Universidad de las Ciencias Informáticas

Facultad 6

Componente de gestión dinámica para la personalización de la Plataforma GeneSIG

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Deivys Ramos Casteleiro.

Tutor(es): Ing. Miosotis Aida Troche.

Ing. Yoan Díaz Cubas.

"Año 58 de la Revolución"

La Habana, Julio 2016

"Para el logro del triunfo siempre ha sido
indispensable pasar por la senda de los sacrificios"

Simón Bolivar

Declaración de Autoría:

Declaro ser el legítimo autor del trabajo titulado: “Componente de gestión dinámica para configuración de la Plataforma GeneSIG”, y reconozco a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____

Firma del Autor

Deivys Ramos Casteleiro.

_____

Firma del Tutor

Ing. Miosotis Aida Troche Rodríguez.

_____

Firma del Tutor

Ing. Yoan Días Cubas.

Datos de Contacto

Síntesis del Tutor
Nombre y Apellidos: Ing. Miosotis Aida Troche Rodríguez
Correo electrónico: miosotisaida@uci.cu.
Año de graduado: 2013.
Profesión: Ingeniero en Ciencias Informáticas.

Nombre y Apellidos: Ing. Yoan Díaz Cubas
Correo electrónico: ycubas@uci.cu.
Año de graduado: 2014.
Profesión: Ingeniero en Ciencias Informáticas.

Dedicatoria

Dedico este Trabajo de Diploma a mis padres, a mi hermana y a mi Chiquitín, por acompañarme en todo momento y brindarme todo el amor y dedicación que he necesitado.

Agradecimientos

A mi padre, por su ejemplo, por ser mi guía y darme el aliento y confianza para seguir adelante durante estos años. A mi madre, por ser la mejor madre del mundo, por su apoyo, su cariño y su amor, por estar siempre ahí para mí a pesar de la distancia. A mi segunda mama, mi hermana, por estar ahí para mí y ayudarme siempre y consentirme tanto. A mi sobrino por exigirme, aunque no lo crean a ser un ejemplo para él. A mis abuelos por el amor, por el cariño y sus consejos. Al resto de mi familia por creer en mí y apoyarme durante mi carrera. A mi Titi por mostrarme el camino indicado, por aguantar todas las pesadeces, mis malcriadeces, gracias por tu amor, por tu amistad, por esa seguridad y confianza que tenías en mí que llegaba a ser más grande incluso que la que yo mismo me tenía, por soportarme durante esta etapa tan difícil que fue el proceso de tesis, sé que para ambos fue un reto del que no salimos ilesos. A mis amigos, Juan Javier, Rafael, Alejandro, Andrés, Migue, Luisma, Rolando, Yosiel, Pepo y Pepe mil gracias por su amistad, por ser como hermanos para mí durante estos años, por los momentos compartidos, por el apoyo incondicional, por los consejos, por aguantar las charlas, en fin, gracias de corazón. Al chino que es como mi segundo padre por todos los consejos. A mis suegros por aconsejarme y acogerme como uno más de la familia. A Lazi, Carlos y familia por su apoyo y consejos. A Dayana, Arianna y Ara mis psicólogas, por aguantar mis pataletas y aconsejarme. A Mayra por ser como una madre más para nosotros y por la obra de arte que hizo con la vestimenta. A mi familia del 107 porque en realidad es eso lo que somos una gran familia. A mi equipo de pelota tanto el de la uci como el de la 6. A aquellos que para mí fueron más que compañeros de grupo y a los que me llevaré siempre en mi corazón. A mis profesores durante estos años, que han sido los encargados de formar a la persona que soy hoy, no solo en el plano académico, sino también en el personal, ha sido un honor ser uno más de sus estudiantes. A mis tutores por su apoyo durante este proceso tan duro y complicado para mí. A todos los que aportaron su granito de arena en el desarrollo de esta investigación. Al Bati, Camilo y Yoan no sé qué hubiera sucedido sin sus ayudas les estaré agradecido siempre. A Neybis y Nelson el dúo dinámico, por sus consejos y ayuda. Al tribunal y al oponente por sus consejos y recomendaciones en aras de lograr un trabajo con la calidad requerida. A todos los que alguna vez me vieron corriendo y me preguntaron, oye como va esa tesis. Gracias a la revolución por darme la oportunidad de estudiar aquí en esta fabulosa universidad.

Resumen

Los Sistemas de Información Geográfica (SIG) son altamente utilizados en el mundo para ubicar los principales puntos de interés. En Cuba, específicamente en la Universidad de las Ciencias Informática (UCI) se desarrolla la Plataforma GeneSIG que permite la generación de proyectos con diferentes objetivos, encaminados a ampliar sus funcionalidades. GeneSIG permite el acceso de usuarios que acceden a la información geográfica que contiene, los que han manifestado poca satisfacción con la manera en que esta se visualiza. Para solucionar la problemática identificada se determinó desarrollar un Componente de Gestión Dinámica que les permitiera a los usuarios modificar los atributos de los *plugins,* para posibilitar que la información se mostrará de acuerdo a sus necesidades. El desarrollo del componente se realizó siguiendo la guía de la metodología PRODESOFT por lo que se realizó el Diagrama de Clases del Dominio, el Modelo del Diseño, el Diagrama de Componentes y el de Despliegue. Se definieron, además los estilos y patrones arquitectónicos, así como el estándar de código utilizado para las clases, atributos y funcionalidades. Se utilizó el Lenguaje de Modelado UML para el diseño, apoyándose en la Herramienta CASE VisualParadigm. El desarrollo se realizó con los lenguajes PHP 5.3 y JavaScript, el framework ExtJS y el IDE PHPStorm. Se aplicaron Pruebas de Caja Negra y Caja Blanca para comprobar el correcto funcionamiento del Componente de Gestión Dinámica. El desarrollo del componente permitió que los usuarios pudieran adaptar la visualización de la información a sus necesidades específicas.

Palabras claves: atributo, componente, satisfacción del cliente, visibilidad.

Abstract

Geographic Information Systems (GIS) are highly used worldwide to locate the main points of interest. In Cuba, specifically at the University of Informatics Sciences (UCI) the GeneSIG platform that allows the creation of projects with different objectives, aimed at expanding its functionality is developed. GeneSIG enables access by users accessing geographic information contained therein, which have expressed some satisfaction with the way this is displayed. In search of a solution to the problems identified was determined to develop a dynamic management component that allows users to modify the attributes of plugins, to enable the information will be displayed according to your needs. The development component was carried out following the guidance of the methodology PRODESOFT so the Class Diagram Domain, the Model Design, Component Diagram Deployment and performed. Defined, plus architectural styles and patterns, as well as the standard code used for classes, attributes and functions. Modeling Language UML for design, relying on VisualParadigm CASE tool was used. The development was done with the languages PHP 5.3 and JavaScript, the framework ExtJS and IDE PhpStorm. Black Box testing and White Box applied to check the correct operation of Dynamic Management Component. Component development allowed users could adapt the display of information to your specific needs.

***Keywords**: attribute, component, customer satisfaction, visibility.*

Índice

Índice de Figuras

Índice de Tablas

Introducción

Desde el inicio de la vida humana la ubicación de los puntos de interés en un área geográfica ha sido una necesidad. De esta forma surgen los mapas con el "objetivo de representar diversos puntos y accidentes de la tierra y la relación que entre ellos establece el hombre" (Líter, y otros, 1992). La representación manual de grandes volúmenes de información dificultaba la lectura y acceso a los datos incorporados en los mapas. Con la llegada de la informática, el desarrollo de software se presenta como una oportunidad ideal para digitalizar los mapas y agruparlos de tal forma que acceder a ellos sea más sencillo.

La manera de alcanzar esta meta fue mediante el desarrollo de los Sistemas de Información Geográfica (SIG), "Estas herramientas van dotadas de procedimientos y aplicaciones para captura, almacenamiento, análisis y visualización de la información georefenciada[1]" (López Cruz, 2013). Por tal motivo han influido considerablemente como elemento complementario para la toma de decisiones. En Cuba con el objetivo de beneficiar dicho proceso se crea la Plataforma GeneSIG, producto realizado en la Universidad de las Ciencias Informáticas, en colaboración con la Empresa XETID y el Grupo Empresarial GEOCUBA como principal proveedor de datos cartográficos en el país (Márquez, 2015).

La Plataforma GeneSIG es una herramienta libre que tiene como objetivo agilizar la implementación de un SIG en la web, a partir de la reutilización de sus módulos y funcionalidades. Además, pone a disposición de los usuarios y desarrolladores servicios de acceso a la información geográfica almacenada en bases de datos geoespaciales. Un aspecto importante del sistema es que logra la integración de mecanismos de edición de mapas y personalización de funcionalidades por roles de usuarios. También admite representar la información *raster*[2] existente (imágenes de satélite, ortofotos o mapas escaneados) y la vectorial. Permite la edición de geometrías, así como las capas de un mapa; y la medición de distancias y superficies basándose en las coordenadas y sistemas de referencia en los que se sirve la información (Márquez, 2015).

GeneSIG tiene como principal ventaja que está compuesto por componentes o *plugins* dedicados a diferentes funcionalidades. Muchos de estos componentes poseen conjuntos de atributos que permiten que se muestren los datos en los mapas y mediante los cuales se realiza la ubicación georeferenciada. Otros se encargan de tramitar información referente a los usuarios y datos generales de la Plataforma GeneSIG. La forma en que los atributos de los *plugins* son mostrados al usuario está predefinida, por lo que no existen diferencias en su visualización. Esto provoca que algunas personas cuando acceden a la Pataforma GeneSIG no se sientan cómodos con la manera en que visualmente se encuentran disponibles los datos, al punto de clasificar la lectura de la información como un proceso engorroso.

[1] Se refiere a un objeto espacial (representado mediante punto, vector, área, volumen) en un sistema de coordenadas.

[2] Se conoce como una matriz de imágenes en *Mapa de bits.*

Individualmente en los componentes que trabajan de manera directa en los mapas puede ocurrir que los colores y los objetos utilizados en estos no contrasten de forma clara, afectando la visualización de los puntos de interés y por tanto los objetivos del usuario. En los componentes de información del usuario la principal dificultad está asociada al gusto de cada uno de ellos y su comodidad para trabajar con la Plataforma GeneSIG. Otro inconveniente se presenta cuando un nuevo desarrollador se enfrenta por primera vez a la configuración de algún componente, e incluso teniendo años de experiencia resulta tedioso localizar en ellos los atributos personalizables. De manera general lo planteado se puede valorar como desventajas pues limita la visibilidad de los datos, la interacción con la Plataforma GeneSIG y afecta la comprensión de la información por parte de los usuarios.

A partir de la problemática descrita se plantea como **problema de la investigación**: ¿Cómo personalizar los componentes de la plataforma GeneSIG?

El presente trabajo de investigación tiene como **objeto de estudio:** la personalización de componentes en los SIG.

Se define como **campo de acción:** la personalización de los componentes en la Plataforma GeneSIG.

Para dar solución al problema planteado se define como **objetivo general:** Desarrollar un componente de gestión dinámica que permita personalizar la información que se visualiza en los componentes de la Plataforma GeneSIG.

Para el desarrollo de la investigación se tienen en cuenta las siguientes **preguntas científicas:**

1. ¿Cuáles son las bases teóricas que fundamentan el desarrollo de un componente para personalizar la Plataforma GeneSIG?
2. ¿Cómo definir las reglas para identificar los componentes personalizables?
3. ¿Cómo permite el componente de gestión dinámica personalizar los componentes que conforman la Plataforma GeneSIG?
4. ¿Cómo favorece el componente de personalización la visibilidad de la información en Plataforma GeneSIG?

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

1. Elaboración del marco-teórico de la investigación asociado a la problemática planteada.
2. Descripción del estado del arte sobre las herramientas que permiten la personalización de componentes en los SIG.
3. Identificación de los atributos configurables en los componentes de la Plataforma GeneSIG.
4. Caracterización de las herramientas, tecnologías y la metodología de desarrollo de software a utilizar en la elaboración de la solución propuesta.

5. Elaboración de los artefactos propuestos por la metodología de software seleccionada.
6. Implementación de la solución propuesta en correspondencia con los requisitos identificados.
7. Aplicación de pruebas de funcionalidad al componente desarrollado.

En la investigación de utilizaron los siguientes **Métodos Científicos**:

Métodos teóricos:

- **Método Analítico - Sintético:** Se utiliza para analizar la bibliografía especializada referente a los componentes de personalización e identificar los elementos claves que contribuyen a la solución del problema planteado. Además, permite sintetizar los conceptos y arribar a conclusiones oportunas para la investigación.
- **Método Histórico-Lógico:** Se utiliza para organizar los elementos teóricos y estructurar la investigación de acuerdo al objetivo planteado. Permite establecer el orden de la información y relacionarla de forma que influya positivamente en el desarrollo de la solución propuesta.

Métodos empíricos:

- **Análisis documental:** Se utiliza en la revisión de la literatura especializada para consultar la información necesaria en el proceso de investigación.
- **Entrevista:** Se utiliza para conocer las consideraciones de los especialistas que desarrollan la Plataforma GeneSIG en relación con las ventajas y desventajas del sistema. Permite comprender la necesidad determinada y las propuestas del equipo de desarrollo para la solución (ver anexo 1).

Resultado Esperado:

Un componente de gestión dinámica para la Plataforma GeneSIG, que le permita al usuario personalizar su entorno de trabajo en el sistema.

El presente trabajo de diploma está estructurado en tres capítulos:

Capítulo 1 “Fundamentos teóricos sobre el componente de gestión dinámica de personalización”: En este capítulo se define el marco teórico de la investigación reflejado a través de los principales conceptos asociados a la temática definida. Se describen las principales herramientas encontradas que se corresponden con la necesidad detectada en la Plataforma GeneSIG para realizar el estudio correspondiente. Además, se describen las tecnologías, herramientas y la metodología de desarrollo que se utiliza en la elaboración del componente para GeneSIG.

Capítulo 2 “Descripción del componente de gestión dinámica de personalización”: Se describe la solución propuesta mediante el Modelo de Dominio del problema. Se identifican los requisitos funcionales y no funcionales del componente y se describen los artefactos de la metodología de desarrollo seleccionada. De manera general se muestra el diseño propuesto para el componente de personalización.

Capítulo 3 “Implementación y pruebas del componente de gestión dinámica de configuración”: Describe las funcionalidades del componente de acuerdo a su desarrollo. Propone las pruebas de software a realizar y muestra los resultados obtenidos, con el fin de validar funcionalmente la solución desarrollada.

Capítulo 1. Fundamentos teóricos sobre el componente de gestión dinámica de personalización

En el presente capítulo se exponen los principales conceptos asociados a la problemática para comprender los fundamentos teóricos de la investigación. Se caracterizan algunas de las herramientas que realizan la personalización de componentes o incluyen funcionalidades que realizan actividades similares con la planteada en el problema de la investigación. Con el estudio de homólogos realizado se define la idea base de la propuesta de solución y se obtiene la información necesaria para dar continuidad al desarrollo de la investigación. Finalmente, se describe el modelo PRODESOFT, como la metodología de desarrollo que se utiliza para guiar la elaboración del componente propuesto como solución; así como las herramientas y tecnologías que intervienen en su implementación.

1.1 Conceptos asociados

La problemática planteada tiene asociado un conjunto de conceptos y definiciones que sustentan teóricamente la investigación. El entendimiento de estos elementos apoya al autor en la búsqueda de la información adecuada y le permite utilizar los términos correctos durante la elaboración de la solución y su descripción.

Sistema de Información Geográfica (SIG)

El Instituto de Investigación de Recursos Biológicos Alexander Von Humboldt en Venezuela, describe un SIG como un conjunto de métodos, herramientas y datos que están diseñados para actuar coordinada y lógicamente para capturar, almacenar, analizar, transformar y presentar toda la información geográfica y de sus atributos con el fin de satisfacer múltiples propósitos (Revista Geoenseñanza, 2006).

Según el artículo “La Plataforma GEOQ como solución alternativa para la representación de datos georeferenciados de los recursos geológicos de Cuba” un SIG es un sistema de hardware, software y procedimientos, diseñados para soportar la captura, el manejo, la manipulación, el análisis, el modelado y el despliegue de datos espacialmente referenciados (geo-referenciados), para la solución de problemas complejos del manejo y planeamiento territorial (Chávez Márquez, y otros, 2013).

De manera más sintetizada se plantea que un SIG es una herramienta fuertemente arraigada dentro de numerosos ámbitos de la sociedad del conocimiento, valorados como un potente instrumento para visualizar, editar y analizar información geográfica” (López, y otros, 2015).

Las definiciones descritas, evidencian la necesidad de que un SIG sea una herramienta que permita la edición de sus datos. El sustento de esta idea se basa, especialmente, en los conceptos del El Instituto de Investigación de Recursos Biológicos Alexander Von Humboldt en Venezuela y el de López 2015, por su relación con la investigación propuesta.

Componente

Desde el punto de vista del desarrollo de software un componente se entiende como "una unidad reutilizable que puede interoperar con otros módulos de software por medio de sus interfaces, además, puede presentarse en forma de código fuente o código objeto" (García Quintela, y otros, 2013).

Según el funcionamiento de los componentes en la Plataforma GeneSIG la definición es similar a la descrita. En GeneSIG los componentes responden a funcionalidades específicas que realiza la Plataforma y agrupa el código fuente que permite su ejecución. Se comprende entonces, desde esta perspectiva, que un componente es una estructura desarrollada para agrupar una funcionalidad o conjunto de ellas, de forma tal que la implementación sea más organizada y el acceso a la información que contiene sea más fácil.

Personalización

Se comprende como la capacidad para proporcionar un contenido que se adapte a los individuos basados en el conocimiento acerca de sus preferencias y el comportamiento (Adomavicius, y otros, 2003).

De acuerdo al concepto planteado, el término personalización se adapta a la investigación como una funcionalidad que permita a los usuarios, que acceden a la Plataforma GeneSIG, modificar la forma en que se muestra la información de acuerdo a sus preferencias.

Atributo personalizable

En la base de conocimiento del Centro de Soporte de IMB[3] se define la personalización de atributos como una manera de añadir un comportamiento a atributos según los tipos de elemento de trabajo (IBM, 2013).

En el contexto de la investigación se puede considerar que un atributo es personalizable cuando la información que representa puede ser manejada de maneras diferentes sin modificar su tipo de dato o su objetivo.

1.2 Sistemas homólogos

El desarrollo de Sistemas de Información Geográfica en el mundo es bastante amplio. Algunos de ellos, incluso, pueden ser configurados por el usuario. A continuación, se describen algunos de estos sistemas por su similitud con la solución propuesta.

1.3.1 MapInfo Professional

MapInfo Professional es una completa herramienta de mapeo por computadoras que permite llevar a cabo análisis geográficos complejos como la redistribución de distritos, el acceso a sus datos a distancia,

[3] International Business Machines Corp. (IBM) es una reconocida empresa multinacional estadounidense que fabrica y comercializa hardware y software para computadoras.

arrastrando y soltando los objetos de mapa en sus aplicaciones, la creación de mapas temáticos que hacen hincapié en los patrones[4] de los datos, y mucho más (MapInfo Professional, 2011). A medida que el campo de la creación de mapas por ordenador ha evolucionado, *MapInfo* se ha situado a la cabeza del sector con productos nuevos diseñados para satisfacer las necesidades de los usuarios.

Una de las funciones más relevantes de este sistema es la de dibujo y edición, que permite crear y personalizar una variedad ilimitada de objetos para los mapas. La edición de estos objetos se realiza, tanto a través de herramientas visibles en la interfaz del sistema como por la línea de comandos, brindando un amplio conjunto de opciones de personalización. El nivel de personalización abarca los colores, patrones de relleno, tipos de línea, símbolos y el texto. Además de ello, el usuario puede dibujar objetos que no se encuentren predefinidos, como pueden ser círculos, polígonos y otros. Los tipos de objetos a dibujar son limitados por lo que *MapInfo* proporciona la búsqueda de registros dentro de esos límites. Es válido destacar que solo se puede realizar la personalización si la capa en la que se encuentra el objeto es editable (MapInfo Corporation, 2003).

En la documentación disponible del software MapInfo, en la página oficial que se encarga de su comercialización *www.pitneybowes.com,* se muestra que el producto es privativo sin revelarse los datos específicos de sus costos. El mecanismo establecido para obtener este tipo de información es directamente con su proveedor en Estados Unidos mediante un formulario de contacto que se activa al acceder por la opción *Buy Now.* Existe una versión de prueba que puede utilizarse para conocer el funcionamiento de *MapInfo* en un plazo de 30 días. Una vez vencido este tiempo se debe comprar la licencia para poder continuar utilizando sus funcionalidades. A pesar que los precios no se encuentran disponibles si se puede determinar que varían de acuerdo a la clasificación de *MapInfo*, según la edición que se adquiera. Estas pueden ser Básica o Avanzada y funcionan para varias arquitecturas[5] del sistema operativo Windows (Pitney Bowes, 2015).

1.3.2 QGIS

QGIS es un Sistema de Información Geográfica de código abierto. El proyecto nació en mayo de 2002 y se establece como un proyecto en *SourceForge*[6] en junio del mismo año. Este software rompe con el tradicionalismo del software SIG (tradicionalmente software propietario caro), pues está al alcance de cualquiera con acceso básico a un ordenador personal. Actualmente funciona en la mayoría de Plataformas Unix, Windows y OS X. QGIS se desarrolla usando el *kit* de herramientas Qt y C++. Se puede definir como

[4] En geografía, un patrón es un conjunto de rasgos esenciales en un diseño gráfico, mapa o escrito.

[5] Se refiere 64 o 32 bits.

[6] Es un sitio web de colaboración que actúa como una central de desarrollos de software para gestionar proyectos de software libre.

un sistema ligero con una interfaz gráfica de usuario agradable y fácil de usar. El objetivo inicial del proyecto, debido al incremento en su utilización, fue proporcionar un visor de datos SIG para las instituciones (QGIS, 2015).

QGIS admite diversos formatos de datos *raster* y vectoriales, puede añadir nuevos formatos al utilizar la arquitectura de complementos. Además, se distribuye bajo la Licencia Pública General GNU (GPL), que garantiza al usuario el acceso a un programa SIG libre de costo y que puede ser libremente modificado. Además, se considera altamente configurable debido a que cuenta con un conjunto de funciones y paneles de dibujo que permiten la creación de gran variedad de objetos; así como la edición para personalizar los existentes. Las herramientas de dibujo son fácilmente utilizables e incluyen la edición de los colores, los patrones de relleno, los tipos de línea, los símbolos y el texto en el mapa. También permiten que se desactiven la mayoría de los elementos en la interfaz de usuario para evitar que se sature la pantalla con datos que el usuario ya no necesita (QGIS, 2015).

1.3.3 Kosmo Plataforma

Kosmo se caracteriza por poseer una interfaz de usuario práctica y amigable, que proporciona todos los elementos necesarios para que el usuario pueda interactuar con el programa de una manera fácil e intuitiva. La interfaz de *Kosmo* está distribuida en cuatro partes fundamentales (SAIG, 2009):

1. **Barra de menús:** se encuentran de forma agrupada algunas de las funcionalidades ofrecidas por *Kosmo*.
2. **Barra de herramientas principal:** ofrece acceso a las funcionalidades más habituales del programa.
3. **Área de visualización:** zona destinada a contener las ventanas que se abren en el programa.
4. **Barra de estado:** ofrece información acerca del estado del programa, visualiza tanto los mensajes informativos, como los mensajes de advertencia.

En temas de edición *Kosmo* tiene implementada la capacidad de editar las capas del sistema, permitiendo almacenar los cambios realizados en ficheros con formato *shape*[7] y en Bases de Datos. De manera general las herramientas de edición que facilita *Kosmo* se pueden dividir en siete grupos: de selección, de manipulación de elementos, de historial, de grupo, de edición puntual, de edición lineal y de edición general. A continuación, se muestra la filosofía bajo la cual opera el sistema *Kosmo*:

[7] Formato vectorial de almacenamiento digital donde se guarda la localización de los elementos geográficos y los atributos asociados a ellos.

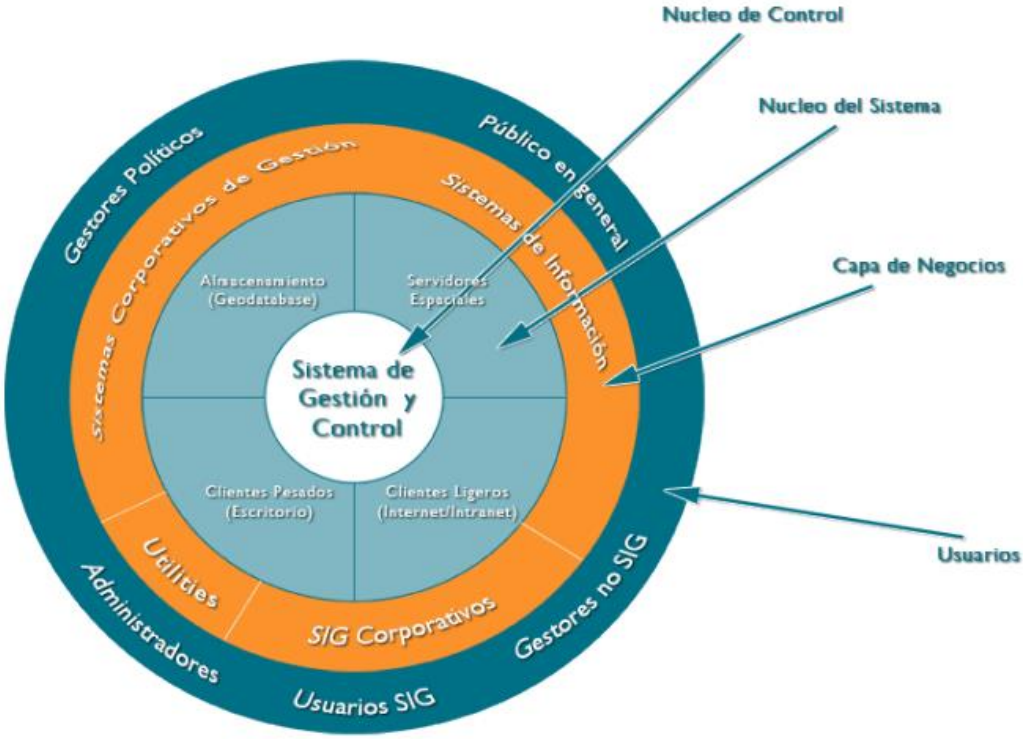


Fig. 1. Arquitectura básica de un SIG Corporativo según *Kosmo*.

Kosmo-Plataforma SIG Libre Corporativa cuenta con las siguientes funcionalidades SIG:

- Almacenamiento y Gestión de la información geográfica en Sistemas de Gestión de Bases de Datos Relacionales (SGBDR), integrada con el resto de información corporativa.
- Edición (multiusuario), consulta y explotación de la información geográfica desde el cliente de escritorio *Kosmo*, junto con funciones de geoprocesamiento, entre muchas otras.
- Publicación de la información geográfica a través de protocolos estándar del OGC (WMS, WFS), lo que permite compartir la información y construir/participar en IDEs[8].
- Publicación de información y funcionalidades específicas en Internet, y acceso mediante el uso de los clientes ligeros *Kosmo*.

1.3.4 ArcGIS

Es un sistema de información geográfica (SIG) integrado que consta de tres partes claves (Kubota, 2014):

- **El software *ArcGISDesktop*:** un conjunto integrado de aplicaciones SIG avanzadas.
- **El *ArcSDETM Gateway*:** es una interfaz para administrar las *geodatabase* (base de datos geográfica) en un sistema de administración de bases de datos (DBMS).
- **El software *ArcIMS*:** es un SIG orientado a Internet para distribuir datos y servicios.

[8] Entornos de Desarrollo Integrado. Software para la compilación de códigos fuentes.

ArcGIS es un sistema de partes que pueden ser organizadas en un *desktop* individual o pueden ser distribuidas en una red de computadores heterogénea de estaciones de trabajo y/o servidores. Ofrecen tres niveles de licencia para su producto, que van aumentando las capacidades de sus tres componentes. Los tres niveles de licencias son, en orden ascendente de capacidad*: ArcView*, *ArcEditor*, y *ArcInfo* (Kubota, 2014)

Específicamente el módulo de *ArcGIS* de interés para la investigación es el *ArcMap* que es "el componente primario del software y es utilizado para todas las tareas que involucren cartografía digital en forma directa, su análisis y edición" (Montaner, y otros, 2008). Dentro de este módulo se encuentra la propiedad *MapLayer* mediante la cual se pueden modificar las características de las coberturas insertadas en los mapas, por ejemplo, los colores de línea de un vector o el grosor de los bordes.

1.3.5 Herramienta para la creación de Sistemas de Información Geográfica

La Herramienta para la creación de Sistemas de Información Geográfica es una aplicación Desktop concebida con el fin de facilitar la personalización de la Plataforma GeneSIG. Fue desarrollado en la Universidad de las Ciencias Informática como resultado práctico del trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas de un miembro del Centro de Desarrollo GEySED. Las tecnologías utilizadas para su implementación fueron el lenguaje de programación *Java* y el *framework Swing* (Labrada, y otros, 2012).

1.3.6 Sistemas de edición de mapas en la web

Durante el estudio de los sistemas homólogos se encontró que existen un amplio número de productos en la web que se encargan de editar la información en los mapas. A partir de que el objetivo planteado para resolver la problemática de la investigación incluye la personalización de los componentes asociados al trabajo con los mapas en la Plataforma GeneSIG se decidió analizarlos también. Durante el estudio se encontró en el sitio web *geotux.tuxfamily.org* una amplia comparación entre más de quince productos que realizan edición de mapas en la web mostrando información referente a (Carrillo, 2012):

- Licencia del producto.
- País de origen del producto.
- Documentación disponible.
- Pertenencia del producto al proyecto OSGeo.
- Categoría del producto.
- Convenciones de disponibilidad (Cliente oficialmente abandonado y Cliente sin versión reciente desde hace más de un año).
- Lenguaje de programación utilizado en el desarrollo.

- Otros datos referentes al desarrollo del producto[9].

A partir de que en la comparación mencionada se muestran datos relevantes para la investigación y que la cantidad de productos analizados son demasiado para incluirlos en el presente trabajo, se consideró oportuno tomar como referente lo que plantea el Msc. en Geoinformática German Carrillo en su artículo "Comparación de clientes web de servicios web geográficos". El estudio de los sistemas se realizará en base a indicadores que estén en correspondencia con las necesidades de la investigación, definiéndose los siguientes:

1. Convenciones de disponibilidad.
2. Lenguaje de programación utilizado en el desarrollo.
3. Categoría del producto.

Los productos encontrados y su relación con los indicadores propuestos se describen en la siguiente tabla:

Tabla 1. Relación producto-indicadores.

	Indicadores		
Productos	**Convenciones de disponibilidad**	**Lenguaje de programación**	**Categoría del producto**
AppForMap	Sin versión reciente	Java Scritp, PHP	Cliente
AtlasMapper	Estable	Java Scritp, Java	Cliente
CartoWeb	Sin versión reciente	PHP	Framework
Chameleon	Sin versión reciente	Java Scritp, PHP	Framework
Dracones	Sin versión reciente	Java Scritp, PHP, Python	Framework
ET - Map	Oficialmente abandonado	Java Scritp, PHP	Cliente
Flamingo	Sin versión reciente	ActionScript	Cliente
FlexLayers	Oficialmente abandonado	ActionScript 3	Biblioteca
Fusion	Estable	Java Scritp, PHP	Framework
GeoExt	Estable	Java Scritp	Conjunto de Herramientas
Geoide	Sin versión reciente	ActionScript 2	Cliente
Geomajas	Estable	Java	Framework
GeoMOOSE	Estable	Java Scritp, PHP	Framework
GisClient	Estable	Java Scritp, PHP	Cliente
GMap	Sin versión reciente	PHP	Cliente
GPAAMP Viewer	Estable	Java Scritp, PHP	Cliente
GWT-OpenLayers	Estable	Java Scritp, PHP	Contenedor
Heron Mapping Client	Estable	JavaScript	Cliente
HSLayers	Sin versión reciente	Java Scritp, PHP, Python	Framework
i3Geo	Estable	Java Scritp, PHP	Cliente

[9] No se incluyen por no ser relevante para la toma de decisiones en la investigación.

iGeoPortal	Estable	Java	Framework
ka-Map	Oficialmente abandonado	Java Scritp, PHP	Biblioteca
kvwmap	Estable	Java Scritp, PHP	Framework
Leaflet	Estable	JavaScript	Biblioteca
Legato	Estable	Java, JavaScript	Cliente
Mapbender	Estable	Java Scritp, PHP	Framework
MapBuilder	Oficialmente abandonado	JavaScript	Biblioteca
MapFaces	Sin versión reciente	Java, JavaScript	Biblioteca
MapFish	Estable	JavaScript; Python	Framework
MapGuide OS Ajax Viewer	Estable	C++	Cliente
MapQuery	Estable	JavaScript	Contenedor
MiraMon	Sin versión reciente	JavaScript	Cliente
msCross	Sin versión reciente	JavaScript	Cliente
OL4JSF	Estable	Java	Contenedor
OpenLayers	Estable	JavaScript	Biblioteca
OpenScales	Estable	ActionScript 3	Framework
p.mapper	Estable	Java Scritp, PHP	Framework
QGIS Web Client	Estable	JavaScript; Python	Cliente
ReadyMap Web SDK	Estable	JavaScript	Conjunto de Herramientas
SLMapViewer	Sin versión reciente	ASP.NET; C#	Cliente
TimeMap	Sin versión reciente	Java	Cliente
UMN MapServer	Estable	C/C++	Biblioteca
WebGIS Public	Oficialmente abandonado	JavaScript	Biblioteca
worldKit	Sin versión reciente	ActionScript	Cliente

1.3.7 Análisis de los sistemas homólogos

Durante la descripción y estudio de las herramientas se verifica que ninguno de los casos expuestos puede ser utilizado como solución a la problemática planteada. Esto se demuestra debido a que, en el caso de *MapInfo,* a pesar de que es un sistema con una amplia variedad de funcionalidades muy positivas tiene como desventaja que es privativo y no se encuentran disponible sus costos. Esto trae como consecuencia que, aun cuando se pudiera adquirir, no es posible obtener el código fuente para modificarlo de forma tal que se ajuste al modelo de desarrollo en la Plataforma GeneSIG. Además, el objetivo de MapInfo se centra solamente en la edición de mapas, por lo que con su obtención solamente se soluciona la parte de la problemática asociada a la personalización de los componentes de GeneSIG relacionados con la edición de mapas.

Otro sistema homólogo descrito es QGIS, que incluye también funcionalidades de edición de información geográfica. En este caso se observó que las tecnologías de desarrollo de QGIS difieren de las utilizadas en el proyecto de GeneSIG y las arquitecturas de ambos productos también son diferentes. Por este motivo,

aunque QGIS se distribuye bajo licencia GPL, sería necesario destinar todo un equipo de trabajo para migrar el código fuente de forma tal que fuera posible integrarlo como un componente a la Plataforma GeneSIG. Similar a *MapInfo* se encuentra QSIG que posee funcionalidades de edición solamente para objetos en los mapas, presentando por lo tanto dificultades para solucionar totalmente el problema de la investigación planteado.

En Kosmo-Plataforma se describe como un producto que opera de forma similar a la Plataforma GeneSIG lo que hace que sea más complejo adaptarlo como un componente Además durante su estudio no se encontró ninguna funcionalidad especifica que realizara la edición de mapas, por lo que no cumple con el objetivo propuesto en su totalidad, solo hace referencia a un subconjunto de él. A pesar de ser descartados como solución, la forma en que *MapInfo* y QGIS describen la edición de la información en los mapas puede ser utilizada como referente para la implementación del componente propuesto.

En el caso de *ArcGIS* se observó cómo ventaja su condición de software modular, lo que facilita la posible integración como un componente de la Plataforma GeneSIG. Sin embargo, el lenguaje de programación utilizado en el desarrollo del sistema es Python y la licencia de comercialización es privativa. En tal caso, aun cuando los precios de cada uno de los niveles no se encuentran disponibles en la web oficial del producto (*desktop.arcgis.com/es/pricing/*), la obtención del módulo no implica acceso al código fuente por lo que no se podría incluir como un componente totalmente funcional de la Plataforma GeneSIG, ni lograr las dependencias necesarias con los demás *plugins*. Por este motivo *ArcGIS* no representa una opción efectiva para solucionar la problemática planteada.

La herramienta para la creación de Sistemas de Información Geográfica desarrollada en el Centro GEySED, aunque cuenta con la ventaja de poseer especialistas preparados en su funcionamiento, tiene como desventajas no estar implementado con tecnologías web y tener una arquitectura Modelo-Vista-Controlador. Ambas características difieren con lo que se utiliza en el desarrollo de los componentes de GeneSIG, por lo que la comunicación entre ambos sistemas debe ser necesariamente de manera remota y no puede trabajar directamente desde la propia plataforma GeneSIG. Esto trae como consecuencia la necesidad de incluir un conjunto de recursos de hardware y software extra como elementos de conectividad, herramientas de conexión y garantizar la disponibilidad de ambos productos en el momento de realizar la personalización. Sin embrago realizar un componente que pueda ser integrado a la Plataforma GeneSIG permite disminuir el consumo de microprocesador, memoria RAM y el número de conexiones a realizar hacia la base de datos permitiendo que la Plataforma GeneSIG no se convierta en un sistema lento y con demasiadas limitantes.

Finalmente, los sistemas de edición de mapas en la web presentados durante el estudio de homólogos parte de la base de que en todos los casos solamente se encargan de la personalización de la información de los mapas. A pesar de ello se decidió analizarlos con el fin de encontrar una solución parcialmente

desarrollada que facilitara la implementación del componente que se propone. Atendiendo a los indicadores seleccionados el análisis arrojó los siguientes resultados:

1. En un primero momento se descartaron los productos que poseían convenciones de disponibilidad diferentes a Estable quedando de la siguiente manera:

Tabla 2. Resultados del análisis del indicador Convenciones de disponibilidad.

	Indicadores		
Productos	**Convenciones de disponibilidad**	**Lenguaje de programación**	**Categoría del producto**
AtlasMapper	Estable	Java Scritp, Java	Cliente
Fusion	Estable	Java Scritp, PHP	Framework
GeoExt	Estable	Java Scritp	Conjunto de Herramientas
Geomajas	Estable	Java	Framework
GeoMOOSE	Estable	Java Scritp, PHP	Framework
GisClient	Estable	Java Scritp, PHP	Cliente
GPAAMP Viewer	Estable	Java Scritp, PHP	Cliente
GWT-OpenLayers	Estable	Java Scritp, PHP	Contenedor
Heron Mapping Client	Estable	JavaScript	Cliente
i3Geo	Estable	Java Scritp, PHP	Cliente
iGeoPortal	Estable	Java	Framework
kvwmap	Estable	Java Scritp, PHP	Framework
Leaflet	Estable	JavaScript	Biblioteca
Legato	Estable	Java, JavaScript	Cliente
Mapbender	Estable	Java Scritp, PHP	Framework
MapFish	Estable	JavaScript; Python	Framework
MapGuide OS Ajax Viewer	Estable	C++	Cliente
MapQuery	Estable	JavaScript	Contenedor
OL4JSF	Estable	Java	Contenedor
OpenLayers	Estable	JavaScript	Biblioteca
OpenScales	Estable	ActionScript 3	Framework
p.mapper	Estable	Java Scritp, PHP	Framework
QGIS Web Client	Estable	JavaScript; Python	Cliente
ReadyMap Web SDK	Estable	JavaScript	Conjunto de Herramientas
UMN MapServer	Estable	C/C++	Biblioteca

2. De los productos restantes se eliminaron aquellos cuyo lenguaje de programación fuera diferente a los utilizados en el desarrollo de la Plataforma GeneSIG:

Tabla 3. Resultados del análisis del indicador Lenguaje de Programación.

	Indicadores		
Productos	**Convenciones de disponibilidad**	**Lenguaje de programación**	**Categoría del producto**
Fusion	Estable	Java Scritp, PHP	Framework
GeoMOOSE	Estable	Java Scritp, PHP	Framework
GisClient	Estable	Java Scritp, PHP	Cliente
GPAAMP Viewer	Estable	Java Scritp, PHP	Cliente
GWT-OpenLayers	Estable	Java Scritp, PHP	Contenedor
i3Geo	Estable	Java Scritp, PHP	Cliente
kvwmap	Estable	Java Scritp, PHP	Framework
Mapbender	Estable	Java Scritp, PHP	Framework
p.mapper	Estable	Java Scritp, PHP	Framework

3. Finalmente se analizó el indicador Categoría del Producto encontrándose que los restantes solamente eran *frameworks*, contenedores o clientes y funcionan como complemento para el desarrollo de sistemas de edición y no como herramientas independientes.

De manera general se descartaron los productos analizados en su totalidad por no cumplir con la adecuada similitud con la propuesta realizada en cuanto a tecnologías de desarrollo, arquitectura y objetivo. Sin embargo, cada uno de ellos sirvió como referencia para comprender la necesidad del cliente y las características básicas de un sistema para personalizar un SIG.

1.3 Tecnologías y herramientas a utilizar

El componente propuesto como solución debe integrarse a la Plataforma GeneSIG por lo que deben ser compatibles. Es por este motivo que las herramientas y tecnologías que se utilizan en su desarrollo son las mismas que las de la Plataforma. Además, como el desarrollo de GeneSIG se realiza dentro de un proyecto ya maduro y consolidado se utiliza la metodología que este propone.

1.3.1 Modelo de Desarrollo de Software PRODESOFT

El Proceso de Desarrollo de Software, PRODESOFT como se conoce sus siglas, se basa en la combinación de varios modelos de procesos: el basado en componentes, el iterativo y el incremental. El proceso se halla disponible en su versión 1.5 y según su especificación, plantea que el ciclo de vida de un proyecto está compuesto por cinco fases: inicio, modelación, construcción, explotación experimental y despliegue (UCID, 2012) (ver figura 3).

Fig. 2. Ciclo de vida de PRODESOFT.

Básicamente las fases de PRODESOFT se encargan de la organización del producto. La descripción de cada una de ellas es la siguiente (UCID, 2012):

- **Inicio:** propicia una visión preliminar de la problemática a resolver y se definen los recursos relevantes para la ejecución del proyecto. Es decir, se describen los objetivos y el alcance del proyecto, se identifican los involucrados y ejecutores (entidades involucradas), se estima de manera general las actividades a realizar durante todo el ciclo de desarrollo del proyecto (Cronograma General), se establece la estrategia a seguir para realizar la modelación del negocio y se capturan de requisitos y de ser necesario se estiman los recursos materiales que deben ser adquiridos.
- **Modelación:** se capturan las partes esenciales del sistema, donde se identifican los procesos de negocio fundamentales y se aceptan los requisitos funcionales, obteniéndose la línea base de la arquitectura y una estrategia de construcción de la aplicación aprobada por los implicados en el proyecto. El hito fundamental de esta fase es la liberación de la arquitectura de sistema, datos y despliegue.
- **Construcción:** se aclaran los requisitos restantes y se completa el desarrollo del sistema sobre una base estable de la arquitectura. Las fases anteriores solo dieron una arquitectura básica que es aquí refinada de manera incremental conforme se construye el producto. En esta fase todas las características, componentes, y requisitos deben ser integrados, implementados, y probados en su totalidad, obteniendo una versión liberada del producto.
- **Explotación Experimental:** se convierte la versión liberada del producto en una solución estable, donde se eliminan los errores que surgen durante las pruebas y se obtiene una certificación funcional y de seguridad del producto.
- **Despliegue:** se instala y configura el sistema para un ambiente de producción real, se capacita al personal que usará la aplicación y se continúa dando soporte durante la explotación del sistema, y se culmina, de ser preciso, con transferencias tecnológicas.

La utilización del modelo PRODESOFT como metodología se debe a que el proyecto de desarrollo de GeneSIG consideró que era el más completo en lo que a documentación y organización respecta. La realización del Componente de Gestión Dinámica para la personalización bajo la guía de este modelo permite que el expediente de proyecto que se genere esté acorde al resto de los expedientes de GeneSIG, para facilitar una única interpretación de su contenido y la centralización de la documentación.

1.3.2 Lenguaje Unificado de Modelado (UML) 2.0

El UML es un lenguaje gráfico para el modelado de sistemas de software que permite representar gráficamente la estructura de un sistema, haciendo posible que cualquier persona ajena o no al proceso de diseño lo pueda entender. Mediante UML se pueden especificar, visualizar y documentar de manera explícita las características de un sistema de software antes y durante su construcción. Con UML se pueden modelar pruebas del sistema, sistemas de hardware, sistemas de negocios, el flujo de trabajo en una empresa, diseño de la estructura de una organización, actividades de planeación de proyectos y hasta sistemas no informáticos. En su versión 2.0 se define un total de 13 diagramas para representar la estructura y la dinámica de los sistemas (Infante O, 2009).

Además de todas las facilidades y características del lenguaje UML, su selección se basa por ser el lenguaje soportado por la herramienta CASE adoptada. También se tiene en cuenta que los diagramas de otros componentes de GeneSIG están realizados con UML, por lo que se considera positivo que los de la solución propuesta estén acorde a estos.

1.3.3 Lenguaje de Programación del lado del Cliente JavaScript

JavaScript es un lenguaje de programación interpretado que se utiliza principalmente para crear páginas web dinámicas por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios (Eguíluz Pérez , 2009).

La utilización de JavaScript como lenguaje del lado del cliente posibilita que la aplicación sea configurable y totalmente dinámica. Por este motivo y la presencia de JavaScript en las interfaces de GeneSIG se decide acogerla para el desarrollo de la solución.

1.3.4 Lenguaje de programación de lado del servidor PHP 5.3

PHP (Procesador de Hipertexto) es un lenguaje de programación de código abierto que se emplea para programar aplicaciones web dinámicas. Se considera un lenguaje del lado del servidor simple de utilizar debido a que existe gran documentación sobre su uso. La versión estable PHP 5 posee como características más significativas (Vázquez Mariño, 2008):

- Mejor soporte para la Programación Orientada a Objetos.

- Mejor soporte para MySQL.
- Mejor soporte a XML (XPath[10], DOM, etc.)
- Soporte nativo para SQLite.
- Soporte integrado para SOAP[11].
- Iteradores de datos.
- Manejo de excepciones.

Además de que GeneSIG emplea PHP en el desarrollo de sus componentes y que el componente a desarrollar debe integrarse a la misma, PHP permite lograr un sistema lo más libre de licencias posible cubriendo todas las necesidades; por lo que se puede aceptar como parte de los lenguajes utilizados para la implementación.

1.3.5 Framework de desarrollo para JavaScript ExtJS 3.0

ExtJS es un framework para desarrollar aplicaciones Web con estándares, patrones, arquitecturas comprobadas en la industria de la programación como POO[12], MVC[13] y MVVM[14] para la creación de estructuras reutilizables y robustas para todos los dispositivos. De manera particular al utilizar ExtJS se reduce la complejidad de la aplicación. No es necesario utilizar dependencias ni versiones de terceras librerías, todo lo encuentras dentro de este poderoso Framework. Todo esto incrementa la velocidad a la vez que se estandariza el estilo de programación (ExtJS, 2016).

El uso de ExtJS posibilita el empleo de un gran número de componentes visuales que mejoran considerablemente la calidad de las aplicaciones. La selección no solo se debe a que es utilizado por GeneSIG, sino que además brinda la posibilidad de validaciones de formularios de todo tipo, basándose en expresiones regulares y tipos de datos.

[10] Lenguaje que permite construir expresiones que recorren y procesan un documento XML.

[11] Protocolo estándar que define como dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos.

[12] Paradigma de Programación Orientado a Objetos.

[13] Arquitectura Modelo-Vista-Controlador.

[14] Arquitectura Modelo-Vista, Vista-Modelo.

1.3.6 Herramientas CASE Visual Paradigm for UML Enterprise Edition 8.0

Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Se integra con varios IDEs (Entorno de Desarrollo Integrado) y soporta múltiples usuarios trabajando sobre un mismo proyecto. Ofrece interoperabilidad entre diagramas, ya que permite a partir de un diagrama obtener otro que guarde relación con el mismo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación (Universidad Carlos III, 2015).

Además de sus múltiples ventajas, *Visual Paradigm* es multiPlataforma, por lo que puede utilizarse en el desarrollo de productos software libre sin generar problemas de licencias. De manera general su adopción para el modelado de la solución propuesta se basa en la amplia experiencia de los especialistas del proyecto GeneSIG en su utilización, por lo que el autor contaba con una fuente confiable de asesoría. Además, se encuentran las características técnicas de la herramienta, que posibilitan que los artefactos propuestos por la metodología estén acordes a lo esperado por el proyecto y la base documental del software esté representada de forma clara y totalmente interpretable.

1.3.7 Entorno de Desarrollo Integrado PhpStorm 9.0

PhpStorm es un Entorno de desarrollo Integrado (IDE) diseñado específicamente para facilitar el desarrollo de aplicaciones web, escritas en PHP. Sus características principales son (Gajda, 2013):

- Fácil Navegación.
- Edición de código y re-factorización[15].
- Buena sincronización de los archivos de despliegue.
- Control de Versiones[16].
- Depuración[17] del código fuente.

Además, permite el completamiento inteligente de código, detecta código duplicado y mezcla diferentes lenguajes (JavaScript, SQL, XML). Se reconoce como un editor de JavaScript avanzado, basado en DOM[18], que es ligero, de fácil instalación, multiPlataforma y de código abierto (JetBrains, 2016).

La elección del *PHPStorm* como IDE para el desarrollo del componente propuesto se basó esencialmente en su característica de compatibilidad con el lenguaje JavaScript y su condición de herramienta ligera. De manera habitual las aplicaciones que se realizan en este lenguaje suelen ser pesadas al ejecutarlas y contar

[15] Técnica de la ingeniería de software para reestructurar un código fuente, alterando su estructura interna sin cambiar su comportamiento externo.
[16] Gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo.
[17] Proceso de identificar y corregir errores de programación.
[18] Interfaz que permite el acceso dinámico a través de la programación para acceder, añadir y cambiar el contenido estructurado para la web.

con un IDE que no sobrecargue los procesos de la computadora, es una ventaja a tener en cuenta. Es importante resaltar que el auge que alcanzado *PHPStorm* en la comunidad de programadores ha permitido que exista documentación disponible sobre su uso, tanto en español como en inglés. Además de comunidades y foros de debates donde pueden realizarse consultas sobre él, por lo que el autor puede documentarse con facilidad.

1.4 Conclusiones parciales

El estudio de los conceptos asociados a la problemática permitió definir los elementos relevantes de los Sistema de Información Geográfica, estructurando la investigación acorde al objeto de estudio y el campo de acción definidos. El análisis de los sistemas *MapInfo*, QGIS y *Kosmo* arrojaron como resultado la necesidad de crear una nueva solución que cumpla con los objetivos propuestos, aunque se toman como referencia algunas características de los sistemas descritos. A partir de la decisión tomada se selecciona PRODESOFT como la metodología de desarrollo a seguir por los resultados alcanzado durante se utilización en el proyecto GeneSIG. De manera similar se acogieron las tecnologías y herramientas con las que se desarrollan los componentes de la Plataforma GeneSIG, siendo estas el lenguaje de modelado UML, los lenguajes de programación PHP y JavaScript, la herramienta CASE *VisualParadigm*, el IDE PHPStorm y el *framework* EXTJS para las interfaces del componente. De esta manera se facilita la integración del Componente de Gestión Dinámico con la Plataforma GeneSIG, lo que permite mayor compatibilidad entre sus funcionalidades.

Capítulo 2. Descripción del Componente de Gestión Dinámica para la personalización.

2.1 Introducción

En el presente capítulo se describe la propuesta de solución a la problemática planteada. A partir del análisis realizado se diseña el Modelo de Dominio que representa los principales ámbitos que incluye el Componente de Gestión Dinámica para la personalización, mostrándose los elementos de mayor importancia que deben tenerse en cuenta en el desarrollo del componente. Se realiza el levantamiento de los requisitos funcionales y no funcionales, los cuales conforman un resumen detallado por la importancia que supone para el componente. Además, mediante el diagrama de diseño se muestran las clases que contienen el componente, sus atributos y funciones, y las relaciones que poseen tanto entre ellas como con la propia Plataforma GeneSIG. De manera general durante la elaboración del capítulo se incluyen los elementos del diseño que permiten organizar el desarrollo del componente.

2.2 Solución Propuesta

El Componente de Gestión Dinámica para la personalización de los demás componentes de la Plataforma GeneSIG tiene como objetivo, en primer lugar, identificar los componentes del sistema que poseen atributos personalizables. Para ello se debe realizar la búsqueda por todos los componentes e identificar cuáles de los atributos que posee, pueden ser personalizados. De acuerdo a los estudiado durante la fundamentación teórica y mediante consultas con especialistas del proyecto GeneSIG se estableció la propiedad *attr_config*[19] que permite seleccionar aquellos atributos que pueden cambiar su valor, por ser esta una de las principales características que deben valorarse para poder modificarlos.

Una vez que los componentes son detectados se muestran al usuario para que pueda seleccionar cuál de ellos desea personalizar. De esta forma se muestran los atributos que posee y se brinda un conjunto de opciones para cada uno de ellos, que pueden ser elegidas en correspondencia con las preferencias del usuario. Una vez que se hayan realizado todos los cambios se brinda la posibilidad de guardarlos, y si se desea, exportarlos con el fin de poder en algún momento volver a utilizar los mismos valores cambiados. Se incluye también la opción de importar la personalización realizada para que, de esta manera, el usuario lo pueda reutilizar en accesos posteriores a la Plataforma.

[19] Se representa con la sintaxis attr_config<nombre_atributo1>:<tipo_dato1>:<valor1><nombre_atributo2>:<tipo_dato2>:<valor2>…<nombre_atributoN>:<tipo_datoN>:<valorN>

2.3 Modelo de dominio

El modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las “cosas” que existen o los eventos que suceden en el entorno en el que trabaja el sistema. A diferencia del modelo de negocio las clases del dominio se obtienen del conocimiento de algunos expertos o de experiencias con sistemas similares al que se está desarrollando (Rumbaugh, y otros, 2000).

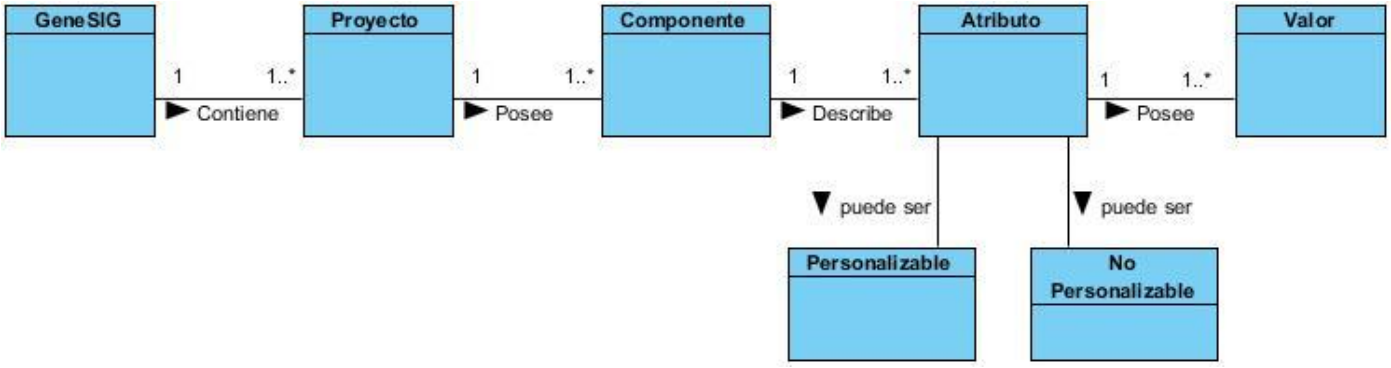


Fig. 3. Diagrama de clases del dominio para el componente de personalización.

1.3.1 Definición de las clases del Modelo de Dominio

Tabla 4. Conceptos que describen el Diagrama de Clases del Dominio para la solución propuesta.

Concepto	Descripción
GeneSIG	Plataforma base sobre la que se desarrollan las personalizaciones y los proyectos.
Proyecto	Representa las personalizaciones SIG del centro, las cuales contienen los módulos cuyos parámetros son editados frecuentemente por los administradores del sistema de forma manual.
Componente	Constituyen los elementos atómicos que conforman los proyectos y suelen tener varios parámetros o atributos de interés para los administradores.
Atributo	Son propiedades que contienen los componentes.
Valor	Son medidas cuantitativas o cualitativas que definen al atributo para un contexto específico
Atributo Personalizable	Es la cualidad del atributo que permite que pueda ser modificado por el usuario sin representar un riesgo para el funcionamiento del componente.
Atributo No Personalizable	Es la cualidad del atributo que indica que no se debe modificar pues afecta el funcionamiento del componente.

1.3.2 Descripción del modelo de dominio

La Plataforma GeneSIG está conformada por diferentes proyectos, donde cada uno de ellos está dedicado a objetivos diferentes en el desarrollo de los SIG. Los proyectos implementan diferentes tipos de componentes que responden a funcionalidades específicas. El objetivo de estos componentes es facilitar el análisis y la representación visual de servicios de acceso a la información geográfica; debido a que se componen de atributos propios que tienen valores definidos. En algunos casos los valores de los atributos responden a características estáticas del componente, por lo que no debe modificarse su contenido, conociéndose como Atributo No Personalizable. En otros casos, el valor del atributo está asociado a particularidades del componente que no necesariamente tienen que comportarse de la misma manera para todos los usuarios, siendo estos los Atributos Personalizables. Estas características permiten conocer si un componente puede personalizarse, atendiendo a la clasificación de sus atributos.

2.4 Requisitos de software

La Especificación de Requisitos de Software (ERS) es una descripción completa del comportamiento del sistema que se va a desarrollar. Un requisito funcional (RF) puede expresarse de dos formas: de alto nivel o de usuario, y de bajo nivel o de sistema. Un requisito de alto y/o bajo nivel puede tener implícito varios requisitos que responden al más general y, por lo tanto, estos también tienen que ser especificados. Además, la ERS también contiene requisitos no funcionales o complementarios (RNF). Los no funcionales son requisitos que imponen restricciones en el diseño o la implementación.

2.4.1 Requisitos Funcionales

Los requisitos funcionales definen los servicios que un sistema debe proveer, su comportamiento a las diferentes entradas y situaciones. Para la aplicación en cuestión se definen los siguientes requisitos, organizados mediante el artefacto propuesto por el modelo PRODESOFT:

Tabla 5. Requisitos Funcionales del Componente de Personalización

No.	Nombre	Descripción	Prioridad	Complejidad
RF1	Listar proyecto	Este requisito permite listar el proyecto activo para el usuario.	Alta	Media
RF2	Listar *plugins* del proyecto	Este requisito permite listar los *plugins* del proyecto activo que contenga atributos personalizables.	Alta	Media
RF3	Mostrar atributos configurables	Este requisito permite visualizar los atributos personalizables de cada *plugin*.	Alta	Media

RF4	Adicionar atributo	Este requisito permite añadir un nuevo atributo al *plugin* seleccionado. (esperar x tutor)	Alta	Media
RF5	Modificar atributo	Este requisito permite modificar los valores de los atributos personalizables del *plugin* seleccionado.	Alta	Media
RF6	Eliminar atributo	Este requisito permite eliminar atributos personalizables que el usuario no desee modificar.	Alta	Media
RF7	Guardar configuración	Este requisito permite salvar en un fichero la personalización realizada por el usuario a cada *plugin*.	Alta	Media
RF8	Importar configuración	Este requisito permite obtener la configuración del *plugin* desde una ubicación especificada por el usuario.	Alta	Media
RF9	Exportar configuración.	Este requisito permite guardar el fichero de configuración en una ubicación especificada por el usuario.	Alta	Media
RF10	Mostrar cadena de configuración en texto plano.	Este requisito permite al programador visualizar el comportamiento interno del componente durante las modificaciones.	Media	Media

El modelo de desarrollo PRODESOFT establece una serie de parámetros básicos para cada uno de los requisitos funcionales. Para ello se genera un informe conocido como Especificación de requisitos que es almacenado en el expediente documental del componente. Alguno de los datos que se manifiestan en este documento son:

- La tabla de especificación de requisitos.
- Prototipo de interfaz del requisito.

A continuación, se muestran algunos de los resultados obtenidos por cada uno de los requisitos definidos para el componente propuesto. El resto de los requisitos se encuentran especificados en la sección de los anexos. (Ver anexo 2)

Requisito “Listar proyecto”

Tabla 6. Requisito Funcional “Listar Proyecto”.

Conceptos tratados	Conceptos	Atributos
	Proyecto	No procede

Precondiciones	**Precondiciones**	**Pre-requisito**
	-Ser usuario del sistema. -Deben existir proyectos creados en la Plataforma GeneSIG.	No procede
Descripción	El usuario debe: 1. Autenticarse en la Plataforma GeneSIG con un usuario válido. 2. Seleccionar la opción del Menú correspondiente al Componente de Personalización Dinámica para acceder a él. Como resultado se debe: 3. Listar, en la parte izquierda de la interfaz, el proyecto activo para el usuario.	
Complejidad	Media.	
Validaciones	-El usuario y la contraseña deben coincidir con algún usuario del sistema GeneSIG o tener el rol de desarrollador. -Solo se lista el proyecto activo para el usuario activado. -Un usuario solo puede acceder a un proyecto a la vez.	
Post-condiciones	-Se debe mostrar estrictamente un proyecto. El resultado no debe ser nulo ni más de un proyecto a la vez.	
Post-requisito	-Listar *plugins* de proyectos	

Tabla 7. Requisito Funcional “Listar *plugins* de proyecto”.

Conceptos tratados	**Conceptos**	**Atributos**
	Plugins	*attr_config*
Precondiciones	**Precondiciones**	**Pre-requisito**
	-Debe haberse listado al menos un proyecto.	Listar Proyecto.
Descripción	El usuario debe: 1. Seleccionar el proyecto listado. Como resultado se debe: 1. Mostrar la lista de *plugins* dentro del proyecto que pueden personalizarse.	
Complejidad	Media.	
Validaciones	-El *plugin* debe cumplir con la propiedad *attr_config*, por ser la que define si contiene o no atributos personalizables.	

	-Se debe seleccionar solo un *plugin* a la vez.
Post-condiciones	-Se pueden listar varios *plugins* o mostrarse un resultado nulo.
Post-requisito	-Mostrar atributos configurables.

Tabla 8. Requisito Funcional “Mostrar atributos configurables”.

Conceptos tratados	**Conceptos**	**Atributos**
	Atributos Configurables	*attr_config*
Precondiciones	**Precondiciones**	**Pre-requisito**
	-Debe haberse seleccionado al menos un *plugin*.	Listar *Plugins.*
Descripción	El usuario debe: 1. Seleccionar el *plugin* que desea modificarse. Como resultado se debe: 2. Mostrar la nueva interfaz con los atributos del *plugin.*	
Complejidad	Media.	
Validaciones	-Se debe mostrar al menos un atributo.	
Post-condiciones	-Se muestra una nueva interfaz con los atributos del *plugin*. Deben coincidir los datos especificados en la propiedad con los atributos mostrados.	
Post-requisito	-Adicionar atributo. -Modificar atributo -Eliminar atributo. -Guardar configuración.	

Tabla 9. Requisito Funcional “Adicionar atributo”.

Conceptos tratados	**Conceptos**	**Atributos**
	Adicionar	Nombre, Tipo de Dato
Precondiciones	**Precondiciones**	**Pre-requisito**
	1. Debe haberse seleccionado un *plugin*. 2. El atributo debe estar correspondencia con el objetivo del *plugin* al que se adiciona.	Mostrar atributos configurables.

Descripción	1. Seleccionar el *plugin* que desea modificarse. 2. Seleccionar la opción Adicionar. 3. Introducir los datos del atributo a adicionar. 4. Dar *enter* para aceptar el nuevo atributo
Complejidad	Media.
Validaciones	-El atributo adicionado debe cumplir con el objetivo del *plugin*.
Post-condiciones	-Se debe mostrar el nuevo atributo añadido.
Post-requisito	-Guardar configuración. -Exportar configuración. -Mostrar cadena de valor en texto plano.

Tabla 10. Requisito Funcional “Modificar atributos”.

Conceptos tratados	**Conceptos**	**Atributos**
	Modificar	*attr_config*
Precondiciones	**Precondiciones**	**Pre-requisito**
	-Debe seleccionarse al menos un atributo.	Mostrar atributos configurables.
Descripción	El usuario debe: 2. Seleccionar el atributo que se desea modificar. 3. Dar doble clic sobre el valor del atributo para activar la edición. 4. Insertar el nuevo valor del atributo. 5. Dar *enter* para aceptar la modificación. Como resultado se debe: 6. Mostrar el atributo con el nuevo valor introducido.	
Complejidad	Media.	
Validaciones	-El valor introducido debe ser del mismo tipo de datos que el atributo modificado.	
Post-condiciones	-Se deben mostrar los nuevos valores del atributo modificado. Además se muestra el cambio realizado en la propiedad *attr_config*.	
Post-requisito	-Guardar configuración. -Exportar configuración. -Mostrar cadena de valor en texto plano.	

Tabla 11. Requisito Funcional “Eliminar atributo”.

Conceptos tratados	**Conceptos**	**Atributos**
	Eliminar	No procede

Precondiciones	**Precondiciones**	**Pre-requisito**
	-Debe haberse seleccionado un *plugin*. -Se deben mostrar los atributos del *plugin* seleccionado. -Se debe seleccionar el atributo que se desea eliminar.	Mostrar atributos configurables.
Descripción	El usuario debe: 1. Seleccionar el *plugin* que desea modificar. 2. Mostrar los atributos del *plugin*. 3. Seleccionar el atributo a eliminar. 4. Seleccionar la opción Eliminar. Como resultado se debe: 5. Dejar de mostrar el atributo eliminado para el usuario autenticado.	
Complejidad	Media.	
Validaciones	-El atributo se elimina solamente de la propiedad *attr_config* y queda descartado por el usuario para ser personalizado.	
Post-condiciones	-Se deja de mostrar el atributo para ese usuario en la propiedad *attr_config*.	
Post-requisito	-Guardar configuración. -Exportar configuración. -Mostrar cadena de valor en texto plano.	

Tabla 12. Requisito Funcional “Guardar configuración”.

Conceptos tratados	**Conceptos**	**Atributos**
	Configuración	*attr_config*
Precondiciones	**Precondiciones**	**Pre-requisito**
	-Se debe haber realizado algún cambio en el *plugin*.	Adicionar atributo. Modificar atributo. Eliminar atributo.
Descripción	El usuario debe: 1. Accionar la opción de Guardar en el menú del componente para almacenar la nueva configuración realizada. Como resultado se debe:	

	2. Sobrescribir los atributos del *plugin* para el usuario autenticado.
Complejidad	Media.
Validaciones	-La configuración inicial debe ser diferente a la que se guarda.
Post-condiciones	-Se deben mostrar los atributos con los nuevos valores y los cambios se deben reflejar en la propiedad *attr_config*.
Post-requisito	-Exportar Configuración. -Mostrar cadena de valor en texto plano.

2.4.2 Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Son las características que hacen al producto atractivo, usable, rápido y confiable. Dichos requisitos resultan fundamentales en la evaluación de las características no funcionales del software como: seguridad y confiabilidad. A continuación, se definen los requisitos no funcionales del módulo.

- **Usabilidad.**

- El módulo puede ser usado por personas con conocimientos básicos en el manejo de computadoras.
- Las funcionalidades principales del módulo deben estar orientadas a íconos para un mayor reconocimiento por parte del usuario.

- **Fiabilidad.**

- El módulo será diseñado para garantizar el correcto tratamiento de los valores de los atributos y su correcta integración con cada uno de los módulos, sin embargo, dependerá del conocimiento del administrador del sistema para la correcta utilización de los valores a insertar.

- **Soporte.**

- La LPS Aplicativos_SIG se hace responsable del soporte del módulo.

- **Restricciones de diseño.**

  - El diseño debe ser sencillo, con pocas entradas, donde no sea necesario mucho entrenamiento para utilizar el sistema.
  - Se debe lograr un producto altamente configurable y extensible, que pueda integrarse con la Plataforma GeneSIG.

- **Requisitos para la documentación de usuarios en línea y ayuda del sistema.**
  - La Plataforma GeneSIG debe brindar a sus clientes finales, manuales de usuarios en los que se especifiquen todas las funciones que pueden realizar sus módulos y cómo hacerlo; incluyendo además la descripción del componente de gestión dinámica para configuración de la Plataforma GeneSIG.
- **Interfaz.**

  Interfaces de usuario.
  - El sistema debe cumplir con las líneas de interfaz de usuarios establecidas por GeneSIG.

  Interfaces de hardware.

  Para las PCs clientes:
  - Se requiere tengan tarjeta de red.
  - Al menos 1 Gb de memoria RAM.
  - Procesador 512 MHz como mínimo.

  Para los servidores:
  - Se requiere tarjeta de red.
  - El Servidor de Mapas debe tener como mínimo 2 GB de RAM y 40 GB de disco duro.
  - El Servidor de base de datos debe tener como mínimo 2 GB de RAM y 40 GB de disco duro.
  - Procesador de 3 GHz como mínimo.

  Interfaces de software. Como requerimientos mínimos de software:

  Para las PCs clientes:
  - Navegador *Mozilla Firefox*, *Safari* u otro que cumpla con los estándares W3C.
  - Sistema operativo: GNU/Linux.

  Para los Servidores:
  - Sistema operativo GNU/Linux Ubuntu Server 11.04 o superior.

2.5 Arquitectura del sistema

La arquitectura del sistema es la organización fundamental de un sistema encarnada en sus componentes, la relación entre ellos y el ambiente y los principios que orientan su diseño y evolución (Kiccillof, 2004).

"La arquitectura del software de un programa o sistema de cómputo es la estructura o estructuras del sistema, lo que comprende a los componentes del software, sus propiedades externas visibles y las relaciones entre ellos". Es decir, es una representación que permite analizar la efectividad del diseño para cumplir los requerimientos establecidos, considerar alternativas arquitectónicas en una etapa en la que hacer cambios al diseño todavía es relativamente fácil y reducir los riesgos asociados con la construcción del software (Pressman, 2010).

Dentro de la arquitectura de la Plataforma GeneSIG se concibe un elemento importante como son los *plugins*. Estos, a su vez, están organizados de manera similar para todos los proyectos de la Plataforma. El componente que se propone debe formar parte de este grupo cuando su desarrollo haya concluido y por tanto debe poseer la misma estructura que los ya existentes. A partir de esto se decide que la estructura del Componente de Gestión Dinámica para la personalización quede conformada de la siguiente manera:

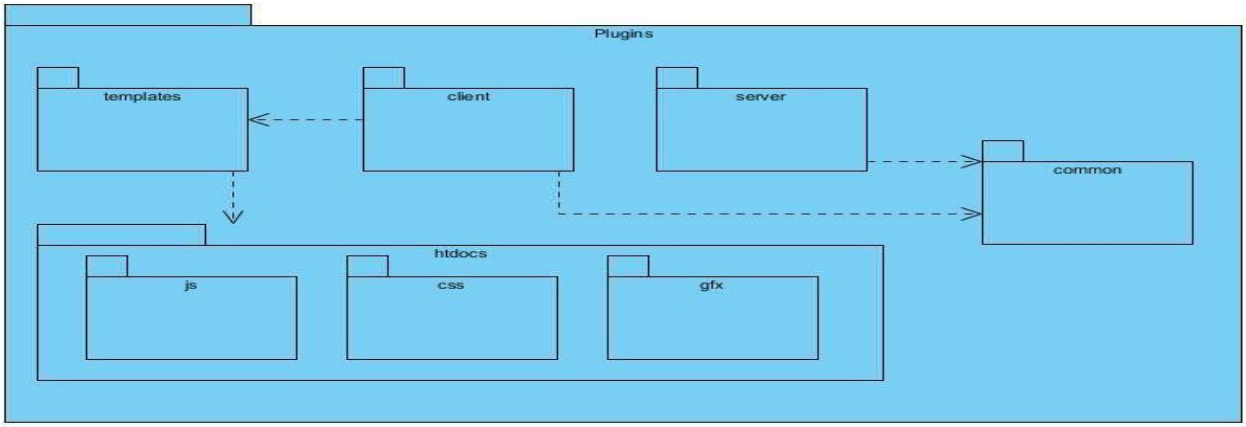


Fig. 4. Estructura de un *plugin* en GeneSIG.

2.5.1 Descripción de la estructura de la solución

El Componente de Gestión Dinámica se estructura similar al resto de los *plugins* de la Plataforma GeneSIG. Esta estructura se establece con el fin de organizar el desarrollo y facilitar la comunicación entre las principales funcionalidades del *plugin*, quedando de la siguiente manera:

- **templates**: agrupa las interfaces o vistas que se muestran al usuario como punto de acceso al *plugin*.
- **client**: organiza las clases del desarrollo para las interfaces, facilitando la relación mediante la recepción de peticiones y el envío de respuestas.
- **server**: organiza las clases del desarrollo de las principales funcionalidades del *plugin*, procesando las peticiones realizadas a través del *client* y generando los resultados de que se muestran en las interfaces.
- **common**: agrupa al *client* y al *server* en un único acceso para facilitar la comunicación con la Plataforma GeneSIG.
- **htdocs**: organiza los elementos que garantizan la interactividad y el diseño en la web. Específicamente agrupa los ficheros css[20], js[21] y gfx[22].

[20] Lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML.

[21] Archivo de texto plano que contiene código *Javascript* que se ejecuta en páginas web.

[22] Término utilizado en el mundo digital para definir gráficos o efectos gráficos.

2.5.2 Estilo Arquitectónico

Un Estilo Arquitectónico es una transformación impuesta al diseño de todo un sistema. El objetivo es establecer una estructura para todos los componentes del sistema (Pressman, 2010). Son un tipo particular de estructura que definen los posibles patrones a usar en la implementación de la aplicación. En el caso de la solución que se propone se considera viable utilizar en su desarrollo los mismos estilos arquitectónicos de GeneSIG para propiciar una mejor integración entre el componente propuesto como solución y los demás *plugins* de la Plataforma. Es por este motivo que se describe a continuación el estilo Cliente-Servidor.

Estilo Cliente-Servidor

La arquitectura Cliente-Servidor definido como un modelo organiza un sistema como un conjunto de servicios y servidores asociados, más unos clientes que acceden y usan los servicios. Los principales componentes de estos servicios son (Sommerville, 2005):

1. Un conjunto se servidores que ofrecen servicios a otros sistemas.
2. Un conjunto de clientes que llaman a los servicios ofrecidos por los servidores.
3. Una red que permite a los clientes acceder a los servicios.

De manera más informal la arquitectura Cliente-Servidor se identifica como un componente servidor, que ofrece ciertos servicios, escucha que algún otro componente requiera uno; un componente cliente solicita ese servicio al servidor a través de un conector. El servidor ejecuta el requerimiento (o lo rechaza) y devuelve una respuesta (Reynoso, 2004). La principal ventaja de este modelo es que pueden insertarse clientes y servidores sin afectar la distribución del sistema.

2.5.3 Arquitecturas Basadas en Componentes

Define la composición de software como el proceso de construir aplicaciones mediante la interconexión de componentes de software a través de sus interfaces (de composición), abogaba por la utilización de componentes prefabricados sin tener que desarrollarlos de nuevo (Robaina, 2008).

2.5.4 Arquitectura Orientada a Objetos

Se basa en los principios de la Programación Orientada a Objetos (POO): encapsulamiento, herencia y polimorfismo. Sus componentes son los objetos, o más bien instancias de los tipos de dato abstracto. Las interfaces están separadas de las implementaciones y se puede modificar la implementación de un objeto sin afectar a sus clientes (Camacho, 2004).

2.5.5 Patrones Arquitectónicos

"Los patrones arquitectónicos para el software definen un enfoque específico para el manejo de algunas características del sistema y definen cierto número de dominios propios del patrón" (Pressman, 2010). Están concebidos como buenas prácticas conocidas en el diseño arquitectónico que tienen como objetivo dar solución a un problema en específico. Normalmente se dividen en Patrones GRASP (General Responsability Assignment Software Patterns) y GOF (Gang of Four), según los objetivos de cada grupo de ellos.

Patrones GRASP

Experto: Se encarga de "asignar una responsabilidad al experto en información: la clase que posee la información necesaria para cumplir con la responsabilidad. Si se hacen en forma adecuada, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, y se presenta la oportunidad de reutilizar los componentes en futuras aplicaciones (Lerman, 1999)." En el componente de personalización dinámica este patrón se evidencia en las clases *ClientToolConfig* y *ServerToolConfig*, siendo la primera de ellas experta en procesar los datos que son enviados a través del navegador web y la segunda en lograr la interacción con el servidor. El uso de este patrón en el componente permite que se conserve el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide.

Creador: Plantea que se debe asignar a la clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos (Lerman, 1999)

- B agrega los objetos A.
- B contiene los objetos A.
- B registra las instancias de los objetos A.
- B utiliza específicamente los objetos A.

B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado (así que B es un Experto respecto a la creación de A).

El patrón Creador se refleja en la solución a través de las clases *ClientToolConfig* y *ServerToolConfig*, encargadas de crear una instancia de las clases *ToolConfigRequest* y *RasterToolConfig*. Estas últimas describen las variables que contienen la información para conformar los valores de entrada de la solicitud que realiza el *ClientToolConfig* al *ServerToolConfig* y viceversa.

Alta Cohesión: En la perspectiva del diseño orientado a objetos, la cohesión (o, más exactamente, la cohesión funcional) es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme (Lerman, 1999). La alta cohesión se garantiza que cada una de las clases del componente *ToolConfig* dando a las clases indicadas las responsabilidades y que se retroalimenten

entre ellas mediante una correcta relación. El uso de este patrón permite que se pueda mejorar la claridad y facilidad en que se entiende el diseño, se simplifique el mantenimiento y existan mejoras en las funcionalidades.

Bajo Acoplamiento: El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con alto (o fuerte) acoplamiento recurre a muchas otras (Lerman, 1999). En correspondencia con la alta cohesión, el bajo acoplamiento se refleja en cada una de las clases del componente *ToolConfig*, con el objetivo de disminuir la dependencia entre ellas. De esta forma, no se afectan las clases por cambios de otros componentes, son fáciles de entender por separado y fáciles de reutilizar.

Controlador: Responsable de recibir o manejar un evento del sistema. En el componente *ToolConfig* esta evidenciado en la clase *PluginManager,* que controla la petición realizada desde la interfaz para poder acceder al *plugin* que atenderá dicha petición.

Patrones GOF

GoF[23] representa un grupo de expertos que se dedicaron a analizar los problemas recurrentes en el desarrollo de software y realizaron una clasificación y agrupación a partir de dos criterios, su propósito y alcance. Según este grupo los patrones describen soluciones simples y elegantes a problemas específicos en el diseño de software orientado a objetos (Guerrero, et al., 2013). Estos patrones se agrupan en tres categorías principales: Creacionales, Estructurales y de Comportamiento.

En el componente de Personalización, a partir de la arquitectura de la Plataforma GeneSIG, se emplearon dos de los patrones GoF conocidos también como “La Banda de los Cuatros”: Singleton y Command.

Singleton: Este patrón permite “garantizar el acceso único a una clase mediante una única instancia. De esta forma se controla el acceso a las clases”. (Kubota, 2014). Se evidencia en la clase ServerContext utilizando el método GetMapObject, para garantizar una única instancia del objeto mapa en toda la ejecución del programa.

Command: Encapsula las peticiones a través de un objeto, lo que permite realizar operaciones como gestionar las acciones de dicho objeto (Larman, 2003). Se utiliza para la comunicación a través de las interfaces de usuario, específicamente a través de la clase AJAXHelper que es la encargada de comunicar las interfaces con el servidor.

[23] Hace referencia al grupo de patrones declarados por lo que se conoce como la Banda de los Cuatros.

2.6 Modelo de Diseño

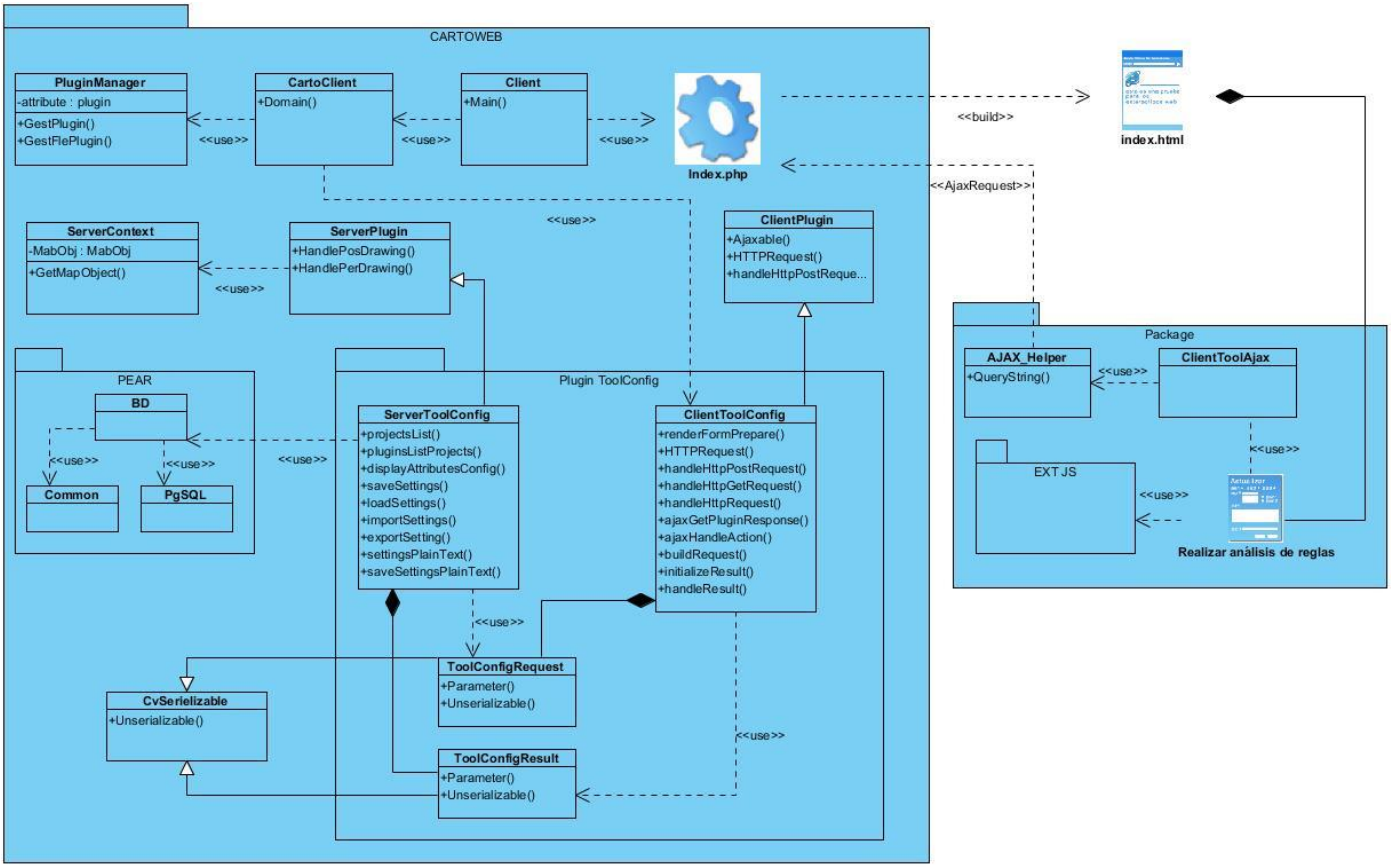


Fig. 5. Modelo de Diseño para la solución propuesta.

2.6.1 Descripción del Modelo del diseño para la solución

El Modelo de Diseño describe las clases que representan las funcionalidades que se desarrollan en la fase de implementación. De manera general se divide en dos paquetes; uno que contiene las clases de la Plataforma, donde se incluye además el Componente de Gestión Dinámica y otro para los elementos que garantizan la interactividad en la web. El paquete CARTOWEB representa las clases *PluginManager*, *CartoClient*, *Client*, *ClientPlugin*, *ServerContext*, *ServerPlugins* y *PEAR* que son clases generales de la Plataforma que el componente necesita para ejecutar sus funcionalidades. Además, muestra las siguientes clases del componente:

- **ServerToolConfig**: incluye el desarrollo de los requisitos funcionales con los que el componente debe cumplir. Estas funcionalidades implementan la base para que las peticiones del usuario se respondan correctamente.
- **ClientToolConfig**: incluye el desarrollo de las funcionalidades que permiten que la información del componente se muestra de forma correcta al usuario. A través de ellas se reciben las peticiones que el *Server* procesa y se les da respuesta.

- **ToolConfigRequest**: recibe las peticiones del usuario.
- **ToolConfigResult**: devuelve los resultados al usuario.

La comunicación del Componente de Gestión Dinámica con la Plataforma se establece mediante la conexión entre los servidores y los clientes de cada uno de ellos. Además, el acceso al componente solo es posible mediante la propia interfaz de la Plataforma GeneSIG utilizando el mismo fichero *Index.php*.

2.7 Estándar de codificación

Los estándares de codificación garantizan la comunicación fluida y directa entre los programadores, permitiendo el aumento de la reutilización y el mantenimiento de los sistemas. (Batista Desdin, y otros, 2014) En el desarrollo del componente se utiliza la Notación *CamelCase.* Esta notación se subclasifica en *UpperCamelCase* cuando la primera palabra comienza con mayúscula y *lowerCamelCase* cuando comienza con minúscula. Ambos casos están presentes en el código fuente de la solución de la siguiente manera:

- **Clases y Objetos**: Utilizan la notación *UpperCamelCase.*
- **Métodos y Atributos***:* Utilizan la notación *lowerCamelCase.*

```
class ServerToolConfig extends ClientResponderAdapter
{
    public $valor;
    public $action;
    public $direccionBase;

    //new
    private $manager;

    public function __construct()
    {
        parent::__construct();
    }

    public function handlePreDrawing($request)
    {
        $result = new ToolConfigResult();
        $result->action = $request->action;
```

Fig. 6. Uso de la Notación *CamelCase*.

2.8 Conclusiones parciales

La descripción de la solución permitió conocer los objetivos del Componente de Gestión Dinámica y establecer los pasos necesarios para el diseño de los artefactos de la metodología de desarrollo PRODESOFT. Se modeló el Diagrama de Clases del Dominio mediante el cual se representaron los conceptos más importantes del entorno que involucra la solución propuesta con la Plataforma GeneSIG. Además, tanto la descripción de la solución como el diagrama mencionado ayudaron a definir ocho

requisitos funcionales que deben ser implementados y siete no funcionales con los que debe cumplir el componente una vez terminado. Los requisitos funcionales determinados fueron incluidos en el Modelo de Clases diseñado que permitió, además, establecer las relaciones entre las clases del componente y como estas deben vincularse a la Plataforma GeneSIG. A partir de la necesidad de que el componente se integre a la Plataforma se decidió sustentar el desarrollo de la solución en la arquitectura que esta utiliza describiendo los estilos arquitectónicos de Llamada y Retornos, y el Basado en Componentes. Se aplicaron patrones de diseño *GRASP* y *GoF* mediante los cuales se posibilitó una mejor organización en el desarrollo del componente y se utilizó la notación *CamelCase*, en sus dos opciones *UpperCamelCase* y *lowerCamelCase*, logrando aumentar la comprensión del código fuente por terceros y por tanto su reutilización.

Capítulo 3. Implementación y pruebas del componente de gestión dinámica de personalización.

3.1 Introducción

En el presente capítulo se describe la implementación de la solución mediante el Diagrama de Componentes, Diagrama de Despliegue y la descripción de las principales funcionalidades del componente de Personalización. Se muestran algunas interfaces importantes que apoyan la comprensión del componente obtenido y responden a las actividades más comunes que el usuario podrá realizar. Se proponen las pruebas que verifican el funcionamiento del componente de Personalización y se muestran los resultados obtenidos. De esta manera se comprueba el cumplimiento de los Requisitos Funcionales definidos y el objetivo propuesto para solucionar la problemática.

3.2 Diagrama de Componentes

El Modelo de Implementación, como también se le conoce al Diagrama de Componentes, es comprendido por “un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Un componente es el empaquetamiento físico de los elementos de un modelo como lo son las clases del diseño”. (Jacobson, y otros, 1999)

El diagrama manifiesta y organiza la relación que existe entre el Componente de Personalización Dinámica con la Plataforma GeneSIG. De manera específica el *plugin* desarrollado posee un *index* a través del cual se obtiene la información y se envían los resultados directamente al *index* del paquete *CartoWeb*. Al mismo *index* se conecta el *ClientToolAjax* para el dinamismo del componente. Otras relaciones importantes se establecen entre el *ClientPlugin* y el *ServerPlugin*, con el *ClientToolConfig* y el *ServerToolConfig* del componente respectivamente. A continuación, se muestra el Diagrama de Componente modelado con el lenguaje UML, mediante la herramienta CASE *Visual Paradigm* (ver figura 7).

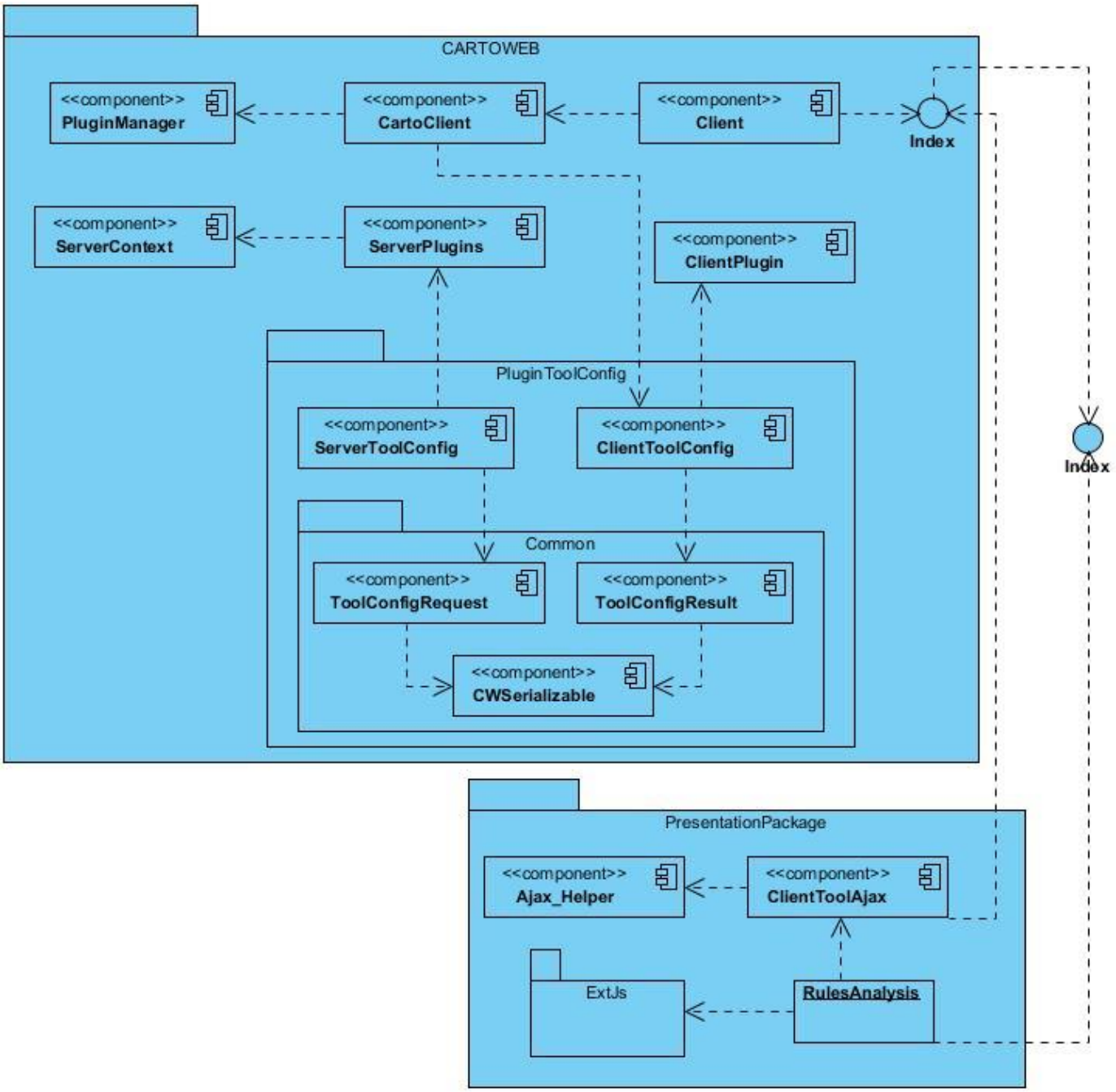


Fig. 7. Diagrama de Componente del Componente de Personalización Dinámica.

3.3 Diagrama de Despliegue

Los Diagramas de Despliegue muestran la disposición física de los distintos nodos que entran en la composición de un sistema y el reparto de los programas ejecutables sobre estos nodos. (Visconti, y otros, 2004)

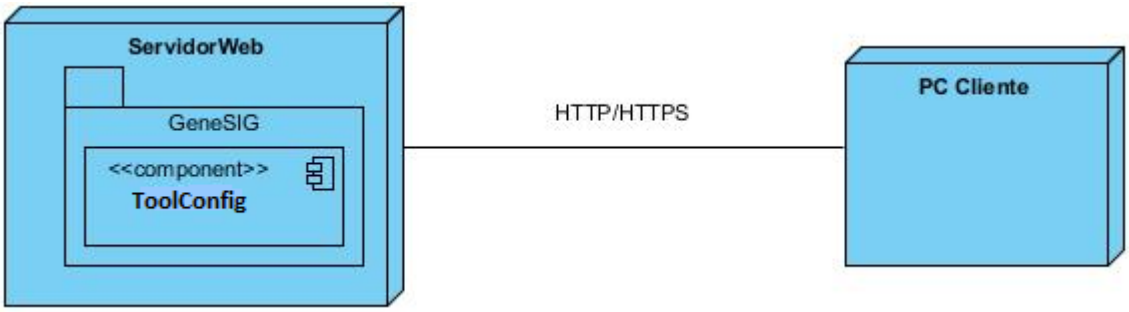


Fig. 8. Diagrama de Despliegue del Componente de Personalización Dinámica.

En la representación modelada se reflejan dos nodos fundamentales, interconectados mediante el protocolo HTTP[24] o HTTPS[25]. Estos nodos responden a las siguientes descripciones:

- **Nodo Servidor Web:** Representa el servidor seleccionado para hospedar, ejecutar y mostrar la Plataforma GeneSIG, con todos los proyectos y *plugins* que contiene. En el caso particular de la investigación se muestra el Componente de Personalización Dinámica desarrollado.

- **Nodo PC Cliente:** Representa cada una de las computadoras desde donde los usuarios pueden conectarse al servidor web y acceder a la Plataforma GeneSIG para, con los permisos establecidos, utilizar las funcionalidades del Componente de Personalización Dinámica.

3.4 Descripción de la Solución

El Componente de Personalización Dinámica está desarrollado como un *plugin* para la Plataforma GeneSIG. Es por este motivo que la arquitectura, las tecnologías de desarrollo y el diseño de interfaz cumplen con lo establecido para poder integrar el componente a la Plataforma, garantizando un mayor nivel de compatibilidad. El objetivo principal de la solución es permitir que los usuarios modifiquen la forma en que se visualiza la información de los *plugins* que utiliza, siempre y cuando los atributos de estos sean modificables. Se define que un *plugin* contiene atributos personalizables mediante la propiedad *attr_config*, la cual cambia cuando los atributos son modificados y establece, además, el formato de la configuración cuando se guarda o se exporta. Mediante el desarrollo del Componente de Personalización Dinámica el usuario puede gestionar las configuraciones de la Plataforma GeneSIG a sus expectativas y necesidades.

[24] Protocolo de comunicación que permite las transferencias de información en internet.

[25] Versión segura del protocolo HTTP.

El acceso al componente se realiza a través de la Plataforma GeneSIG. Una vez que el usuario se autentica aparece en el menú la opción de Configurar (ver figura 9) mediante la cual se inicia el componente de personalización.

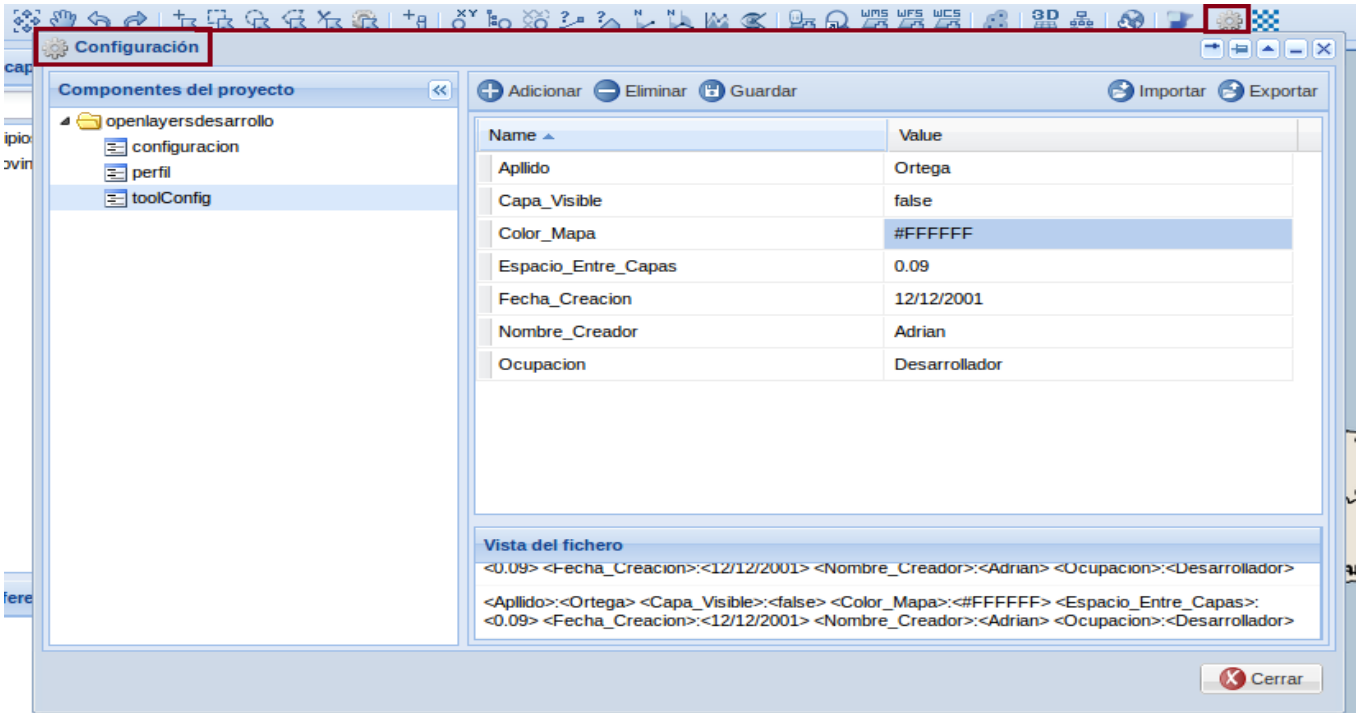


Fig. 9. Acceder al Componente de Personalización Dinámica.

Una vez seleccionada la opción se muestra en la Plataforma la interfaz del componente (Ver figura 10). En la parte izquierda se encuentra el Árbol de Proyectos en el cual se listan aquellos proyectos que contienen *plugins* personalizables. El parte derecha superior se muestra el menú de edición del componente con las opciones de Adicionar, Modificar, Eliminar, Guardar, Importar y Exportar. Estas opciones pueden ser aplicadas a la configuración del *plugin* que se desee modificar. Debajo del menú está establecido el espacio para listar los atributos configurables a los que se le aplicaran los cambios. En la parte inferior de la ventana se encuentran las Vista del fichero en el componente donde el usuario puede verificar los cambios que realiza en la propiedad *attr_config*. En esta área se puede verificar como quedará la configuración ante de guardarla o exportarla.

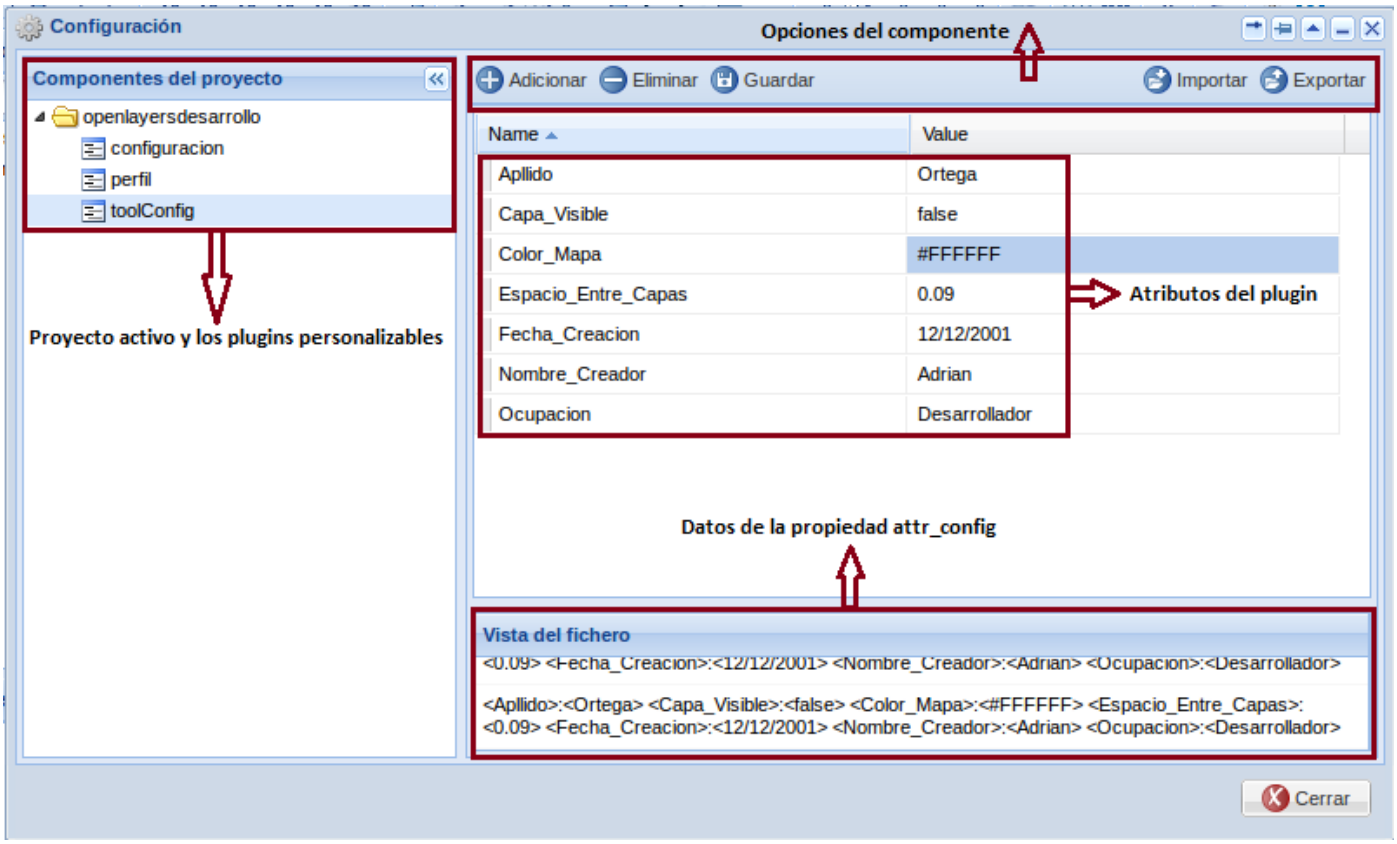


Fig. 10. Interfaz del componente.

El primer paso a realizar para comenzar la personalización es desplegar el proyecto que se muestra en el árbol de Componentes del proyecto. Una vez seleccionado se muestran los *plugins* configurables que este posee (ver figura 11). Es importante destacar que solo se muestra al usuario un proyecto a la vez que necesariamente será el que se activó en la Plataforma GeneSIG cuando se autentica. Además, el proyecto se muestra siempre, aun cuando no contenga ningún *plugin* que cumpla con la propiedad *attr_config*. La personalización de estos *plugins* debe hacerse seleccionándolos de uno en uno.

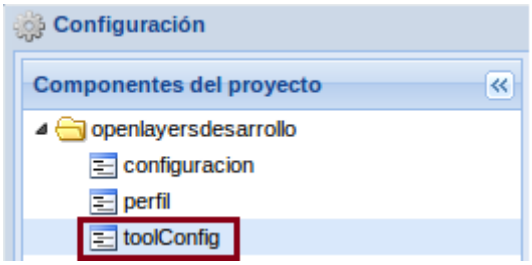


Fig. 11. Listar *plugins* del proyecto activo.

Cuando se selecciona el *plugin* que se desea personalizar, en el área de edición se muestran todos sus atributos configurables. Cada uno de ellos posee, en el campo *Value*, la opción de editar el valor del atributo

para introducir el cambio que el usuario necesita realizar (ver figura 12). Los atributos deben modificarse de uno en uno, seleccionado con el clic el valor o mediante las opciones de edición que brinda el componente.

Fig. 12. Área de edición de atributos.

Además de la opción de modificar los valores de los atributos, el componente permite adicionar un nuevo atributo o eliminarlos en caso de ser necesario. Para añadir un atributo se debe seleccionar la opción Adicionar en el Menú del componente. Una vez seleccionada esta opción se muestra una nueva pantalla donde se pueden introducir los datos del atributo que se adiciona (ver figura 13). En el caso de que el usuario desee eliminar un atributo de los existentes debe seleccionarlo y luego dar clic en la opción Eliminar. A continuación, se muestra un mensaje de confirmación para concluir el proceso de eliminación. Si se decide interrumpir este proceso se debe seleccionar la opción Cancelar (ver figura 14).

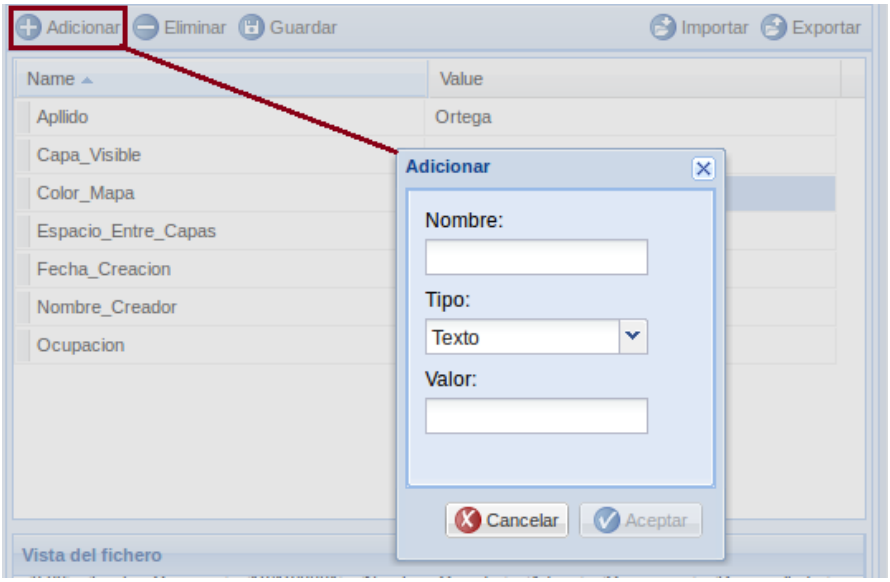


Fig. 13. Adicionar atributo.

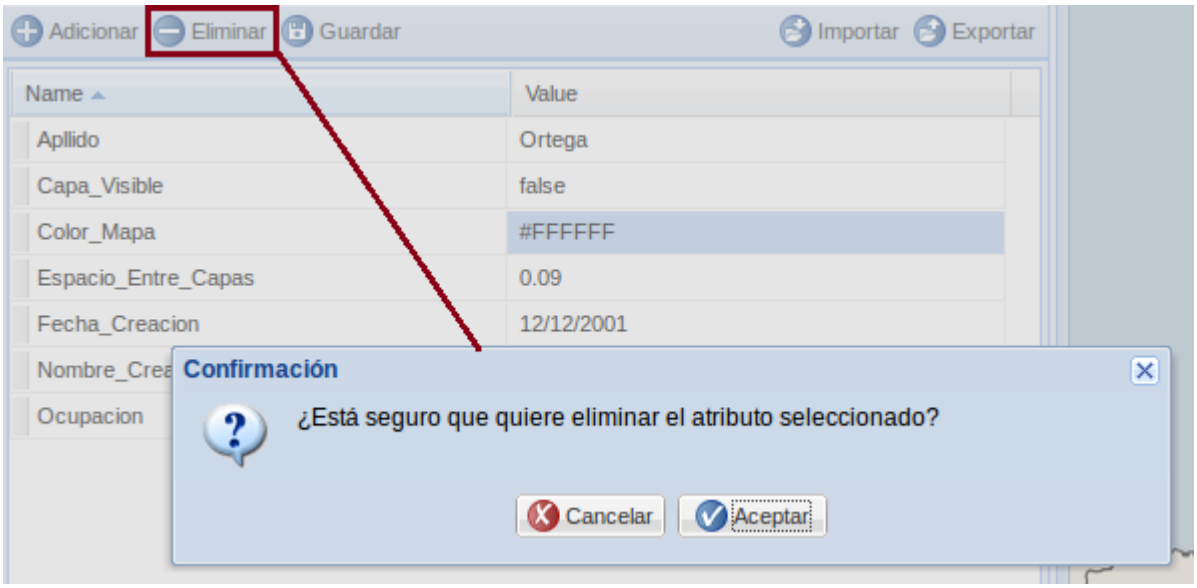


Fig. 14. Eliminar atributo.

Una vez que el usuario realiza la personalización del *plugin* puede guardar la nueva configuración mediante la opción Guardar del menú (ver figura 15). Esta acción implica que en el próximo acceso a la Plataforma GeneSIG el *plugin* personalizado muestra la última configuración realizada en el perfil autenticado. Además, la configuración puede ser exportada del sistema como un fichero que el usuario pueda almacenar de forma externa al sistema e importada cuando se desee, incluso por otro usuario al que se le haya concedido el fichero correspondiente (ver figura 16).

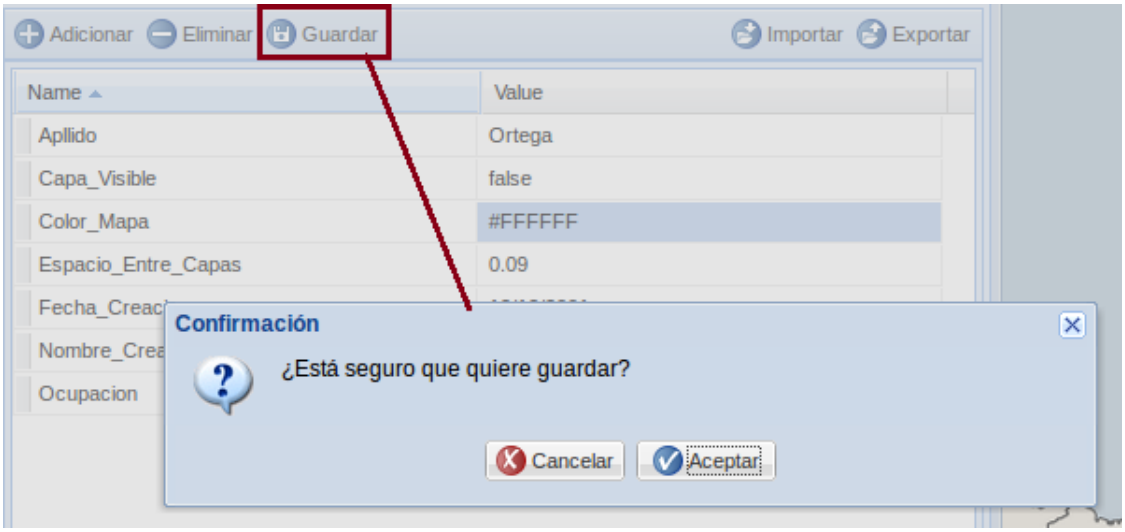


Fig. 15. Guardar configuración.

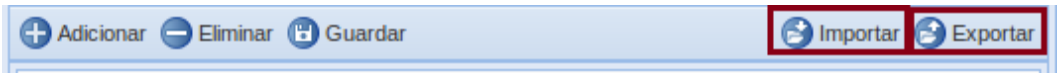


Fig. 16. Importar o Exportar configuración.

Cada edición realizada o acción ejecutada en el componente se muestra en tiempo real en la parte inferior de la interfaz. En esta área se debe reflejar, además, la regla de validación que los atributos deben cumplir para que el *plugin* pueda considerarse como personalizable (ver figura 17).

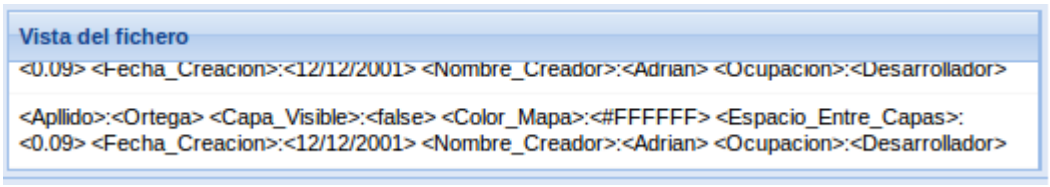


Fig. 17. Eventos del componente.

Cuando se determine concluir el trabajo en el componente se puede cerrar al hacer clic en la opción correspondiente al final de la interfaz o en la parte superior derecha (ver figura 18).

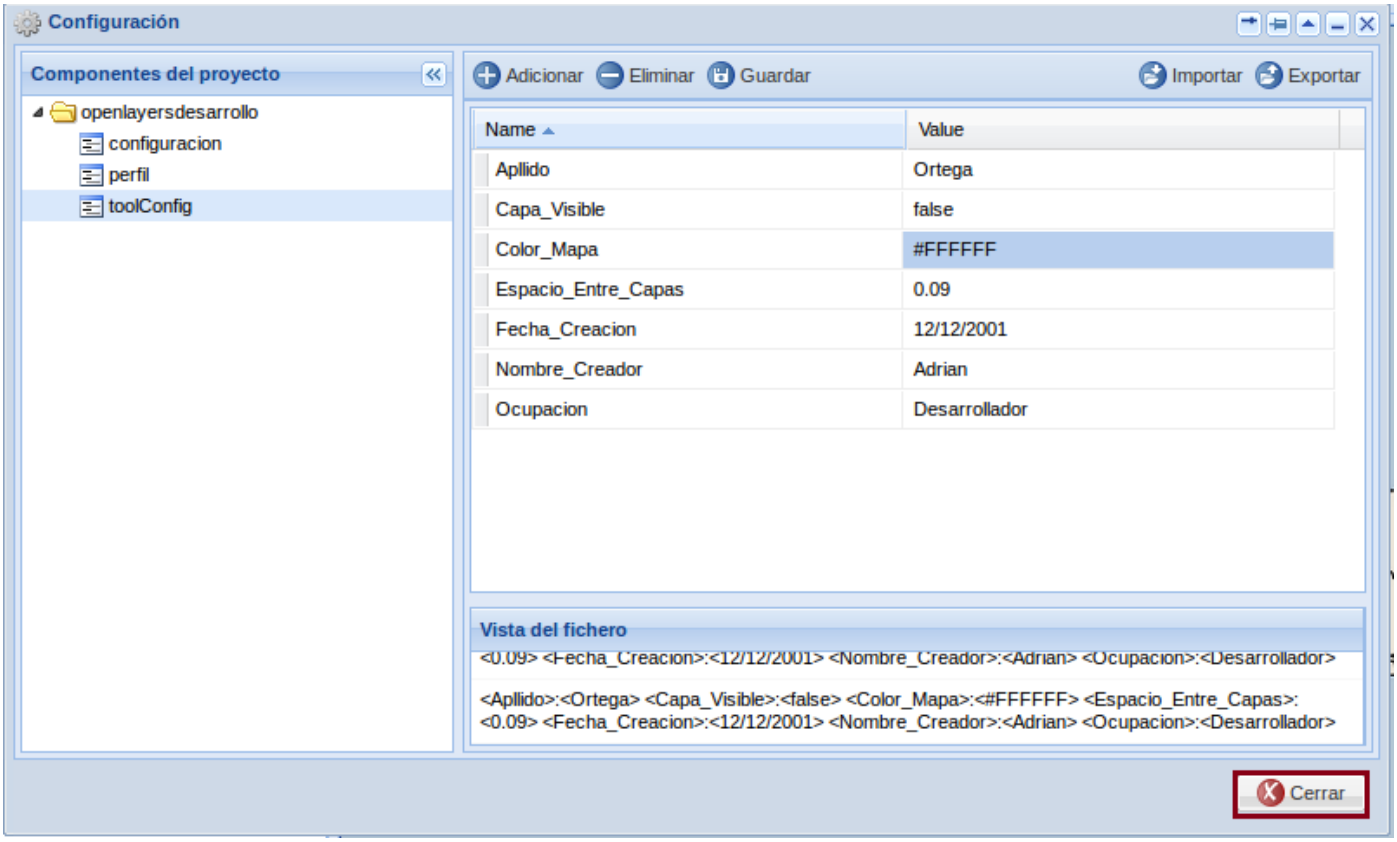


Fig. 18. Salir del componente.

3.5 Pruebas al componente

Las pruebas de software permiten medir la calidad de un software en términos de los defectos detectados por lo que respecta a requisitos y características funcionales y no funcionales (tales como fiabilidad, usabilidad, eficiencia, mantenimiento y portabilidad). Someter los sistemas a pruebas rigurosas puede ayudar a reducir riesgo de complicaciones durante las operaciones y contribuir a la calidad del sistema de

software, siempre que los defectos detectados se corrijan antes de que se ponga el sistema a disposición para su uso operativo. (ISTQB, 2010)

Las pruebas aplicadas al Componente de Personalización Dinámica se encuentran dentro de las pruebas de Caja Negra, se utiliza en la investigación la técnica de Partición Equivalente. Mediante estas pruebas se pretende detectar los defectos al revisar las funcionalidades principales del componente y verificar que los requisitos funcionales generan las salidas esperadas.

2.8.1 Pruebas de Caja Negra

Un tipo de prueba aplicada al Componente de Personalización Dinámica se encuentra dentro de las pruebas de Caja Negra, utilizando la técnica de Partición Equivalente. Mediante estas pruebas se pretende detectar los defectos al revisar las funcionalidades principales del componente y verificar que los requisitos funcionales generan las salidas esperadas. Por el nivel de relevancia para el funcionamiento del componente se seleccionaron seis requisitos de un total de ocho, lo que representa el 75% de los requisitos funcionales definidos.

Tabla 13. Diseño de casos de prueba del requisito “Listar proyectos”.

Nombre del requisito	**Descripción general**	**Escenarios de pruebas (EP)**	**Flujo del escenario**
Listar proyectos.	Este requisito permite listar los proyectos activos en la Plataforma GeneSIG que contienen *plugins* personalizables.	EP 1.1: Se muestra uno o más proyectos.	- Se accede al componente de personalización en la Plataforma. - Se muestra en la región izquierda de la interfaz un árbol con el proyecto o los proyectos activos en la Plataforma GeneSIG.
		EP 1.2: No se muestran proyectos.	- Se accede al componente de personalización en la Plataforma. - En la región izquierda de la interfaz no se muestra ningún proyecto.

Tabla 14. Definición de variables para el requisito “Listar proyectos”.

No	Nombre de campo	Clasificación	Puede ser nulo	Descripción
1	Existe Proyecto	Booleano	No	Representa la existencia del proyecto en la Base

				de Datos de la Plataforma GeneSIG. A partir del valor de esta variable se determina si se muestra o no algún proyecto en el componente de personalización.

Tabla 15. Datos probados para el requisito “Listar proyectos”.

Id	Escenario	V1	Respuesta	Resultado
1	Se muestra uno o más proyectos.	Si	Se listó un proyecto activo en el árbol de proyectos en la interfaz del componente.	Correcto
2	No se muestran proyectos.	No	No aparece ningún proyecto activo en el componente.	Correcto

Tabla 16. Diseño de casos de prueba del requisito “Listar *plugin*”.

Nombre del requisito	**Descripción general**	**Escenarios de pruebas (EP)**	**Flujo del escenario**
Listar *plugin*.	Este requisito permite listar los *plugins* de un proyecto, que son considerados como personalizables.	EP 1.1: Se muestra uno o más *plugins*.	- Se despliega uno de los proyectos mostrados en la interfaz del componente. - Se muestra en la región izquierda de la interfaz un árbol con los *plugins* activos en el proyecto.
		EP 1.2: No se muestran *plugins*.	- Se despliega uno de los proyectos mostrados en la interfaz del componente. - No aparece dentro del proyecto ningún *plugin* activo

Tabla 17. Definición de variables para el requisito “Listar *plugin*”.

No	Nombre de campo	Clasificación	Puede ser nulo	Descripción
1	attr_Conf	Booleano	No	Representa la propiedad que deben cumplir los *plugins* para ser considerados como personalizables. Los *plugins* se muestran en el componente de personalización si el resultado es positivo.

Tabla 18. Datos probados para el requisito “*Listar plugin*”.

Id	Escenario	V1	Respuesta	Resultado
1	Se muestra uno o más *plugins*.	Si	Se listaron dos *plugins* dentro del proyecto activo en el componente.	Correcto
2	No se muestran *plugins*.	No	No aparece ningún *plugin* en el proyecto activo en el componente.	Correcto

Tabla 19. Diseño de casos de prueba del requisito “Mostrar atributos configurables”.

Nombre del requisito	**Descripción general**	**Escenarios de pruebas (EP)**	**Flujo del escenario**
Mostrar atributos configurables.	Este requisito se encarga de mostrar los atributos de un *plugin*, que pueden ser modificados.	EP 1.1: Se muestra al menos un atributo.	- Se selecciona un *plugin* del árbol de proyectos del componente. - Se muestran uno o más atributos del *plugin*, que permiten que su valor sea modificado.

		EP 1.2: No existen atributos personalizables.	- Se selecciona un *plugin* del árbol de proyectos del componente. - No existe ningún atributo en el *plugin* que cumpla con la propiedad de ser personalizable.

Tabla 20. Definición de variables para el requisito “Mostrar atributos configurables”.

No	Nombre de campo	Clasificación	Puede ser nulo	Descripción
1	attr_Conf	Booleano	No	Representa la propiedad que deben cumplir los atributos para ser considerados como personalizables. Los atributos se muestran en el componente de personalización si el resultado es positivo.

Tabla 21. Datos probados para el requisito “Mostrar atributos configurables”.

Id	Escenario	V1	Respuesta	Resultado
1	EP 1.1: Se muestra al menos un atributo.	Si	Se mostraron cuatro atributos en el *plugin*.	Correcto
2	EP 1.2: No existen atributos personalizables.	No	No existe ningún atributo que cumpla con la propiedad *attr_Conf*.	Correcto

Tabla 22. Diseño de casos de prueba del requisito “Adicionar atributo”.

Nombre del requisito	**Descripción general**	**Escenarios de pruebas (EP)**	**Flujo del escenario**
Adicionar atributo.	Este requisito se encarga de adicionar un nuevo atributo en el *plugin* activo.	EP 1.1: El atributo es adicionado correctamente.	- Se selecciona un *plugin* del árbol de proyectos del componente. - Se selecciona la opción Adicionar. - Se introduce el nombre y el tipo de dato del atributo que se adiciona. - Se muestra el nuevo atributo en la lista del *plugin*.
		EP 1.2: No se pudo adicionar el atributo deseado.	- Se selecciona un *plugin* del árbol de proyectos del componente. - Se selecciona la opción Adicionar. - Se introduce el nombre y el tipo de dato del atributo que se adiciona. - El atributo no se adiciona a la lista del *plugin*.

Tabla 23. Definición de variables para el requisito “Adicionar atributo”.

No	Nombre de campo	Clasificación	Puede ser nulo	Descripción
1	Nombre	*string*	No	Identifica al atributo que se desea adicionar.
2	Tipo de Dato	Cualquier tipo de dato nativo o creado en el proyecto.	No	Identifica que tipo de dato admite el atributo como valor.

Tabla 24. Datos probados para el requisito “Adicionar atributo”.

Id	Escenario	V1	V2	Respuesta	Resultado
1	EP 1.1: El atributo es adicionado correctamente.	Color	String	Se adicionó el atributo Color en el *plugin*.	Correcto
		Coordenada	Vector	Se adicionó el atributo Coordenada en el *plugin*.	Correcto

2	EP 1.2: No se pudo adicionar el atributo deseado.	Edad	int	No se adicionó el atributo Edad porque no cumple con la propiedad attr_Conf.	Correcto

Tabla 25. Diseño de casos de prueba del requisito “Eliminar atributo”.

Nombre del requisito	**Descripción general**	**Escenarios de pruebas (EP)**	**Flujo del escenario**
Eliminar atributo.	Este requisito se encarga de eliminar un atributo del *plugin* activo.	EP 1.1: El atributo es eliminado correctamente.	- Se selecciona un *plugin* del árbol de proyectos del componente. - Se selecciona el atributo que se desee eliminar. - Se selecciona la opción Eliminar. - El atributo deje de existir en la lista del *plugin*.
		EP 1.2: No se pudo eliminar el atributo deseado.	- Se selecciona un *plugin* del árbol de proyectos del componente. - Se selecciona el atributo que se desee eliminar. - Se selecciona la opción Eliminar. - El atributo continua en la lista del *plugin*.

Tabla 26. Definición de variables para el requisito “Eliminar atributo”.

No	Nombre de campo	Clasificación	Puede ser nulo	Descripción
1	Archivo eliminado	*booleano*	No	Orienta la acción de borrar el atributo de la configuración del *plugin*.

Tabla 27. Datos probados para el requisito “Eliminar atributo”.

Id	Escenario	V1	Respuesta	Resultado
1	EP 1.1: El atributo es eliminado correctamente.	Si	Se eliminó el atributo seleccionado.	Correcto
2	EP 1.2: No se pudo eliminar el atributo deseado.	No	El atributo continua mostrándose en la configuración del *plugin*.	Correcto

Tabla 28. Diseño de casos de prueba del requisito “Guardar configuración”.

Nombre del requisito	**Descripción general**	**Escenarios de pruebas (EP)**	**Flujo del escenario**
Guardar configuración	Este requisito se encarga de guardar la nueva configuración realizada por el usuario a los *plugins* de un proyecto.	EP 1.1: La configuración se guarda correctamente.	- Se selecciona un *plugin* del árbol de proyectos del componente. - Se modifica el valor del atributo o los atributos que se deseen. - Se realizan todas las configuraciones que el usuario desee sobre los atributos del *plugin* (adicionar, eliminar) - Se selecciona la opción Guardar. - En el área de los logs se muestra que la configuración guardada.
		EP 1.2: No se pudo guardar la configuración.	- Se selecciona un *plugin* del árbol de proyectos del componente. - Se modifica el valor del atributo o los atributos que se deseen. - Se realizan todas las configuraciones que el usuario desee sobre los atributos del *plugin* (adicionar, eliminar) - Se selecciona la opción Guardar. - En el área de los logs se muestra la configuración anterior.

Tabla 29. Definición de variables para el requisito “Guardar configuración”.

No	Nombre de campo	Clasificación	Puede ser nulo	Descripción
1		*booleano*	No	Orienta la acción de borrar el atributo de la configuración del *plugin*.

Tabla 30. Datos probados para el requisito “Guardar configuración”.

Id	Escenario	V1	Respuesta	Resultado
1	EP 1.1: El atributo es eliminado correctamente.	Si	Se eliminó el atributo seleccionado.	Correcto
2	EP 1.2: No se pudo eliminar el atributo deseado.	No	El atributo continua mostrándose en la configuración del *plugin*.	Correcto

2.8.2 Pruebas de Caja Blanca

Con el fin de probar el componente desde el código fuente se decidió aplicar pruebas de Caja Blanca, utilizando específicamente la técnica del Camino Básico. Estas pruebas fueron aplicadas a 4 funcionalidades que responden a 4 requisitos funcionales de los 8 especificados para el desarrollo del componente de personalización, representando el 50% del total. A partir de los resultados observados en la prueba se podrá determinar el nivel de riesgo en la implementación de los requisitos.

La ejecución de las pruebas se realizó de forma manual, teniendo en cuenta la cantidad de requisitos. Para ello se realizaron los grafos de flujo de cada funcionalidad y se calculó la complejidad ciclomática de cada uno de ellos a partir de la formula $CC = A - V + 2$ donde:

- CC representa la Complejidad Ciclomática calculada.
- A es el número de aristas del grafo.
- V es el número de vértices o nodos del grafo.

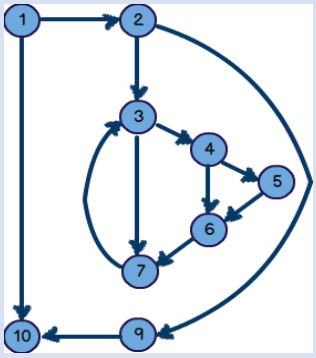
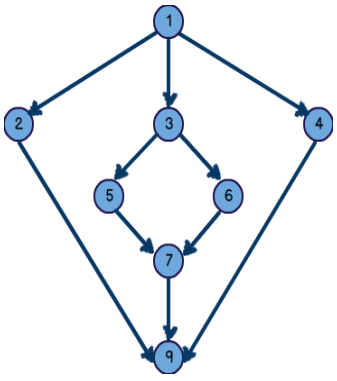
Cuando el valor de CC es calculado se establece un resultado cualitativo teniendo en cuenta el análisis realizado, utilizando como guía lo propuesto por Eduardo Martínez en su trabajo de diploma en 2012 que plantea la siguiente tabla (Martínez Salazar, 2012):

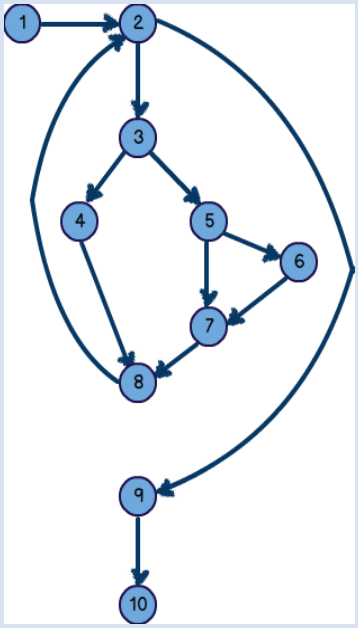
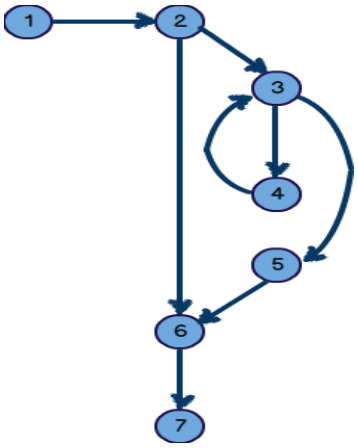
Tabla 31. Análisis de la Complejidad Ciclomática.

Complejidad Ciclomática	Evaluación del Riesgo
1-10	Programa simple, sin mucho riesgo.
11-20	Más complejo, riesgo moderado.
21-50	Complejo, programa de alto riesgo.
50 en adelante	Programa no testeable, muy alto riesgo.

Los grafos de flujo de los métodos analizados y la complejidad ciclomática de cada uno de ellos se evidencian en la siguiente tabla:

Tabla 32. Pruebas de Caja Blanca.

Nombre del método	Grafo de Flujo	Complejidad Ciclomática	Evaluación del Riesgo
GetProjects()		$CC = A - V + 2$ $= 12 - 10 + 2$ $= 4$	Programa simple, sin mucho riesgo.
GetPluginsByProject()		$CC = A - V + 2$ $= 10 - 9 + 2$ $= 3$	Programa simple, sin mucho riesgo.

GetPluginsConfigurable()		$CC = A - V + 2$ $= 12 - 10 + 2$ $= 4$	Programa simple, sin mucho riesgo.
GetDataConfigByPlugins()		$CC = A - V + 2$ $= 8 - 7 + 2$ $= 3$	Programa simple, sin mucho riesgo.

3.6 Resultados obtenidos

Una vez terminado el desarrollo del Componente de Personalización Dinámica se realizaron las pruebas de funcionalidad y las pruebas al código fuente. Las pruebas de Caja Negra se realizaron en 2 iteraciones, encontrándose en la primera de ellas 2 no conformidades en las funcionalidades “Mostrar atributos configurables” y “Guardar Configuración”. En una segunda iteración se comprobó que las no conformidades fueron corregidas y no se encontró ninguna nueva, a través de lo cual se demostró, que, desde el punto de vista del funcionamiento, el componente está listo para ser utilizado en la Plataforma GeneSIG. Una vez concluidas las Pruebas de Caja Negra se verifico el código fuente mediante la técnica de Camino Básico como parte de las Pruebas de Caja Blanca. En este caso los métodos evaluados resultaron con poco riesgo, por lo que se considera que el componente ha desarrollado satisfactoriamente.

3.7 Conclusiones parciales

Durante el presente capítulo se diseñó el Diagrama de Componentes y el de Despliegue como sustento de la implementación de la solución al problema de la investigación. Estos diagramas posibilitaron organizar los elementos que intervienen en el desarrollo y especificar el entorno básico para el funcionamiento del Componente de Personalización Dinámica. Una vez culminado el desarrollo se pudieron describir las principales funcionalidades del componente mediante las interfaces principales, permitiendo evidenciar las opciones disponibles para personalizar la configuración de los *plugins* de la Plataforma GeneSIG. Finalmente, se seleccionaron las pruebas de Caja Negra y de Caja Blanca para probar la solución, concluyéndose que el componente se encuentra listo para ser utilizado.

Conclusiones Generales

- Los fundamentos teóricos planteados facilitaron el entendimiento de los principales conceptos asociados, lo que generó el conocimiento imprescindible para identificar la bibliografía útil y encaminar la investigación. El análisis de los sistemas homólogos MapInfo, QGIS, Kosmo, ArcGIS, "La herramienta para la creación de Sistemas de Información Geográfica" y la comparación entre más de veinticinco herramientas para la edición de mapas en la web mostró que no se encontraron soluciones viables para el problema planteado; sin embargo, le brindó al autor ideas relevantes que se tuvieron en cuenta para el desarrollo del Componente de Personalización Dinámica.

- El análisis teórico realizado permitió seleccionar los elementos que convierten a un atributo en personalizable mediante los cuales se estructuró la propiedad *attr_config*. Se describió la solución propuesta y se modeló el Diagrama de Clases del Dominio a través del cual se identificaron los principales requisitos funcionales y no funcionales del componente. La elaboración del Diagrama de Clases posibilitó estructurar la implementación a partir de la arquitectura utilizada en la Plataforma GeneSIG, basada en los estilos Cliente-Servidor y Basado en Componentes. Se identificaron los patrones del diseño GRASP y Gof, que en conjunto con la definición de la notación *CamelCase* permitieron aumentar la comprensión y reutilización del código fuente.

- El diseño y descripción del Diagrama de Componentes y el de Despliegue, apoyó la implementación del Componente de Gestión Dinámica, logrando que todos los elementos asociados al desarrollo fueran previstos e incluidos. Una vez concluido el desarrollo se diseñaron y aplicaron las pruebas de Caja Negra y Caja Blanca, respectivamente, mediante las cuales se identificaron los defectos que fueron corregidos durante una primera iteración. Una vez culminado el proceso se prueba se demostró que el Componente de Personalización Dinámica se encuentra listo para ser integrado a la Plataforma GeneSIG.

- El desarrollo del componente de gestión dinámica favoreció las actividades de personalización de los plugins que componen la plataforma GeneSIG permitiendo que los usuarios trabajen en un entorno más cercano a sus expectativas. Incrementó la cualidad de la Plataforma GeneSIG de ser configurable y brindó una herramienta de personalización que se integre sin necesidad de aumentar los recursos necesarios para su correcto funcionamiento. Además, aportó un conjunto de funcionalidades como Importar y Exportar Configuración, y Mostrar cadena de configuración en texto plano que generaron valor agregado al propio componente y a la Plataforma GeneSIG.

Recomendaciones

Se recomienda:

- Establecer por cada *plugins* perteneciente a la Plataforma GeneSIG un estándar de configuración donde estén agrupados los atributos configurables de cada uno de ellos.

Referencias Bibliográficas

Adomavicius, Gediminas y Tuzhilin, Alexander. 2003. *Personalization Technologies: A Process-Oriented Perspective.* Minnesota : Univesity of Minnesota, 2003.

Batista Desdin, Yorgüy Antonio y Meriño Coronada, Andy Daniel. 2014. *Módulo de Gestión de Problemas para el Centro de Soporte de la Universidad de las Ciencias Informáticas.* La Habana : Universidad de las Ciencias Informáticas, 2014.

Camacho, Erika, Cardeso, Fabio y Núñez,. 2004. *Gabriel. Arquitecturas de Software, Guía de estudio.* 2004.

Carrillo, German. 2012. GeoTux. *GeoTux.* [En línea] Enero de 2012. http://geotux.tuxfamily.org/index.php/es/geo-blogs/item/291-comparacion-clientes-web-v6/291-comparacion-clientes-web-v6.

Chávez Márquez, Nilberto Caridad y Ramírez Ramírez, Dainy. 2013. *La plataforma GEOQ como solución alternativa para la representación de datos georeferenciados de los recursos geológicos de Cuba.* La Habana. Cuba : XI Congreso Cubano de Informática y Geociencias. Universidad de las Ciencias Informáticas, 2013. ISSN 2307-499X.

Eguíluz Pérez , Javier . 2009. *Introducción a JavaScript.* s.l. : Comunidad de JavaScript, 2009.

ExtJS, Comunidad. 2016. Programadores ExtJS. *Programadores ExtJS.* [En línea] 2016. http://www.extjs.mx/.

Gajda, Włodzimierz. 2013. *Learn professional PHP development with PhpStorm.* Ucrania : Livery Place, 2013. ISBN 978-1-84969-394-3.

García Quintela, Sheyla y González Martínez, Félix. 2013. *Extensión del NetBeans IDE para el diseño de Interfaces Gráficas de Usuario con Ext JS.* La Habana. Cuba : Universidad de las Ciencias Informáticas, 2013.

Guerrero, Carlos A, Suárez, Johanna M y Gutiérrez, Luz E. 2013. No. 3, Colombia : La Serena, 2013, Vol. Vol. 24. ISSN 0718-0764..

IBM, Knowledge Center. 2013. IBM Knowledge Center. *IBM Knowledge Center.* [En línea] IBM, 2013. http://www.ibm.com/support/knowledgecenter/SSR27Q_6.0.2/com.ibm.team.workitem.doc/topics/c_customizing_attributes.html?lang=es.

Infante O, Kevin J. . 2009. *Desarrollo de un Sistema de Información Web Centralizado para la CANTV del Estado Mérida.* Venezuela : Universidad de Los Andes, 2009.

ISTQB. 2010. *Programa de Estudios de nivel básico.* s.l. : Fundation Level Syllabus, 2010.

Jacobson, Ivar, Rumbaugh, James y Booch, Grady. 1999. *El lenguaje unificado de modelado.* Madrid : Caps. 16 y 17, 1999.

JetBrains. 2016. JetBrains. *JetBrains.* [En línea] 2016. [Citado el: 12 de 1 de 2016.] https://www.jetbrains.com/phpstorm/whatsnew/.

Kiccillof, Carlos Reynoso and Nicolás. 2004. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* 2004.

Kubota, Roberto Carlos. 2014. *Componente de edición de símbolos cartográficos.* La Habana. : Universidad de las Ciencias Informática., 2014.

Labrada, Ariel y Páez, Raidel. 2012. *Herramienta para la creación de Sistemas de Información Geográfica.* La Habana : Universidad de las Ciencias Informática, 2012.

Larman, Craig y Hall, Prentice. 2003. *UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado.* s.l. : Pearson Educación, 2003. 8420534382.

Lerman, Craig. 1999. *UML y Patrones.* México : s.n., 1999. 970-17-0261-1.

Líter, Carmen , Sanchis, Francisca y Herrero, Ana. 1992. Historia de la Ciencia y de la Técnica. [aut. libro] Francisco Javier Puerto Sarmiento. *Geografía y Cartografía renacentista.* Madrid : Akal SA, 1992.

López Cruz, Erika Yhoana . 2013. *Diagnóstico de estabilidad de taludes en la localidad de Usme a través de los Sistema de Información Geográficas de Libre Distribución.* Bogotá DC : Universidad Católica de Colombia, 2013.

López, A. Arranz, y otros. 2015. *Web PLOTEG: una herramienta SIG web para el análisis espacial en la ciudad de Zaragoza.* Zaragoza : Universidad de Zaragoza, 2015. ISBN: 978-84-92522-95-8.

MapInfo Corporation. 2003. *Guía del usuario v7.5.* Nueva York : Conversor universal de Safe Software, Inc., 2003.

MapInfo Professional. 2011. *User Guide V. 11.* s.l. : Pitney Bowes Software Inc, 2011.

Márquez, Odalys Rosa Falcón. 2015. Portal UCI. [En línea] 05 de Abril de 2015. http://www.uci.cu/node/18147.

Martínez Salazar, Ing. Eduardo . 2012. *Propuesta de procedimiento para realizar pruebas de Caja Blanca a la aplicaciones que se desarrollan en el lenguaje Python.* Cuba : Facultad Regional de Granma. Universidad de las CIencias Informática., 2012.

Montaner, Daniel y Hernández, Jaime. 2008. *Manual ArcGIS 9.2.* 2008.

Pitney Bowes. 2015. www.pitneybowes.com. *www.pitneybowes.com.* [En línea] Octubre de 2015. http://www.pitneybowes.com/content/dam/pitneybowes/us/en/location-intelligence/geographic-information-systems/mapinfo-pro-15-2/15_DCS_07206-mapinfo-pro-v15-2-datasheet.pdf.

Pressman, Roger S. 2010. *Ingenieria del Sofware.* México : s.n., 2010. 978-607-15-0314-5.

QGIS Project. 2015. *QGIS User Guide.* s.l. : QGIS Project, 2015.

Revista Geoenseñanza. **Instituto de Investigación de Recursos Biológicos. 2006.** No. 1, Venezuela : Red de Revistas Científicas de América Latina, el Caribe, España y Portugal, 2006, Vol. Vol. 11. ISSN 1316-60-77.

Reynoso, Carlos Billy. 2004. *Introducción a la Arquitectura de Software.* Argentina : Universidad de Buenos Aires., 2004.

Robaina, Irilys Ldeon. 2008. *Propuesta del Diseño Arquitectónico del Simulador de Sistemas Biológicos.* La Habana,Cuba : s.n., 2008.

Rumbaugh, James, Jacobson, Ivar y Booch, Grady. 2000. *El proceso unificado de desarrollo de software .* Madrid : Pearson Education, 2000. 84-78-29-036-2.

SAIG S.L. 2009. *Manuales de uso.* s.l. : SAIG, 2009.

Sommerville, Ian. 2005. *Ingeniería de Software.* Madrid : Pearson Educación S.A, 2005. ISBN: 84-7829-074-5.

UCID. 2012. *Proceso de Desarrollo y Gestión de Proyectos de Software.* La Habana : XETID, 2012.

Universidad Carlos III. 2015. Sitio Oficial de la Universidad de Carlos III de Madrid. *Repositorio de clases de la Universidad de Carlos III de Madrid.* [En línea] 2015. [Citado el: 12 de 1 de 2016.] http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf.

Vázquez Mariño, Carlos. 2008. *PROGRAMACIÓN EN LENGUAJE PHP5. NIVEL BÁSICO.* 2008.

Visconti, Marcello y Astudillo, Hernán. 2004. *Fundamentos de Ingeniería de Software.* s.l. : Universidad Técnica Federico Santa María, 2004.

Bibliografía

Adomavicius, Gediminas y Tuzhilin, Alexander. 2003. *Personalization Technologies: A Process-Oriented Perspective.* Minnesota : Univesity of Minnesota, 2003.

Batista Desdin, Yorgüy Antonio y Meriño Coronada, Andy Daniel. 2014. *Módulo de Gestión de Problemas para el Centro de Soporte de la Universidad de las Ciencias Informáticas.* La Habana : Universidad de las Ciencias Informáticas, 2014.

Camacho, Erika, Cardeso, Fabio y Núñez,. 2004. *Gabriel. Arquitecturas de Software, Guía de estudio.* 2004.

Carrillo, German. 2012. GeoTux. *GeoTux.* [En línea] Enero de 2012. http://geotux.tuxfamily.org/index.php/es/geo-blogs/item/291-comparacion-clientes-web-v6/291-comparacion-clientes-web-v6.

Chávez Márquez, Nilberto Caridad y Ramírez Ramírez, Dainy. 2013. *La plataforma GEOQ como solución alternativa para la representación de datos georeferenciados de los recursos geológicos de Cuba.* La Habana. Cuba : XI Congreso Cubano de Informática y Geociencias. Universidad de las Ciencias Informáticas, 2013. ISSN 2307-499X.

Eguíluz Pérez , Javier . 2009. *Introducción a JavaScript.* s.l. : Comunidad de JavaScript, 2009.

ExtJS, Comunidad. 2016. Programadores ExtJS. *Programadores ExtJS.* [En línea] 2016. http://www.extjs.mx/.

Gajda, Włodzimierz. 2013. *Learn professional PHP development with PhpStorm.* Ucrania : Livery Place, 2013. ISBN 978-1-84969-394-3.

García Quintela, Sheyla y González Martínez, Félix. 2013. *Extensión del NetBeans IDE para el diseño de Interfaces Gráficas de Usuario con Ext JS.* La Habana. Cuba : Universidad de las Ciencias Informáticas, 2013.

Guerrero, Carlos A, Suárez, Johanna M y Gutiérrez, Luz E. 2013. No. 3, Colombia : La Serena, 2013, Vol. Vol. 24. ISSN 0718-0764..

IBM, Knowledge Center. 2013. IBM Knowledge Center. *IBM Knowledge Center.* [En línea] IBM, 2013. http://www.ibm.com/support/knowledgecenter/SSR27Q_6.0.2/com.ibm.team.workitem.doc/topics/c_customizing_attributes.html?lang=es.

Infante O, Kevin J. . 2009. *Desarrollo de un Sistema de Información Web Centralizado para la CANTV del Estado Mérida.* Venezuela : Universidad de Los Andes, 2009.

ISTQB. 2010. *Programa de Estudios de nivel básico.* s.l. : Fundation Level Syllabus, 2010.

Jacobson, Ivar, Rumbaugh, James y Booch, Grady. 1999. *El lenguaje unificado de modelado.* Madrid : Caps. 16 y 17, 1999.

JetBrains. 2016. JetBrains. *JetBrains.* [En línea] 2016. [Citado el: 12 de 1 de 2016.] https://www.jetbrains.com/phpstorm/whatsnew/.

Kiccillof, Carlos Reynoso and Nicolás. 2004. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* 2004.

Kubota, Roberto Carlos. 2014. *Componente de edición de símbolos cartográficos.* La Habana. : Universidad de las Ciencias Informática., 2014.

Labrada, Ariel y Páez, Raidel. 2012. *Herramienta para la creación de Sistemas de Información Geográfica.* La Habana : Universidad de las Ciencias Informática, 2012.

Larman, Craig y Hall, Prentice. 2003. *UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado.* s.l. : Pearson Educación, 2003. 8420534382.

Lerman, Craig. 1999. *UML y Patrones.* México : s.n., 1999. 970-17-0261-1.

Líter, Carmen , Sanchis, Francisca y Herrero, Ana. 1992. Historia de la Ciencia y de la Técnica. [aut. libro] Francisco Javier Puerto Sarmiento. *Geografía y Cartografía renacentista.* Madrid : Akal SA, 1992.

López Cruz, Erika Yhoana . 2013. *Diagnóstico de estabilidad de taludes en la localidad de Usme a través de los Sistema de Información Geográficas de Libre Distribución.* Bogotá DC : Universidad Católica de Colombia, 2013.

López, A. Arranz, y otros. 2015. *Web PLOTEG: una herramienta SIG web para el análisis espacial en la ciudad de Zaragoza.* Zaragoza : Universidad de Zaragoza, 2015. ISBN: 978-84-92522-95-8.

MapInfo Corporation. 2003. *Guía del usuario v7.5.* Nueva York : Conversor universal de Safe Software, Inc., 2003.

MapInfo Professional. 2011. *User Guide V. 11.* s.l. : Pitney Bowes Software Inc, 2011.

Márquez, Odalys Rosa Falcón. 2015. Portal UCI. [En línea] 05 de Abril de 2015. http://www.uci.cu/node/18147.

Martínez Salazar, Ing. Eduardo . 2012. *Propuesta de procedimiento para realizar pruebas de Caja Blanca a la aplicaciones que se desarrollan en el lenguaje Python.* Cuba : Facultad Regional de Granma. Universidad de las CIencias Informática., 2012.

Montaner, Daniel y Hernández, Jaime. 2008. *Manual ArcGIS 9.2.* 2008.

Pitney Bowes. 2015. www.pitneybowes.com. *www.pitneybowes.com.* [En línea] Octubre de 2015. http://www.pitneybowes.com/content/dam/pitneybowes/us/en/location-intelligence/geographic-information-systems/mapinfo-pro-15-2/15_DCS_07206-mapinfo-pro-v15-2-datasheet.pdf.

Pressman, Roger S. 2010. *Ingenieria del Sofware.* México : s.n., 2010. 978-607-15-0314-5.

QGIS Project. 2015. *QGIS User Guide.* s.l. : QGIS Project, 2015.

Revista Geoenseñanza. **Instituto de Investigación de Recursos Biológicos. 2006.** No. 1, Venezuela : Red de Revistas Científicas de América Latina, el Caribe, España y Portugal, 2006, Vol. Vol. 11. ISSN 1316-60-77.

Reynoso, Carlos Billy. 2004. *Introducción a la Arquitectura de Software.* Argentina : Universidad de Buenos Aires., 2004.

Robaina, Irilys Ldeon. 2008. *Propuesta del Diseño Arquitectónico del Simulador de Sistemas Biológicos.* La Habana,Cuba : s.n., 2008.

Rumbaugh, James, Jacobson, Ivar y Booch, Grady. 2000. *El proceso unificado de desarrollo de software .* Madrid : Pearson Education, 2000. 84-78-29-036-2.

SAIG S.L. 2009. *Manuales de uso.* s.l. : SAIG, 2009.

Sommerville, Ian. 2005. *Ingeniería de Software.* Madrid : Pearson Educación S.A, 2005. ISBN: 84-7829-074-5.

UCID. 2012. *Proceso de Desarrollo y Gestión de Proyectos de Software.* La Habana : XETID, 2012.

Universidad Carlos III. 2015. Sitio Oficial de la Universidad de Carlos III de Madrid. *Repositorio de clases de la Universidad de Carlos III de Madrid.* [En línea] 2015. [Citado el: 12 de 1 de 2016.] http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf.

Vázquez Mariño, Carlos. 2008. *PROGRAMACIÓN EN LENGUAJE PHP5. NIVEL BÁSICO.* 2008.

Visconti, Marcello y Astudillo, Hernán. 2004. *Fundamentos de Ingeniería de Software.* s.l. : Universidad Técnica Federico Santa María, 2004.

Anexos

Anexo #1: Entrevista

(Realizada a desarrolladores y clientes de la Plataforma GeneSIG).

Objetivo: comprobar la satisfacción de clientes y desarrolladores con la manera en que se muestra la información en la Plataforma GeneSIG.

Compañero o compañera.

Se necesita su valiosa colaboración para la realización de esta investigación. Contamos con su colaboración. **MUCHAS GRACIAS**.

Datos generales.

- Nombre y Apellidos_____
- Rol _____
- Años de experiencia_____

Aspectos a encuestar

1. ¿Considera la interfaz de la Plataforma GeneSIG amigables?
2. ¿Con que facilidad encuentra el componente con el que desea trabajar?
3. ¿Considera que los símbolos existentes para los mapas son suficientes?
4. ¿Considera que la combinación de colores de los mapas posibilita que se muestre la información correctamente?
5. ¿Con que frecuencia accede a la Plataforma GeneSIG?
6. ¿Qué elementos influyen para que utilice la Plataforma GeneSIG?
7. ¿Cuál es su nivel de satisfacción al utilizar la Plataforma GeneSIG?
8. ¿Qué recomienda para mejorar la información geográfica que se muestra en la Plataforma GeneSIG?

Anexo #2: Especificación de Requisitos

RF: "Importar Configuración"

Conceptos tratados	Conceptos	Atributos
	Importar	fichero
Precondiciones	**Precondiciones**	**Pre-requisito**
	- Ser usuario del sistema. - Poseer un fichero con la extensión *.ini*. - Acceder a un *plugin* del proyecto activo	Exportar configuración.
Descripción	El usuario debe: 1. Seleccionar la opción Importar del menú del componente. 2. Acceder a la dirección donde se encuentra almacenado el fichero con la configuración que se desea obtener. Como resultado se debe: 3. Obtener la configuración que el usuario, guardada en el fichero *.ini* importado.	
Complejidad	Media.	
Validaciones	-La configuración importada debe encontrarse en un fichero *.ini*.	
Post-condiciones	-Se debe mostrar el *plugin* con las configuraciones almacenada en el fichero importado.	
Post-requisito	- Guardar Configuración. - Exportar Configuración.	

RF: "Exportar Configuración"

Conceptos tratados	Conceptos	Atributos
	Exportar	No procede
Precondiciones	**Precondiciones**	**Pre-requisito**
	-Ser usuario del sistema. -Acceder a un *plugin* del proyecto activo.	No procede.
Descripción	El usuario debe: 1. Seleccionar la opción de Exportar del menú del componente.	

	2. Seleccionar la ubicación donde se desea almacenar el fichero que contiene la configuración realizada. Como resultado se debe: 3. Guardar en la ubicación seleccionada un fichero con la configuración realizada por el usuario.
Complejidad	Media.
Validaciones	-La configuración debe exportarse en un fichero *.ini*.
Post-condiciones	-Se debe obtener un fichero con las personalizaciones realizadas.
Post-requisito	- Importar Configuración.

RF: "Mostrar cadena de configuración en texto plano".

Conceptos tratados	**Conceptos**	**Atributos**
	-Cadena de configuración. - Texto plano.	attr_config
Precondiciones	**Precondiciones**	**Pre-requisito**
	-Ser usuario del sistema. -Realizar alguna acción en el sistema.	- Listar proyectos. - Listar *plugins*. - Mostrar atributos configurables. - Adicionar atributo. - Modificar atributo. - Eliminar atributo. - Guardar configuración.
Descripción	El usuario debe: 1. Realizar alguna acción sobre el componente. 2. Visualizar la propiedad con los cambios realizado. Como resultado se debe: 3. Mostrar la propiedad *attr_config* con los cambios realizados.	
Complejidad	Media.	
Validaciones	No procede	
Post-condiciones	-Se debe en tiempo real los cambios realizado.	
Post-requisito	No procede	