Se entiende por "Sistema de Información" la conjunción de información con herramientas informáticas, es decir, con programas informáticos o software. Si el objeto concreto de un sistema de información (información + software) es la obtención de datos relacionados con el espacio físico, entonces se estará hablando de un Sistema de Información Geográfica o SIG (*GIS en su acrónimo inglés*, *Geographic Information System*). Así pues, un SIG es un software específico que permite a los usuarios crear consultas interactivas, integrar, analizar y representar de una forma eficiente cualquier tipo de información geográfica referenciada asociada a un territorio, conectando mapas con bases de datos. (1)

Eventualmente el auge de los SIG en la nueva sociedad de la información ha tenido una enorme importancia en el incremento de la capacidad organizacional frente al cambio del entorno. La voluntad de lograr un sistema de información útil, que permita obtener una ventaja competitiva, implica la posibilidad de ofrecer múltiples, frecuentes, oportunas y relevantes informaciones, así como facilitar el proceso de

toma de decisiones jugando un papel fundamental.

Los SIG se emplean para almacenar la información geográfica, además implementan funcionalidades para el análisis y manipulación de datos geográficos así como la modelación de la realidad a través de la creación de imágenes abstractas de una realidad más compleja, permitiendo su uso, estudio, análisis y gestión. Cuba no se ha quedado atrás en el desarrollo de la tecnología, involucrando a múltiples sectores y esfuerzos diversos, a pesar de factores internos y externos en contra avanza con un programa de informatización de la sociedad, dentro del cual se puede destacar el estudio y desarrollo de los SIG.

La Universidad de las Ciencias Informáticas se ha convertido en la principal desarrolladora de Sistemas de Información Geográfica en Cuba, en el departamento Geoinformática, perteneciente al centro de desarrollo y producción de software Geoinformática y Señales Digitales, específicamente en la Línea de Productos Aplicativos SIG, se desarrollan SIG para la Web, basados en la versión 1.5 de la Plataforma Genesig. La Plataforma Genesig (versión 1.5), tiene como principal objetivo realizar la representación geoespacial de la información asociada a negocios específicos, permitiendo además realizar análisis sobre dicha información. Por otra parte su estructura arquitectónica permite personalizar sus funcionalidades y adaptarlas a cualquier negocio que lo requiera a través de la reutilización de sus componentes. Puede ser considerado como un SIG Web único y extensible, basado en estándares OpenGIS que incluye funcionalidades operativas de las aplicaciones de esta tecnología. (2)

La herramienta ATOM v1.0 facilita el proceso de personalización de la plataforma Genesig al centralizar los procesos de gestión y configuración de los proyectos, componentes, mapas, conexiones a base de datos y otros ficheros de configuración de la misma. Esto posibilita al equipo de desarrollo poder configurar la plataforma Genesig de forma rápida sin necesidad de tener instalados en la estación de trabajo los servidores necesarios para cargar la plataforma, sin embargo, no permite visualizar los cambios realizados en la interfaz de usuario en los proyectos instantáneamente, para esto sí se deben tener configuradas las estaciones de trabajo para que permita visualizar la plataforma en un entorno Web.

Para dar una explicación más simple de lo que se necesita se debe tener en cuenta que la herramienta ATOM no cuenta con un entorno en el cual se pueda observar la interfaz con que se está formando el SIG, para poder ver esta interfaz es necesario salir de la herramienta, acceder a un navegador y tener configurada la plataforma Genesig. Por lo tanto causa pérdida de tiempo, consumo de recursos de

hardware y se pierde simplicidad en la herramienta a la hora de la modificación de las interfaces.

Teniendo en cuenta lo planteado anteriormente se define como **problema a resolver**: ¿Cómo simplificar el proceso de personalización de la plataforma Genesig en la herramienta ATOM, de manera que sea posible visualizar los cambios realizados en la interfaz de usuario instantáneamente? El **objeto de estudio** es la personalización de la plataforma Genesig, en el desarrollo de aplicativos SIG con la herramienta ATOM. Enmarcado en el **campo de acción:** el proceso para el desarrollo de la interfaz de usuario de la plataforma Genesig en la herramienta ATOM. Para dar solución al problema se define como **objetivo general:** Implementar un módulo para la herramienta ATOM que permita representar instantáneamente los cambios realizados en la interfaz de usuario.

**Idea a defender:** Con la implementación de un módulo para la herramienta ATOM que permita representar instantáneamente los cambios realizados en la interfaz de usuario se simplificará los procesos de personalización de la plataforma Genesig.

Para dar cumplimiento al objetivo general se definen las siguientes tareas de la investigación:

1. Seleccionar y fundamentar las tendencias y tecnologías actuales a utilizar.
2. Identificar los conceptos asociados al entorno donde está enmarcado el problema.
3. Definir o identificar las principales funcionalidades a implementar en módulo.
4. Modelar el módulo propuesto.
5. Implementar las funcionalidades del módulo propuesto.
6. Probar los resultados obtenidos.

Al concluir la investigación se esperan los siguientes resultados:

- El módulo para la visualización de interfaz de usuario para la herramienta ATOM.
- La documentación técnica asociada al desarrollo del módulo para la visualización de interfaz de usuario para la herramienta ATOM.

**Métodos Científicos**

Dentro de los **Métodos Teóricos:**

- ✓ **Histórico-lógico:** Este método es un procedimiento por el cual se analizan diversos hechos o fenómenos de manera secuencial, aludiendo a la aparición y desarrollo de los factores presentes. (3). Se utiliza con el objetivo de definir las tendencias actuales para el trabajo con interfaces de usuario.
- ✓ **Análisis y Síntesis:** Este método será utilizado en el estudio y análisis de bibliografías relacionadas con el argumento, para ocupar algunas posiciones teóricas vinculadas al objetivo de la investigación.
- ✓ **Modelación:** Se utiliza para la modelación de diagramas, representar el proceso de desarrollo y propiciar un mejor entendimiento de la solución a implementar.

Dentro de los **Métodos Empíricos:**

- ✓ **Observación:** Este método será utilizado para poder percibir los cambios y características que se presenten durante el proceso de desarrollo. Para definir cuáles son las principales deficiencias desde el punto de vista visual de la aplicación.

**Estructura de la investigación:**

**Capítulo 1: Fundamentación Teórica**. Este capítulo mostrará la descripción del negocio, se encontrará descrita la situación problemática y se analizarán otras posibles soluciones para el problema de investigación que se plantea en este trabajo.

**Capítulo 2: Actuales tendencias y tecnologías para el desarrollo**. Este capítulo caracteriza las herramientas, tecnologías y lenguaje de programación que se necesitan y utilizan para el desarrollo de la solución propuesta, además de las ventajas de su utilización.

**Capítulo 3. Presentación de la solución propuesta:** Se abordará las fases iniciales de la Programación Extrema, se definirán las historias de usuario y las necesidades del cliente. En la segunda fase, planificación, se estima el esfuerzo que se requiere para cada historia de usuario y se realiza una programación de acuerdo a esa estimación.

**Capítulo 4. Construcción y validación de la solución propuesta**: Este capítulo abordará las fases restantes que contiene XP, la fase de construcción y la de prueba. Serán descritas las tarjetas CRC (Clase + Responsabilidad + Colaboración) y se describen las iteraciones hechas en la etapa de construcción de la aplicación, así como las actividades generadas por las historias de usuarios, también se describirán las pruebas efectuadas al módulo.

# Capítulo 1. Fundamentación Teórica.

En este capítulo se exponen los elementos relacionados con la fundamentación teórica. Está destinado a abarcar mediante una breve descripción los conceptos asociados al dominio del problema, así como la descripción del objeto de estudio, referente a la creación del módulo de interfaz de usuario para la herramienta ATOM, del centro de desarrollo de Geoinformática y Señales Digitales de la Universidad de las Ciencias Informáticas. También se encontrarán una serie de soluciones que en la actualidad se utilizan para la creación de interfaces de usuarios y se expondrán los principales problemas que condujeron al desarrollo de la investigación.

## 1.1 Conceptos asociados al dominio del problema

**Interfaz de Usuario**

Se define interfaz como "la presentación en pantalla que un sistema informático ofrece al usuario para que este pueda interactuar con él", es lo que el usuario ve del sistema, no el funcionamiento interno del mismo.

Es el medio mediante el cual la información se transfiere del usuario al ordenador y al contrario. En general, existe consenso en reconocer que se trata de un elemento fundamental puesto que conecta al usuario con la fuente de información, permitiendo tanto la búsqueda o acceso como la presentación, organización o almacenamiento de la información (4).

**Sistema de Información Geográfica**

Se entiende como la 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 programa informáticos específico; el conjunto es soportado por un sistema de computadores y por un personal especializado. (5)

**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. (6)

## 1.2 Objeto de Estudio

### 1.2.1 Descripción General

El desarrollo en que se encuentra el mundo en la actualidad está dado principalmente al surgimiento sistemático de nuevas herramientas y tecnologías, por lo cual han surgido varios métodos y estrategias para crear aplicaciones informáticas o software, especialmente a la hora de construir una interfaz práctica y con la apariencia deseada.

El diseño de la interfaz de usuario, es un proceso iterativo donde los usuarios interactúan con los diseñadores de interfaces y realizan acciones sobre los prototipos de la interfaz para decidir las características y funcionamiento de la interfaz de usuario del sistema.

Estas interfaces deben ser capaces de brindar comodidad al usuario, además de contribuir en la manera que sea posible a la productividad y a la imagen del software, ya que es el producto con que más interactúa. Durante el proceso de creación de estas y la posterior terminación de las mismas se suelen utilizar gran cantidad de programas. Estos programas o herramientas son principalmente manejados por personal profesional, con conocimientos previos en su uso y con experiencia en el estudio de las principales o más comunes formas de diseño debido a que en muchas ocasiones es necesario un conocimiento mínimo del lenguaje de programación con el que trabaja la herramienta y el nivel de complejidad que presenta.

### 1.2.2 Descripción General del Dominio del Problema

La Universidad de las Ciencias Informáticas (UCI) en el país es una universidad vanguardia y de referencia que vincula el estudio con el trabajo como modelo de formación; que tiene como principio producir software y servicios informáticos. Estos servicios están principalmente destinados a los sectores económicos y sociales de Cuba, desarrollando soluciones con calidad y eficacia que abalan este trabajo.

Adjunto a la universidad se encuentra el centro de desarrollo de software GEySED dividido en dos

departamentos (Geoinformática y Señales Digitales). En el departamento de Geoinformática, se define como uno de sus principales productos la plataforma Genesig, que brinda una solución inmediata a la necesidad de tener una plataforma de este tipo en el país, ya que no se contaba con la existencia de una. Otros aspectos que influyeron en la creación de Genesig fueron la demanda e incremento del uso de los SIG a nivel nacional e internacional, vale destacar también que de manera efectiva apoya la disminución de costos por trabajar sobre tecnologías libres.

La representación geoespacial de la información que presentan los SIG relacionada con negocios específicos es una de las principales tareas a desarrollar en el centro GEySED, este se incorpora a la búsqueda de nuevos mecanismos y soluciones que permitan mejorar los sistemas y las técnicas que existen para su creación.

La herramienta ATOM desarrollada por el proyecto Aplicativos SIG fue creada con el objetivo de personalizar la plataforma Genesig para poder ajustar las funcionalidades de esta herramienta a las necesarias para un SIG destinado a un negocio en específico, así el usuario podrá interactuar con un software más cercano y ajustado a sus necesidades. La herramienta será capaz de utilizar la información y ficheros presentes en la plataforma, cambiando su posición y contenido dependiendo del trabajo realizado y las configuraciones ejecutadas. Por lo cual al culminar un ciclo de trabajo se podrán observar cambios en la plataforma, debido a la personalización realizada en la misma.

### 1.2.3 Situación problemática

El proyecto Aplicativos SIG enfrenta como uno de los principales problemas la personalización de la plataforma Genesig a la hora de obtener nuevos productos, el problema radica principalmente en la necesidad de que los especialistas tengan un amplio conocimiento sobre el trabajo con dicha plataforma, para comprender su funcionamiento interno. Este trabajo trae consigo la descentralización, debido a que las tareas son realizadas en distintas herramientas, lo que genera un difícil manejo, organización, control y nueva utilización de datos y componentes necesarios.

Como solución a los problemas antes mencionados se desarrolló la herramienta ATOM capaz de personalizar la plataforma Genesig, evitando retrasos en el proceso de desarrollo y logrando centralizar los procesos de gestión y configuración de los proyectos, componentes, mapas, conexiones a base de

datos y otros ficheros de configuración de la misma, sin tener la necesidad de instalar en la estación de trabajo los servidores necesarios para el trabajo con la plataforma, sin embargo, los cambios que se realizan en la interfaz de usuario no se pueden visualizar instantáneamente, ya que la herramienta no cuenta con un entorno o sector en el cual se muestren estos cambios realizados al SIG, para que esto suceda si se deben configurar las estaciones de trabajo.

## 1.3 Soluciones Existentes

En el mundo existen varias aplicaciones para el trabajo con interfaces de usuarios o para la creación de las mismas, debido a la gran importancia que tienen estas en el éxito de los diferentes softwares[1], aplicaciones o sistemas. Cuando se habla de tendencias de diseño, se refiere hacia dónde se dirige el diseño en estos días; como son las nuevas interfaces gráficas, su estética y usabilidad sumada a la capacidad de funcionamiento.

Cuando los diseñadores hacen su trabajo, siempre tienen presente que están diseñando para los usuarios, por lo tanto, están realizando una interfaz que será usada por cientos, miles o millones de usuarios y que el éxito de su trabajo, dependerá casi exclusivamente de su capacidad para seducir a la mayor cantidad de usuarios. (7)

Tomando en cuenta que no existe una solución para el problema planteado, se hace el estudio de las principales formas o tendencias utilizadas en el mundo para el desarrollo de interfaces de usuarios y se generan propuestas e ideas para la creación de la solución propuesta. A continuación se presentan algunas herramientas con las cuales se logran desarrollar interfaces de usuarios para distintas aplicaciones y con características diferentes.

[1] Conjunto de programas, instrucciones y reglas informáticas que permiten ejecutar distintas tareas en una computadora.

.

### 1.3.1 NetBeans 6.9

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java pero puede servir para cualquier otro lenguaje de programación. Para la creación de las interfaces de usuarios cuenta con un formulario que contiene barras de herramientas con elementos visuales para su manipulación, con estos elementos se crea la base de lo que será el aspecto visual de la aplicación, todo a un nivel de selección y arrastre a la posición deseada y con la existencia de una barra de propiedades se puede cambiar o configurar esos elementos para su adaptación a la idea principal del diseño.

Además cuenta también con la posibilidad de hacer un cambio de entorno y dirigirse directamente al código que se desarrolla en segundo plano y ajustar o hacer cambios de propiedades y comportamiento a los diferentes componentes seleccionados.

### 1.3.2 ExtDesigner

ExtDesigner es una aplicación de escritorio muy útil a la hora de crear interfaces, con un entorno de arrastrar y soltar es una de las aplicaciones más sencillas e intuitivas a la hora de su uso, permitiendo crear una interfaz rápidamente y con una excelente apariencia. Por tal motivo muchos desarrolladores se apoyan en este software de diseño para darle un aspecto fresco y diferente a sus trabajos. Como inconveniente presenta que no permite editar el código generado por el mismo, solamente copiarlo para su posterior uso.

### 1.3.3 Adobe Dreamweaver CS6

Dreamweaver también es un software que permite trabajar en la creación de interfaces, en este caso para la web, o bien a nivel de código, o bien sólo en modo diseño (la apariencia de la web), pero también cuenta con la opción de trabajar en los dos tipos de entornos a la vez, dividiendo la zona de trabajo en dos partes y permitiendo acceder de una a la otra con rapidez, de esta manera se visualiza diseño y código al mismo tiempo y se brinda una mejor movilidad y fluidez en el trabajo.

### 1.3.4 Qt Designer

Es una herramienta muy potente mediante la cual se puede realizar el diseño de aplicaciones de forma gráfica y muy intuitiva. Muy parecida a la utilizada en los demás entornos de desarrollo, con un panel en el cual aparecen los componentes a utilizar en la construcción de la interfaz, no presenta la posibilidad de dirigirse al código, simplemente genera un archivo *.ui al que no es recomendable cambiarle el código generado.

### 1.3.5 Artister 3.1

La herramienta Artister fue creada para el diseño de páginas web, presenta una interfaz amigable e intuitiva para la creación y construcción de las mismas, como técnica o tendencia presenta el uso de las plantillas como base fundamental para el trabajo, permitiendo el cambio de contenido y formato, esto permite al usuario poder variar entre las plantillas existentes y lograr así un producto final acorde a lo deseado. En su interfaz cuenta con un menú de opciones en la parte superior que permite un manejo más profundo del contenido, permitiendo la edición de las plantillas predefinidas y la transformación total de las mimas.

### 1.3.6 Visual Studio 2010

Es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) que presenta un excelente diseño a la hora de crear interfaces de usuarios. Cuenta con un entorno de trabajo muy organizado en el cual se pueden observar y tener al alcance varios paneles con opciones de selección y adición de componentes o propiedades. Se puede trabajar creando la interfaz desde la sección de código o simplemente arrastrando los componentes deseados y organizándolos de una manera conveniente. Por estas características es considerado un IDE vanguardia en la creación de interfaces, destacando su facilidad y funcionalidad.

### 1.3.7 Análisis y selección de las prácticas de trabajo

Para el trabajo de configuración de la interfaz de usuario para Genesig en la herramienta ATOM se creará un entorno en modo diseño, es decir, se podrá visualizar el diseño utilizado para la creación de los SIG y este entorno contará con un sector, el cual contiene los diferentes componentes que se podrán adicionar al menú. Además contará también con oportunidades de transformación en estos y la posibilidad de

cambio entre los mismos, todo esto al alcance del puntero de selección y con pequeños menús de opciones y configuración. Sin embargo no se creará un entorno para la gestión mediante el código ya que el entorno en modo diseño cumple con las necesidades del software.

## 1.4 Conclusiones del capítulo

Tras el análisis realizado teniendo en cuenta la fundamentación teórica de la investigación en curso se ha determinado que actualmente en el departamento de Geoinformática, la herramienta ATOM no cuenta con un entorno o sector en el cual se pueda observar la interfaz de usuario de los SIG, por lo que se hace necesario el desarrollo de la misma. Además se obtuvo un formato de diseño para la construcción del módulo propuesto, mediante el estudio y el análisis de las diferentes y más usadas tendencias en la actualidad.

# Capítulo 2. Tecnologías a utilizar.

Para analizar las diferentes tecnologías y herramientas que se utilizarán en el desarrollo de la solución deseada, estará dirigido este capítulo, para poder hacer un estudio detallado de las características y mejores opciones con respecto a las mismas. Mediante este análisis se logra determinar cuál es el verdadero nivel de importancia y necesidad de trabajar con dichas tecnologías y tendencias.

## 2.1 Metodología de desarrollo a utilizar

Las metodologías surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto (software). Las más utilizadas son: La Programación Extrema (en lo adelante XP), Scrum y el Proceso Unificado de desarrollo de Software (en lo adelante RUP). (8)

**Metodologías de desarrollo:** Conjunto de procedimientos, técnicas, herramientas y soporte documental que deben seguirse para el desarrollo del software.

**Procedimiento:** descomponer el proceso a nivel de tareas, para cada tarea definir un procedimiento sobre cómo llevarla a cabo.

**Técnicas:** Se puede usar una o más técnicas para aplicar el procedimiento.

**Herramientas:** Software que automatiza la aplicación de una técnica. (9)

**Especifica:**

• Cómo se debe dividir un proyecto en etapas

• Qué tareas hay que realizar en cada etapa

• Qué salidas se producen y cuándo

• Qué restricciones se aplican

• Qué herramientas se utilizan

• Cómo se gestiona y controla un proyecto

Sin dudas es de gran importancia hacer una selección de la metodología de desarrollo, de acuerdo a las características del proyecto o software. Actualmente existen varios tipos de metodologías, por lo que se necesita hacer una búsqueda cuidadosa a la hora de apegarse a una de estas, para ello se deben tener en cuenta varios puntos que influyen en el desarrollo del sistema: tiempo de desarrollo esperado o disponible, recursos humanos y técnicos con los que se cuenta.

Las propuestas metodológicas se dividen en “pesadas o tradicionales” y las “ágiles o ligeras”, las primeras son más usadas cuando los proyectos son de gran envergadura y requieran tiempo y recursos, en cambio para proyectos de tiempo restringido y entornos cambiantes se recomienda la metodología ágil.

**Metodologías ágiles**

En las metodologías agiles se puede apreciar que se valora:

Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. El personal es el principal factor de éxito de un proyecto software. Es más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente. Es mejor crear el equipo y que éste configure su propio entorno de desarrollo en base a sus necesidades.

Desarrollar software funciona más que conseguir una buena documentación. La regla a seguir es no producir documentos a menos que sean necesarios de forma inmediata para tomar una decisión importante. Estos documentos deben ser cortos y centrarse en lo fundamental.

La colaboración con el cliente más que la negociación de un contrato. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.

Responder a los cambios más que seguir estrictamente un plan. La habilidad de responder a los cambios que puedan surgir a lo largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino

flexible y abierta. (10)

Debido a que el trabajo a realizar es la implementación de un módulo para una herramienta ya existente solamente se realizará el estudio de las metodologías ágiles, que son las metodologías más utilizadas para proyectos pequeños o de poca envergadura donde el proceso de creación es enfocado al desarrollo pleno de la tarea y donde se genera poca documentación.

### 2.1.1 Scrum

Scrum es un marco de referencia para desarrollo ágil de productos software. Actualmente SCRUM es uno de los métodos ágiles que está creciendo más y en el 2011 lo usaba el 75% de equipos de desarrollo ágiles alrededor del mundo. Muchos equipos que están usando Scrum reportan mejoras importantes y en algunos casos transformaciones completas, sobre todo en la productividad.

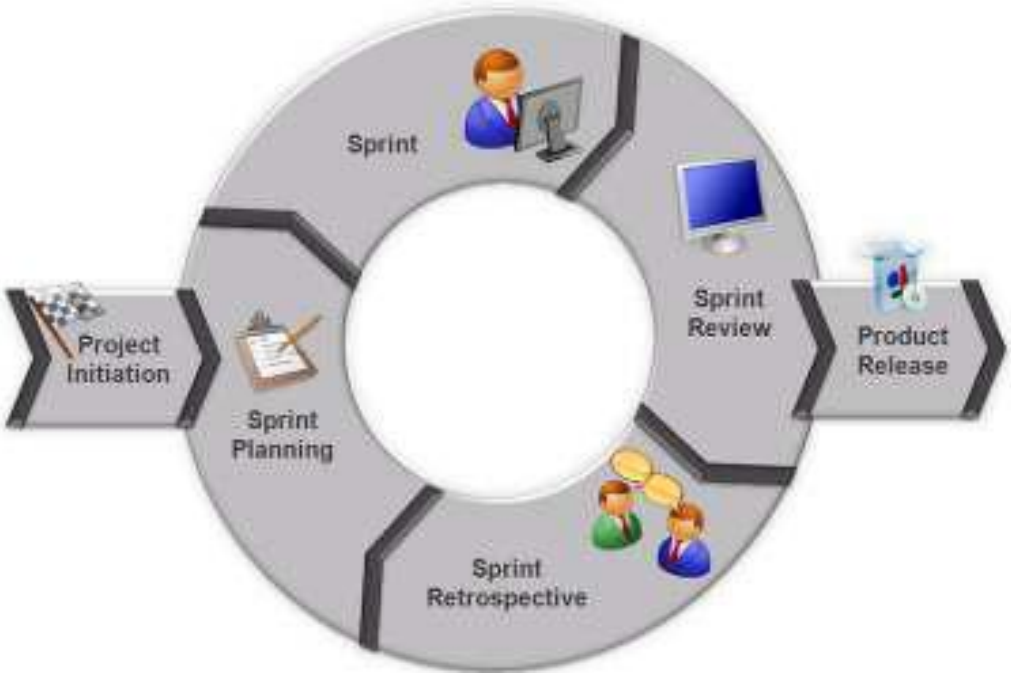
El período de desarrollo es de una iteración de 30 días llamada Sprint, si bien podría trabajarse con sprints de menor duración. El marco de trabajo de Scrum tiene tres componentes o cuatro según como se los mire. (11)

• Pre-sprint: planificación del sprint.

• Sprint: ciclo de trabajo

• Post-sprint: revisión y retrospectiva del sprint (pueden considerarse como dos componentes diferentes posteriores al sprint)

El punto focal es el sprint, donde se desarrolla software que funcione. (Ver Siguiente Figura)




Ventajas

- ✓ Programación organizada.
- ✓ Menor taza de errores.
- ✓ Satisfacción del programador.

Desventajas

- ✓ Es recomendable emplearlo solo en proyectos a corto plazo.

### 2.1.2 XP

Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. XP se basa en la realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y facilidades para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes y donde existe un alto riesgo

técnico. Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre. (12)

Comunicación

Es muy importante que haya una comunicación constante con el cliente y dentro de todo el equipo de trabajo, de esto dependerá que el desarrollo se lleve a cabo de una manera sencilla, entendible y que se entregue al cliente lo que necesita.

Simplicidad

En la XP se refiere que ante todo y sin importar qué funcionalidad requiera el usuario en su sistema, éste debe ser fácil. El diseño debe ser sencillo y amigable al usuario, el código debe ser simple y entendible, programando sólo lo necesario y lo que se utilizará.

Retroalimentación

Es la comunicación constante entre el desarrollador y el usuario.

Coraje

Se refiere a la valentía que se debe tener al modificar o eliminar el código que se realizó con tanto esfuerzo; el desarrollador debe saber cuándo el código que desarrolló no es útil en el sistema y, por lo mismo, debe ser eliminado. También se refiere a tener la persistencia para resolver los errores en la programación.

Herramientas de XP:

**Historias de usuarios:** Son tarjetas físicas en las cuales se anota una descripción de una funcionalidad del sistema, en una oración, se le da un número y un título para ser identificada.

**Casos de prueba de aceptación:** Son tarjetas que se elaboran para realizar las pruebas de cada historia de usuario.

**Tarea de ingeniería:** Son tarjetas que se elaboran para ayudar y simplificar la programación de una historia de usuario.

**Tarjetas (Clase, Responsabilidad y Colaboración) CRC:** Describen las clases utilizadas en la programación de una historia.

Ventajas y desventajas:

Una de las ventajas de la programación extrema es que se adapta al desarrollo de sistemas pequeños y grandes; optimiza el tiempo de desarrollo; permite realizar el desarrollo del sistema en parejas para complementar los conocimientos; el código es sencillo y entendible, además de la poca documentación a elaborar para el desarrollo del sistema.

Las desventajas son que no se tiene la definición del costo y el tiempo de desarrollo; el sistema va creciendo después de cada entrega al cliente y nadie puede decir que el cliente no querrá una función más; se necesita de la presencia constante del usuario, lo cual en la realidad es muy difícil de lograr.

Otra desventaja es la programación en parejas, algunos desarrolladores son celosos del código que escriben y no les es grato que alguien más modifique las funciones que realizó o que su código sea desechado por no cubrir el estándar. (13)

El ciclo de vida ideal consta de 4 fases: (14)

**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.

### 2.1.3 Selección de la metodología a utilizar.

Luego de hacer el análisis de las metodologías más usadas a nivel mundial, se selecciona XP para desarrollar el trabajo previsto ya que se adapta y ofrece características que se acoplan al proyecto en función.

El proyecto a realizar es pequeño y necesariamente cambiante en su etapa de desarrollo, a la hora de la selección de los principales requisitos y esta metodología ofrece una estrategia de comunicación fluida entre el cliente y el desarrollador, además de la facilidad para enfrentar los cambios necesarios y continuos. Tampoco se cuenta con un contrato que especifique el tiempo, el alcance o los recursos necesarios, que son tomados en cuenta principalmente en metodologías como RUP para trabajos de gran alcance o tamaño.

## 2.2 Lenguajes de programación.

Es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana.

Está formado de un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa informático se le llama programación. (15)

### 2.2.1 Java.

El lenguaje para la programación en Java, es un lenguaje orientado a objeto, de una plataforma

independiente, fue desarrollado por la compañía Sun Microsystems, con la idea original de usarlo para la creación de páginas WEB (World Wide Web). Esta programación Java tiene muchas similitudes con los lenguajes C y C++, así que si se tiene conocimiento de este lenguaje, el aprendizaje de la programación Java será de fácil comprensión por un programador que haya realizado programas en estos lenguajes. (16)

**Características del lenguaje Java**

**Lenguaje simple**

Java posee una curva de aprendizaje muy rápida. Resulta relativamente sencillo escribir applets[2] interesantes desde el principio. Debido a su semejanza con C y C++ y dado que la mayoría de las personas 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.

**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[3], 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).

**Seguro**

---

[2] Es un componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo un navegador web.

Dada la naturaleza distribuida de Java, donde las applets se bajan desde cualquier punto de la Red, la seguridad se impuso como una necesidad de vital importancia. A nadie le gustaría ejecutar en su ordenador programas con acceso total a su sistema, procedentes de fuentes desconocidas. Así que se implementaron barreras de seguridad en el lenguaje y en el sistema de ejecución en tiempo real.

**Indiferente a la arquitectura**

Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows Nt, pasando por Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos. El compilador de Java genera bytecodes: un formato intermedio indiferente a la arquitectura diseñada para transportar el código eficientemente a múltiples plataformas hardware y software. El resto de los problemas los soluciona el intérprete de Java.

**Multihilo**

Hoy en día ya se ven limitadas las aplicaciones que sólo pueden ejecutar una acción a la vez. Java soporta sincronización de múltiples hilos de ejecución (*multithreading*) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Así, mientras un hilo se encarga de la comunicación, otro puede interactuar con el usuario mientras otro presenta una animación en pantalla y otro realiza cálculos.

**Produce applets**

Java puede ser usado para crear dos tipos de programas: aplicaciones independientes y applets. Las aplicaciones independientes se comportan como cualquier otro programa escrito en cualquier lenguaje, como por ejemplo el navegador de *Web HotJava*, escrito íntegramente en Java. Por su parte, las applets son pequeños programas que aparecen embebidos en las páginas Web, como aparecen los gráficos o el texto, pero con la capacidad de ejecutar acciones muy complejas, como animar imágenes, establecer conexiones de red, presentar menús y cuadros de diálogo para luego emprender acciones. (17)

### 2.2.2 C++

**Características del Lenguaje C++** (18)

- 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. (19)

### 2.2.3 C#

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++.
- Se ahorra 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. (20)

### 2.2.4 Selección del lenguaje de programación a utilizar

Tras el estudio e investigación realizada sobre los lenguajes de programación orientados a objetos, se toma como decisión la utilización del lenguaje Java debido a que el módulo a desarrollar es la continuidad y mejora de una aplicación creada en este lenguaje de programación, es necesario utilizarlo para la construcción del mismo, dependiendo de su integración y posterior uso sobre la herramienta. No obstante este lenguaje cuenta con varias características que son el motivo por el cual se usó primeramente, muchas de estas acordes a las políticas y necesidades del proyecto encargado.

Es un lenguaje multiplataforma que ha tomado una gran importancia en el mundo del desarrollo de aplicaciones, tanto a nivel de ordenadores como a nivel de tecnología móvil, también cuenta con código abierto y los programas hechos en Java podrán ejecutarse en cualquier ordenador ya que es independiente de la plataforma.

Además de su portabilidad, potencia, seguridad y estabilidad es fácil al aprendizaje y la producción, cuenta también con millones de usuarios a nivel internacional y con gran diversidad de apoyo en documentación y ayuda.

## 2.3 Entornos de desarrollo.

Un Entorno de Desarrollo Integrado, traducido del inglés Integrated Development Environment (IDE) es un programa informático compuesto por un conjunto de herramientas de programación. Los IDE proveen de un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, Java, C#, Delphi, Visual Basic, etc. Es posible que un mismo desarrollo integrado funcione con varios lenguajes de programación. (21)

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, o sea, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. (22)

### 2.3.1 NetBeans.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de

componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. El IDE NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java - pero puede servir para cualquier otro lenguaje de programación, soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Existe además un número importante de módulos para extender el IDE NetBeans, es un producto de Software Libre y gratuito sin restricciones de uso. (23)

Aparte de la filosofía de distribución y desarrollo que respalda a NetBeans, el IDE ofrece a los desarrolladores numerosas ventajas, en la creación de nuevas aplicaciones multiplataforma. Este IDE de Java es de mucha confianza para los desarrolladores y sumamente útil por su manual de instrucciones, es uno de los IDE más utilizado y poderoso a la hora del desarrollo de aplicaciones.

### 2.3.2 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 componentes se pueden utilizar tanto Subversion para realizar el control de versiones como Hibernate para desarrollar las funciones de un motor de persistencia. (24)

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.
- 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 componentes.

### 2.3.3 Selección del entorno de desarrollo a utilizar.

Se utilizará NetBeans en sus versiones 6.8 o 6.9 como entorno de desarrollo ya que fue utilizado con anterioridad a la hora de la creación de la herramienta ATOM, además de ser las versiones que cuentan con el framework[4] Swing para la visualización y utilización de las interfaces de la herramienta, este framework no está presente en las versiones posteriores del IDE por lo que no se pueden utilizar en la creación del módulo para la posterior integración.

Se tiene en cuenta también que es un IDE libre y gratuito sin restricciones de uso, multilenguaje, multiplataforma y permite el desarrollo de aplicaciones completas para la entrega al cliente, permite también la creación de ventanas, menús, barras de herramientas y otros elementos básicos y fundamentales a la hora de la construcción de un proyecto. A pesar de que Eclipse también cumple con las características expuestas anteriormente, NetBeans cuenta un completamiento de código más eficiente, inteligente y resaltado y mayores facilidades para el diseño de aplicaciones de escritorio.

## 2.4 Framework a utilizar:

## Swing

Swing es un framework de interfaz gráfica para Java. Incluye componentes para interfaz gráfica de usuario

[4] Es un esquema, esqueleto o patrón para el desarrollo y/o la implementación de una aplicación.

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 una interfaz atractiva 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.

## 2.5 Arquitectura de software.

Se entiende por Arquitectura de Software la representación de alto nivel de la estructura de un sistema o aplicación, que describe las partes que la integran, las interacciones entre ellas, los patrones que supervisan su composición y las restricciones a la hora de aplicar esos patrones. (25)

En el campo de la ingeniería de software, el concepto de arquitectura ha sido fundamental como estrategia para enfrentar la complejidad del desarrollo de soluciones informáticas y establecer acuerdos de diseño de alto nivel para estas. En una arquitectura de software confluyen tres elementos fundamentales:

- Los modelos que definen la estructura, topología y dinámica del sistema.
- La trazabilidad o correspondencia de dichos modelos con los requisitos o necesidades establecidas en el contexto que va a operar la solución.
- Las reglas, los principios y justificaciones que rigen la arquitectura y que sustentan las decisiones que se tomaron.

La arquitectura de software es fundamental a la hora de garantizar que la solución cumpla con los criterios de calidad establecidos en los requisitos. (26) El arquitecto es quien define las tecnologías y herramientas a utilizar, como frameworks, estilos arquitectónicos, lenguajes de implementación y patrones de diseño.

Así se estará construyendo una base sobre la cual se desarrollará un software fiable, económico y correspondiente a las necesidades y requisitos del cliente.

### 2.5.1 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[5] 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. (27)

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. (28)

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.

Las vistas:

Una vista obtiene del modelo solamente la información que necesita para desplegar y se actualiza cada

[5] Es un lenguaje de programación que se utiliza para el desarrollo de páginas de Internet.

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. (29)

¿Qué ventajas trae utilizar el MVC?

Es posible tener diferentes vistas para un mismo modelo.

Es posible construir nuevas vistas sin necesidad de modificar el modelo subyacente.

Proporciona un mecanismo de configuración a componentes complejos muchos más tratable que el puramente basado en eventos (el modelo puede verse como una representación estructurada del estado de la interacción). (30)

## 2.6 Conclusiones del Capítulo

Para la realización de este capítulo se realizó un estudio detallado y profundo de las herramientas, arquitectura, lenguaje y metodología que se utilizará para la construcción de la aplicación. Se adquiere Java como lenguaje de programación para el desarrollo de las funcionalidades y NetBeans como entorno de desarrollo.

Como metodología será conveniente el uso de Programación Extrema para contrarrestar los diferentes cambios que puedan ocurrir en el desarrollo del sistema, además de que mantiene una constante comunicación con el cliente.

# Capítulo 3 Presentación de la Solución Propuesta

En este capítulo se estará realizando una valoración de las primeras dos fases del ciclo de vida de la metodología XP, exploración y planificación. Se establecen las historias de usuarios, las cuales servirán para un mejor entendimiento y conocimiento del software. Estas historias de usuarios son escritas por el cliente representando en estas las principales necesidades del sistema.

## 3.1 Fase de 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. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología. (31)

### 3.1.1 Historias de Usuarios

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.

Respecto de la información contenida en la historia de usuario, existen varias plantillas sugeridas pero no existe un consenso al respecto. En muchos casos sólo se propone utilizar un nombre y una descripción o sólo una descripción, más quizás una estimación de esfuerzo en días. Beck en su libro presenta un ejemplo de ficha (*customer story and task card*) en la cual pueden reconocerse los siguientes contenidos: fecha, tipo de actividad (nueva, corrección, mejora), prueba funcional, número de historia, prioridad técnica y del cliente, referencia a otra historia previa, riesgo, estimación técnica, descripción, notas y una lista de

seguimiento con la fecha, estado, elementos por terminar y comentarios. (31)

### 3.1.2 Relación de las Historias de Usuarios

Como parte del proceso de trabajo dentro de la fase de exploración se identificaron las siguientes Historias de Usuarios:

| Historia de Usuario | |
|---|---|
| **Número:** 1. | **Nombre de historia:** Seleccionar Proyecto. |
| **Usuario:** Cliente. | |
| **Prioridad en el negocio:** Alta. | **Riesgo en Desarrollo:** Alto. |
| **Puntos estimados:** 1. | **Iteración asignada:** 1. |
| **Programador responsable:** Emir Mutajar Salej Hernández**.** | |
| **Descripción**: Permite al usuario al seleccionar un proyecto observar la interfaz visual que presenta el mismo. | |
| **Observaciones:** | |

***Tabla # 1*** **HU Seleccionar Proyecto.**

## 3.2 Fase de 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. 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. (31)

### 3.2.1 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.

| Historias de Usuarios | Puntos de Estimación |
|---|---|
| Seleccionar Proyecto. | 1 semana. |
| Mostrar una barra de componentes. | 1 semana. |

| | |
|---|---|
| Adicionar Componentes. | 1 semana. |
| Cambiar posición de los Componentes. | 1 semana. |
| Cambiar ícono de los Componentes. | ½ semana. |

***Tabla # 2*** **Estimación de Esfuerzos**

### 3.2.2 Plan de Iteraciones

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. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción.

Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores. (31)

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 iteraciones, las cuales se detallan a continuación:

**Iteración # 1:**

La primera iteración tiene como objetivo la implementación de las HU #1, #2 con prioridad para el cliente de Alta, Alta y Alta respectivamente. En esta fase la HU Seleccionar Proyecto es la base para implementar las demás iteraciones. Se dispone de 3 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 #3, #4, #5 las cuales hacen alusión a la configuración y cambio de los componentes. 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.

### 3.2.3 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.

| No Iteración | Historias de Usuario | Duración Total de Iteraciones |
|---|---|---|
| Iteración # 1 | Seleccionar Proyecto. | 2 semanas. |
| | Mostrar una barra de componentes. | |
| Iteración # 2 | Adicionar Componentes. | 3 semanas. |
| | Cambiar posición de los Componentes. | |
| | Cambiar ícono de los Componentes. | |

***Tabla # 3*** **Plan de duración de Iteraciones**

### 3.2.4 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.

| Nro. Iteración | Duración (Iter.) | Fecha Inicio. | Fecha Final. |
|---|---|---|---|
| Iteración # 1 | 3 semanas. | 1-2-2013. | 21-2-2013. |
| Iteración # 2 | 2 semanas. | 22-2-2013. | 6-3-2013. |

## 3.3 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, abordando la información correspondiente a la fase de exploración y planificación. Se generó la mayor cantidad de artefactos posibles, siendo así, la fase de exploración permitió representar brevemente el comportamiento del sistema, concluyendo a una correcta y fluida comunicación entre el proyecto Aplicativos SIG como cliente y el equipo de desarrollo. Lo anterior desarrollado permite lograr el sistema deseado y poder satisfacer la necesidad del cliente.

# Capítulo 4 Construcción y validación de la solución propuesta.

En el presente capítulo se realiza una valoración de las fases de construcción y prueba de la metodología XP abordadas anteriormente, incidiendo fundamentalmente en las restantes fases de la metodología, se describen las tarjetas CRC (Contenido, Responsabilidad y Colaboración) para un mejor entendimiento del sistema. Además se exponen las tareas de programación o ingeniería generadas por cada historia de usuario y las pruebas de aceptación efectuadas sobre el sistema.

## 4.1 Diseño de la solución propuesta.

La metodología XP plantea que el desarrollo de una aplicación o software debe realizarse de forma iterativa y obtener al término de cada iteración un producto funcional que debe ser probado; además debe ser mostrado al cliente y al emitir una opinión se pueda obtener una mejor visión de lo que desea. Esta metodología está dirigida a potenciar las relaciones interpersonales como premisa para lograr el mejor resultado posible en el desarrollo del software, proponiendo un trabajo en equipo y buscando la preparación de los desarrolladores para lograr un buen ambiente de trabajo.

En las aplicaciones desarrolladas bajo las pautas de XP no se requiere la representación del sistema mediante diagramas de clases utilizando notación UML, al utilizarse otras técnicas como las tarjetas CRC. Sin embargo se puede hacer uso de los diagramas cuando estos aporten mejoras a la hora de la comunicación entre los desarrolladores y su enfoque en la información importante.

### 4.1.1 Tarjetas CRC.

XP estimula el uso de tarjetas CRC como un mecanismo eficaz para pensar en el software en un contexto orientado a objetos. Las tarjetas CRC identifican y organizan las clases orientadas a objetos que son relevantes para el incremento actual de software. Las tarjetas CRC son el único producto de trabajo de diseño que se generan como parte del este proceso. (32)

Cada tarjeta contiene el nombre de la clase que representa, además existe una descripción de las responsabilidades (métodos) asociados con la clase, así como una lista de otras clases relacionados mediante el envío de mensajes. El sistema CRC se usa principalmente como herramienta didáctica y

como metodología para estudiar la conducta de los diseñadores orientados a objetos. Las tarjetas CRC son también un recordatorio y ayudan a los programadores experimentados y principiantes a comunicarse entre si acerca de la modelación del entorno con objetos. (33)

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.

| 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 #28: Plantilla para Tarjetas CRC.***

| **Tarjeta CRC** | |
|---|---|
| **Clase:** Plugin | |
| **Responsabilidades:**<br>getAjaxFile<br>getCssFile<br>getPluginName<br>getPluginPath<br>Separator<br>isOutOfProjectPath<br>setToolButtons | **Colaboraciones:** |

| |
|---|
| getIconPath<br>getCssClassName |

***Tabla #10:*** **Plugin.**

## 4.2 Desarrollo de las iteraciones.

En la fase de planificación de la metodología XP se detallaron cada una de las HU correspondientes a cada iteración según la selección del cliente. Durante el desarrollo de las iteraciones se va realizando una revisión del plan de iteraciones producto de que el mismo puede o no ser modificado. Como parte de este plan se crean tareas para ayudar a organizar la implementación exitosa de la HU.

Estas tareas en la que se descomponen las HU son las llamadas tareas de ingeniería, que son asignadas a una persona perteneciente al equipo de desarrollo, que el mismo sea quien responda por la implementación asignada. Estas tareas son escritas por los programadores y no por el cliente, dado que son para uso estricto de los programadores.

La planificación que se llevó a cabo para el desarrollo del sistema, está compuesta por dos iteraciones, permitiendo que al final de la última iteración se logre un producto con todas las restricciones y características deseadas por el cliente. A continuación se detallan cada una de las tareas de la ingeniería por iteraciones.

### 4.2.1 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.

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

| | | |
|---|---|---|
| Seleccionar Proyecto | 1 | 1 |
| Mostrar una barra de componentes | 1 | 1 |

***Tabla #15: HU abordadas en la primera iteración.***

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

| Tareas de la Ingeniería | |
|---|---|
| **No. de la tarea:** 1. | **No. de la HU:** 1. |
| **Nombre de la tarea:** Cargar configuración de la interfaz visual. | |
| **Tipo de tarea:** Desarrollo. | **Puntos estimados**: 1 |
| **Fecha inicio**: 1-2-2013. | **Fecha fin:** 8-2-2013. |
| **Programador responsable:** Emir Mutajar Salej Hernández. | |
| **Descripción:** Cada proyecto seleccionado debe mostrar la interfaz de usuario que presenta ese en hasta ese momento. | |

***Tabla #16: Tarea #1 de la HU #1.***

### 4.2.2 Iteración 2:

Esta iteración tiene como finalidad desarrollar las HU #3, #4 y #5.

| Historias de Usuario | Tiempo de implementación (semanas) | |
|---|---|---|
| | **Estimación** | **Real** |
| Adicionar Componentes | 1 | 1 |
| Cambiar posición de los Componentes | 1 | 1 |
| Cambiar ícono de los Componentes | $^1/_2$ | $^1/_2$ |

***Tabla #18: HU abordadas en la segunda iteración.***

| **Tareas de la Ingeniería** | |
|---|---|
| **No. de la tarea:** 1. | **No. de la HU:** 3. |
| **Nombre de la tarea:** Seleccionar Componentes. | |
| **Tipo de tarea:** Desarrollo. | **Puntos estimados:** $^1/_2$ |
| **Fecha inicio:** 5-2-2013. | **Fecha fin:** 7-2-2013. |
| **Programador responsable:** Emir Mutajar Salej Hernández. | |
| **Descripción:** Permite al usuario seleccionar de una lista de componentes los que desea incluir en el proyecto seleccionado. | |

***Tabla #19: Tarea #1 de la HU #3.***

## 4.3 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.

### Requisitos de usabilidad

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

### Fiabilidad

**RNF 2.** El módulo no modifica perjudicialmente los archivos y datos implicados en el proceso de personalización.

### Soporte

**RNF 3.** Cualquier cambio ocurrido en la plataforma GeneSIG que no cumpla con los estándares de la aplicación, es responsabilidad del administrador del sistema actualizar los cambios realizados.

### Restricciones de diseño

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

### Interfaz

**RNF 5.** El módulo debe tener un diseño sencillo, donde no sea necesario mucho entrenamiento para ser utilizado.

**RNF 6.** Los botones utilizados en la aplicación deben tener un nombre o descripción entendible para el usuario.

### Requisitos de hardware

**RNF 7.** La PC de trabajo debe tener como mínimo 256 de RAM y 10 GB de disco duro.

### Requisitos de software

**RNF 8.** Sistemas operativos Ubuntu versión 9.4 en adelante.

**RNF 9.** Tener instalada la máquina virtual de Java.

**RNF 10.** Tener al menos una plataforma GeneSIG.

### Requisitos de Licencia

**RNF 11.** La arquitectura, de acuerdo a los tipos de licencias de las herramientas que se van a utilizar, 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 12.** La mayoría de las herramientas de desarrollo son libres y las licencias están avaladas.

### Estándares Aplicables

**RNF 13.** El módulo será desarrollado bajo estándares internacionales como la normativa ISO19115.

## 4.4 Pruebas.

Uno de los pilares de la Programación Extrema es el proceso de pruebas. XP anima a probar constantemente tanto como sea posible. 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. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones.

XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final. (34)

### 4.4.1 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: (34)

- 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.

### 4.4.2 Pruebas de aceptación.

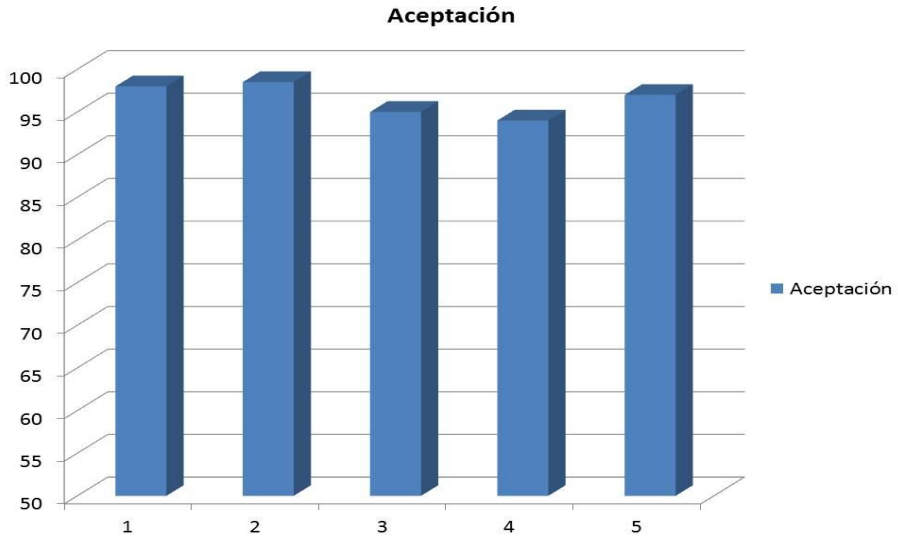
Debido a las características del trabajo a realizar, se determina que las pruebas de aceptación son más importantes que las pruebas unitarias ya que el módulo a desarrollar presenta como objetivo principal el trabajo con interfaces de usuarios y se obtendrá una mejor resultado en función de la aceptación del cliente realizando este tipo de pruebas, además marcarán 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. (34)

| Caso de prueba de aceptación | |
|---|---|
| **Código:** HU1_PA1 | **No. De la HU:** 1 |

| |
|---|
| **Nombre:** Seleccionar Proyecto. |
| **Descripción:** Permite al usuario al seleccionar un proyecto observar la interfaz visual que presenta el mismo. |
| **Condiciones de Ejecución:** Debe existir al menos un proyecto. |
| **Entrada/ Pasos de ejecución:** Se selecciona un proyecto de los cargados en la herramienta ATOM. |
| **Resultado Esperado:** Es mostrada la interfaz visual que presenta el proyecto seleccionado hasta ese momento. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

***Tabla #22:*** **HU1_PA1**

Se realizó una prueba piloto a un total de 7 usuarios para comprobar el nivel de aceptación de las principales funcionalidades, arrojando los siguientes datos:

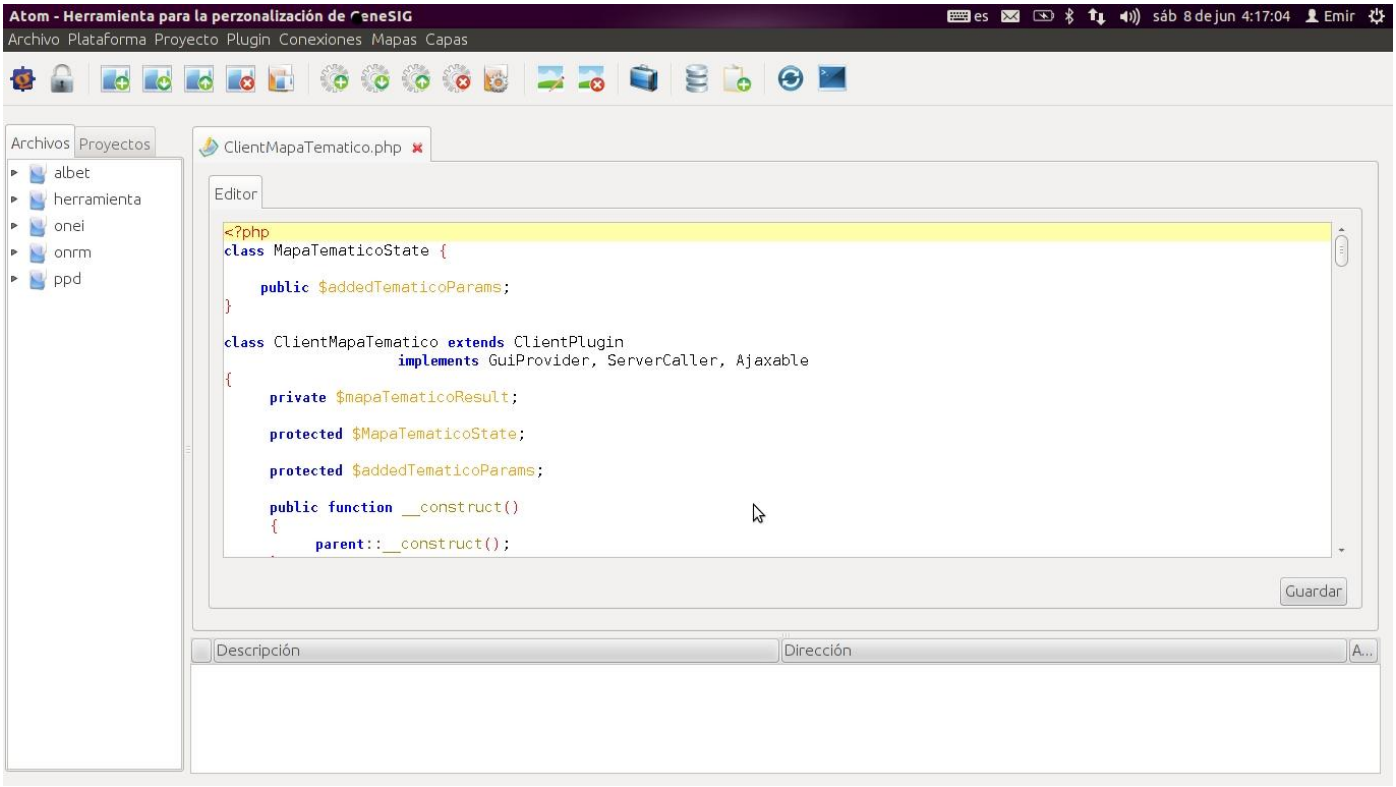



1: Seleccionar Proyecto. 2: Mostrar una barra de componentes. 3: Adicionar componentes. 4: Cambiar posición de los componentes. 5: Cambiar íconos de los componentes.

Como se puede observar en la gráfica los resultados obtenidos en la prueba fueron satisfactorios, teniendo un nivel de aceptación del 95,2%.

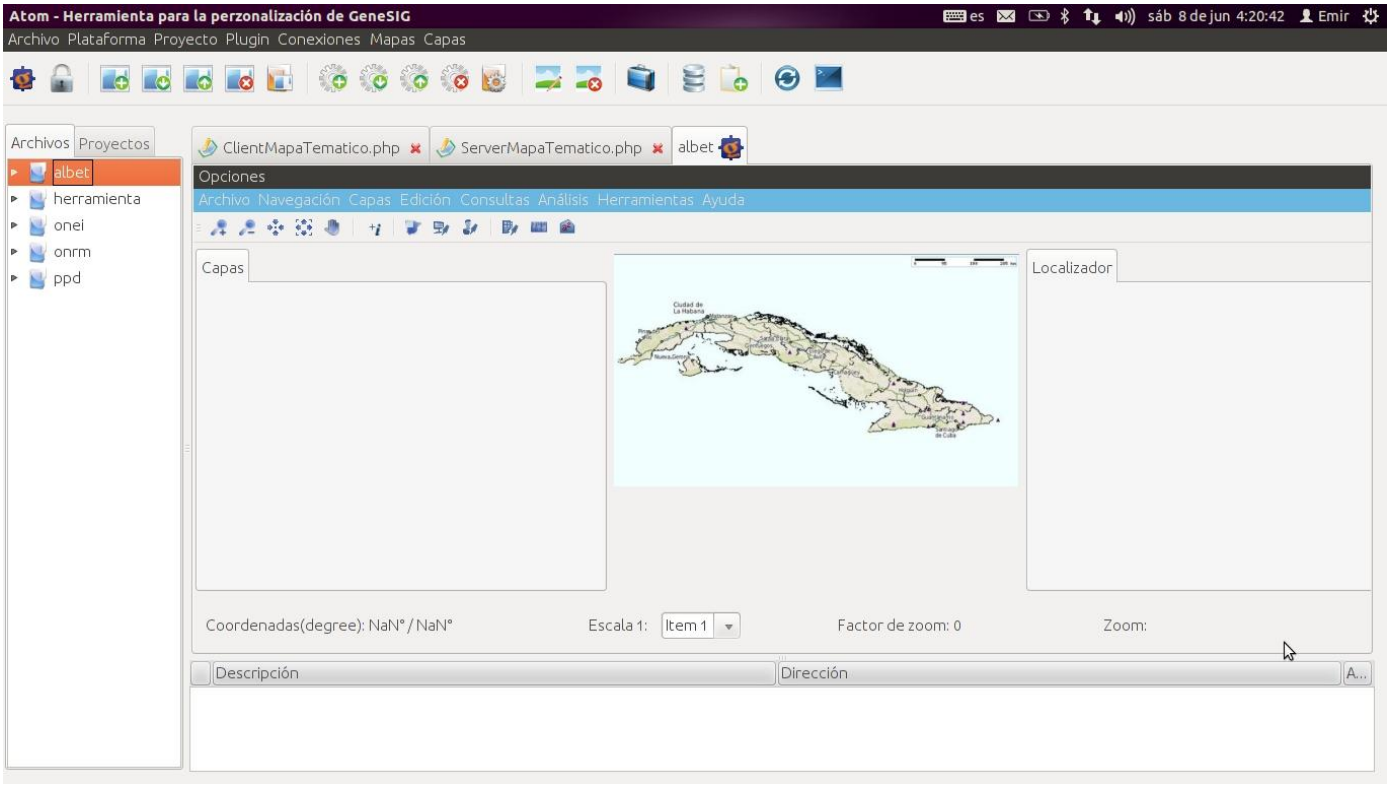
Luego de la última entrega y posterior proceso de pruebas, se cuenta con un módulo para la visualización de la interfaz de usuario de los SIG creados en la herramienta ATOM. A continuación se muestra en imágenes el resultado obtenido luego de la investigación y el proceso de desarrollo del módulo antes mencionado:

Imagen 1. Herramienta ATOM antes de la implementación del módulo.



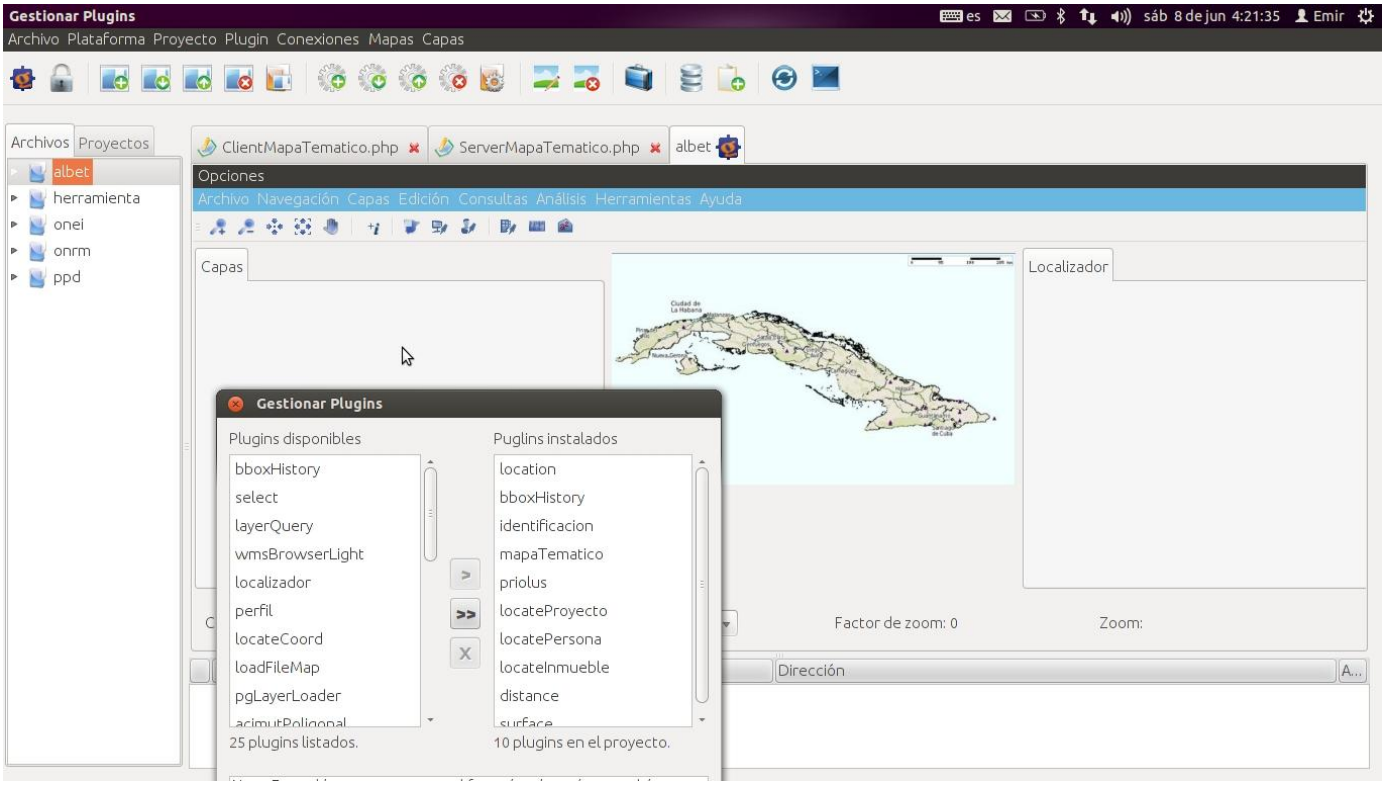

Como se puede apreciar, durante el proceso de creación de los SIG no se podía observar la interfaz visual que presentaba el mismo.

Imagen 2. Herramienta ATOM luego de la implementación del módulo.




Luego del desarrollo del módulo para la visualización de la interfaz visual de los SIG en la herramienta, se puede observar la estructura que presentan, además de contar con una serie de funcionalidades que brindarán una mayor comodidad al usuario, todas ellas desde el entorno creado por el módulo.

Imagen 3. Ejemplo de funcionalidad desde el módulo.




La imagen muestra la funcionalidad de gestión de *Plugins* o Componentes, donde se pueden seleccionar los mismos y adicionarlos al proyecto en el que se trabaja.

## 4.5 Conclusiones Parciales.

Se realizaron construcciones y validaciones en una versión de la solución propuesta que corresponde con el 95% de la aplicación final. Se desarrollaron las fases de construcción y prueba de la metodología XP, describiéndose las tarjetas CRC para lograr el mayor entendimiento posible del sistema. Se muestran en detalle las 2 iteraciones ejecutadas en la fase de construcción de la aplicación, así como sus tareas de programación e ingeniería. Para la comprobación del funcionamiento del software se realizó un estudio de las pruebas unitarias y de aceptación, siendo las pruebas de aceptación las de mayor importancia ya que en el módulo desarrollado se necesita tener la aprobación del cliente respecto a su apariencia y funcionalidad.

## Conclusiones Generales

Se cumplen los objetivos planteados al contar con un módulo para la visualización de la interfaz de usuario para la herramienta ATOM, el mismo cumple con todos los requisitos previstos para su desarrollo.

Mediante el estudio de las distintas aplicaciones o entornos de desarrollo que son capaces de crear interfaces de usuarios se logró establecer un prototipo de interfaz para el módulo desarrollado, el mismo presenta un diseño simple e intuitivo y logra mantener un equilibrio entre comodidad y profesionalidad en su uso.

El uso de las tecnologías y lenguajes de programación seleccionados para el desarrollo del módulo fue satisfactorio, además de las pruebas de aceptación realizadas al producto final, logrando un 100% de satisfacción del cliente.

Se creó además la documentación técnica relacionada con la herramienta, en la misma se detallan todas las fases desarrolladas a partir de la aplicación de la metodología de desarrollo de software XP.

La implementación del módulo significa un aporte importante para el proyecto de AplicativosSIG del Centro de Desarrollo de Geoinformática y Señales Digitales (GEySED) de la UCI, debido a la simplificación y comodidad que brinda este a la hora del desarrollo de SIG con la herramienta ATOM.

# Bibliografía

## Trabajos citados


1. **Confederación de Empresarios de Andalucía.** Sistemas de información geográfica y aplicaciones empresariales. [En línea] 2010. http://sig.cea.es/SIG.

2. **Espinosa, Membrides.** *Plataforma GeneSIG.* La Habana, 2008.

3. **Villafuerte, Deymor B. Centty.** BIBLIOTECA VIRTUAL de Derecho, Economía y Ciencias Sociales. [En línea] 2001. http://www.eumed.net/libros-gratis/2010e/816/METODO%20LOGICO%20HISTORICO.htm.

4. **López, Genaro Luis García.** Los sistemas automatizados de acceso a la información bibliográfica: Evaluación y tendencias en la era de internet. pág. 183.

5. **Bosques, Sandra.** *Sistemas de Información Geográfica.* 1997.

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

7. Diseño y Programación. [En línea] 2013. http://www.abcdisegno.com/nuevas-tendencias-de-diseno/.

8. Ingeniería en Software. [En línea] 2013. http://ejemploparaingenieria.blogspot.de/2013/02/ingenieria-en-software-primera.html.

9. **Universidad Rey Juan Carlos.** escet.urjc. [En línea] 2013. www.escet.urjc.es/~gtazon/IS/Metodologias.pdf.

10. *Metodologías Ágiles en el Desarrollo de Software.* **José H. Canós, Patricio Letelier yMª Carmen Penadés.** Valencia : s.n., 2013.

11. **Gimson, Lic. Loraine.** Metodologías ágiles y desarrollo basado en conocimiento. s.l. : UNIVERSIDAD NACIONAL DE LA PLATA, junio 2012.

12. *Metodologías ágiles para el desarrollo de software: Xtreme Programming (XP).* **Penadés, Patricio Letelier y Mª Carmen.** Valencia : Universidad Politécnica de Valencia.

13. Universo. [En línea] Universidad de Veracruz- México, 11 de junio de 2012. http://www.uv.mx/universo/486/infgral/infgral_15.html.

14. **J.J. Gutierrez, M.J. Escalona, M.Mejias, J.Torres.** Sistema de Programación Extrema. [En línea] 2012. http://www.lsi.us.es/~javierj/investigacion_ficheros/PSISEXTREMA.pdf.

# Bibliografía


15. **Ruíz, David.** Red del Conocimiento. *www.reddelconocimiento.org.* [En línea] 2013. http://www.reddelconocimiento.org/profiles/blogs/lenguaje-de-programaci-n.

16. Lenguajes Programación. [En línea] 2013. http://www.lenguajes-de-programacion.com/programacion-java.shtml.

17. IEC-CSIC. [En línea] 2013. http://www.iec.csic.es/criptonomicon/java/quesjava.html.

18. **Delgado, Yordan.** Teoría de Programación. [En línea] 2012. http://teoria-de-programacion.globered.com/categoria.asp?idcat=34..

19. Lenguaje C++. [En línea] 2012. http://www.zator.com/Cpp/E1_2.htm..

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

21. Programación y Desarrollo. [En línea] 2013. http://programaciondesarrollo.es/que-es-un-entorno-de-desarrollo-integrado-ide/.

22. Grupo de Programación del plantel 34. [En línea] 2013. http://infowebc.net46.net/entornos-de-desarrollo.html.

23. Dosideas. [En línea] 2013. http://www.dosideas.com/wiki/NetBeans.

24. Entornos de Desarrollo Inrtegrado. [En línea] 2012. http://petra.euitio.uniovi.es/~i1667065/HD/documentos/Entornos%20de%20Desarrollo%20Integrado.pdf.

25. **Isidro Ramos Salavert, María Dolores Lozano Pérez.** *Ingeniería Del Software Y Bases de Datos: Tendencias Actuales.* 2000. pág. 62.

26. **Atuesta, Claudia Maria Zea Restrepo y Maria del Rosario.** *Hacia Una Comunidad Educativa Interactiva.* Medellin : s.n., 2007. pág. 97.

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

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

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

30. **Sebastián, Ing. Juan.** Comusoft. [En línea] 2013. http://www.comusoft.com/modelo-vista-controlador-

definicion-y-caracteristicas.

31. *Metodologías ágiles para el desarrollo de software: Extreme Programming (XP).* **Penadés, Patricio Letelier y Mª Carmen.** Valencia : Universidad Politécnica de Valencia, 2013.

32. Sities Google XP_Metodologías. [En línea] 2013. https://sites.google.com/site/xpmetodologia/marco-teorico/funcionamiento.

33. Prezi. [En línea] 2013. http://prezi.com/9hc9opeav71b/tarjetas-crc/.

34. **Javier J. Gutierrez, M. J. Escalona, M. Mejías, J. Torres.** Sistemas de Programación Extrema. [En línea] http://www.lsi.us.es/~javierj/investigacion_ficheros/PSISEXTREMA.pdf.

# Bibliografía

1. **Confederación de Empresarios de Andalucía.** Sistemas de información geográfica y aplicaciones empresariales. [En línea] 2010. http://sig.cea.es/SIG.

2. **Espinosa, Membrides.** *Plataforma GeneSIG.* La Habana, 2008.

3. **Villafuerte, Deymor B. Centty.** BIBLIOTECA VIRTUAL de Derecho, Economía y Ciencias Sociales. [En línea] 2001. http://www.eumed.net/libros-gratis/2010e/816/METODO%20LOGICO%20HISTORICO.htm.

4. **López, Genaro Luis García.** Los sistemas automatizados de acceso a la información bibliográfica: Evaluación y tendencias en la era de internet. pág. 183.

5. **Bosques, Sandra.** *Sistemas de Información Geográfica.* 1997.

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

7. Diseño y Programación. [En línea] 2013. http://www.abcdisegno.com/nuevas-tendencias-de-diseno/.

8. Ingeniería en Software. [En línea] 2013. http://ejemploparaingenieria.blogspot.de/2013/02/ingenieria-en-software-primera.html.

9. **Universidad Rey Juan Carlos.** escet.urjc. [En línea] 2013. www.escet.urjc.es/~gtazon/IS/Metodologias.pdf.

10. *Metodologías Ágiles en el Desarrollo de Software.* **José H. Canós, Patricio Letelier yMª Carmen Penadés.** Valencia : s.n., 2013.

11. **Gimson, Lic. Loraine.** Metodologías ágiles y desarrollo basado en conocimiento. s.l. : UNIVERSIDAD NACIONAL DE LA PLATA, junio 2012.

12. *Metodologías ágiles para el desarrollo de software: Xtreme Programming (XP).* **Penadés, Patricio Letelier y Mª Carmen.** Valencia : Universidad Politécnica de Valencia.

13. Universo. [En línea] Universidad de Veracruz- México, 11 de junio de 2012. http://www.uv.mx/universo/486/infgral/infgral_15.html.

14. **J.J. Gutierrez, M.J. Escalona, M.Mejias, J.Torres.** Sistema de Programación Extrema. [En línea] 2012. http://www.lsi.us.es/~javierj/investigacion_ficheros/PSISEXTREMA.pdf.

# Bibliografía

15. **Ruíz, David.** Red del Conocimiento. *www.reddelconocimiento.org.* [En línea] 2013. http://www.reddelconocimiento.org/profiles/blogs/lenguaje-de-programaci-n.

16. Lenguajes Programación. [En línea] 2013. http://www.lenguajes-de-programacion.com/programacion-java.shtml.

17. IEC-CSIC. [En línea] 2013. http://www.iec.csic.es/criptonomicon/java/quesjava.html.

18. **Delgado, Yordan.** Teoría de Programación. [En línea] 2012. http://teoria-de-programacion.globered.com/categoria.asp?idcat=34..

19. Lenguaje C++. [En línea] 2012. http://www.zator.com/Cpp/E1_2.htm..

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

21. Programación y Desarrollo. [En línea] 2013. http://programaciondesarrollo.es/que-es-un-entorno-de-desarrollo-integrado-ide/.

22. Grupo de Programación del plantel 34. [En línea] 2013. http://infowebc.net46.net/entornos-de-desarrollo.html.

23. Dosideas. [En línea] 2013. http://www.dosideas.com/wiki/NetBeans.

24. Entornos de Desarrollo Inrtegrado. [En línea] 2012. http://petra.euitio.uniovi.es/~i1667065/HD/documentos/Entornos%20de%20Desarrollo%20Integrado.pdf.

25. **Isidro Ramos Salavert, María Dolores Lozano Pérez.** *Ingeniería Del Software Y Bases de Datos: Tendencias Actuales.* 2000. pág. 62.

26. **Atuesta, Claudia Maria Zea Restrepo y Maria del Rosario.** *Hacia Una Comunidad Educativa Interactiva.* Medellin : s.n., 2007. pág. 97.

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

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

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

30. **Sebastián, Ing. Juan.** Comusoft. [En línea] 2013. http://www.comusoft.com/modelo-vista-controlador-

definicion-y-caracteristicas.

31. *Metodologías ágiles para el desarrollo de software: Extreme Programming (XP).* **Penadés, Patricio Letelier y Mª Carmen.** Valencia : Universidad Politécnica de Valencia, 2013.

32. Sities Google XP_Metodologías. [En línea] 2013. https://sites.google.com/site/xpmetodologia/marco-teorico/funcionamiento.

33. Prezi. [En línea] 2013. http://prezi.com/9hc9opeav71b/tarjetas-crc/.

34. **Javier J. Gutierrez, M. J. Escalona, M. Mejías, J. Torres.** Sistemas de Programación Extrema. [En línea] http://www.lsi.us.es/~javierj/investigacion_ficheros/PSISEXTREMA.pdf.

35. eumed (Biblioteca). [En línea] 2013. http://www.eumed.net/libros-gratis/2009c/584/Metodologias%20tradicionales%20y%20metodologias%20agiles.htm.

36. **Roberth G. Figueroa, Camilo J. Solís, Armando A. Cabrera.** *Metodologias Tradicionales Vs Metodologías Ágiles.* Valencia : Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación.

37. Comunidad virtual del departamento de sistemas de la Universidad del Cauca (Colombia). [En línea] 2013. http://pepemxl.zxq.net/cursos/lenguaje_2012/slides/slide_18.pdf.

38. Ecured (Metodologías_desarrollo). [En línea] 2013. http://www.ecured.cu/index.php/Metodolog%C3%ADas_de_desarrollo_de_software.

39. Ecured (Metodología_ágil). [En línea] 2013. http://www.ecured.cu/index.php/Metodolog%C3%ADa_%C3%A1gil.

40. IBM. [En línea] 2013. www.ibm.com/software/awdtools/rup/.

41. Ecured (L.Prog). [En línea] 2013. http://www.ecured.cu/index.php/Lenguaje_de_Programaci%C3%B3n.

42. Ecured (Entorno Prog). [En línea] 2013. http://www.ecured.cu/index.php/IDE_de_Programaci%C3%B3n.

# Anexos

| Historia de Usuario | |
|---|---|
| **Número:** 1. | **Nombre de historia:** Seleccionar Proyecto. |
| **Usuario:** Cliente. | |
| **Prioridad en el negocio:** Alta. | **Riesgo en Desarrollo:** Alto. |
| **Puntos estimados:** 1. | **Iteración asignada:** 1. |
| **Programador responsable:** Emir Mutajar Salej Hernández**.** | |
| **Descripción**: Permite al usuario al seleccionar un proyecto observar la interfaz visual que presenta el mismo. | |
| **Observaciones:** | |

***Tabla # 1*** **HU Seleccionar Proyecto.**

| Historia de Usuario | |
|---|---|
| **Número:** 2. | **Nombre de historia:** Mostrar una barra de componentes. |

| | |
|---|---|
| **Usuario:** Cliente**.** | |
| **Prioridad en el negocio:** Alta. | **Riesgo en Desarrollo:** Alto. |
| **Puntos estimados:** 1**.** | **Iteración asignada:** 1. |
| **Programador responsable:** Emir Mutajar Salej Hernández. | |
| **Descripción:** Permite al usuario observar la lista de componentes disponibles, para luego incorporarlos al proyecto. | |
| **Observaciones**: Se mostrarán los componentes que no contenga el proyecto seleccionado. | |

***Tabla # 2*** **HU Mostrar una barra de componentes.**

| **Historia de Usuario** | |
|---|---|
| **Número:** 3. | **Nombre de historia:** Adicionar componentes. |
| **Usuario:** Cliente. | |
| **Prioridad en el negocio:** Alta. | **Riesgo en Desarrollo:** Alto. |
| **Puntos estimados:** 1. | **Iteración asignada:** 2. |

| |
|---|
| **Programador responsable:** Emir Mutajar Salej Hernández. |
| **Descripción:** Permite al usuario adicionar componentes desde la barra que los contiene al proyecto en el cual se trabaja. |
| **Observaciones:** |

***Tabla # 3* HU Cargar Plataforma.**

| Historia de Usuario | |
|---|---|
| **Número:** 4. | **Nombre de historia:** Cambiar posición de los componentes. |
| **Usuario:** Cliente. | |
| **Prioridad en el negocio:** Media. | **Riesgo en Desarrollo:** Alto. |
| **Puntos estimados:** 1. | **Iteración asignada:** 2. |
| **Programador responsable:** Emir Mutajar Salej Hernández. | |
| **Descripción:** Permite al usuario cambiar la posición de los componentes que se están utilizando. Esto permite que se organicen de la manera más conveniente o cómoda para el usuario. | |
| **Observaciones:** Si un componente cuenta en su composición con más de un ícono se moverán todos los íconos en conjunto. | |

***Tabla # 4* HU Crear Proyecto.**

| Historia de Usuario | |
|---|---|
| **Número:** 5. | **Nombre de historia:** Cambiar íconos de los componentes. |
| **Usuario:** Cliente. | |
| **Prioridad en el negocio:** Baja. | **Riesgo en Desarrollo:** Medio. |
| **Puntos estimados:** 0,5. | **Iteración asignada:** 2. |
| **Programador responsable:** Emir Mutajar Salej Hernández. | |
| **Descripción:** Permite al usuario cambiar la imagen de cualquier ícono. | |
| **Observaciones:** | |

***Tabla # 5* HU Eliminar Proyecto.**

| Historias de Usuarios | Puntos de Estimación |
|---|---|
| Seleccionar Proyecto. | 1 semana. |
| Mostrar una barra de componentes. | 1 semana. |
| Adicionar componentes. | 1 semana. |

| | |
|---|---|
| Cambiar posición de los componentes. | 1 semana. |
| Cambiar íconos de los componentes. | ½ semana. |

***Tabla # 6* Tabla de Estimación de esfuerzos**

| No Iteración | Historias de Usuario | Duración Total de Iteraciones |
|---|---|---|
| Iteración # 1 | Seleccionar Proyecto.<br>Mostrar una barra de componentes. | 2 semanas. |
| Iteración # 2 | Adicionar componentes.<br>Cambiar posición de los componentes.<br>Cambiar íconos de los componentes. | 3 semanas. |

***Tabla # 7* Duración de la Iteraciones**

| **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 #8: Plantilla para Tarjetas CRC.***

| **Tarjeta CRC** |
|---|

| **Clase:** VistaPreviaProyecto | |
|---|---|
| **Responsabilidades:** | **Colaboraciones:** |
| CargarCorePlugin<br>CargarPlugins<br>GoodString<br>GetIconPath<br>GetToolBarButtons<br>Reestructurar<br>CambiarIcono<br>updateCssFile<br>copy<br>LoadToolBarButtons<br>LoadMenuBar<br>isMenu<br>GestionarPlugin | Plugin<br>ToolBarButton<br>vMenuItem<br>vMenu<br>PopUp |

***Tabla #9:*** **VistaPreviaProyecto.**

| **Tarjeta CRC** | |
|---|---|
| **Clase:** Plugin | |
| **Responsabilidades:** | **Colaboraciones:** |
| getAjaxFile<br>getCssFile<br>getPluginName<br>getPluginPath | ToolBarButton |

| |
|---|
| Separator |
| isOutOfProjectPath |
| setToolButtons |
| getIconPath |
| getCssClassName |

***Tabla #10:*** **Plugin.**

| Tarjeta CRC | |
|---|---|
| **Clase:** ToolBarButton | |
| **Responsabilidades:** | **Colaboraciones:** |
| getPlugin | |
| isSeparator | |
| getNombre | |

***Tabla #11:*** **ToolBarButton.**

| Tarjeta CRC | |
|---|---|
| **Clase:** vMenuItem | |
| **Responsabilidades:** | **Colaboraciones:** |
| getPadre | |
| getNombre | |

***Tabla #12:*** **vMenuItem.**

| Tarjeta CRC | |
|---|---|
| **Clase:** vMenu | |
| **Responsabilidades:**<br>getPadre<br>getNombre | **Colaboraciones:** |

***Tabla #13:*** **vMenu.**

| Tarjeta CRC | |
|---|---|
| **Clase:** PopUp | |
| **Responsabilidades:**<br>isIzq<br>getPlugin<br>getToolButton | **Colaboraciones:** |

***Tabla #14:*** **PopUp.**

| Historias de Usuario | Tiempo de implementación (semanas) | |
|---|---|---|
| | **Estimación** | **Real** |
| Seleccionar Proyecto | 1 | 1 |
| Mostrar una barra de componentes | 1 | 1 |

***Tabla #15: HU abordadas en la primera iteración.***

| Tareas de la Ingeniería | |
|---|---|
| **No. de la tarea:** 1. | **No. de la HU:** 1. |
| **Nombre de la tarea:** Cargar configuración de la interfaz visual. | |
| **Tipo de tarea:** Desarrollo. | **Puntos estimados**: 1 |
| **Fecha inicio**: 1-2-2013. | **Fecha fin:** 8-2-2013. |
| **Programador responsable:** Emir Mutajar Salej Hernández. | |
| **Descripción:** Cada proyecto seleccionado debe mostrar la interfaz de usuario que presenta en ese momento. | |

***Tabla #16: Tarea #1 de la HU #1.***

| Tareas de la Ingeniería | |
|---|---|
| **No. de la tarea:** 1 | **No. de la HU:** 2 |
| **Nombre de la tarea:** Mostrar un panel lateral. | |
| **Tipo de tarea:** Desarrollo. | **Puntos estimados:** 1 |
| **Fecha inicio:** 8-2-2013. | **Fecha fin:** 15-2-2013. |
| **Programador responsable:** Emir Mutajar Salej Hernández. | |

| |
|---|
| **Descripción:** Permite al usuario visualizar un panel con los componentes que no estén en el proyecto seleccionado. |

***Tabla #17: Tarea #1 de la HU #2.***

| Historias de Usuario | Tiempo de implementación (semanas) | |
|---|---|---|
| | **Estimación** | **Real** |
| Adicionar Componentes | 1 | 1 |
| Cambiar posición de los Componentes | 1 | 1 |
| Cambiar ícono de los Componentes | $^1/_2$ | $^1/_2$ |

***Tabla #18: HU abordadas en la segunda iteración.***

| **Tareas de la Ingeniería** | |
|---|---|
| **No. de la tarea:** 1. | **No. de la HU:** 3. |
| **Nombre de la tarea:** Selección de los Componentes. | |
| **Tipo de tarea:** Desarrollo. | **Puntos estimados:** 1 |
| **Fecha inicio:** 15-2-2013. | **Fecha fin:** 22-2-2013. |
| **Programador responsable:** Emir Mutajar Salej Hernández. | |

| |
|---|
| **Descripción:** Permite al usuario seleccionar de una lista de componentes lo que desea incluir en el proyecto. |

***Tabla #19: Tarea #1 de la HU #3.***

| Tareas de la Ingeniería | |
|---|---|
| **No. de la tarea:** 2. | **No. de la HU:** 4. |
| **Nombre de la tarea:** Menú de desplazamiento. | |
| **Tipo de tarea:** Desarrollo. | **Puntos estimados:** 1 |
| **Fecha inicio:** 22-2-2013 | **Fecha fin:** 1-3-2013 |
| **Programador responsable:** Emir Mutajar Salej Hernández. | |
| **Descripción:** Permite al usuario seleccionar un componentes y reorganizarlo en la barra de componentes del proyecto dependiendo de su interés. | |

***Tabla #20: Tarea #2 de la HU #4.***

| Tareas de la Ingeniería | |
|---|---|
| **No. de la tarea:** 1 | **No. de la HU:** 5 |
| **Nombre de la tarea:** Seleccionar imagen. | |
| **Tipo de tarea:** Desarrollo. | **Puntos estimados:** $^{1}/_{2}$ |

| | |
|---|---|
| **Fecha inicio:** 1-3-2013 | **Fecha fin:** 4-3-2013 |
| **Programador responsable:** Emir Mutajar Salej Hernández. | |
| **Descripción:** Permite al usuario seleccionar la imagen que desea para un componente determinado. | |

***Tabla #21: Tarea #1 de la HU #5.***

| Caso de prueba de aceptación | |
|---|---|
| **Código:** HU1_PA1 | **No. De la HU:** 1 |
| **Nombre:** Seleccionar Proyecto. | |
| **Descripción:** Permite al usuario al seleccionar un proyecto observar la interfaz visual que presenta el mismo. | |
| **Condiciones de Ejecución:** Debe existir al menos un proyecto. | |
| **Entrada/ Pasos de ejecución:** Se selecciona un proyecto de los cargados en la herramienta ATOM. | |
| **Resultado Esperado:** Es mostrada la interfaz visual que presenta el proyecto seleccionado hasta ese momento. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

***Tabla #22:*** **HU1_PA1**

| Caso de prueba de aceptación | |
|---|---|
| **Código:** HU2_PA1 | **No. De la HU:** 2 |

| |
|---|
| **Nombre:** Mostrar una barra de componentes. |
| **Descripción:** Permite al usuario observar la lista de componentes disponibles, para luego incorporarlos al proyecto. |
| **Condiciones de Ejecución:** Debe existir al menos un componente en la plataforma que no se esté usando por el proyecto seleccionado. |
| **Entrada/ Pasos de ejecución:** Se muestra un panel en la parte derecha del área de trabajo para visualizar los componentes a adicionar. |
| **Resultado Esperado:** El panel debe mostrar solamente los componentes que no se encuentren en el proyecto. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

***Tabla #23:*** **HU2_PA1**

| Caso de prueba de aceptación | |
|---|---|
| **Código:** HU3_PA1 | **No. De la HU:** 3 |
| **Nombre:** Adicionar Componente. | |
| **Descripción:** Permite al usuario adicionar componentes desde la barra que los contiene al proyecto en el cual se trabaja. | |
| **Condiciones de Ejecución:** Debe existir la barra que contiene los componentes no usados y mostrar al menos uno. | |
| **Entrada/ Pasos de ejecución:** Se selecciona uno o varios componentes para añadir al proyecto y se oprime el botón de aceptación de la tarea. | |

| |
|---|
| **Resultado Esperado:** Son incorporados los componentes seleccionados a la barra de componentes del proyecto en función. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

***Tabla #23:*** **HU3_PA1**

| Caso de prueba de aceptación | |
|---|---|
| **Código:** HU4_PA1 | **No. De la HU:** 4 |
| **Nombre:** Cambiar posición de los componentes. | |
| **Descripción:** Permite al usuario cambiar la posición de los componentes que se están utilizando. Esto permite que se organicen de la manera más conveniente y cómoda para el usuario. | |
| **Condiciones de Ejecución:** Deben existir al menos dos componentes. | |
| **Entrada/ Pasos de ejecución:** Se oprime clic derecho encima del componente que se desea cambiar de posición y se accede a la opción de desplazamiento. | |
| **Resultado Esperado:** Es cambiado de posición el componente seleccionado. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

***Tabla #24:*** **HU4_PA1**

| **Caso de prueba de aceptación** | |
|---|---|
| **Código:** HU5_PA1. | **No. De la HU:** 5. |
| **Nombre:** Cambiar íconos de los componentes. | |

| |
|---|
| **Descripción:** Permite al usuario cambiar el ícono de cualquier componentes existente en el proyecto. |
| **Condiciones de Ejecución:** Debe existir al menos un componente en el proyecto. |
| **Entrada/ Pasos de ejecución:** Se oprime clic derecho encima del componente que se desea cambiar de posición y se accede a la opción de cambiar ícono. |
| **Resultado Esperado:** Debe aparecer una ventana para que se proceda a la búsqueda de la imagen por la cual se va a cambiar. |
| **Evaluación de la Prueba:** Prueba satisfactoria. |

***Tabla #25:*** **HU5_PA1**

| **Caso de prueba de aceptación** | |
|---|---|
| **Código:** HU5_PA2. | **No. De la HU:** 5. |
| **Nombre:** Seleccionar imagen. | |
| **Descripción:** Permite al usuario seleccionar la imagen que desea para un componente determinado. | |
| **Condiciones de Ejecución:** Debe existir una imagen por la cual se va a cambiar. | |
| **Entrada/ Pasos de ejecución:** Luego de presentarse la ventana para la búsqueda de una imagen deberá seleccionar una imagen en algún punto del ordenador. | |
| **Resultado Esperado:** Se cambia el pasado ícono del componente por el seleccionado recientemente. | |
| **Evaluación de la Prueba:** Prueba satisfactoria. | |

***Tabla #26:*** **HU5_PA2**

# Glosario de términos

**Software:** Conjunto de programas, instrucciones y reglas informáticas que permiten ejecutar distintas tareas en una computadora.

**Applet:** Es un componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo un navegador web.

**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.

**ISO:** Organización Internacional para la Estandarización o ISO, cuyo nombre en inglés es International Organization for Standardization.

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

**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.

**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.

**Sun Microsystems:** empresa informática fabricante de semiconductores y software.