Universidad de las Ciencias Informáticas
FACULTAD 6

Título: Marco de Trabajo para la implementación de Interfaces de Usuario en Sistemas de Información Geográfica “Geomap UI”.

Autor(es): Aryam Ernesto Guerra Caleo
Rafael Ernesto Vázquez Villar
Tutor(es): MSc. Adonis Ricardo Rosales García
Ing. Pedro Enrique Palau Isaac

La Habana, Junio de 2014
“Año del 56 Aniversario de la Revolución”

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____	_____
Firma del autor	Firma del autor
Aryam Ernesto Guerra Caleo	Rafael Ernesto Vázquez Villar
_____	_____
Firma del tutor	Firma del tutor
MSc. Adonis Ricardo Rosales García	Ing. Pedro Enrique Palau Isaac

Datos de contacto

Tutores:

Nombre y apellidos: Adonis Ricardo Rosales García

Sexo: Masculino

Institución: Universidad de las Ciencias Informáticas (UCI)

Dirección de la institución: Carretera San Antonio de Baños Km 2 ½, Boyero

Correo electrónico: arrosales@uci.cu

Teléfono del trabajo: 837 2553 **Teléfono particular:** 837 2197

Categoría docente: Asistente

Título de la especialidad de graduado: Master en Informática Aplicada

Año de graduación: 2008

Institución donde se graduó: Universidad de las Ciencias Informáticas

Nombre y apellidos: Pedro Enrique Palau Isaac

Sexo: Masculino

Institución: Universidad de las Ciencias Informáticas (UCI)

Dirección de la institución: Carretera San Antonio de Baños Km 2 ½, Boyero

Correo electrónico: pepalau@uci.cu

Teléfono del trabajo: 837 3781 **Teléfono particular:** 837 2100

Categoría docente: Ninguna

Título de la especialidad de graduado: Ingeniero en Ciencias Informáticas

Año de graduación: 2011

Institución donde se graduó: Universidad de las Ciencias Informáticas

Agradecimientos

A mis padres, por estar siempre a mi lado, por su comprensión y guía durante toda la vida. Son lo más grande que tengo y mi modelo a seguir.

A mis abuelos y a mi hermano, por darme todo su cariño.

A mis tíos y primos Marielena, Jesús, Yehilin, Yehicel, el flaco y Daryl, por su preocupación y apoyo.

A mi novia por estar a mi lado en todo momento, por su paciencia, por su amor.

A mis suegros por preocuparse constantemente y brindarme su ayuda.

A nuestros tutores, por aconsejarnos y ayudarnos en todo momento.

A todas mis amistades fuera y dentro de la Universidad, sin las cuales no sería la persona que soy hoy. Nunca olvidaré los momentos buenos y malos que compartimos, cuando estudiábamos para cada prueba, o cuando jugábamos futbol en Capua y mucho menos cuando conversábamos horas de cualquier bobería o mentira que surgiera.

A todos los que de una forma u otra formaron parte de mi vida y pusieron su granito de arena en cada momento de este largo camino.

Aryam

Agradezco a mi mamá por todo el amor y el apoyo que me ha dado en todo momento, por todo lo que ha hecho para que se cumpliera este sueño.

A mi hermano por ser mi guía y por darme todo su apoyo cuando me ha hecho falta en las circunstancias más difíciles.

A mis tíos Luis y Gabriel ya que siempre están preocupados por mis estudios y por tratarme siempre como a un hijo.

A mis amistades Yolanda Lecsy y Yordi por brindarme su ayuda incondicional a través de sus consejos y velar por la calidad de esta tesis.

A todos mis amigos de mi antiguo grupo 6103 por compartir estos 5 años en esta universidad, donde pasamos buenos y malos momentos pero al final crecimos como personas y como profesionales… gracias, no los voy a olvidar nunca.

A nuestro tutor Pedro por dedicar todo su esfuerzo, tiempo y empeño en la confección del presente trabajo.

A todos aquellos que, de una forma u otra, colaboraron y me extendieron su mano para seguir adelante y culminar esta tesis.

Rafael

Dedicatoria

A mis padres, Mayra y Jorge.

A mis abuelos, Acela y Mariano

A mi novia, Karla.

A toda mi familia y a mis amistades fuera y dentro de la Universidad.

Aryam

A mi hermano Ramiro.

A mi madre Dulce.

A mi padre Rafe.

A toda mi familia y mis amigos de la UCI y de la infancia.

Rafael

Resumen

El presente trabajo surge por la necesidad del departamento Geoinformática perteneciente al Centro de Desarrollo Geoinformática y Señales Digitales, de contar con una herramienta que facilite y agilice el desarrollo y personalización de los Sistemas de Información Geográfica que se implementan en la Línea de Productos de Software Aplicativos SIG de la Universidad de las Ciencias Informáticas. El objetivo que se persigue es desarrollar un marco de trabajo para la implementación de Interfaces de Usuario para SIG en entornos Web. Dicho marco de trabajo está basado en la arquitectura Modelo Vista Controlador, y para guiar el proceso de desarrollo del software se empleó como metodología el Proceso Unificado de Desarrollo. Mediante el lenguaje de modelado Lenguaje Unificado de Modelado fueron generados todos los artefactos, apoyados por la herramienta de Ingeniería de Software asistida por Computadora Visual Paradigm. Para garantizar la calidad de las distintas funcionalidades de la aplicación se aplicaron a esta una serie de pruebas (Pruebas de Caja Negra, Pruebas de Aceptación Alfa), las cuales arrojaron resultados satisfactorios.

Palabras claves: Marco de trabajo, Personalización, SIG.

Abstract

This paper arises from the need of Geoinformatics department of the Center for Development and Geoinformatics Digital Signals, to have a tool to facilitate and accelerate the development and customization of Geographic Information Systems that are deployed on the Line Software Products University of Informatics Sciences. The objective pursued is to develop a framework for implementing GIS in Web UI environments. This framework is based on the Model View Controller architecture, and to guide the software development process methodology was used as the Rational Unified Process. By modeling language Unified Modeling Language were generated all artifacts, supported by the Engineering Tool Software Computer assisted Visual Paradigm. To ensure the quality of the different functionalities of the application are applied to this series of tests (Black Box Testing, Acceptance Testing Alpha), which yielded satisfactory results.

Keywords: Framework, GIS, Personalization.

Índice de contenido

Índice de contenido

Índice de tablas:

Índice de figuras

Índice de figuras:

Introducción

En Cuba, la Universidad de las Ciencias Informáticas (UCI) ofrece servicios de personalizaciones de Sistemas de Información Geográfica (SIG), desarrollados por la Línea de Productos de Software (LPS) Aplicativos SIG. Un SIG es un sistema que permite crear consultas interactivas, integrar, analizar y representar información georeferenciada asociada a cualquier tipo de fenómeno geográfico, con el fin de resolver problemas complejos de planificación y gestión para la toma de decisiones.

Durante los últimos años, las implementaciones de esas personalizaciones, se han basado principalmente en el uso del marco de trabajo GeneSIG, una herramienta desarrollada en la Universidad y que recibe un soporte continuo por un equipo de profesionales. La utilización de dicha herramienta en la LPS, ha facilitado en gran medida la obtención de productos en menores intervalos de tiempo, ya que ha permitido a los desarrolladores centrar los esfuerzos en la implementación de funcionalidades relacionadas con los negocios para los cuales, se realizan esas personalizaciones y la reutilización de otras, comunes para este tipo de sistemas.

Sin embargo, dado por la cantidad de solicitudes de negocio realizadas a la LPS y el incremento notorio por parte de numerosas entidades en la utilización de SIG en sus procesos de negocio, el equipo de trabajo de la LPS se está enfrentando a diversos inconvenientes durante el desarrollo de personalizaciones que cumplan con ciertos requisitos que imponen los clientes y que son muy comunes entre dichas solicitudes. Estas deficiencias están dadas, principalmente, por la implementación de Interfaz de Usuario (UI, *del inglés User Interface*) orientadas a las pautas de identidad visual (*posicionamiento de logotipos o símbolos, colores, tipografías, entre otras*) que poseen estas entidades, lo que influye negativamente en los índices de satisfacción de los clientes. Por ejemplo, en GeneSIG la Interfaz de Usuario está implementada de una forma estática, que impide establecer una plantilla de visualización diferente para cada personalización y que está compuesta además por regiones y bloques ya predefinidos, difícilmente personalizables o modificables en alguna de sus características.

Asimismo, entre sus herramientas base, GeneSIG utiliza la librería ExtJS para la construcción de la interfaz de usuario y la visualización de la información que se gestiona en dicha plataforma, lo que dificulta la implementación de personalizaciones que se acoplen a las pautas que se mencionaron anteriormente, ya que permite presentar el contenido utilizando cualquier estructura y personalizarla a través de un tema o

plantilla, pero desarrollar plantillas para las personalizaciones, requiere de un tiempo y conocimiento elevados, debido a la complejidad que implica su implementación.

Otra desventaja, está relacionada con los resultados de estudios realizados sobre usabilidad en interfaces de usuarios para SIG. Algunos de estos estudios, como (*Usabilidad: Hacer la Web pensando en el usuario* (Barcelona Activa Cibernarium, 2012), *User-Centered Graphical User Interface Design for GIS* (Lanter and Essinger, 1991), *User Interface Issues in Geographic Information Systems* (Lopes de Oliveira and Bauzer Medeiros, 1996)), han permitido la definición y creación de soluciones no estándares, pero que incrementan las Experiencias de los Usuarios (UX, por sus siglas en inglés) a niveles superiores, pero se dificulta su aplicación en las personalizaciones de estos sistemas usando GeneSIG.

Todas estas deficiencias influyen negativamente en los cronogramas de entrega de los productos y conllevan a la creación de otros más extensos. Además, si por cuestiones propias de cualquier negocio, se necesita entregar un producto sin acceso al código fuente, se estaría violando la licencia GPLv3 de ExtJS, que obliga a entregar el código fuente de cualquier aplicación que la utilice.

Teniendo en cuenta la situación descrita anteriormente, se define entonces, como **problema a resolver**:

El esquema actual de desarrollo de interfaces de usuario en la LPS Aplicativos SIG no responde adecuadamente a las pautas de identidad visual exigidas por los clientes.

Se define como **objeto de estudio**: Marcos de trabajo para la implementación de interfaces de usuarios en entornos Web, estableciéndose como **campo de acción** específicamente en los marcos de trabajo que permitan implementar interfaces de usuarios para SIG en entornos Web.

Para dar solución al problema planteado se define como **objetivo general**: Desarrollar un marco de trabajo para la implementación de interfaces de usuarios para SIG en entornos Web.

Para el cumplimiento del objetivo general se definieron los **objetivos específicos** que se mencionan a continuación:

- Determinar el marco teórico de la investigación que permita conocer las principales tendencias de cómo crear un marco de trabajo para interfaces de usuario en SIG.

- Fundamentar la selección de la metodología, herramientas y tecnologías a utilizar en el desarrollo del marco de trabajo.
- Realizar el análisis, diseño e implementación del marco de trabajo.
- Realizar pruebas al marco de trabajo.

Preguntas de la investigación:

- ¿Cuáles son los fundamentos teóricos de los marcos de trabajo para la implementación de UI en entornos web para Sistemas de Información Geográfica?
- ¿Cuáles son las características que debe cumplir el marco de trabajo para que permitan la estandarización de las UI en los Sistemas de Información Geográfica de la LPS Aplicativos SIG?
- ¿Cómo estructurar el proceso de diseño e implementación del marco de trabajo para la LPS Aplicativos SIG?
- ¿La versión desarrollada por la LPS Aplicativos SIG permite el diseño de UI en entornos web para la LPS Aplicativos SIG?

Tareas de la investigación:

- Estudio y análisis de la información relacionada con los marcos de trabajo y del proceso mediante el cual se desarrollan, para conocer sus características y funcionalidades.
- Comparación de los marcos de trabajo para la implementación de interfaces de usuario que aporten de manera parcial o total una solución al problema a resolver definido.
- Caracterización de las herramientas y tecnologías a utilizar durante el desarrollo de la solución propuesta para entender mejor su funcionamiento y explotar sus funcionalidades.
- Identificación de las características funcionales y no funcionales que deberá cumplir la solución propuesta para su posterior diseño e implementación.
- Implementación del marco de trabajo en correspondencia a los requisitos identificados para dar cumplimiento al objetivo propuesto.
- Realización de pruebas de software a la solución propuesta para comprobar su correcto funcionamiento y calidad.

Durante el desarrollo de la investigación se utilizan para dirigir la investigación los **Métodos Científicos**:

Métodos teóricos a utilizar

Método histórico-lógico: Este método permitirá analizar la trayectoria histórica referente a los Sistemas de Información Geográfica para entornos Web, a los diferentes marcos de trabajo para la implementación de interfaz de usuario que se analizarán, entre otros aspectos importantes para la investigación de modo que agilice y simplifique la realización posterior del Software.

Método sintético-analítico: Este método se utilizará con el objetivo de poder analizar y comprender la información obtenida, para poder seleccionar lo mejor de la misma de forma tal que sea de mayor utilidad y se relacione en gran medida con el objeto de estudio.

Modelación: La modelación se empleará para la realización de los diagramas asociados a la metodología de desarrollo que se adoptará en la solución a desarrollar.

Métodos empíricos a utilizar:

Observación: Se utilizará en todo momento durante la investigación, para analizar herramientas que de cierta forma proveen una solución parcial o total al problema planteado en la investigación, con el fin de determinar algunas características y requisitos funcionales, en la selección de los patrones que organizarán la implementación de la solución que se propone y en la aplicación de las pruebas al sistema para verificar su calidad.

Entrevista: Esta técnica es la vía mediante la cual consultando con los especialistas de la LPS Aplicativos SIG se recopilará información sobre los problemas que se presentan hoy en la implementación de interfaces de usuario, así como posibles ideas o vías para darle solución.

Pruebas: Se realizarán una serie de pruebas a la solución propuesta para comprobar si se ajustan a los resultados esperados.

Para una mejor comprensión el trabajo de diploma está divido por capítulos, específicamente en 3, los cuales están desglosados en epígrafes y subepígrafes, sobre el contenido de estos capítulos, se presenta un resumen a continuación.

Capítulo 1: Fundamentación Teórica del Marco de Trabajo “Geomap UI”: En este primer capítulo se realiza un análisis y descripción de la situación problemática y del objeto de estudio para lograr una mejor comprensión de la solución propuesta. También se analizarán soluciones existentes que se utilizan tanto a nivel mundial como en Cuba. Además contendrá las herramientas y tecnologías que serán utilizadas para darle cumplimiento al objetivo propuesto.

Capítulo 2: Análisis y Diseño del Marco de Trabajo “Geomap UI”: En este capítulo se expondrán los artefactos obtenidos en el desarrollo de los procesos de análisis y diseño. Se generarán los diagramas necesarios para la obtención de la solución, los cuales ofrecerán una mayor claridad en la confección de la misma. Por último se especifica el estilo y el patrón arquitectónico que definirá al marco de trabajo, así como los patrones de diseño y los estándares de codificación que se utilizarán en la implementación de la solución propuesta.

Capítulo 3: Implementación y Prueba del Marco de Trabajo “Geomap UI”: En este último capítulo se describirá todo el proceso de implementación de la solución y los diagramas correspondientes a esta fase, además se mostrará una descripción de las pruebas que se realizarán al sistema una vez terminado y los resultados arrojados por las mismas.

Capítulo 1: Fundamentación teórica del Marco de Trabajo “Geomap UI”

1.1 Introducción

En el presente capitulo se abordan temas que comprenden la fundamentación teórica de la investigación, primeramente se plasman una serie de conceptos asociados al dominio del problema, para que se tenga mayor conocimiento de los aspectos que se tratarán durante todo el transcurso de la investigación. Por otra parte se realiza un estudio del estado del arte para analizar las diferentes librerías que permiten la implementación de interfaces de usuario para Sistemas de Información Geográfica. Se argumentan los motivos de la utilización o no de los marcos de trabajo analizados. Además se profundizará en la selección de las metodologías, tecnologías y herramientas de soporte al desarrollo que se utilizarán, las cuales permitirán comenzar con el análisis y diseño de la solución.

1.2 Conceptos asociados al dominio del problema

Línea de Productos de Software:

Según Cabello “*Una línea de productos Software constituye básicamente el ensamblaje por partes, previamente elaboradas de un Software, se inspira en las líneas de producción de sistemas físicos, manteniendo muchas de sus características. Una LPS fundamenta la reutilización del Software así como la reducción de tiempo de su construcción* (Cabello, 2012).”

Según Northrop y Clements “*Una línea de productos de software se refiere a un conjunto de sistemas de software que comparten características y que son desarrollados a partir de un conjunto común de bienes núcleo. Donde los productos dentro de la línea son los distintos sistemas y que los bienes núcleo son las partes reutilizables que permitirán desarrollar los productos* (L.Northrop y P.Clements, 2002).”

Por lo que se pude decir que una Línea de Productos de Software no es más que, un sistema que construye un software a partir de la unión de las distintas partes previamente elaboradas de dicho software, reduciendo así el tiempo de confección del mismo.

Interfaz de Usuario:

Según la Real Academia Española “*Se conoce como Interfaz de Usuario al medio que permite a una persona comunicarse con una máquina. La interfaz, en este caso, está compuesta por los puntos de contacto entre un usuario y el equipo* (Real Academia Española, 2010).”

Según el libro El País “*Cuando se habla de la interfaz de un programa o la interfaz de usuario, uno se refiere al diseño que tiene el programa y la forma en que permite al usuario comunicarse con él y llevar a cabo ciertas tareas. Así, se dice de ciertos programas que tienen una interfaz agradable, amigable (calco) o intuitiva. A la hora de informar de esto se puede traducir como el diseño del programa, aunque implica algo más que eso; concretamente el diseño destinado a que el uso sea más o menos fácil* (El País, 2002).

Por lo que se establece que una Interfaz de Usuario no es más que, la forma en que se comunica un usuario con la computadora, siendo esta interacción más o menos compleja en dependencia del diseño que posea dicha interfaz.

Marco de trabajo:

Según Molina “*Estructura de Software compuesta por componentes personalizables e intercambiables para el desarrollo de una aplicación. Marco de trabajo o Framework se puede considerar como una aplicación genérica incompleta y configurable a la que se puede añadir las últimas piezas para construir una aplicación concreta* (Molina, 2012).”

Según Gutierrez “*En el desarrollo de software, un marco de trabajo es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un marco de trabajo puede incluir soporte de programas, librerías y un lenguaje descriptivo entre otros software, para ayudar a desarrollar y unir los diferentes componentes de un proyecto. De igual forma representa una arquitectura de software que modela las relaciones generales de las entidades del dominio de trabajo, la cual extiende o utiliza las aplicaciones del dominio* (Gutierrez, 2007).”

Queda definido un marco de trabajo como una estructura de software genérica e incompleta, con la cual se desarrollan distintos software, siempre que estos pertenezcan al ámbito o al dominio del propio marco de trabajo.

Personalización:

Según la Real Academia Española “*Personalización, es dar carácter personal a algo, incurrir en personalidades hablando o escribiendo. Aplicada a la informática en el ámbito de la web se entiende como el hecho de que los usuarios modifiquen o personalicen la interfaz de usuario y el comportamiento de determinadas aplicaciones web* (Real Academia Española, 2010).”

Según EDUC “Personalización, es la confección de un contenido Web por un usuario. Se puede llevar a cabo cuando el usuario introduce sus preferencias, pero también cuando el computador adivina estas preferencias (EDUC, 2013).”

Por lo que se puede decir que personalizar, es adaptar algo (negocio, interfaz gráfica, u otro) al gusto o necesidades del usuario, la empresa o entidad que lo requiera y/o exija.

Identidad visual corporativa:

Según eO2 “*Es la representación visual de una organización, incluyendo su logo (marca), diseño, tipografía y colores. Y refleja la filosofía y valores de la organización* (eO2, 2013).”

Según LUISANNET “*La identidad corporativa es la representación o imagen que un espectador tiene de una organización o de una empresa, no se trata sólo del logotipo de la empresa, la identidad corporativa es la imagen que la empresa transmite al exterior y la representación que nosotros como espectadores nos hacemos de ella. La identidad corporativa abarca tanto aspectos tangibles como son el diseño del logotipo y el diseño gráfico_corporativo (su representación visual), su símbolo o logotipo, tipografías, colores, papelería corporativa, los elementos de comunicación externa e interna, publicidad, protocolo, arquitectura corporativa, entre otros; como aspectos intangibles, por ejemplo la filosofía de la propia organización o empresa, su misión y sus valores* (LUISANNET, 2014).”

Se puede expresar que una identidad visual corporativa no es otra cosa que, la imagen o el rostro que muestran las empresas al público, está conformada por aspectos como los colores que la representarán, el

logotipo que las identificará, las tipografías que utilizarán, así como las metas, filosofía, valores u otros aspectos que las definen.

Una vez definidos los conceptos asociados al dominio del problema se describe todo lo asociado al objeto de estudio.

1.3 Descripción del objeto de estudio

El término “marco de trabajo”, traducido del inglés “*framework*”, es un concepto que hace mención a una estructura de software compuesta por componentes personalizables e intercambiables para el desarrollo de una aplicación. Se caracteriza por estar asociado a un determinado tipo de aplicaciones, lo que implica que su alcance esté limitado; o sea que está vinculado a un dominio concreto (Real Academia Española, 2010).

Los objetivos principales que persigue un marco de trabajo son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones. Un marco de trabajo Web, por tanto, podemos definirlo como un conjunto de componentes que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas Web. Las principales características de estos, así como algunas de sus ventajas se describen a continuación (Gutiérrez, 2010).

Ventajas de la utilización de un marco de trabajo:

- El desarrollo rápido de aplicaciones. Los componentes incluidos en un marco de trabajo constituyen una capa que libera al programador de la escritura de código de bajo nivel.
- La reutilización de componentes de software al por mayor. Los marcos de trabajo son los paradigmas de la reutilización.
- El uso y la programación de componentes que siguen una política de diseño uniforme. Un marco de trabajo orientado a objetos logra que los componentes sean clases que pertenezcan a una gran jerarquía de clases, lo que resulta en bibliotecas más fáciles de aprender a usar.

Características principales que poseen los marcos de trabajo:

Tabla 1: Características de los marcos de trabajo.

Abstracción de URLs y sesiones	No es necesario manipular directamente las URLs ni las sesiones, el marco de trabajo ya se encarga de hacerlo.
Acceso a datos	Incluyen las herramientas e interfaces necesarias para integrarse con herramientas de acceso a datos, en BBDD, XML, entre otras.
Controladores	La mayoría de los marcos de trabajo implementan una serie de controladores para gestionar eventos, como una introducción de datos mediante un formulario o el acceso a una página. Estos controladores suelen ser fácilmente adaptables a las necesidades de un proyecto concreto.
Autentificación y control identificación de usuarios	Incluyen mecanismos para la identificación de usuarios mediante *login* y *password,* y permiten restringir el acceso a determinas páginas a determinados usuarios.

Quedando definido qué es un marco de trabajo, así como sus principales características y ventajas que ofrecen, se dará paso a analizar los marcos de trabajo existentes que pueden solucionar los problemas presentes en la LPS Aplicativos SIG.

1.4 Análisis de marcos de trabajo existentes para el diseño de interfaces de usuarios

1.4.1 ExtJS

ExtJS es un framework para JavaScript muy utilizado en el desarrollo de aplicaciones Web. Tiene una librería inmensa que permite configurar la interfaces Web de manera semejante a las aplicaciones desktop. Permite construir aplicaciones complejas en internet además de flexibilizar el manejo de componentes de la página como el DOM, peticiones AJAX, DHTML, tiene la funcionalidad de crear interfaces de usuario bastante funcionales. Las desventajas, así como algunos aspectos importantes que incluye se muestran a continuación (García, 2009)

Este marco incluye:

- Componentes de interfaces de usuarios personalizables.
- Modelo de componentes extensibles.
- Un API fácil de usar.
- Licencias Open Source (GPL) y comerciales.

Desventajas:

- Necesidad de una plataforma: Pues se dependerá del paquete ExtJS para obtener los resultados que se desean.
- No cuenta con una licencia LGPL.
- No existe una forma fácil de realizar encuadernación (*binding*) entre los componentes visuales con el respectivo modelo, lo cual genera que el programador tenga que escribir más código para validar y enlazar los formularios.

Licencias: ExtJS divide su licencia en dos “frentes”. La primera es la versión comercial, con el apoyo y el acceso directo (a través de Subversion[1] (SVN)) al código fuente, con la posibilidad de realizar cambios al marco casi a diario. La versión “libre” sigue a la licencia GPLv3, como elemento más importante de esta es que se debe proporcionar el código fuente de la aplicación desarrollada. Es decir, se creará una aplicación que utiliza ExtJS “libre”, también debe utilizar la licencia GPLv3. Se podrá incluso cobrar por el Software, pero se debe proporcionar el código fuente de forma gratuita. Por lo tanto, la venta de la aplicación dependerá de la buena voluntad de que el cliente quiera pagar por ella o no (Sencha Inc, 2013).

1.4.2 GeoExt

La librería GeoExt es de código abierto y permite la creación de aplicaciones SIG de escritorio, como a través de la Web. Se trata de un marco de JavaScript que combina la funcionalidad GIS de OpenLayers con la interfaz de usuario de la biblioteca ExtJS proporcionada por Sencha[2]. Forma parte de los proyectos de la

[1] **Subversion:** Subversion es un sistema de control de versiones de código abierto (The Apache Software Foundation, 2011).

[2] **Sencha:** Sencha Ext JS es una plataforma de desarrollo de aplicaciones de escritorio con compatibilidad entre navegadores (Sencha Inc, 2014).

Open Source Geospatial Foundation (OSGeo)[3], está bajo una licencia BSD. Las aplicaciones de mapas contienen capas de mapa (ráster o vectoriales, procedentes de una gran variedad de fuentes), y los controles para operar sobre esas capas. GeoExt combina los controles geoespaciales de OpenLayers con los componentes de interfaz de usuario de ExtJS en un framework que permite construir aplicaciones SIG de estilo similar a las de escritorio, pero en un navegador. A continuación se muestran algunos de sus principales componentes (GeoExt2, 2014).

Componentes que implementa GeoExt:

- **GeoExt.FeatureRenderer:** Crea un componente de caja para la prestación de un vector con distintas características.
- **GeoExt.MapPanel:** Crea un contenedor de panel para un mapa.
- **GeoExt.Popup:** Ventana especializada que apoya el anclaje a un lugar determinado en un panel de mapa. Cuando una ventana emergente que está anclada a un lugar, eso significa que el popup debe apuntar visiblemente a la ubicación en el mapa, y se mueven en consecuencia cuando el mapa se desplaza o ampliada.

GeoExt.SliderTip: Crea un regulador que muestra una ventana emergente.

Licencia: La fuente GeoExt puede ser distribuida con una licencia BSD. Ext proporciona una excepción de licencia para los marcos de código abierto utilizando Ext que permite esta licencia. Prácticamente, esto significa que usted puede modificar y redistribuir GeoExt sin requisitos especiales. (Trac, 2014).

1.4.3 OpenLayers

OpenLayers es una librería de JavaScript de código abierto que posibilita mostrar mapas interactivos en los navegadores Web. OpenLayers ofrece un API para acceder a diferentes fuentes de información cartográfica como pueden ser los WMS Además cuenta con una interfaz amigable[4] muy parecida a la de Google Maps[5].

[3] **OSGeo:** Fue creada para apoyar el desarrollo colaborativo de software de código abierto geoespacial, y promover su uso generalizado (OSGeo, 2014).
[4] **Interfaz amigable:** Capacidad intuitiva de la interfaz de usuario (TEC, 2007).

[5] **Google Maps:** Es un sistema de búsqueda de localizaciones geográficas (Arnal, 2007)

Sus principales características, así como las ventajas de su utilización son especificadas a continuación (Aurelio, 2012).

Características:

- Soporta Capas vectoriales, Simple Interface, OGC WMS, OGC WFS, GeoRSS, Canvas.
- Posee un conjunto de controles para interactuar con el mapa como: *Zoom* panorámico, Barra *zoom*, *Controls Mouse*, *Scale Ratio*, Marcadores, *Popups*, Barra escala.
- Es de código libre lo que permite utilizarlo o modificarlo en cualquier momento.
- Puede utilizar múltiples servidores de datos.

¿Por qué elegir OpenLayers?

- Superposición de múltiples capas de mapa en una sola aplicación.
- Muestra tiles/imágenes de WMS, WMTS, TMS, WMS-C, WMTS.
- Representación de elementos vectoriales y estilo con soporte para KML, GeoJSON, WKT, GML, WFS y GeoRSS.
- Conectable con cualquier kit de herramientas JavaScript (jQuery, Ext, Dojo, MooTools).
- Elementos de clúster y paginación.

Ventajas:

- OpenLayers no requiere instalación.
- Menor procesamiento en el servidor.
- Puede ampliar fácilmente el código para su aplicación en particular.

Por los aspectos mencionados anteriormente OpenLayers permitirá interactuar con servicios de mapas, importados ya sea de Google Maps, Bing Maps y otros. También permitirá crear mapas interactivos, así como la inclusión y superposición de distintos tipos de capas. Además posibilitará la visualización de información espacial/geográfica, así como la edición de la misma.

Licencia: OpenLayers tiene como licencia la 2-cláusula Licencia BSD, también conocida como la licencia FreeBSD. Esta licencia se aplica a todo el código y el contenido en las " ramas ", " tronco ", y directorios

"proyecto" del repositorio de código OpenLayers en svn.openlayers.org, y se aplica a todo lanzamiento de OpenLayers a partir de su versión 2.5. La redistribución y el uso en formato fuente y binario, con o sin modificaciones, están permitidos siempre que se cumplan las siguientes condiciones (Trac, 2011):

- Las redistribuciones del código fuente deben conservar el aviso de copyright, esta lista de condiciones y el descargo de responsabilidad.
- Las redistribuciones en formato binario deben reproducir el aviso de copyright, esta lista de condiciones y el descargo de responsabilidad en la documentación y / u otros materiales proporcionados con la distribución.

El aviso de copyright, así como el descargo de la responsabilidad puede ser consultado visitando el sitio Web del cual se referencia.

1.4.4 jQuery

La biblioteca o framework de JavaScript jQuery es rápida y concisa, permite simplificar el trabajo con documentos HTML, trabajar con DOM, manejar eventos, desarrollar efectos dinámicos e interfaces de usuarios avanzadas. Esta ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código ya que con las funciones propias de esta biblioteca se logran grandes resultados, reduciendo en gran escala los tiempos de desarrollo de las aplicaciones. Es un Software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privativos. Las principales características y ventajas de dicho marco de trabajo se pueden ver a continuación. Sus principales características, así como sus las ventajas de su utilización son especificadas a continuación (The jQuery Foundation, 2013).

Características de jQuery son:

- jQuery consiste en un único fichero JavaScript que contiene las funcionalidades comunes de DOM, eventos, efectos y AJAX.
- La característica principal de la biblioteca es que permite cambiar el contenido de una página Web sin necesidad de recargarla, mediante la manipulación del árbol DOM y peticiones AJAX. Para ello utiliza las funciones $() o jQuery ().

Ventajas:

- Como buen JavaScript se pude trabajar separado del HTML, haciéndolo más fácil de mantener y enriquecer, aumentando la productividad del proyecto.
- Se considera su sintaxis liviana, teniendo en cuenta que la sencillez y poca extensión de código es fundamental para los desarrollos.
- Funciona independientemente del navegador sobre el que se visualiza el sitio.
- Al servir para todos, el desarrollador tiene menos trabajo, menos código, menos espacio, menos problemas.
- Es un producto serio, estable, bien documentado y con un gran equipo de desarrolladores a cargo de la mejora y actualización del marco de trabajo.
- jQuery es un producto con una aceptación por parte de los programadores muy buena y un grado de penetración en el mercado muy amplio, lo que hace suponer que es una de las mejores opciones.
- La comunidad de creadores de componentes (*plugins*) es muy amplia, lo que hace fácil encontrar soluciones ya creadas en jQuery para implementar interfaces de usuario, galerías, votaciones, efectos diversos, entre otros.

jQuery UI es un complemento que permite implementar componentes diversos para generar interfaces de usuario en páginas Web, además de otras funcionalidades básicas para crear aplicaciones Web enriquecidas. Como su propio nombre indica, es una librería JavaScript basado en el popular framework jQuery y se pueden encontrar vínculos (*links*), explicaciones, así como demos y descargas a partir del sitio Web oficial de jQuery. Al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio. La librería se divide en módulos, y ofrece una API para las transiciones animadas, estos se muestran a continuación (jQuery UI Team, 2010).

La librería se divide en cuatro módulos:

- Núcleo:

  Contiene las funciones básicas para el resto de módulos.

- Interacciones:

Añade comportamientos complejos a los elementos:

  - Draggable: Hace al elemento arrastrable.
  - Droppable: Permite que el elemento responda a elementos arrastrables.
  - Resizable: Permite redimensionar el elemento.
  - Selectable: Permite seleccionar entre una lista de elementos.
  - Sortable: Ordena una lista de elementos.
- Widgets:

Es un conjunto completo de controles de interfaz de usuario. Cada control tiene un conjunto de opciones configurables y se les pueden aplicar estilos CSS.

  - Accordion: Menú con efecto acordeón.
  - Autocomplete: Caja con autocompletado.
  - Button: Botón.
  - Dialog: Ventanas con contenido.
  - Slider: Elemento para elegir en un rango de valores.
  - Tabs: Pestañas.
  - Datepicker: Calendario gráfico.
  - Progressbar: Barra de progreso.
- Efectos:

La librería jQuery ofrece una API para añadir transiciones animadas y facilidades para interacciones:

Tabla 2: Transiciones animadas de jQuery UI.

Core	**Fold**
Blind	Highlight
Bounce	Pulsate
Clip	Scale
Drop	Shake
Explode	Slide
Fade	Transfer

Según (NorthWare, 2013) las principales características de jQuery son:

- Selección de elementos.
- Eventos.
- Manipulación de la hoja de estilos CSS.
- Efectos y animaciones.
- Animaciones personalizadas.
- AJAX.
- Soporta extensiones.
- Utilidades varias como obtener información del navegador, operar con objetos y vectores.

Licencia: Utiliza la licencia MIT, por lo que se puede utilizar libremente en cualquier proyecto jQuery UI, en cualquier otro proyecto (incluso proyectos comerciales), siempre y cuando la cabecera de los derechos de autor se deje intacta (The jQuery Foundation, 2013).

Kendo UI es un paquete de librerías listas para usar por los desarrolladores en páginas Web dinámicas, enteramente realizadas con JavaScript y basadas en jQuery. A pesar de usar jQuery como base en su desarrollo, podríamos denominarlo framework JavaScript, ya que ofrece diversas funcionalidades básicas, como arrastrar y soltar o un sistema de plantillas JavaScript, pero su fuerte son las interfaces de usuarios listas para incorporar en los proyectos Web.

Kendo UI es muy similar a lo que ofrece jQuery, con la diferencia de que ambas librerías implementan algunos componentes (*widgets*) distintos. En realidad la mayoría de lo que podemos hacer en jQuery UI lo podemos hacer también con Kendo UI. La variedad de componentes para interfaces de usuario de Kendo UI incluye menús dinámicos, gráficas, paneles, rejillas de datos, árboles, ventanas, sistemas para subir (*upload*) archivos, y así hasta llegar a 13 distintos tipos de *widgets* altamente personalizables y con funcionalidades realmente avanzadas.

En Kendo se ha cuidado perfectamente la compatibilidad en todos los navegadores más habituales, tal como acostumbran este tipo de productos. De este modo, tanto las funcionalidades del framework como los distintos componentes, corren perfectamente en Internet Explorer (versión 7 en adelante), Safari, Chrome o

Firefox, tanto en las versiones creadas para ordenadores personales, como dispositivos móviles. Sus principales características, así como sus desventajas se muestran a continuación (Telerik Inc, 2013).

Las principales características de Kendo UI son:

- Construido y pensado en HTML5
- Framework basado en jQuery.
- Prometen gran velocidad tanto en la carga de la interfaz como en la carga de datos.
- Gran variedad de componentes especialmente diseñados para Kendo (grillas, gráficos (lineales, barras, tortas), *slider*, *splitter*, *treeview*).
- Listo para ser usado con dispositivos móviles y dispositivos con pantallas táctiles (*touchscreens*).
- Soporte técnico provisto por Telerik[6].
- Soporte para Internet Explorer 7+, Firefox 3+, Safari 4+, Chrome, Opera 10+.
- Gran cantidad de demos para iniciarse en el desarrollo.
- Foro y Blog para estar en contacto con la comunidad.

Desventajas:

- La licencia Open Source no tiene soporte, si se quiere soporte se deberá adquirir la licencia comercial.
- Limitaciones de usar como se estime conveniente el producto, debido a su licencia pública

Licencia**:** Si se ha comprado una licencia comercial, se puede distribuir los programas como incrustados en productos integrados a usuarios finales solamente en virtud de una licencia de usuario final que cumple con los requisitos de esta sección. No se permite distribuir el software conforme a esta sección: como producto independiente o como parte de cualquier producto que no sea de este producto integrado. El contrato de licencia debe: limitar la responsabilidad de sus licenciantes o proveedores en la medida máxima permitida por la ley; y prohibir cualquier intento de desmontar, descompilar o "desbloquear", decodificar o reversión por otro medio traductor o ingeniero, e intentar de cualquier manera reconstruir o descubrir cualquier código

[6] **Telerik:** Es una empresa proveedora líder en el mercado de los controles de interfaz de usuario (Telerik, 2014).

fuente o algoritmos del software subyacente. No se permite y está expresamente prohibido conceder a los usuarios finales autorizados cualquier derecho a sublicenciar más el software.

Por su parte la redistribución de Kendo UI Web bajo licencia Open Source. Únicamente puede distribuir el Software Web Kendo UI bajo licencia GPL v3.0. En caso de que se distribuya los productos integrados, incluyendo el Software Web Kendo UI, bajo la GPL v3.0, el código fuente del Software puede ser distribuido de conformidad con sus obligaciones en virtud de dicha licencia. Si se distribuye un producto integrado en virtud de cualquier otra licencia, no se puede distribuir ninguna parte del Software Web Kendo UI en forma de código fuente (Telerik Inc, 2013).

Como se puede apreciar existen varios Marcos de trabajo y librerías destinadas a la creación de interfaces de usuario personalizables, pero ninguna de estas propuestas se ajusta a las necesidades de la investigación. Pues por distintas razones la utilización de ellas al menos por si solas no conllevan a la solución deseada. En el caso de Extjs debido a que es la fuente principal de los problemas que atraviesa la LPS como bien se describe en la situación problemática, además de ser el motivo por el cual no se utiliza el marco de trabajo GeoExt. Kendo UI por motivos de licencia, ya que su licencia pública es GPLv3, lo cual descarta la posibilidad de su utilización por los argumentos dados cuando se hizo referencia a esta para ExtJS. Por otra parte el uso de jQuery UI por sí solo no ofrece los elementos de SIG que se requieren, ya que la solución deberá trabajar con SIG permitiendo el trabajo con mapas, entre otros aspectos, aunque sí ofrece todas las características que se necesitan en cuanto a la implementación de interfaces de usuarios personalizables. Por último OpenLayers tampoco es viable utilizarlo ya que no posee las facilidades para la implementación interfaces de usuarios personalizables que son requeridas, aunque permite trabajar mapas interactivos y ofrezca una serie de controles para interactuar con los mapas y demás recursos característicos de los SIG que son necesarios para la solución.

Por lo anteriormente planteado, se decide implementar un marco de trabajo el cual se basará en las librerías jQuery UI y OpenLayers ya que juntas poseen todas las características que son deseadas y requeridas para esta aplicación. Primeramente jQuery UI ya que permite implementar componentes diversos para generar interfaces de usuario en páginas Web, además de otras funcionalidades básicas para crear aplicaciones Web enriquecidas. Estos elementos son fundamentales e imprescindibles para la confección de un marco de trabajo personalizable. Por otra parte OpenLayers posibilita mostrar mapas interactivos en los navegadores Web, se conecta con cualquier *kit* de herramientas JavaScript (o sea se puede conectar con

jQuery UI). Posee un conjunto de controles para interactuar con el mapa. Interactúa con servicios SIG externos. Además, ambas librerías son de código libre lo que permite utilizarlas, comercializarlas o modificarlas en cualquier momento. Por estos motivos se toman las características necesarias de cada una de ellas para la conformación del marco de trabajo Geomap UI.

1.5 Tecnologías y herramientas de soporte al desarrollo

1.5.1 Metodología de desarrollo a utilizar

La tendencia de crear sistemas más sofisticados, adaptados a las nuevas tecnologías, a las necesidades de los usuarios que cambian constantemente, de mejorar los productos de una versión a otra, y de realizar todo este proceso de una forma más rápida, hace cada vez más complejo el proceso de desarrollo de Software. Existen varias metodologías de desarrollo de Software aplicadas en los Centros de Desarrollo de Software de la UCI, tales como RUP, XP y SCRUM. El objetivo de estas es guiar y organizar el proceso de desarrollo de Software, garantizando un producto final de calidad, y para que se desarrolle en el tiempo establecido es necesario la aplicación de una de estas metodologías. Se debe realizar un análisis de las características de cada proyecto informático para determinar cuál es la más factible para evitar clientes y desarrolladores insatisfechos con el producto final.

Se utilizará la metodología RUP ya que permite seleccionar fácilmente el conjunto de componentes de proceso que se ajustan a las necesidades específicas del marco de trabajo Geomap UI. RUP sigue un proceso adaptable y un ciclo de vida iterativo, ajustándose a las necesidades del proyecto. Por otra parte el marco de trabajo Geomap UI continuará desarrollándose en la LPS, o sea, se continuará perfeccionando con posteriores versiones por otro equipo de desarrollo. Por esto se hace necesario generar documentación que permita a dicho equipo comprender y realizar esta tarea con la mayor eficacia. De igual forma se generan los artefactos que se determinen necesarios por lo anteriormente planteado. Otro aspecto tomado en consideración es que en la Universidad se ha incrementado el uso de RUP, lo que ha posibilitado el surgimiento de profesionales expertos en la metodología . A continuación se describe el mismo brevemente y se brindan una serie de características para una mayor comprensión de la metodología.

RUP:

El proceso unificado conocido como RUP, es un modelo de Software que permite el desarrollo de Software a gran escala, mediante un proceso continuo de pruebas y retroalimentación, garantizando el cumplimiento de ciertos estándares de calidad. Aunque con el inconveniente de generar mayor complejidad en los controles de administración del mismo. Sin embargo, los beneficios obtenidos recompensan el esfuerzo invertido en este aspecto.

El proceso de desarrollo constituye un marco metodológico que define en términos de metas estratégicas, objetivos, actividades y artefactos (documentación) requerido en cada fase de desarrollo. Esto permite enfocar esfuerzo de los recursos humanos en términos de habilidades, competencias y capacidades a asumir roles específicos con responsabilidades bien definidas.

Sus características permiten que esta metodología sea adaptable a una gran variedad de sistemas para diferentes áreas de aplicación, diferentes tipos de organización y diferentes tamaños de proyecto. La particularidad de que cada ciclo de iteración exige el uso de artefactos, es el motivo que hace que sea una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del Software. Sus principales características se detallan a continuación (Galves, y otros, 2004).

Características de RUP:

Dirigido por casos de uso (CU): Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requisitos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso.

Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los CU relevantes desde el punto de vista de la arquitectura.

Iterativo e Incremental: Aunque pueda parecer que los flujos de trabajo se desarrollan en cascada, RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Por ejemplo, una iteración de

elaboración centra su atención en el análisis y diseño, aunque refina los requisitos y obtiene un producto con un determinado nivel, pero que irá creciendo e incrementándose en cada iteración.

RUP divide su ciclo de vida en cuatro fases y nueve flujos de trabajo, de ellos seis de ingeniería y tres de soporte, tal como se muestra en la figura 1.

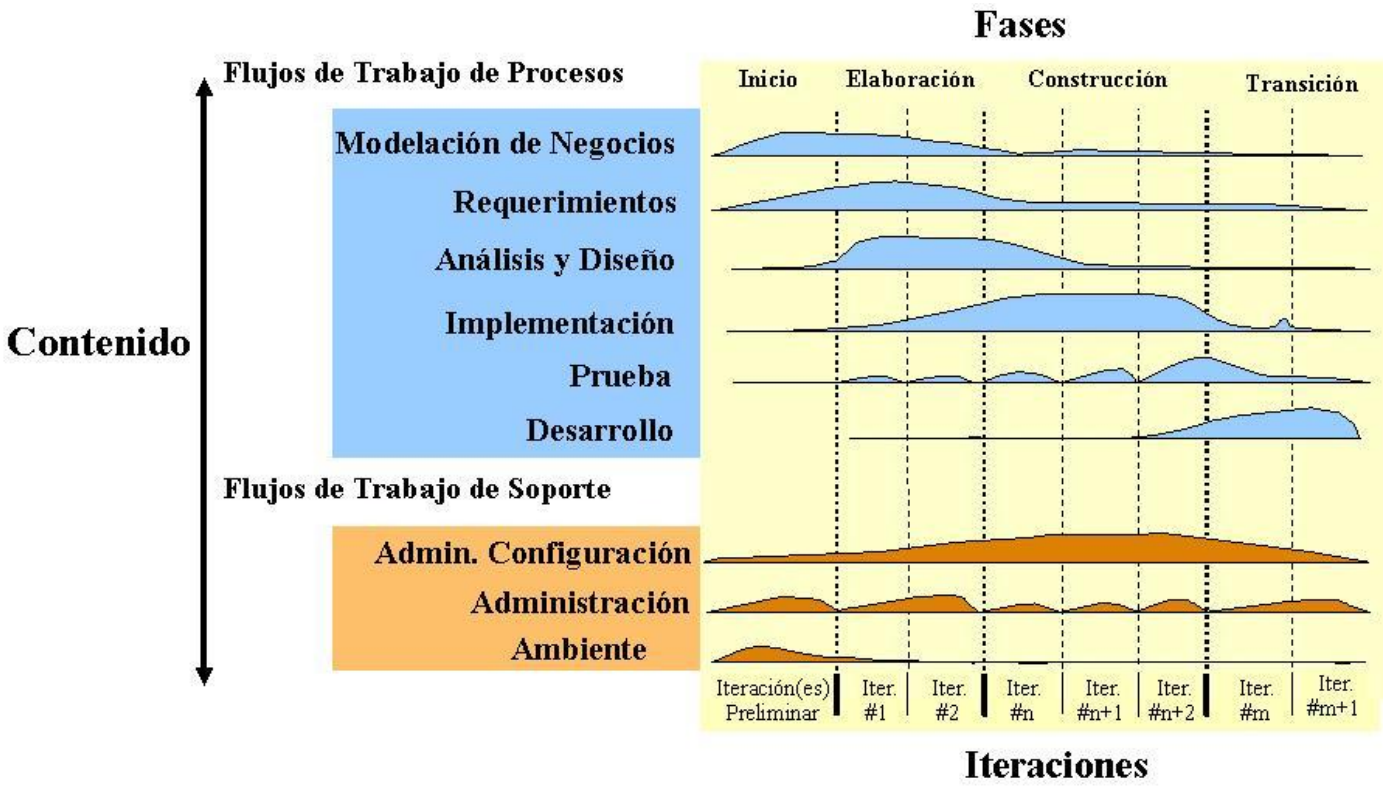


Fig. 1: Fases y Flujos de Trabajo de RUP (Benzadón y otros, 2007).

Una vez definida la metodología a utilizar, la cual guiará todo el proceso de desarrollo del marco de trabajo, se determinan las tecnologías y herramientas para el desarrollo que se utilizarán.

1.5.2 Tecnologías para el desarrollo

Las tecnologías para el desarrollo se seleccionaron primeramente de acuerdo al lenguaje en el que están basados los marcos de trabajo que serán utilizados por la librería que se desea crear, así como otros lenguajes que permitirá enriquecer la misma desde el punto de vista visual y profesional. Asimismo, debido

a que no existe una obligación de trabajar con uno específico, la selección del entorno de desarrollo se basara en la utilización de uno conocido por el equipo de desarrollo.

Lenguajes informáticos a utilizar:

Un lenguaje informático es usado por, o asociado con ordenadores. Muchas veces, este término es usado como sinónimo de lenguaje de programación, pero un lenguaje informático no tiene por qué ser un lenguaje de programación. En general, como cualquier otro lenguaje, un lenguaje de ordenador es creado cuando hay que transmitir una información de algo a alguien basado en computadora. El lenguaje de programación es el medio que utilizan los programadores para crear un programa de ordenador; un lenguaje de marcas es el medio para describir a un ordenador el formato o la estructura de un documento; y así con los demás tipos de lenguajes. Los Lenguajes informáticos pueden ser clasificados en varias clases, entre las que se incluyen las siguientes (Damián Pérez Valdés, 2007).

- Lenguaje de programación.
- Lenguaje de especificación.
- Lenguaje de consulta, como SQL o XQuery.
- Lenguaje de marcas, como XML y otros más ligeros.
- Lenguaje de transformación, como XSLT.
- Lenguaje de sonido, para crear sonidos.
- Lenguaje gráfico, para crear figuras y dibujos.
- Pseudocódigo.

Lenguaje JavaScript:

JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript[7]. Posee mecanismos que simulan la programación orientada a objetos; basado en prototipos, imperativo y dinámico. Se utiliza principalmente en su forma del lado del cliente (*client-side*), implementado como parte de un navegador Web permitiendo mejoras en la interfaz de usuario y páginas Web dinámicas, aunque existe una forma de JavaScript del lado del servidor (*Server-side JavaScript* o SSJS).

[7] **ECMAScript:** Es una especificación de lenguaje de programación publicada por ECMA International. El desarrollo empezó en 1996 y estuvo basado en el popular lenguaje JavaScript (Academic, 2013).

JavaScript se diseñó con una sintaxis similar al C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo Java y JavaScript no están relacionados y tienen semánticas y propósitos diferentes. Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas Web. Para interactuar con una página Web se provee al lenguaje JavaScript de una implementación del Modelo de Objetos del Documento (DOM por sus siglas en inglés, del inglés *Document Object Model*). Tradicionalmente se utiliza en páginas Web HTML para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a funciones del servidor. JavaScript se interpreta en el agente de usuario, al mismo tiempo que las sentencias van descargándose junto con el código HTML. Sus principales ventajas se muestran a continuación (ClubEnsayos.com, 2013).

Ventajas:

- Es un lenguaje de programación sencillo y muy liviano.
- Se recomienda para la creación de aplicaciones Web, muy útil para el desarrollo de páginas Web dinámicas.
- Utiliza poca memoria y ligero en cuanto a carga.
- Tiene gran cantidad de efectos visuales.
- Fácil manejo de datos.
- Es soportado por los más populares navegadores.
- Fácil de integrar.
- Cientos de aplicaciones disponibles para su uso.
- Puede agregar interactividad a elementos Web.

CSS3:

Hojas de Estilo en Cascada (*Cascading Style Sheets*), es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos. CSS se utiliza para dar estilo a documentos HTML y XML, separando el contenido de la presentación. Los estilos definen la forma de mostrar los elementos HTML y XML. CSS permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo. Cualquier cambio en el estilo para un elemento del

documento en el CSS, afectará a todas las páginas en las que aparezca el elemento y que estén vinculadas a esa hoja de estilos. CSS funciona a base de reglas, es decir, declaraciones sobre el estilo de uno o más elementos. Las hojas de estilo están compuestas por una o más de esas reglas aplicadas a un documento HTML o XML. La regla tiene dos partes: un selector y la declaración. A su vez la declaración está compuesta por una propiedad y el valor que se le asigne. CSS3 está compuesto por distintos módulos, los cuales se muestran a continuación (Eguíluz Pérez, Javier, 2009).

Principales módulos que ofrece CSS3:

- Selectores
- Modelo de caja.
- Fondos y Bordes.
- Valores de imagen y contenido reemplazado.
- Efectos de texto.
- Animaciones
- Diseño de columna múltiple.
- Interfaz de usuario

HTML5:

HTML5 es la nueva versión de HTML. Este proporciona una plataforma con la que desarrollar aplicaciones Web más parecidas a las aplicaciones de escritorio, donde su ejecución dentro de un navegador no implique falta de recursos o facilidades para resolver las necesidades reales de los desarrolladores. Para ello se crearon interfaces de programación que permiten trabajar con cualquiera de los elementos en la página y realizar acciones que antes era necesario realizar por medio de tecnologías accesorias.

Estas interfaces de programación se están documentando con minuciosidad y tendrán que ser implementadas por los distintos navegadores del mercado. El objetivo de esta tarea, es permitir que todos estos navegadores utilicen estándares y disminuir las incompatibilidades con la Web. A continuación se muestran sus principales características (Freddy Vega, y otros, 2011):

- Estructura del cuerpo: La mayoría de los sitios y aplicaciones web tienen un formato común, formado por elementos como cabecera, pie, navegadores, entre otros. HTML 5 permite agrupar todas estas partes de una Web en nuevas etiquetas que representarán cada uno de las partes típicas de una página.

- Etiquetas para contenido específico: Hasta ahora se utilizaba una única etiqueta para incorporar diversos tipos de contenido enriquecido, como animaciones Flash o vídeo. Ahora se utilizarán etiquetas específicas para cada tipo de contenido en particular, como audio, vídeo, etc.

- Canvas: Este componente permite dibujar en una página web todo tipo de formas, que podrán estar animados y responder a la interacción del usuario. Es una tecnología similar a Adobe Flash pero que no será necesario la utilización de un agregado en el navegador para visualizar esas formas.

Lenguaje de Modelado v2.2 (UML):

UML, es un lenguaje que permite modelar, construir y documentar los elementos que forman una Aplicación Informática orientada a objetos. UML ha puesto fin a las llamadas “guerras de métodos” que se han mantenido a lo largo de los 90, en las que los principales métodos sacaban nuevas versiones que incorporaban las técnicas de los demás, dentro de las cuales se puede mencionar las técnicas en el modelado CMM. Con UML se fusiona la notación de estas técnicas para formar una herramienta compartida entre todos los ingenieros de Software que trabajan en el desarrollo orientado a objetos (Orallo, 2001).

Según Rumbauhg[8] “UML es ya un estándar de la industria del Software, pero no solo de la industria, sino que en general, de cualquier industria que requiera la construcción de modelos como condición previa para el diseño y posterior construcción de prototipos”.

Debido a estas características, se decide utilizar UML en el ciclo de construcción de la solución propuesta, ya que la modelación de los artefactos durante las fases del ciclo de construcción, posibilitara que en las fases posteriores los desarrolladores puedan tener un mayor dominio y comprensión sobre qué es lo que

[8] **James Rumbauhg:** Científico Estadounidense de la computación orientada a objetos metodólogo, es más conocido por su trabajo en la creación de la técnica de modelado de objetos (OMT) y el Lenguaje Unificado de Modelado (UML).

se debe implementar. También permitirá tener una representación real del alcance y la factibilidad que puede o no llegar a tener el producto.

1.5.3 Herramientas a utilizar

Visual Paradigm v8.0:

Visual Paradigm es una plataforma de modelado diseñado para apoyar a los arquitectos de sistemas, desarrolladores y diseñadores UML, para acelerar el proceso de análisis y diseño de aplicaciones empresariales complejas (Soft112, 2014).

Según Jacobson[9] las principales características que influyen en la selección de Visual Paradigm como herramienta CASE son:

- Facilita a los ingenieros de Software diseñar, integrar y modelar visualmente los distintos diagramas que se generan a lo largo del desarrollo del Software.
- Permite construir sistemas de Software a gran escala de manera confiable a través del uso de un enfoque Orientado al Objeto.
- Presenta un generador de código que soporta más de diez lenguajes y proporciona la ingeniería inversa.

Se selecciona esta herramienta para generar todos los diagramas correspondientes a los diferentes estados del ciclo de vida del Software que van a sustentar el marco de trabajo. Además presenta características que son favorables para el trabajo con tecnologías libres, con el modelado UML y con la estandarización de la documentación.

Entorno de Desarrollo Integrado (IDE) NetBeans v8.0:

NetBeans es una herramienta libre y gratuita, que en su versión 8.0 da soporte a HTML5, principal característica por la cual se selecciona para la implementación del marco de trabajo. Además en el

[9] **Ivar Jacobson:** Ingeniero sueco en Ciencias de la Computación. Inventó el diagrama de secuencia y desarrolló los diagramas de colaboración.

expediente de proyecto de la LPS Aplicativos SIG, se especifica que debe ser el IDE a utilizar en la implementación de soluciones que se desarrollan en la línea

1.6 Conclusiones parciales

- El análisis realizado sobre marcos de trabajo para la creación de interfaces de usuario para plataformas Web demostró que estos no dan cumplimiento al objetivo de esta investigación ya que no aportan una solución totalmente viable para la creación de interfaces de usuario para SIG.
- Las librerías jQuery UI y OpenLayers a utilizar en conjunto para la implementación de la solución propuesta permitirán implementar un marco de trabajo único de su tipo, pues no existe alguna otra solución similar para la creación de interfaces de usuario para SIG.
- La documentación de todo el proceso de desarrollo de la solución propuesta utilizando la metodología RUP permitirá un mejor entendimiento del marco de trabajo para la implementación de futuras versiones.

Capítulo 2: Análisis y diseño del Marco de Trabajo “Geomap UI”

2.1 Introducción

En este capítulo se fundamenta la estructura del marco de trabajo, así como sus diferentes características y componentes. La descripción de los conceptos asociados al dominio permite estructurar la librería y definir sus necesidades. Se profundiza en la selección de los componentes del mismo teniendo en cuenta el uso de estos y la importancia de su utilización en la librería. Quedan plasmados los requisitos funcionales por cada uno de los componentes, los requisitos no funcionales y el diagrama de casos de uso del sistema. Además se define la arquitectura y los patrones de diseño a utilizar, elementos indispensables en la implementación de la solución. Por último se especifican los diagramas de clases del diseño para cada uno de los casos de uso obtenidos.

2.2 Características del sistema

2.2.1 Modelo del dominio

Debido a que el negocio no está bien definido y que el marco de trabajo que conformará la solución será genérico, se hace necesario definir un modelo de dominio para describir actualmente cómo se desarrolla el proceso de implementación de interfaces de usuarios personalizables en la LPS Aplicativos SIG.

Según Larman[10] “Un modelo del dominio se utiliza con frecuencia como fuente de inspiración para el diseño de los objetos Software. El modelo del dominio muestra (a los modeladores) clases conceptuales significativas en un dominio del problema; es el artefacto más importante que se crea durante el análisis orientado a objetos”.

A continuación se muestra el modelo de dominio, que describe cómo se desarrolla el proceso de personalización de interfaces de usuario en SIG en la LPS Aplicativos SIG.

[10] **Craig Larman:** Científico Informático canadiense, especializado en el desarrollo iterativo e incremental, el desarrollo ágil de software, el análisis orientado a objetos, diseño orientado a objetos, y el modelado ágil.

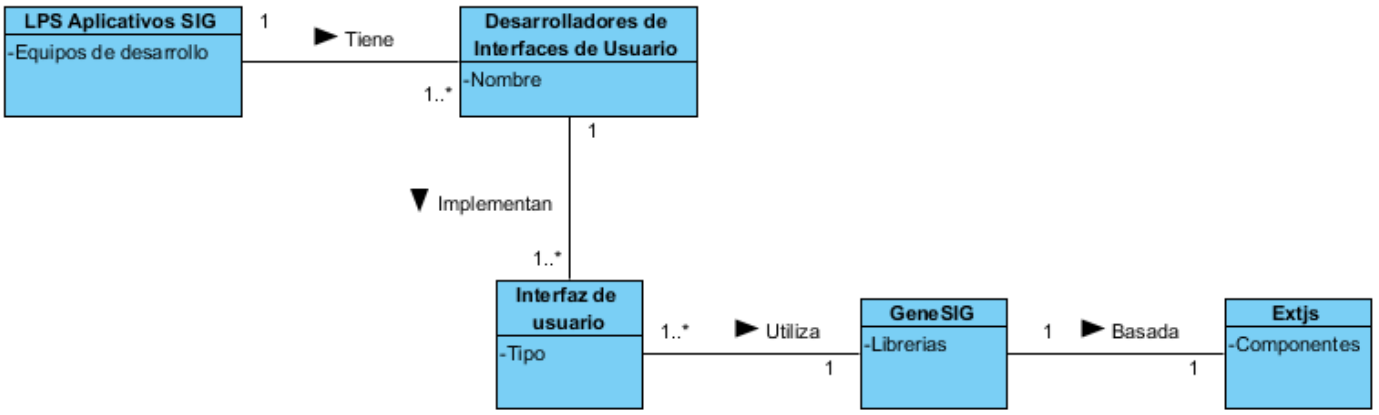


Fig. 2: Diagrama del modelo del dominio

Descripción del modelo del dominio:

En la figura 2, se explica cómo en la LPS Aplicativos SIG, los Desarrolladores de Interfaces de Usuario que conforman la misma implementan Interfaces de usuario utilizando la plataforma GeneSIG, la cual se basa en la librería ExtJS para dichas implementaciones.

Definiciones, acrónimos y abreviaturas del modelo de dominio:

Aplicativos SIG: Es una LPS perteneciente al Centro GEYSED de la UCI que se dedica a la realización de SIG para la Web.

Desarrolladores de Interfaces de Usuario: Personas de la LPS Aplicativo SIG que se encarga de implementar las interfaces de usuarios en la LPS.

Interfaz de Usuario: Es el medio que permite a una persona comunicarse con una máquina.

GeneSIG: Es un SIG para la Web utilizado por la LPS Aplicativos SIG como base para el desarrollo de otros SIG para la Web.

Extjs: Librería en la cual se basan las interfaces de usuarios Implementadas por los desarrolladores de interfaz de usuario de la LPS.

Una vez definido el modelo de domino, así como la descripción del mismo y otros aspectos de interés, se pueden definir cada uno de los componentes que requiere el marco de trabajo para cumplir con los objetivos propuestos.

2.2.2 Descripción de los componentes del marco de trabajo

En esencia, un componente es una pieza de código pre-elaborado que encapsula alguna funcionalidad expuesta a través de interfaces estándar. Los componentes son los "ingredientes de las aplicaciones", que se juntan y combinan para llevar a cabo una tarea (Terreros, 2014). A continuación se definen los componentes que conformarán el marco de trabajo Geomap UI.

Barra de herramientas (*Toolbar*): Es un componente que se muestra usualmente en pantalla a modo de fila, columna, o bloque. Contiene botones que al ser presionados, ejecutan funciones de una aplicación. Una barra de herramientas estándar contiene los botones para los comandos más usados. Muchas de las aplicaciones y sistemas operativos desarrollados recientemente permiten a los usuarios personalizar las barras de herramientas y ajustarlas a sus necesidades.

Barra de herramientas para controles del mapa (*ToolbarMap*): Componente que utiliza como base una barra de herramientas, con la diferencia de que los botones que lo componen, serán en sí controles que interactúan con un mapa.

En la biblioteca, el componente **ToolbarMap** estará directamente relacionado a la interacción con los mapas, ya que contendrá los controles de navegación que la aplicación requiera. Dichos controles a su vez son componentes que se describen a continuación.

- **Mover el mapa (*Drag pan*):** Este componente permite navegar sobre un mapa desplazándolo hacia la dirección deseada con la acción del ratón (mouse).
- **Acercar un nivel (*Zoom in*)**: Permite aumentar la escala de un mapa a un nivel superior, para visualizar más detalles.
- **Alejar un nivel (*Zoom out*):** Permite disminuir la escala de un mapa a un nivel menor para tener una vista completa o más panorámica.

- **Acercar mapa y alejar el mapa (*Zoom box in y Zoom box out*):** Este componente realiza un zoom libre para acercarse o alejarse de un mapa, permitiendo conocer exactamente todas las zonas a las que se puede acceder.
- **Historial de navegación (*Navigation history*):** Componente que permite navegar por el historial de navegación, que crea dos controles dependientes: anterior y posterior. Llama al método de disparo en los controles anterior y siguiente para restaurar los estados históricos anteriores y siguientes. Los controles anteriores y siguientes se activan cuando hay estados disponibles para restablecer y se desactivan cuando no hay estados para restaurar.
- **Medir distancia (*Measure distance*):** Una vez seleccionada, haciendo clic sobre el mapa aparece un punto, cuando se vuelva a hacer clic (siempre sobre la ventana del mapa), aparece un segundo punto y se muestra la distancia entre el primer punto y el segundo. Mientras se tenga seleccionada esta herramienta y a medida que se vayan haciendo sucesivos clics, se calcula la distancia entre los dos últimos puntos. Mientras se esté realizando una medición, se pueden realizar desplazamientos y zooms, para después continuar con la medición.
- **Calcular área (*Measure area*):** Este componente permite calcular un área determinada por el usuario, estableciendo una serie de puntos o marcadores en el mapa (mínimo 3 puntos) que conformarán el área que se desea calcular.
- **Dibujar vector (*Draw feature*)**: Las unidades básicas de información geográfica en los datos vectoriales son puntos, líneas, arcos y polígonos. Este componente posibilita dibujar estos vectores en el mapa sobre el cual se trabaja en la medida que dichos vectores sean necesarios en el negocio.
- **Mover vector (*Drag feature*):** Este componente permite mover un vector previamente creado hacia cualquier posición del mapa.
- **Modificar vector (*Modify feature*):** Este componente posibilita modificar un vector creado con anterioridad, o sea cambiar cualquier punto o puntos, así como la forma del vector.
- **Eliminar vector (*Remove feature*):** Este componente permite eliminar un vector creado con anterioridad.

Control de capas (*Layer control*): Este componente posibilita la utilización de las distintas capas que le pueden ser aplicadas al mapa sobre el que se trabaja, permitiendo visualizar y ocultar determinadas capas en el momento que se requiera en dependencia de las necesidades del cliente.

Panel de pestañas (*Tabpanel*): Los paneles con pestañas proporcionan una forma de utilizar una pequeña porción de espacio de la aplicación para contener varios "paneles" de información para que el usuario haga clics entre ellos. Se puede colocar imágenes, texto y otros contenidos en dichos paneles. Esto permitirá cargar varios elementos separados dentro de una misma ventana y así es posible alternar entre ellos con una mayor comodidad, además de evitar tener multitud de ventanas abiertas.

Ventanas (*Windows*): Este componente brinda la posibilidad de abrir o mostrar en una ventana contenidos, mensajes de alerta, e incluso un mapa al cual se le podrán aplicar las funcionalidades de los demás controles según se necesiten.

Mapa general (*Overview map*): El componente crea un pequeño mapa general, útil para mostrar la extensión de un mapa con zoom y el mapa principal, y proporcionar opciones de navegación adicionales para el usuario.

Ventana emergente (*Popup*): Este componente proporciona una manera de mostrar contenido en una ventana independiente que flota sobre la interfaz de la aplicación actual de manera relativa a un elemento designado o a una coordenada de la pantalla. Es utilizado principalmente para mostrar información sobre elementos que se muestran en un mapa.

Mapa (*Map*): El componente permite crear o construir un mapa.

Capa (*Layer*): El componente permite crear una capa para agregar al mapa.

Escala (S*cale*): El componente permite agregarle escala a los mapas.

Botón (*Button*): El componente permite crear diferentes tipos de botones.

Acordeón (*Accordion*): El componente permite crear un menú con efecto acordeón.

Autocompletado (*Autocomplete*): El componente permite crear una caja con autocompletado.

Calendario gráfico (*Datepicker*): EL componente permite mostrar un calendario con el que se pueda seleccionar una fecha.

Arrastrable (*Draggable*): El componente permite arrastrar un elemento dentro de un área.

Menú (*Menu*): El componente permite crear de una forma fácil y sencilla un menú de opciones.

Barra de progreso (*Progressbar*): El componente permite crear una barra de progreso.

Seleccionable (*Selectable*): El componente permite seleccionar entre una lista de elementos.

Deslizable (*Slider*): El componente permite elegir un elemento en un rango de valores.

Ordenable (*Sortable*): El componente permite ordenar una lista de elementos.

Descripción emergente (*Tooltip*): Es un componente de ayuda visual, funciona al situar el cursor sobre algún elemento gráfico, mostrando una ayuda adicional para informar al usuario de la finalidad del elemento sobre el que se encuentra.

Una vez definidos los componentes que conformarán el marco de trabajo, consecuentemente se podrán obtener los requisitos funcionales de cada uno de ellos.

2.2.3 Especificación de requisitos

En los contextos de los sistemas basados en computadoras (y en *software*), el término especificación tiene significados diferentes para personas distintas. Una especificación pude ser un documento escrito, un conjunto de modelos gráficos, un modelo matemático, una colección de escenarios de uso, un prototipo o cualquier combinación de estos.

Algunos sugieren que para una especificación se debe desarrollar y utilizar una “plantilla estándar”, argumentan que esto conduce a que los requisitos sean presentados de una manera más consistente y por ende más entendible. Sin embargo, algunas veces es necesario ser flexible mientras se desarrolla una especificación. Respecto a sistemas grandes el mejor enfoque podría ser un documento escrito que combinara descripciones en el lenguaje natural y modelos gráficos. Por otro lado en cuanto a productos o sistemas más pequeños, podría ser que no se necesite más que escenarios de uso, cuando dichos sistemas residan en ambientes técnicos que se comprendan bien.

La especificación es el producto de trabajo final que genera la ingeniería de requisitos. Sirve como base para las actividades de ingeniería de software subsecuentes. Describe la función y el desempeño de un sistema basado en computadora y las restricciones que regirán su desarrollo (Pressman, 2005).

Requisitos Funcionales por componentes:

Los Requisitos Funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. Estos describen lo que el sistema debe hacer, en algunos casos, también pueden declarar explícitamente lo que no debe hacer. Estos requisitos dependen del tipo de Software que se desarrolle, de los posibles usuarios y del enfoque general tomado por la organización al redactar requisitos. Cuando se expresan como requisitos del usuario, habitualmente se describen de una forma bastante abstracta. Sin embargo los requisitos funcionales del sistema describen con detalle la función de este, sus entradas y salidas, excepciones, funcionalidades entre otras (Sommerville, 2005).

Se identificaron un total de 33 componentes que conformarán el marco de trabajo y en relación a estos quedaron definidos 119 requisitos funcionales. Estos componentes fueron agrupados en 16 casos de uso debido a las similitudes entre ellos. A continuación se describen algunos de los requisitos funcionales que conforman los casos de uso de Geomap UI, pueden consultarse en su totalidad en el artefacto generado “Marco de Trabajo Geomap UI Especificación de Requisitos de Software v2.0.doc” que se encuentra en el repositorio de documentos de la LPS Aplicativos SIG.

Definir barras de herramientas:

RF.1 Crear barra de herramientas de botones.
RF.2 Especificar botones para las barras de herramientas.
RF.3 Agregar botones para las barras de herramientas.
RF.4 Eliminar botones de la barra de herramientas.
RF.5 Eliminar barras de herramientas de la interfaz.
RF.6 Actualizar propiedades de las barras de herramientas en tiempo de ejecución.

Los requisitos anteriores fueron obtenidos mediante técnicas de recopilación de requisitos, entre ellas la técnica entrevista y la tormenta de ideas. La entrevista se realizó de acuerdo a una serie de preguntas que propone la sexta edición del Pressman (en su capítulo 7: Ingeniería de Requisitos), las cuales están enfocadas en las metas generales, la comprensión del problema, percepción del cliente respecto a la solución, entre otros aspectos importantes. Otra técnica ya mencionada fue la tormenta de ideas, esta se desarrolló en varias reuniones donde el equipo de desarrollo definió otras funcionalidades que no habían

sido identificadas. Cualquier duda con respecto a estas, se pueden consultar detalladamente en el artefacto “Marco de Trabajo Geomap UI Técnicas de Recopilación de Requisitos.doc” que se encuentra en el repositorio de documentos de la LPS Aplicativos SIG. A continuación se definen los requisitos no funcionales una vez definidos los requisitos funcionales que componen cada uno de los casos de uso.

Requisitos no Funcionales:

Un requisito no funcional especifica los criterios que se deben usar para juzgar el funcionamiento de un sistema, en lugar de un comportamiento específico. En general, los requisitos funcionales definen lo que el sistema debería de hacer, mientras que los requisitos no funcionales verifican cómo un sistema debería ser. Estos son a menudo llamados las “cualidades de un sistema”. Puede dividirse en dos categorías (Pressman, 2005):

- **Cualidades de ejecución:** como por ejemplo la seguridad y facilidad de uso, que son observables en tiempo de ejecución.
- **Cualidades de evolución:** como por ejemplo la sostenibilidad, extensibilidad y escalabilidad, que están más vinculados a la estructura del sistema de Software.

Se definieron un total de 16 requisitos no funcionales al marco Geomap UI, se plasman en el informe algunos de los más importantes, para consultarlos todos detalladamente se puede auxiliarse del documento “Marco de Trabajo Geomap UI Especificación de Requisitos de Software v2.0.doc” que se encuentra en el repositorio de documentos de la LPS Aplicativos SIG:

RNF de Usabilidad

- **RNF.1** El marco de trabajo podrá ser usada por usuarios que tengan conocimientos en JavaScript, teniendo experiencia creando interfaces de usuario, así como conocimientos de jQuery y OpenLayers.

RNF de Interfaces de software

- **RNF.11** Geomap UI es un marco de trabajo basado en el lenguaje JavaScript, por lo tanto para que funcione se requiere un navegador Web con JavaScript habilitado, y que soporte todas las características de HTML5 y CSS3, preferiblemente (utilizar una versión de los navegadores inferior

a la recomendada, puede ocasionar problemas en la utilización de determinados componentes de Geomap UI, de manera parcial o total):

- Mozilla Firefox 25 o superior.
- Chrome 29.0.154 o superior.
- Internet Explorer 9 o superior

Ya definidos todos los requisitos del sistema, tanto funcionales como los no funcionales se comienza con la modelación de los casos de uso del sistema.

2.2.3 Modelo de casos de uso del sistema

El modelo de casos de uso ayuda al cliente, a los usuarios y a los desarrolladores a llegar a un acuerdo sobre cómo utilizar el sistema. La mayoría de los sistemas tienen muchos tipos de usuarios. Cada tipo de usuario se representa como un actor. Los actores utilizan el sistema al interactuar con los casos de uso. Todos los actores y casos de uso del sistema forman un modelo de casos de uso (Rumbaugh, y otros, 1999).

Descripción de los actores del sistema:

Tabla 2: Descripción del actor

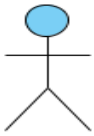
Actor		Descripción
	**Desarrollador de interfaz de usuario**	**Persona que implementa Interfaces de Usuario en la LPS Aplicativos SIG.**

Diagrama de casos de uso del sistema:

En un diagrama de casos de uso, no se muestran los casos de uso en detalle; solamente se resumen algunas de las relaciones entre los casos de uso, los actores y los sistemas. En concreto, en el diagrama no se muestra el orden en el que se llevan a cabo los pasos para lograr los objetivos de cada caso de uso.

Esos detalles pueden describirse en otros diagramas y documentos, que pueden vincularse a cada caso de uso (Microsoft, 2013).

Un diagrama de casos de uso describe parte del modelo de casos de uso y muestra un conjunto de casos de uso y actores con una asociación entre cada par actor/caso de uso que interactúan (Rumbaugh, y otros, 1999). A continuación se muestra el diagrama de casos de uso del sistema, el cual cuenta con 16 casos de uso, los 9 que están a la derecha son para componentes individuales, en relación a la complejidad de los mismos, mientras que los 7 de la izquierda están compuestos al menos por dos componentes, debido a la similitud entre estos:

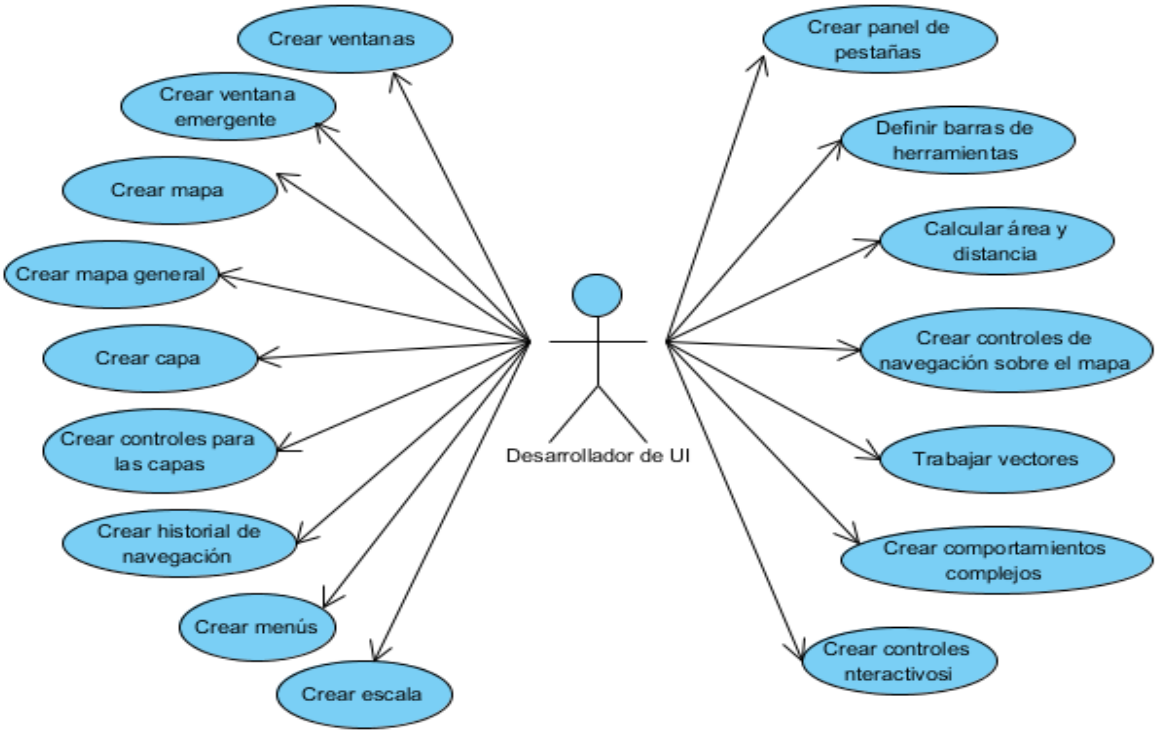


Fig. 3: Diagrama de casos de uso del sistema

Descripción textual de los casos de uso del sistema:

Tabla 3: Descripción del caso de uso “Definir barra de herramientas”

Caso de Uso	Definir barras de herramientas
Actores	Desarrollador de interfaz de usuario
Resumen	El caso de uso inicia cuando el Desarrollador de interfaz de usuario crea un objeto de tipo “Toolbar” o “Map Toolbar”, luego especifica los parámetros por los cuales se generará el objeto. El caso de uso termina cuando el sistema genera la interfaz del objeto de acuerdo a las propiedades especificadas por el actor.
Precondiciones	El usuario debe haber incluido la librería Geomap UI en su proyecto.
Referencia	RF.1, RF.2, RF.3 RF.4, RF.5, RF.6, RF.7, RF.8, RF.9, RF.10, RF.11, RF.12
Prioridad	Crítico.
Postcondiciones	Se genera la interfaz del Toolbar o del Map Toolbar.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso inicia cuando el Desarrollador de interfaz de usuario declara un objeto de tipo Toolbar o Map Toolbar.	2. En caso de que el actor declare el objeto de tipo Toolbar ir a la sección Toolbar. 3. En caso de que el actor declare el objeto de tipo Map Toolbar ir a la sección Map Toolbar. 4. El caso de uso termina cuando el sistema genera la interfaz del Objeto.
Sección Toolbar	
	2.1. El sistema crea de tipo Toolbar el objeto declarado por el actor.
2.2. El Desarrollador de interfaz de usuario especifica los parámetros que desea para el Toolbar.	2.3. El sistema le aplica las propiedades entradas por el actor al objeto de tipo Toolbar previamente creado.
	2.4. Ir a la acción 4.
Sección Map Toolbar	
	3.1. El sistema crea de tipo Map Toolbar el objeto declarado por el actor.
3.2. El Desarrollador de interfaz de usuario especifica los parámetros que desea para el Map Toolbar.	3.3. El sistema le aplica las propiedades entradas por el actor al objeto de tipo Map Toolbar previamente creado.
	3.4. Ir a la acción 4.

Se ha descrito el caso de uso “Definir barra de herramientas”. Para consultar las restantes descripciones auxiliarse del artefacto “Marco de Trabajo Geomap UI Modelo del Sistema v2.0.doc” que se encuentra en el repositorio de documentos de la LPS Aplicativos SIG.

De los 16 casos de uso definidos, 9 de estos son arquitectónicamente significativos, ya que su utilización se hace imprescindibles al crear un SIG. Los otros 7 son para agregar características o funcionalidades que enriquecen el SIG pero su utilización es prescindible. Una vez confeccionados los diagramas de casos de uso se define el estilo y el patrón arquitectónico que se empleará, así como los patrones de diseño que permitieron implementar aplicando buenas prácticas de programación.

2.3 Descripción arquitectónica

La Arquitectura de Software permite agregar flexibilidad y adaptabilidad a los sistemas informáticos, en mercados cambiantes. Ofrece una herramienta de venta y comercialización, ayuda a reducir los costos de mantenimiento, amortiza los costos de desarrollo, ofrece una guía clara para el desarrollo favoreciendo el control de los proyectos, establece un vocabulario común entre los participantes, entre otros aspectos de interés (CPAP, 2013).

En este epígrafe se definirá la estructura de carpetas que tendrá la librería, así como la descripción de cada una de estas carpetas. De igual forma quedará detallada la arquitectura que se utilizará y otros aspectos importantes sobre la arquitectura del marco de trabajo Geomap UI.

2.3.1 Estructura de carpetas

El marco de trabajo Geomap UI estará estructurado de la siguiente manera:

Carpeta Doc: En esta carpeta se almacenarán todos los documentos referentes a la biblioteca, manuales para los usuarios de la misma u otros.

Carpeta Lib: En su interior estarán todas la librerias que son necesarias para el funcionamiento del marco de trabajo en desarrollo.

Carpeta Themes: Contiene distintos temas, los cuales se podrán utilizar sin perjudicar la estructura de los componentes.

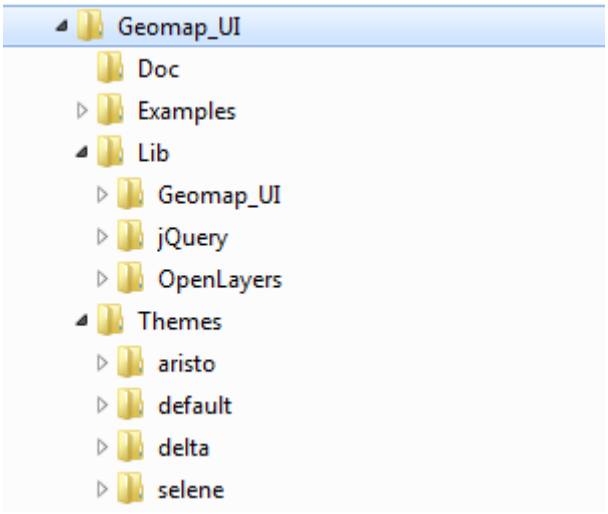


Fig. 4 Estructura de carpeta de Geomap UI**.**

Se decide incluir dentro del marco de trabajo Geomap UI las librerías jQuery (la cual está compuesta por su complemento para implementar componentes diversos para generar interfaces de usuario jQuery UI) y OpenLayers por las siguientes razones:

- Con tener la librería Geomap UI, ya se puede hacer uso de todas las funcionalidades de esta.
- Se puede utilizar el marco de trabajo, sin depender de ningún servicio externo, ni la necesidad de la Internet.

2.3.2. Estilo y patrón arquitectónico

Un estilo arquitectónico son un conjunto de reglas y restricciones que definen según (Bass, 1998):

- Cuáles tipos de componentes, interfaces y conectores pueden ser usados en un sistema (Vocabulario/Metáforas).
- Posible inclusión de tipos de dominio-especifico cómo los componentes y conectores pueden ser combinados (estructura)
- Cómo se comporta el sistema

- Un conjunto de guías que soportan la aplicación del estilo (Cómo lograr ciertas propiedades del sistema)
- Un estilo arquitectónico define una familia de sistemas en términos de un patrón de organización estructural.

Por otra parte los patrones arquitectónicos son los que definen la estructura de un software, los cuales a su vez se componen de subsistemas con sus responsabilidades, también tienen una serie de directivas para organizar los distintos componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño de este.

El estilo arquitectónico seleccionado es **Llamada y Retorno** de acuerdo a las clasificaciones dadas por Bass.

Esta familia de estilos enfatiza la modificabilidad y la escalabilidad. Son los estilos más generalizados en sistemas en gran escala. Los miembros de la familia son las arquitecturas de programa principal y subrutina, los sistemas basados en llamadas a procedimientos remotos, los sistemas orientados a objeto y los sistemas jerárquicos en capas.

Por otra parte se utilizará el patrón arquitectónico **Modelo Vista Controlador (MVC)** en relación a las clasificaciones dadas por Buschmann (Buschmann, 1996).

El MVC es un patrón de arquitectura de Software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones Web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio y el controlador es el responsable de recibir los eventos de entrada desde la vista. La finalidad del modelo es mejorar la reusabilidad por medio del desacople entre la vista y el modelo. Los elementos del patrón son los siguientes:

El modelo es el responsable de:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.

- Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: “Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor”.
- Lleva un registro de las vistas y controladores del sistema.
- Si se está ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero bath que actualiza los datos, un temporizador que desencadena una inserción, entre otros).

El controlador es el responsable de:

- Recibe los eventos de entrada (un clic, un cambio en un campo de texto, entre otros).
- Contiene reglas de gestión de eventos, del tipo “SI Evento Z, entonces Acción W”. Estas acciones pueden suponer peticiones al modelo o a las vistas.

Las vistas son responsables de:

- Recibir datos del modelo y mostrarlos al usuario.
- Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
- Pueden dar el servicio de “Actualización ()”, para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

Es seleccionada dicha arquitectura por su forma de separar los datos de una aplicación, la interfaz de usuario, y la lógica de control, permitiendo hacer cambios en cada una de estas partes sin la necesidad de hacerlos también en los demás. La forma en que se utiliza la misma es la siguiente:

Geomap UI ofrece un API para la creación de plantillas durante la ejecución que servirán de presentación para cada uno de los componentes. Los componentes de jQuery UI que se utilizan internamente en la librería, ya generan sus propias estructuras HTML pero Geomap UI aplica las estructuras básicas para esos componentes de forma dinámica, utilizando su propia estructura interna. A través de la función **`Geomap_UI.Theme`**, se definen los temas por componente para presentar cada contenido, que funciona como las vistas de cada uno de esos componentes, luego de ser generados, se guarda una referencia al componente que controla esa vista a través de la función **`jQuery.data`** que ofrece jQuery a los elementos del DOM.

Los controladores de cada clase que implementan los componentes, se basan en la utilización de **`behaviors`** (comportamientos) que ofrece Geomap UI en el core. Un **`behavior`** en Geomap UI es una función de llamada que se ejecutará cuando el DOM esté listo para utilizarse. Los *behaviors* son usados por los componentes para aplicar a las estructuras HTML que define el desarrollador de interfaz de usuario, según los roles de cada componente su propio comportamiento. La ventaja de este tipo de utilización, es que el desarrollador puede aplicar a un elemento HTML específico, un rol que responde a un componente obteniendo el funcionamiento de dicho componente en ese elemento sin necesidad de crear una instancia directa, y seria ese elemento la vista utilizada por el modelo.

Los modelos, en Geomap UI, se corresponden con las clases que implementan los componentes, especificando su comportamiento y lógica de funcionamiento según las propiedades que define para su utilización.

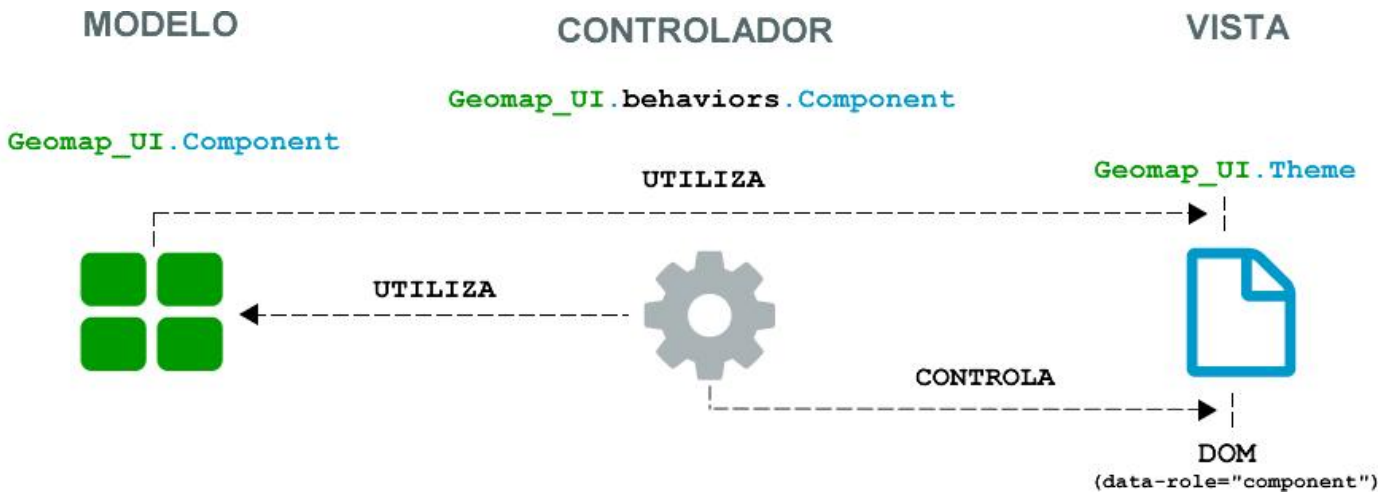


Fig. 5 Evidencia del patrón arquitectónico MVC en el marco de trabajo Geomap UI**.**

Esta estructura arquitectónica se aplica a cada componente, utilizando un único archivo que contiene la vista, el modelo y el controlador. Se ha definido de esta forma, ya que en términos de rendimiento y teniendo en cuenta las características de JavaScript, es más eficiente cargar un solo archivo que varios que implementen un componente del marco de trabajo.

Ya seleccionada la arquitectura se determinarán a continuación los patrones de diseño que se utilizarán en el marco de trabajo Geomap UI.

2.3.3 Patrones de diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de Software. En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de Software que están sujetos a contextos similares. Debemos tener presente los siguientes elementos de un patrón: su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios).

Patrones GRASP:

Creador: Es utilizado en Geomap_UI para que cada clase se especialice en la información que maneja, e implemente los métodos que solo ella puede responder de forma especializada. Por ejemplo, cuando se crea un mapa, y a este se le dice que cree una barra de herramientas con atributos cualquieras (botones y propiedades), la clase **`Geomap_UI.Map`** crearía la barra de herramientas, ya que tiene la información necesaria para crearla, pero los atributos serían creados por la clase **`Geomap_UI.Toolbar,`** ya que esta es quien conoce y lleva el control de estos.

Experto en información: Este patrón es utilizado en Geomap UI para que una clase implemente solo aquellas funcionalidades de las cuales conoce su información exacta. Por ejemplo, una barra de herramientas tiene botones, si se activa un botón, la clase **`Geomap_UI.Button.Map`** es la encargada de implementar el método activar, pero la clase **`Geomap_UI.Toolbar.Map`** es la encargada de que al activarse un botón, se desactive el botón que estuviese activo anteriormente.

Patrones GOF:

Fábrica simple: Este patrón es utilizado en Geomap UI para la creación dinámica de las clases que dan respuesta a los controles que se utilizan en un mapa generado por **`Geomap_UI.Map`**. Por ejemplo, el componente **`Geomap_UI.Toolbar.Map`**, está compuesto por sub componentes **`Geomap_UI.Button.Map`**, cada uno de estos botones utiliza un control que se agrega al mapa. El **`behavior`** de **`Geomap_UI.Toolbar.Map`** se encarga entonces de asignar cada control dinámicamente a cada botón, creando las instancias de dichas clases a través de **`Geomap_UI.Factory`**, que se encarga de determinar el tipo de control y devolver el objeto específico para cada control.

Herencia múltiple: Para utilizar este tipo de patrón de diseño, Geomap UI posee una clase que se encarga de imitar la POO, ya que JavaScript no es un lenguaje orientado a objetos, **`Geomap_UI.Class`** es utilizado para la definición de clases que pueden heredar de una o múltiples clases. Aprovechando esta característica, cada componente es separado en subcomponentes que engloban las propiedades y métodos básicos para su funcionamiento, por ejemplo, **`Geomap_UI.Object`**, que es la base de cada clase que se utiliza en Geomap UI, constituye la clase padre de **`Geomap_UI.Component`**, utilizada a tu vez, como padre por cada componente que implementa Geomap UI. Asimismo, cada componente esta implementado de forma tal, que se pueda utilizar como base para la implementación de otro componente con un funcionamiento similar, pero que constituya un tipo diferente de componente. Este tipo de comportamiento, se ve reflejado en el componente **`Geomap_UI.Toolbar`**, que a su vez es la clase padre de **`Geomap_UI.Toolbar.Map`**, la diferencia entre estas dos clases, es que **`Geomap_UI.Toolbar.Map`**, es un componente que se acopla a un mapa, para utilizarlo según los controles de mapa que definen los botones que la componen.

Para consultar donde se evidencian los demás patrones de diseño utilizados, dirigirse al artefacto “Marco de Trabajo Geomap UI Patrones de Diseño.doc” que se encuentra en el repositorio de documentos de la LPS Aplicativos SIG.

2.3.4 Estándares de codificación

Un estándar de codificación no es más que un conjunto de reglas que se siguen para la generación de código fuente, con el propósito de reflejar un estilo armonioso, con vista a que el código sea igual aunque haya sido programado por más de una persona. Al comenzar un proyecto de software, es necesario establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software. La sostenibilidad del código es la facilidad con que el sistema de software puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento. Aunque la legibilidad y la sostenibilidad son el resultado de muchos factores, una faceta del desarrollo de software en la que todos los programadores influyen especialmente es en la técnica de codificación. El mejor método para asegurarse de que un equipo de programadores mantenga un código de calidad es establecer un estándar de codificación sobre el que se efectuarán luego revisiones del código de rutinas. Usar técnicas de codificación sólidas y realizar buenas

prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento, todo esto posibilita que se obtenga un sistema fácil de comprender y mantener (Microsoft, 2014).

Estilo de codificación que será utilizado en la implementación del marco de trabajo Geomap UI:

- Todas las funciones y clases estarán comentadas, para explicar el flujo del código y el propósito de las mismas.
- Los nombres de las clases estarán escritos con el estándar “UpperCamelCase” que se utiliza cuando la primera letra de cada una de las palabras es mayúscula.
- Los nombres de las variables y funciones estarán escritos con el estándar “lowerCamelCase” que se utiliza cuando la primera letra de cada una de las palabras es minúscula.

2.4 Diagrama de clases del diseño

Una clase de diseño y sus objetivos, así como los subsistemas que contienen las clases del diseño, a menudo participan en varias realizaciones de casos de uso. También puede darse el caso de algunas operaciones, atributos y asociaciones sobre una clase específica que son relevantes para solo una realización de casos de uso. Esto es importante para coordinar todos los requisitos que diferentes realizaciones de casos de uso imponen a una clase, a un objeto y a los subsistemas que contiene. Para manejar todo esto, se utilizan diagramas de clases conectados a una realización de caso de uso, mostrando sus casos participantes, subsistemas y sus relaciones. De esta forma se puede guardar la pista de los elementos participantes en una realización de caso de uso (Rumbaugh, y otros, 1999).

A continuación se presenta el diagrama correspondiente al caso de uso “Definir barras de herramientas”. Para consultar los restantes diagramas auxiliarse del artefacto “Marco de Trabajo Geomap UI Modelo de Diseño v2.0.doc”.

Este diagrama está compuesto por 8 clases, se puede observar en el mismo la relación entre ellas, así como los atributos y métodos que estas tienen.

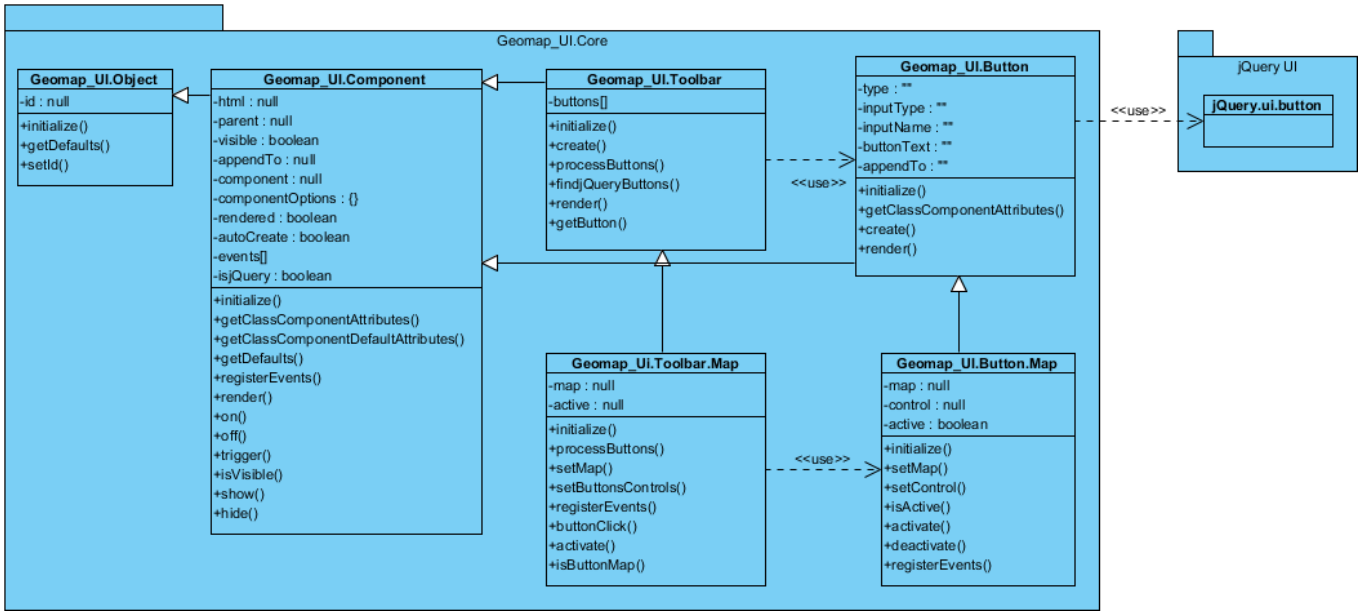


Fig. 6: Diagrama de clases del diseño. Caso de uso “Definir barras de herramientas”.

A continuación se describen las clases que intervienen en el diagrama de clases del diseño anterior, este está compuesto por 8 clases, se puede observar en el mismo la relación entre ellas, así como los atributos y métodos que estas tienen:

Tabla 3: Descripción del diagrama de clases del diseño del caso de uso “Definir barras de herramientas”

Clases	Propósito
Geomap_UI.Core	Es la clase encargada de cargar dinámicamente las clases que conforman el caso de uso “Definir barras de herramientas”.
Geomap_UI.Object	Esta clase contiene las propiedades y funcionalidades básicas para trabajar con elementos del DOM de las clases que implementan los componentes Geomap_UI.Toolbar y Geomap_UI.Toolbar.Map.
Geomap_UI.Component	Esta clase contiene las propiedades y funcionalidades básicas de los componentes Geomap_UI.Toolbar y Geomap_UI.Toolbar.Map. Hereda de Geomap_UI.Object.

Geomap_UI.Toolbar	Esta clase permite crear barras de herramientas. Utiliza Geomap_UI.Button para construir los botones que la componen. Hereda de Geomap_UI.Component.
Geomap_UI.Toolbar.Map	Esta clase permite crear barras de herramientas donde los botones que la conforman son componentes que interactúan con los mapas. Hereda de Geomap_UI.Toolbar y utiliza Geomap_UI.Button.Map.
Geomap_UI.Button	Esta clase hereda de Geomap_UI.Component y utiliza la clase jquery.ui.button. La clase Geomap_UI.Button.Map hereda de ella y es usada por Geomap_UI.Toolbar para crear los botones por los que estará conformada.
Geomap_UI.Button.Map	Esta clase hereda de Geomap_UI.Button y es utilizada por la clase Geomap_UI.Toolbar.Map para crear los botones por los que estará conformada.
jquery.ui.button	Esta clase es usada por Geomap_UI.Button.

2.5 Conclusiones parciales

- El uso de un patrón arquitectónico, patrones de diseño y estándares de codificación para la organización e implementación de la solución propuesta permitirán mejorar su rendimiento, escalabilidad y mantenibilidad.
- La confección de los diagramas de clases del diseño para representar la estructura interna del sistema, facilitarán la ejecución de tareas de mantenimiento del código fuente.

Capítulo 3: Implementación y pruebas del Marco de Trabajo “Geomap UI”

3.1 Introducción

En este capítulo se describe todo el proceso de implementación de la solución propuesta y de las pruebas que se le realizan una vez terminada la misma. Se plasman los diagramas de clases y de componentes, así como los resultados arrojados por las pruebas aplicadas.

3.2 Diagrama de componentes

El diagrama de componentes se utiliza para modelar la vista estática de un sistema. Muestra la organización y las dependencias entre un conjunto de componentes. No es necesario que un diagrama incluya todos los componentes, normalmente se realizan por partes. Cada diagrama describe un apartado del sistema. En él se situarán librerías, tablas, archivos, ejecutables y documentos que formen parte de la aplicación. Uno de los usos principales es que puede servir para ver que componentes pueden compartirse entre sistemas o entre diferentes partes de estos (Rumbaugh, y otros, 1999).

El diagrama de componentes del marco de trabajo Geomap UI quedó conformado por un paquete principal que representa a dicho marco, dentro de este se encuentran en el lado derecho dos paquetes que hacen referencia a las librerías jQuery UI y OpenLayers, dentro de estas se encuentran una serie de componentes que utiliza Geomap UI. En el lado izquierdo se encuentran los componentes que genera el marco, y estos se conectan con los de jQuery UI u OpenLayers en los cuales se basan. Para un mejor entendimiento del mismo se divide dicho diagrama en dos, una parte con la relación entre Geomap UI y jQuery UI, y la otra con OpenLayers.

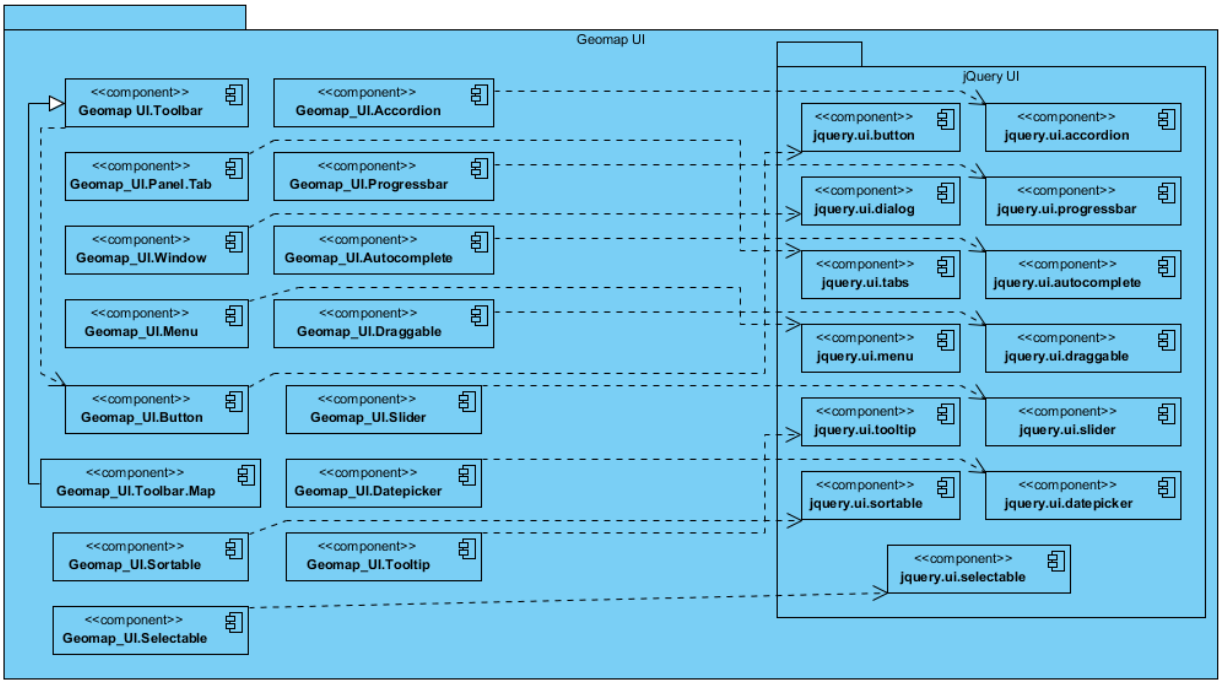


Fig. 7 Diagrama de componentes del marco de trabajo, referente a los componentes que utilizan jQuery UI**.**

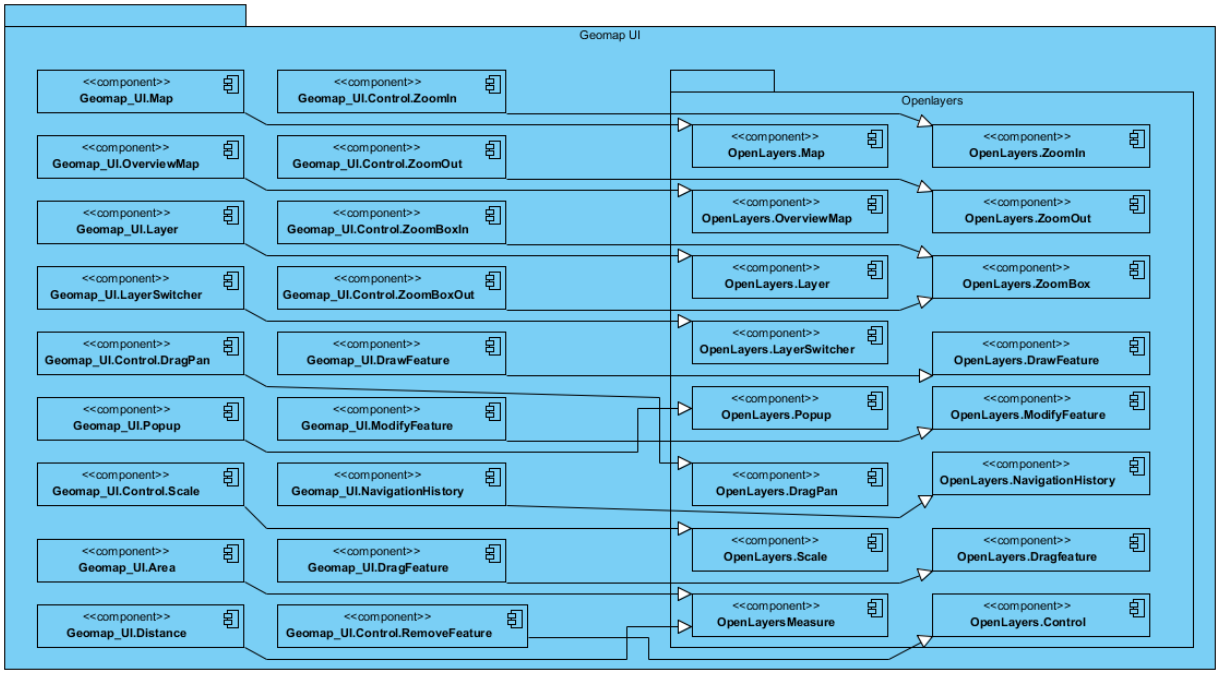


Fig. 8 Diagrama de componentes del marco de trabajo, referente a los componentes que utilizan OpenLayers**.**

Una vez definidos todos los diagramas en cuanto análisis y diseño se refiere, y definido el diagrama de componentes, resulta necesario especificar los estándares de codificación utilizados durante el transcurso de la implementación.

3.3 Validación y pruebas

Las pruebas de software es una investigación llevada a cabo para proporcionar a los interesados información sobre la calidad del producto o servicio bajo prueba. Las pruebas de software también pueden proporcionar una visión objetiva, independiente del software, para conocer y entender los riesgos de implementación de software. Las técnicas de pruebas incluyen, pero no se limitan, al proceso de ejecución de un programa o aplicación con la intención de encontrar errores u otros defectos en el software.

Se puede afirmar que las pruebas de software son el proceso de validación y verificación de que un programa de software / aplicaciones / producto:

- Cumple con los requisitos que guiaron su diseño y desarrollo.
- Funciona como se espera.
- Puede implementarse con las mismas características.

Las pruebas de software, dependiendo del método de análisis empleado, se puede implementar en cualquier momento en el proceso de desarrollo. Sin embargo, la mayor parte del esfuerzo de la prueba se produce después de que los requisitos se han definido y el proceso de codificación se ha completado. Como tal, la metodología de la prueba se rige por la metodología de desarrollo de software adoptado (Raygain, 2013).

Pruebas aplicadas al marco de trabajo Geomap UI:

Se aplican dos tipos de prueba, primeramente las pruebas funcionales de caja negra, utilizando la técnica particiones de equivalencia y las pruebas de aceptación alfa, desarrollando estas últimas en un ambiente controlado con la presencia de los desarrolladores del software.

3.3.1 Pruebas de caja negra

Las pruebas de cajas negras son pruebas de sistema, específicamente funcionales. Es una técnica de pruebas de software que se centra en el análisis de la funcionalidad del software, en comparación con los

mecanismos internos del sistema. Fue desarrollada como un método de análisis de las necesidades del cliente, las especificaciones y el diseño de estrategias de alto nivel. Esta técnica se aplica seleccionando un conjunto de condiciones y los controles de salida de las respuestas válidas de entrada y de ejecución de códigos válidos y no válidos (Janalta Interactive Inc, 2014).

Nivel de prueba: Sistema.
Tipo de Prueba: Funcionalidad.
Método de prueba: Caja Negra.
Técnica: Particiones de equivalencia.

A continuación se realiza el diseño de caso de prueba correspondiente al caso de uso Definir Barras de Herramientas.

Descripción General: El caso de uso se inicia cuando el usuario desee crear una barra de herramienta del tipo que desee y de la forma que decida, y termina cuando el sistema muestra la interfaz de la barra de herramientas creada.

Condiciones de Ejecución: El usuario debe haber incluido la librería Geomap UI en su proyecto.

Diseño de los casos de pruebas:

Tabla 4: Descripción por escenarios. Caso de uso “Definir barras de herramientas”.

Nombre del caso de uso	Nombre del componente	Descripción general	Formas de crear el componente	Escenario de pruebas	Flujo del escenario
Definir Barra de Herramientas	Toolbar	El caso de uso crea una barra de herramientas.	1. Definiendo una estructura HTML en la interfaz e indicándole a la librería que esa estructura asumirá el rol del componente.	EC 1.1: Definir una barra de herramientas con sus parámetros por defecto.	1. El usuario define una estructura HTML indicando que va a asumir el rol de componente a través de la propiedad *data-role* en la etiqueta. 2. La biblioteca crea una barra de herramientas aplicando su comportamiento a la estructura HTML especificada.

				EC 1.2: Definir una barra de herramientas con parámetros especificados.	1. El usuario define una estructura HTML indicando que va a asumir el rol de componente a través de la propiedad *data-role* en la etiqueta, además especifica las propiedades que definirán al componente, con el estilo *data-[nombre de la propiedad]*. 2. La biblioteca crea una barra de herramientas aplicando su comportamiento a la estructura HTML especificada.
			2. Creando instancia de la clase que implementa el componente.	EC 2.1: Definir una barra de herramientas con sus parámetros por defecto.	1. El usuario crea una instancia de la clase que implementa el componente. 2. La biblioteca crea una barra de herramientas aplicando su comportamiento según la instancia creada.

				EC 2.2: Definir una barra de herramientas con parámetros especificados.	1. El usuario crea una instancia de la clase que implementa el componente, además especifica las propiedades que definirán al componente, con el estilo *data-[nombre de la propiedad].* 2. La biblioteca crea una barra de herramientas aplicando su comportamiento según la instancia creada.
	Map Toolbar	El caso de uso crea una barra de herramientas donde los botones que lo conforman son componentes para interactuar con el mapa.	3. Definiendo una estructura HTML en la interfaz e indicándole a la librería que esa estructura asumirá el rol del componente.	EC 3.1: Definir una barra de herramientas para mapas con sus parámetros por defecto.	1. El usuario define una estructura HTML indicando que va a asumir el rol de componente a través de la propiedad data-role en la etiqueta. 2. La biblioteca crea una barra de herramientas para mapas, aplicando su comportamiento a la estructura HTML especificada.

				EC 3.2: Definir una barra de herramientas para mapas con parámetros especificados.	1. El usuario define una estructura HTML indicando que va a asumir el rol de componente a través de la propiedad data-role en la etiqueta, además especifica las propiedades que definirán al componente, con el estilo *data-[nombre de la propiedad].* 2. La biblioteca crea una barra de herramientas para mapas, aplicando su comportamiento a la estructura HTML especificada.
			4. Creando instancia de la clase que implementa el componente.	EC 4.1: Definir una barra de herramientas para mapas con sus parámetros por defecto.	1. El usuario crea una instancia de la clase que implementa el componente, con el estilo *data-[nombre de la propiedad].* 2. La biblioteca crea una barra de herramientas para mapas aplicando su comportamiento según la instancia creada.
				EC 4.2: Definir una barra de herramientas para mapas con parámetros especificados.	1. El usuario crea una instancia de la clase que implementa el componente, además especifica las propiedades que definirán al componente. 2. La biblioteca crea una barra de herramientas para mapas aplicando su comportamiento según la instancia creada.

Una vez descritos los distintos escenarios se describen las variables que conforman los mismos según la tabla siguiente.

Tabla 5: Descripción de las variables. Caso de uso “Definir barras de herramientas”.

No	Propiedad	Clasificación	Valor nulo	Descripción
1	buttons: []	Lista de botones	Si	Se pueden especificar las propiedades que definirán al componente.

Una vez descritas las variables de los distintos escenarios se describe como intervienen estas dentro de los escenarios.

Tabla 6: Matriz de datos. Caso de uso “Definir barras de herramientas”.

ID del EC	Escenario	V1	Respuesta del sistema	Resultado de la prueba
EC 1.1	Definir una barra de herramientas con sus parámetros por defecto.	NA	La biblioteca crea una barra de herramientas aplicando su comportamiento a la estructura HTML especificada.	Satisfactoria
EC 1.2	Definir una barra de herramientas con parámetros especificados.	V(Lista de botones)	La biblioteca crea una barra de herramientas aplicando su comportamiento a la estructura HTML especificada.	Satisfactoria
EC 2.1	Definir una barra de herramientas con sus parámetros por defecto.	NA	La biblioteca crea una barra de herramientas aplicando su comportamiento según la instancia creada.	Satisfactoria
EC 2.2	Definir una barra de herramientas con parámetros especificados.	V(Lista de botones)	La biblioteca crea una barra de herramientas aplicando su comportamiento según la instancia creada.	Satisfactoria
EC 3.1	Definir una barra de herramientas para mapas	NA	La biblioteca crea una barra de herramientas para mapas aplicando su	Satisfactoria

	con sus parámetros por defecto.		comportamiento a la estructura HTML especificada.	
EC 3.2	Definir una barra de herramientas para mapas con parámetros especificados.	V(Lista de botones)	La biblioteca crea una barra de herramientas para mapas aplicando su comportamiento a la estructura HTML especificada.	Satisfactoria
EC 4.1	Definir una barra de herramientas para mapas con sus parámetros por defecto.	NA	La biblioteca crea una barra de herramientas para mapas aplicando su comportamiento según la instancia creada.	Satisfactoria
EC 4.2	Definir una barra de herramientas para mapas con parámetros especificados.	V(Lista de botones)	La biblioteca crea una barra de herramientas para mapas aplicando su comportamiento según la instancia creada.	Satisfactoria

Resultados de las pruebas de caja negra:

Una vez diseñado los casos de prueba, se efectúan los mismos al sistema con el objetivo de verificar que el marco cumple con las funcionalidades establecidas en la fase de análisis y que está en condiciones de ser entregado a los usuarios finales. A continuación se muestran los resultados de la prueba basada en la técnica particiones de equivalencia para el caso de prueba definido para el caso de uso “Definir barra de herramientas”, los resultados de las pruebas aplicadas a los restantes casos de prueba pueden consultarse en el artefacto “Marco de Trabajo Geomap UI Pruebas al Sistema v1.2.doc”. El caso de uso fue sometido a una primera iteración de pruebas, la que arrojó cinco no conformidades, luego de corregidas las mismas se procede a una segunda iteración la cual devuelve dos no conformidades, después de ser corregidas estas, se realiza una tercera iteración en la cual no se evidencian no conformidades, obteniéndose de estas forma resultados satisfactorios. A continuación se muestra una gráfica con los resultados obtenidos en cada una de las iteraciones realizadas.

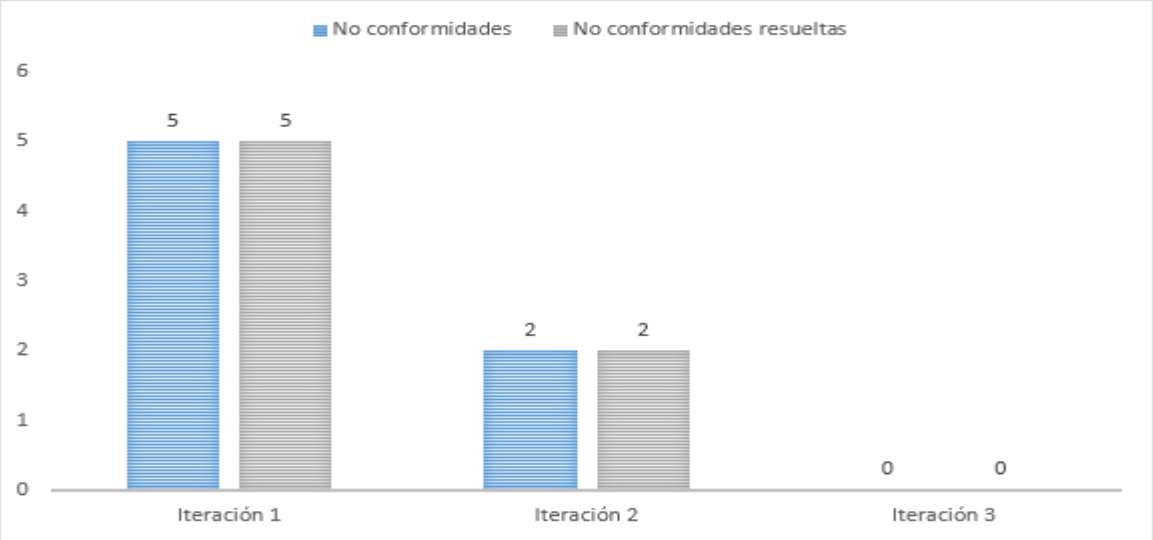


Fig. 9 Resultado de las pruebas de caja negra.

3.3.2 Pruebas alfa

Las pruebas alfa generalmente se producen después de las pruebas del sistema, y afectan tanto a las técnicas de pruebas de caja blanca, como a las de caja negra. Los clientes prueban las funcionalidades y se establece la retroalimentación. Después de esta fase de pruebas, las funciones y características pueden ser agregadas al software.

Las principales características de las pruebas alfa son (Sommerville, 2007):

- Los usuarios externos no están involucrados durante las pruebas.
- Se llevan a cabo en el lugar en donde fue desarrollada la aplicación.
- Se utilizan prácticas de caja blanca y de caja negra.
- Los desarrolladores están involucrados.

Nivel de prueba: Validación de aplicaciones genéricas.
Tipo de Prueba: Aceptación.
Método de prueba: Alfa.
Técnica: Se lleva a cabo en el lugar en donde fue desarrollada la aplicación, en un ambiente controlado, en el cual el desarrollador está presente.

Para aplicar las pruebas de aceptación por el método de prueba alfa se seleccionaron 3 profesionales del diseño de interfaces de la LPS Aplicativos SIG, con años de experiencia desarrollando interfaces. Se escogió el laboratorio en el que radica la LPS como ambiente para ejecutar la prueba. Se definió un documento, donde se hace referencia a las variables que se tomaron en cuenta para este tipo de prueba (Tiempo, Escalabilidad, Inconformidades, Documentación, Tamaño y Consumo de Recursos), los resultados que arrojen estas variables se reflejan en una variable denominada “Calidad”, la cual muestra el nivel de aceptación del software (dan su criterio en porciento) por parte de los especialistas. De acuerdo al promedio de aceptación de cada profesional reflejado en la variable “Calidad”, se define el estado actual del software en una de las categorías siguientes:

- Calidad del software malo: El estado del software es malo cuando el valor de la variable “Calidad” se encuentra entre 0% y 49%.
- Calidad del software regular: El estado del software es cuando el valor de la variable “Calidad” se encuentra entre 50% y 69%.
- Calidad del software bueno: El estado del software es cuando el valor de la variable “Calidad” se encuentra entre 70% y 89%.
- Calidad del software excelente: El estado del software es excelente cuando el valor de la variable “Calidad” se encuentra entre 90% y 100%.

Este documento fue entregado a los participantes para asegurar la calidad y el buen desarrollo del test. Para consulta dicho documento se debe consultar el artefacto “Marco de Trabajo Geomap UI Test de Aceptabilidad.doc” que se encuentra en el repositorio de documentos de la LPS Aplicativos SIG.

Resultados de las pruebas alfa:

Para un mejor análisis de los resultados de las pruebas alfa aplicadas al marco Geomap UI, el resultado de estas se enfocan de acuerdo al estado del software una vez terminado el test, según la valoración que dieron cada uno de los especialistas involucrados en la prueba. Estos estados son Malo, Regular, Bueno y Excelente, y están estrechamente ligados a los porcientos de aceptabilidad del marco por parte de los profesionales. Este vínculo entre porciento de aceptabilidad y estado del software se puede consultar en el anexo 3. Los resultados arrojados por la prueba fueron satisfactorios, debido a la clasificación por parte de los tres especialistas de Bueno como estado del software, los porcientos de aceptabilidad superaron el 80

por ciento en los tres casos. Para más detalle de los resultados de la prueba se debe consultar el artefacto “Marco de Trabajo Geomap UI Pruebas al Sistema v1.2.doc” que se encuentra en el repositorio de documentos de la LPS Aplicativos SIG. A continuación se muestra una gráfica con los resultados obtenidos en cada una de las iteraciones realizadas.

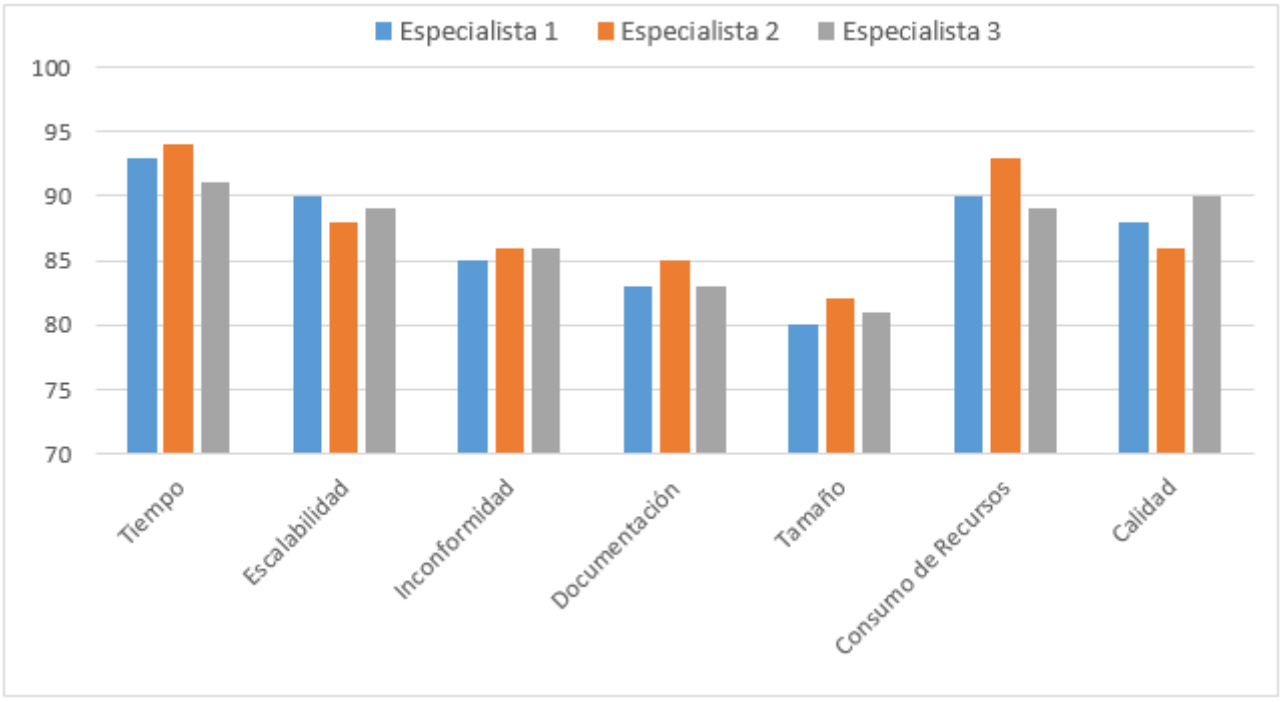


Fig. 10 Resultado de las pruebas alfa.

3.4 Conclusiones parciales

- El modelo del diagrama de componentes al reflejar la estructura general de la solución a través de la interacción entre sus diferentes componentes, permite comprender como debe funcionar el sistema en tiempo de ejecución, durante la creación de los componentes para interfaces de usuario.
- Los resultados de las pruebas de software aplicadas a la solución, demostraron que en diferentes escenarios las funcionalidades implementadas se ejecutan correctamente y que en la LPS Aplicativos SIG dicha solución posee un buen nivel de aceptación.

Conclusiones generales

Durante el desarrollo de la presente investigación, se demostró que:

- El marco de trabajo Geomap UI para la creación de Interfaces de Usuario para Sistemas de Información Geográfica, compuesto por un total de 33 componentes para esta primera versión, constituye una herramienta única implementada con la utilización de las librerías jQuery y OpenLayers, que aporta una solución viable para las personalizaciones de este tipo de sistemas en la LPS Aplicativos SIG.
- El desarrollo de la solución estuvo basado sobre herramientas y tecnologías de código abierto, resultando en la obtención del marco de trabajo Geomap UI que cumple con la política llevada a cabo por la Universidad y el país en cuanto a la soberanía tecnológica.
- La utilización del patrón arquitectónico Modelo-Vista-Controlador y patrones de diseño, permitió que se implementaran componentes de Interfaz de Usuario reutilizables para cualquier tipo de sistema en el que se desee utilizar la biblioteca de modo parcial o total.
- La documentación de todo el proceso de desarrollo de la solución, según las fases de desarrollo que propone la metodología *Rational Unified Process* utilizada, aportará un mejor entendimiento para la implementación de futuras versiones de esta solución.
- Las pruebas Alfa aplicadas a la solución y los resultados que arrojaron, a pesar de haber sido seleccionado un pequeño número de ejecutores, constituyen una muestra positiva del nivel de aceptación hacia la herramienta en la LPS Aplicativos SIG.

Recomendaciones

Los autores recomiendan:

- Definir un equipo de desarrollo que le brinde soporte a la solución para la creación de nuevas versiones.
- Definir o diseñar nuevos componentes que puedan ser útiles para las interfaces.
- Crear temas de presentación que puedan ser utilizados por los desarrolladores de interfaz de usuario que utilicen esta solución.
- Utilizar un Motor de Plantillas en las versiones posteriores para lograr un mejor rendimiento en la creación de las estructuras HTML que define cada componente.
- Aplicar pruebas de caja blanca por especialistas que comprueben el correcto funcionamiento interno del marco.

Referencias bibliográficas:

1. **Academic. 2013.** academic. *academic.* [En línea] 2013. [Citado el: 21 de 4 de 2014.] http://en.academic.ru/dic.nsf/enwiki/125487.

2. **Arnal, Dídac Margaix. 2007.** E-Prints in Library & Information Science. *E-Prints in Library & Information Science.* [En línea] 2007. [Citado el: 31 de 3 de 2014.] http://eprints.rclis.org/9521/1/kx5j65q110j51203.pdf.

3. **Aurelio. 2012.** MappingGIS. *MappingGIS.* [En línea] 2012. [Citado el: 10 de 12 de 2013.] http://mappinggis.com/2012/11/por-que-utilizar-openlayers-y-geoext/.

4. **Avendaño, Raúl Nolasco. 2010.** scribd. *scribd.* [En línea] 2010. [Citado el: 13 de 12 de 2013.] http://es.scribd.com/doc/40075065/Investigacion-Bibliotecas-Estaticas-y-Dinamicas.

5. **Barcelona Activa Cibernarium. 2012.** *Usabilidad: Hacer la Web pensando en el usuario.* Barcelona : s.n., 2012.

6. **Bass, Len, Paul Clements, and Rick Kazman. 1998.** *Software Architecture in Practice.* 1998.

7. **Benzadón y otros, Jorge Duarte, Marcela Hernández. 2007.** Revista Infraestructura Vial. *Revista Infraestructura Vial.* [En línea] 2007. [Citado el: 26 de 3 de 2014.] http://www.lanamme.ucr.ac.cr/riv/index.php?option=com_content&view=article&id=246&Itemid=301 .

8. **Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. & Stal, M. A. 1996.** *System of Patterns: Pattern-Oriented Software Architecture.* West Sussex, England : John Wiley & Sons, 1996.

9. **Cabello, M. E. 2012.** Departamento de Sistemas Informáticos y Computación. *Departamento de Sistemas Informáticos y Computación.* [En línea] 2012. [Citado el: 23 de 11 de 2013.] http://www.dsic.upv.es/docs/bib-dig/informes/etd-06232008-105901/InformeTecnicoMASTER.doc.

Referencias bibliográficas

10. **ClubEnsayos.com. 2013.** clubensayos. *clubensayos.* [En línea] 2013. [Citado el: 12 de 4 de 2014.] http://clubensayos.com/Temas-Variados/JAVA-SCRIPT-VENTAJAS-Y-DESVENTAJAS/222066.html.

11. **Confederación de Empresarios de Andalucía. 2010.** Sistemas de Información Geográfica, tipos y aplicaciones empresariales. *Sistemas de Información Geográfica, tipos y aplicaciones empresariales.* [En línea] 2010. [Citado el: 12 de 2 de 2014.] http://sig.cea.es/SIG.

12. **CPAP. 2013.** CPAP. *CPAP.* [En línea] 2013. [Citado el: 25 de 3 de 2014.] http://www.fing.edu.uy/cpap/cursos/arquitectura-de-software.

13. **Damián Pérez Valdés. 2007.** Los diferentes lenguajes de programación para la web. [En línea] 2007. [Citado el: 14 de Enero de 2014.] Disponible en: http://www.maestrosdelweb.com/principiantes/los-diferentes-lenguajes-de-programacion-para-la-web/.

14. **Diane, Papalia E. 2011.** Psicología. *Psicología.* [En línea] 2011. [Citado el: 5 de 12 de 2013.] http://evebonillap8.blogspot.com/2011/10/sensopercepcion.html.

15. **EDUC. 2013.** educ. *educ.* [En línea] 2013. [Citado el: 3 de 6 de 2014.] http://www.educ.cl/glosario.htm.

16. **Eguíluz Pérez, Javier. 2009.** Cascading Style Sheets home page. [En línea] 2 de Enero de 2009. Disponible en: http://www.librosweb.es/css_avanzado.

17. **El País. 2002.** *El País: Libro de estilo.* Madrid : s.n., 2002.

18. **eO2. 2013.** eO2. *eO2.* [En línea] 2013. [Citado el: 6 de 12 de 2013.] http://www.e02.es/cubic/ap/cubic.php/doc/Identidad-Visual-Corporativa-296.html.

19. **Erich, Gamma, y otros. 2003.** *Patrones de diseño.* s.l. : Pearson Educación, 2003. ISBN 9788478290598.

20. **Freddy Vega, John y Van Der Henst, Christian. 2011.** *Guía de HTML. El presente de la web.* 2011.

Referencias bibliográficas

21. **Free Software Foundation . 2013.** GNU. *GNU.* [En línea] 2013. [Citado el: 1 de 12 de 2013.] http://www.gnu.org/licenses/gpl.html.

22. **Galves, Jorge y Gómez Gallego, Juán Pablo. 2004.** *Fundamentos de la Metodología RUP.* Prereira : Universidad Tegnológica de Pereira, 2004.

23. **García, Jesús D. 2009.** *ExtJS in Action.* s.l. : Manning Publications, 2009.

24. **GeoExt2. 2014.** GeoExt 2. *GeoExt 2.* [En línea] 2014. [Citado el: 7 de 5 de 2014.] http://geoext.github.io/geoext2/.

25. **Gutierrez, Andres Felipe. 2007.** *Kumbia PHP Framework.* 2007.

26. **Gutiérrez, Javier J. 2010.** Departamento de lenguage y Sistemas Informáticos. Universidad de Sevilla. *Departamento de lenguage y Sistemas Informáticos. Universidad de Sevilla.* [En línea] 2010. [Citado el: 4 de 5 de 2014.] http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

27. **Hernández León, Rolando Alfredo y Coello González, Saida. 2011.** *El proceso de la Investigación Científica.* La Habana : Editorial Universitaria del Ministerio de Educación Superior, 2011.

28. **Jachym Cepicky, Stepan Kafka, Premysl Vohnout. 2010.** GIS Ostrava. *GIS Ostrava.* [Online] 2010. [Cited: 5 8, 2014.] http://gis.vsb.cz/GIS_Ostrava/GIS_Ova_2010/sbornik/Lists/Papers/EN_2_11.pdf.

29. **Janalta Interactive Inc. 2014.** techopedia. *techopedia.* [En línea] 2014. [Citado el: 5 de 5 de 2014.] http://www.techopedia.com/definition/3552/black-box-testing.

30. **jQuery UI Team. 2010.** jQuery User Interface. [En línea] 2010. [Citado el: 15 de Enero de 2014.] Disponible en: http://jqueryui.com/about.

31. **L.Northrop y P.Clements, L. 2002.** *"Software Product Lines: Practices and Patterns", SEI Series in Software Engineering.* 2002.

32. **Lanter and Essinger, David Lanter, Rupert Essinger. 1991.** *User-Centered Graphical User Interface Design for GIS.* 1991.

Referencias bibliográficas

33. **Lopes de Oliveira and Bauzer Medeiros, Claudia Bauzer Medeiros and Juliano Lopes de Oliveira. 1996.** *User Interface Issues in Geographic Information Systems.* 1996.

34. **LUISANNET. 2014.** Luisannet Arte y Tecnología. *Luisannet Arte y Tecnología.* [En línea] 2014. [Citado el: 4 de 5 de 2014.] http://www.luisan.net/identidad-corporativa/identidad-corporativa.html.

35. **Microsoft. 2013.** Microsoft. *Microsoft.* [En línea] 2013. [Citado el: 19 de 3 de 2014.] http://msdn.microsoft.com/es-es/library/dd409432.aspx.

36. **—. 2014.** Microsoft Devoloper Network. *Microsoft Devoloper Network.* [En línea] 2014. [Citado el: 22 de 5 de 2014.] http://msdn.microsoft.com/es-es/library/aa291591%28v=vs.71%29.aspx.

37. **Molina, C. D. 2012.** universidadecotec. *universidadecotec.* [En línea] 2012. [Citado el: 2 de 12 de 2013.] http://www.docs.universidadecotec.edu.ec.

38. **NorthWare, Software Development. 2013.** NorthWare. *NorthWare.* [En línea] 2013. [Citado el: 12 de 3 de 2013.] http://www.northware.mx/blog/blog-single-author-full/.

39. **Oracle Corporation and/or its affiliates. 2013.** NetBeans. *NetBeans.* [En línea] 2013. [Citado el: 24 de 4 de 2014.] https://netbeans.org/community/releases/80/index.html.

40. **Orallo, Enrique Hernández. 2001.** disca. *disca.* [En línea] 2001. [Citado el: 20 de 4 de 2014.] http://www.disca.upv.es/enheror/pdf/ActaUML.PDF.

41. **OSGeo. 2014.** OSGeo. *OSGeo.* [En línea] 2014. [Citado el: 8 de 5 de 2014.] http://www.osgeo.org/.

42. **Pressman. 2005.** *Ingeniería del Software.* s.l. : McGrawHill, 2005.

43. **Raygain. 2013.** Raygain Businnes Ideas into Reality. *Raygain Businnes Ideas into Reality.* [En línea] 2013. [Citado el: 2 de 5 de 2014.] http://www.raygain.com/IT-Services/Analyze-Explore.html.

44. **Real Academia Española. 2010.** Diccionario de la lengua española - Vigésima segunda edición. [En línea] 07 de 2010. Disponible en: http://buscon.rae.es/draeI/.

45. **Rumbaugh, Jacobson y Rumbaugh, and Booch. 1999.** *El Proceso Unificado de Desarrollo de Software.* s.l. : Addison Wesley, 1999.

46. **Sencha Inc. 2013.** Sencha. *Sencha.* [En línea] 2013. [Citado el: 21 de 11 de 2013.] http://docs.sencha.com.

47. **—. 2014.** Sencha. *Sencha.* [Online] 2014. [Cited: 5 6, 2014.] http://www.sencha.com/products/extjs/.

48. **Sierra, María.** *Trabajando con Visual Paradigm for UML.* Cantabria : Universidad de Cantabria – Facultad de Ciencias.

49. **Soft112. 2014.** Soft112. *Soft112.* [En línea] 2014. [Citado el: 21 de 4 de 2014.] http://visual-paradigm-for-uml-standard.soft112.com/.

50. **Sommerville, Ian. 2005.** *Igeniería del Software.* s.l. : Addison Wesley, 2005. 7ma Edición.

51. **—. 2007.** *Software Engineering .* s.l. : China Machine Press, 2007. 8va Edición.

52. **TEC. 2007.** Technology Evaluation Center. *Technology Evaluation Center.* [En línea] 2007. [Citado el: 1 de 4 de 2014.] http://health-care-social-work.technologyevaluation.com/es/software/interfaz-de-usuario-amigable-software.html.

53. **Telerik Inc. 2013.** KendoUI. *KendoUI.* [En línea] 2013. [Citado el: 3 de 12 de 2013.] https://www.kendoui.com/purchase/license-agreement/kendo-ui-complete-commercial.aspx.

54. **Telerik. 2014.** Telerik. *Telerik.* [En línea] 2014. [Citado el: 8 de 4 de 2014.] http://www.telerik.com/.

55. **Terreros, Julio Casal. 2014.** Microsoft. *Microsoft.* [En línea] 2014. [Citado el: 19 de 3 de 2014.] http://msdn.microsoft.com/es-es/library/bb972268.aspx.

56. **The Apache Software Foundation. 2011.** subversion. *subversion.* [En línea] 2011. [Citado el: 11 de 2 de 2014.] http://subversion.apache.org/.

57. **The jQuery Foundation. 2013.** JQuery. *JQuery.* [En línea] 2013. [Citado el: 27 de 11 de 2013.] http://docs.jquery.com/Licensing.

Referencias bibliográficas

58. **Trac. 2014.** geoext. *geoext.* [Online] 2014. [Cited: 6 5, 2014.] http://trac.geoext.org/wiki/license.

59. **—. 2011.** Trac. *Trac.* [En línea] 2011. [Citado el: 8 de 4 de 2014.] http://trac.osgeo.org/openlayers/browser/trunk/openlayers/license.txt?rev=11971.

60. **Universidad Nacional del Litoral. 2010.** Emprendedores. *Emprendedores.* [En línea] 2010. [Citado el: 26 de 11 de 2013.] http://www.unl.edu.ar/emprendedores/?p=4776.

61. **WordReference.com.** WordReference. *WordReference.* [En línea] [Citado el: 8 de 12 de 2013.] http://www.wordreference.com/definicion/kit.