
Universidad de las Ciencias Informáticas
Facultad 6


# Generador automático de CRUD para la LPS Aplicativos SIG

**Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas**


Autores:

**Dainy Ramirez Ramirez**

**Reinaldo Antonio Amarán Pérez**

Tutores:

**M.Sc. Ing. Vladimir Martell Fernández**

**Ing. Miosotis A. Troche Rodríguez**

**Ing. Nilberto C. Chávez Márquez**



**La Habana, junio de 2015**
**"Año 57 de la Revolución"**

## Dainy

A mi mamá Dania, que me ha dado todo en la vida.

A mi abuela Caridad que me ha acompañado y ha estado conmigo en todo momento.

A mi papá Armando por su apoyo incondicional.

A mi novio Nilberto que ha confiado en mí incondicionalmente.

A toda mi familia que de una forma u otra han estado presentes en mi formación.

A mis amigos que siempre están presentes en las situaciones buenas y malas de la vida.

A ti que hoy estas leyendo esta investigación.

## Reinaldo Antonio

A mi madre, que me ha dado todo en esta vida.

A mi papá.

A una de las personas más importantes de mi vida que me ha convertido en lo que hoy yo soy, y que ha sido más que un padre para mí y al que con mucho cariño le llamo ¨El papa¨.

A mi abuela Maminona, que siempre ha estado ahí para toda la familia, dando lo que tiene con mucho amor y cariño.

A toda mi familia.

## Dainy

Me gustaría agradecerle en primer lugar a esta Revolución y a Fidel que me han dado la posibilidad de estudiar en una universidad tan magnífica como esta.

En un segundo pero primer lugar le agradezco a mi mamá porque ha estado a mi lado en los momentos más difíciles y más importantes de mi vida, ha sabido guiarme por buen camino y ha estado ahí para hacerme entender en cada momento de flaqueza que mi deber es estudiar y ser una buena profesional. Le doy gracias además por comprenderme en cada minuto, por todos los sacrificios que has hecho por mí y por escuchar una y otra vez la exposición sin cansarse.

Le agradezco a mi abuela, la cual me ha criado con mucho amor y cariño desde muy pequeña y ha estado a mi lado en todos los momentos importantes de mi vida.

Le agradezco a mi Papá porque siempre, a pesar de no vernos mucho, ha estado presente y me ha apoyado en cualquier decisión que he tomado, tanto buena o mala, dándome en todo momento su cariño y comprensión.

Le agradezco a mi novio por haberme soportado por casi 4 años con mis malcriadeces y lloraderas por cualquier cosa, le agradezco además por guiarme y demostrarme en todo momento que puedo hacer más y que yo soy la única que puedo limitar mis conocimientos.

Le doy gracias a mi compañero de tesis el cual me ha acompañado en todo momento dándome su apoyo y confianza.

Muchas Gracias a mis tutores Vladimir, Miosotis y Nilberto por ser ejemplos de profesionalidad en todo momento, por darnos su apoyo, confianza y ayuda: tienen toda mi admiración y mis respetos.

Agradezco a todos los profesores y especialistas del proyecto que siempre han estado dispuestos a ayudar y despejar mis dudas.

A los miembros del tribunal por su preocupación, dedicación y exigencia durante el periodo de desarrollo del trabajo de diploma.

A mis amigos Yasel, Ilianet, Andy y Yailín, por estar siempre a mi lado.

Le doy Gracias a todas aquellas personas que de una forma u otra han aportado su granito de arena en mi formación como profesional y que han estado presentes en todas las etapas de mi vida.

## Reinaldo Antonio

Quiero agradecer a mis padres por el apoyo incondicional que siempre me han brindado, a mis tíos que siempre estaban pendientes de mis estudios dándome consejos y energías, a mis primas y primos.

A mi novia, que ha sido mi princesa, mi musa, mi tesoro. Gracias por haber llegado a mi vida y estar siempre a mi lado para hacerme feliz.

A mi compañera de tesis, eres lo máximo, gracias por tu dedicación, confianza y apoyo, te quiero agradecer todo.

A mis tutores por toda la ayuda, dedicación, sabiduría que nos brindaron. Mi respeto y admiración para ustedes.

A todos los amigos que han estado de una forma u otra junto a mí, a los que han compartido todos estos años maravillosos de estudio.

A todos los profesores que formaron parte de nuestra formación académica y profesional, a los profesores del proyecto Aplicativos SIG, gracias a todos.

# DECLARACIÓN DE AUTORÍA

Declaramos por este medio que los estudiantes Dainy Ramirez Ramirez, con carnet de identidad 93010811290 y Reinaldo Antonio Amarán Pérez, con carnet de identidad 88120406508 somos los autores de este trabajo y que autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste, firmamos la presente declaración jurada de autoría, en La Habana, a los 23 días del mes de junio del año 2015.

**Dainy Ramirez Ramirez**
Autor

**Reinaldo Antonio Amarán Pérez**
Autor

M.Sc. Ing. Vladimir Martell Fernández
Tutor

Ing. Miosotis A. Troche Rodríguez
Tutor

Ing. Nilberto C. Chávez Márquez.
Tutor

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# ÍNDICE DE CONTENIDO

# RESUMEN

La Universidad de las Ciencias Informáticas promueve la informatización que se lleva a cabo en el país desde los años 90, mediante la utilización y producción de nuevas tecnologías que ayudan al desarrollo tecnológico posibilitando el intercambio y la colaboración con disímiles entidades nacionales e internacionales. La Línea de Productos de Software Aplicativos SIG, perteneciente a esta Universidad, se especializa en el desarrollo de productos que favorecen la toma de decisiones y contribuyen al control de la información a partir de su representación y análisis espacial en la web. El proceso de desarrollo de software en esta Línea, se realiza a partir de la plataforma GeneSIG, y en él se producen varias deficiencias que limitan las capacidades del equipo de trabajo así como la calidad de las soluciones obtenidas, tales como la duplicidad de código, la introducción de errores humanos, riesgos de incumplimiento del cronograma pactado con el cliente, entre otras. La presente investigación describe una solución a la problemática planteada mediante la implementación de un módulo que permita generar automáticamente las funcionalidades de crear, leer, actualizar y eliminar información socioeconómica, con el objetivo de agilizar el tiempo de desarrollo de los Sistemas de Información Geográfica y reducir las posibles oportunidades de error en su implementación. La investigación refiere el análisis, diseño e implementación del módulo así como el proceso de pruebas al sistema.

**Palabras claves:** CRUD; GeneSIG, Líneas de Productos de Software; Sistema de Información Geográfica.

# Generador automático de CRUD para la LPS Aplicativos SIG

## INTRODUCCIÓN

En las últimas dos décadas las Tecnologías de la Información y las Comunicaciones (TIC) han tenido una gran aceptación a nivel mundial debido a los disímiles avances obtenidos en los distintos sectores de la sociedad. Las TIC ofrecen servicios, productos e incrementan los niveles de organización, provocando así una nueva forma para llevar los negocios que facilita la toma de decisiones.

El avance de las TIC provocó el surgimiento de los Sistemas de Información Geográfica (SIG), definidos como "*aplicaciones que integran tecnología informática, personas e información geográfica con la función de capturar, analizar, almacenar, editar y representar datos georeferenciados" (OLAYA 2010).*Estos sistemas, al ser implementados en una organización, generan un impacto potencial que condiciona las actividades humanas, al cambiar los procedimientos tradicionales de procesar información geográfica, a nuevos procedimientos automatizados que persiguen optimizar el flujo organizacional mediante la eficacia en la toma de decisiones y el análisis espacial (MONTILVA 1994).

Los SIG son utilizados en las organizaciones para manejar los datos georeferenciados en disímiles esferas: la agricultura, la apicultura, el medio ambiente, la socio-económica, entre otras, posibilitando una mayor facilidad a la hora de resolver problemas complejos de planificación y gestión de datos georeferenciados.

A principios de los 90' es creado el primer SIG cubano (GARCÍA and SUÁREZ 2002), lo cual marcó el inicio de un amplio desarrollo de estos sistemas con el objetivo de extender la informatización a y desde todo país. En la actualidad, se continúan desarrollando estas aplicaciones informáticas con destino al mercado nacional del software.

La Línea de Productos de Software (LPS) Aplicativos SIG, tiene como objetivo fundamental desarrollar aplicaciones informáticas de apoyo a la toma de decisiones, que contribuyan al seguimiento y control de la información socioeconómica de cualquier sector de la sociedad a partir de su representación y análisis espacial. Esta LPS forma parte de la estructura organizativa del Centro Geoinformática y Señales Digitales (GEYSED) de la Universidad de las Ciencias Informáticas (UCI).

# Generador automático de CRUD para la LPS Aplicativos SIG

Aplicativos SIG cuenta con la plataforma de desarrollo de Sistemas de Información Geográfica “GeneSIG”. Esta plataforma es un producto encaminado a realizar la representación y análisis geoespacial de información geográfica y su estructura arquitectónica permite personalizar sus funcionalidades a cualquier negocio que lo requiera a través de la reutilización de sus componentes (ZALDÍVAR 2012).

La experiencia práctica de más de 5 años de desarrollo de SIG en la LPS Aplicativos SIG ha mostrado y demostrado que existen varias funcionalidades comunes en todas las aplicaciones obtenidas, las cuales varían de un caso a otro solo por su dominio de aplicación (el negocio), pero conservan idéntica la naturaleza informática, tales son los casos de las acciones de crear, modificar, eliminar y leer información específica de acuerdo a un caso particular, que son agrupadas –casi por criterio unánime- dentro de la Ingeniería del Software, dando lugar a los conocidos Casos de Uso “Gestionar” (del inglés *Create, Read, Update* y *Delete*, CRUD).

Aún hoy, el personal dedicado a la implementación (codificación) de estas acciones dentro de la LPS Aplicativos SIG las realizan completamente desde el inicio para cada caso particular lo cual supone una limitación importante en la respuesta del equipo y ocasiona algunas dificultades tanto internas como externas entre ellas:

1. Duplicidad del código: provoca mayor volumen de información aún cuando en muchos casos las funciones implementadas son idénticas entre sí.
2. Introducción del error humano: producto de la actividad del desarrollador, se introducen errores en las funciones obtenidas que en ocasiones son detectados en etapas tardías del desarrollo del SIG.
3. Riesgo de incumplimiento de los cronogramas: debido a lo anterior, en ocasiones la planificación se encuentra afectada pues persisten errores sin resolver aún cuando las fechas de entrega se encuentran próximas.
4. Empleo de tiempo en los cronogramas para el desarrollo de estas acciones: cada SIG que se construye en la LPS cuenta dentro de su planificación con un tiempo destinado a la implementación de estos casos de uso lo cual convierte en mayor el tiempo total pactado en cada caso.
5. Mayor esfuerzo en el desarrollo de un SIG: el tiempo dedicado a estas actividades involucran a algunos miembros del equipo los cuales emplean tiempo y recursos (esfuerzo) en estas actividades.

6. Mayor esfuerzo en el mantenimiento de los SIG desarrollados: producto de la variación en la codificación de estas acciones de un SIG a otro, el esfuerzo al momento de su mantenimiento y escalabilidad es mayor, en ocasiones, solo quien lo codifica puede ofrecer el mantenimiento en el período acordado.
7. Aumento en la curva de aprendizaje para nuevos miembros del equipo: similar al caso anterior, la variación provoca que los nuevos miembros necesiten un mayor esfuerzo para asimilar el contenido de un SIG.

Partiendo del análisis de la problemática planteada se define como **problema a resolver**: ¿Cómo disminuir el tiempo de desarrollo y las oportunidades de error en la implementación de los SIG en la LPS Aplicativos SIG?

Se define como **objeto de estudio** de la investigación el proceso de implementación de los SIG y como **campo de acción** asociado al objeto la generación automática de CRUD en la LPS Aplicativos SIG.

El presente trabajo de diploma se estructura y desarrolla en función del siguiente **objetivo general:** desarrollar el módulo Generador automático de CRUD para la LPS Aplicativos SIG que posibilite la disminución del tiempo de desarrollo y de las oportunidades de error.

En función de obtener el objetivo general propuesto se desarrollan una serie de **preguntas científicas** que guían la investigación, a continuación se exponen:

- ¿Cómo definir el proceso de desarrollo de los SIG en la LPS Aplicativos SIG?
- ¿Cómo diseñar y desarrollar un módulo para la generación automática de CRUD que posibilite disminuir el tiempo de desarrollo y las oportunidades de error en la LPS Aplicativos SIG?
- ¿Cómo integrar el módulo de generación automática de CRUD a la plataforma GeneSIG?
- ¿Cómo obtener un módulo libre de errores?

En función de las preguntas científicas formuladas se trazaron las siguientes **tareas de investigación:**

1. Caracterizar el proceso de desarrollo de SIG en la LPS Aplicativos SIG.

2. Valorar las soluciones existentes que responden al campo de acción de la investigación en alguna medida, sus limitaciones y fortalezas.
3. Caracterizar las principales herramientas, tecnologías, lenguajes y metodologías a utilizar para la construcción de la propuesta de solución.
4. Realizar el análisis y el diseño de la solución propuesta.
5. Implementar la solución propuesta.
6. Desarrollar el proceso de pruebas al módulo implementado.

A lo largo del proceso investigativo, se utilizan una serie de métodos científicos, teóricos y empíricos, los cuales se describen a continuación:

**Métodos teóricos:**

Los métodos teóricos que se utilizan en esta investigación son:

- **Histórico - lógico:** En la primera fase de la investigación se utiliza para desarrollar un estudio del estado del arte de la problemática analizada, revisando minuciosamente cada uno de los documentos a investigar con el fin de documentar la existencia de soluciones similares de generadores de CRUD que puedan contribuir al desarrollo de la investigación.
- **Analítico – sintético:** Se utiliza para estudiar y analizar la documentación bibliográfica de diferentes autores y extraer la información que permita sustentar desde el punto de vista teórico y práctico los elementos que se relacionan con el proceso de implementación de los SIG y su situación actual en la LPS Aplicativos SIG.

**Métodos empíricos:**

Los métodos empíricos utilizados en esta investigación fueron:

- **Observación:** Permite realizar valoraciones y obtener información a partir de lo observado. Esto se utilizará principalmente para caracterizar el proceso de desarrollo de los SIG en la LPS de referencia de la cual los autores forman parte.

- **Análisis Documental:** Permite valorar, comparar y argumentar la información referida al estado del arte obtenida de fuentes bibliográficas.

El presente documento se estructura como se especifica a continuación:

**Capitulo 1:** En este capítulo se explican los principales conceptos asociados al objeto de estudio CRUD, Sistemas de información geográfica y Línea de Producto de Software. A su vez se describe el proceso de desarrollo de los Sistemas de Información Geográfica en la Línea de Productos de Software Aplicativos SIG. Además se definen, argumentan y valoran las principales herramientas, tecnologías, metodologías y lenguajes que se utilizan en la construcción de la solución. Finalmente se valoran y critican las soluciones actuales que de alguna manera ofrecen respuesta al problema en cuestión.

**Capítulo 2:** En este capítulo se comienza la construcción de la solución según la metodología de desarrollo seleccionada. Se definen los requisitos funcionales y no funcionales del módulo propuesto y se describen los casos de uso el sistema. Finalmente se concluye la construcción de la solución y se especifican los artefactos referidos a las etapas de diseño e implementación, se define el diagrama de despliegue y el de implementación con su respectivo diagrama de componentes.

**Capítulo 3:** En este capítulo se describe el proceso de pruebas de la aplicación, se desarrolla el diseño de Casos de Prueba pertinentes que permitan disminuir los errores de la implementación y se muestran los resultados obtenidos en el proceso de pruebas.

# Generador automático de CRUD para la LPS Aplicativos SIG

# CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA DE LA SOLUCIÓN PROPUESTA

## 1.1 Introducción al capítulo

En el presente capítulo se describen los conceptos asociados al dominio del problema así como el objeto de estudio y el análisis de soluciones informáticas que solucionen, al menos parcialmente, el problema propuesto. Se analizan, además, las distintas herramientas, tecnologías y la metodología a utilizar para dar solución al objetivo propuesto.

## 1.2 Conceptos asociados al dominio del problema

En cada investigación surgen conceptos esenciales asociados al problema a tratar, dichos conceptos constituyen la base fundamental y es de suma importancia su comprensión para el desarrollo del trabajo en cuestión. A continuación se plantean un conjunto de definiciones, las cuales están estrechamente relacionadas con el objeto de estudio y la problemática tratada en la investigación, esto facilitará la comprensión y entendimiento de la presente investigación.

### 1.2.1 Create – Read – Update – Delete (CRUD)

CRUD es el acrónimo de las palabras del inglés *Create*, *Read*, *Update* y *Delete.* Para la Ingeniería de Software no es más que un patrón que se utiliza en los casos donde se desean realizar altas, bajas, cambios y consultas a alguna entidad del sistema.

Los patrones de diseño no son más que comportamientos que deben existir en el sistema, los cuales ayudan a describir qué es lo que el sistema debe hacer, o sea, muestran la interacción entre el sistema y los usuarios. Estos patrones son utilizados generalmente como plantillas que describen cómo deberían ser estructurados y organizados los Casos de Uso. Son patrones que capturan mejores prácticas para modelar Casos de Uso (CUESTA 2013).

El patrón CRUD se divide en dos tipos fundamentales denominados CRUD parcial y CRUD completo:

**CRUD Completo**: el patrón CRUD Completo propone estructurar un Caso de Uso para administrar información que modele las operaciones tales como crear, leer, cambiar y dar de baja. Se usa cuando todas las operaciones contribuyen al mismo valor de negocio y todas son cortas y simples (ver Figura 1).

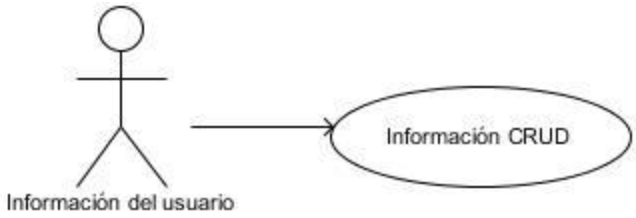



**Figura 1: CRUD Completo.**

**CRUD Parcial:** Constituye una alternativa del CRUD Completo, cuando algunas de las acciones puede ser modelada como Caso de Uso independiente. Es utilizado preferentemente cuando uno de los flujos alternativos del Caso de Uso es más significativo, extenso o mucho más complejo que el resto de los flujos (ver Figura 2).

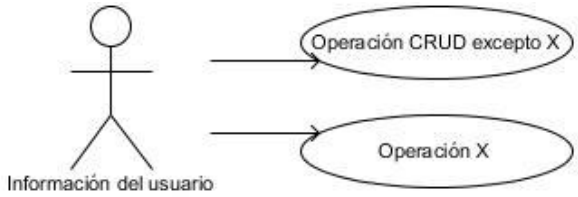



**Figura 2: CRUD Parcial.**

En la programación de computadoras se entiende que CRUD es el acrónimo de Crear, Obtener, Actualizar y Borrar, y se utiliza para referirse a las funciones básicas en las Bases de Datos o la capa de persistencia de un software.

Según la página oficial GlosarioIT, un CRUD se refiere a las operaciones que se pueden realizar en un almacén de datos. En la terminología del lenguaje de consultas SQL se refiere a Insertar (INSERT), Seleccionar (SELECT), Actualizar (UPDATE) y Eliminar (DELETE) operaciones respectivamente (GLOSARIOIT 2014).

### 1.2.2 Línea de Productos de Software

En el concepto dado por Clement and Northrhop se define a las Líneas de Productos de Software (en lo adelante LPS) como “un conjunto de sistemas de software, que comparten un conjunto común de características las cuales satisfacen las necesidades específicas de un dominio o segmento particular de mercado, y que se desarrollan a partir de un sistema común de activos base de una manera preestablecida”(P. CLEMENT AND L. NORTHRHOP 2002).

Además se define una LPS como el conjunto de sistemas de software que comparten un conjunto común y gestionado de aspectos que satisfacen las necesidades específicas de un segmento de mercado o misión y que son desarrollados a partir de un conjunto común de activos fundamentales del software de una manera prescrita (GÓMEZ 2000).

Otra de las definiciones que reflejan claramente el concepto es la de Charles W. Krueger en sus bibliografías académicas referidas a los modelos de desarrollo, donde destaca que las LPS: “se refieren a las técnicas de ingeniería para la creación de una cartera de sistemas de software similares a partir de un conjunto común de activos de software, utilizando un medio común de producción” (KRUEGER 2006).

Los autores de la presente investigación coinciden con Rabiser y Richardson los cuales definen una LPS como el “conjunto de elementos clave para producir sistemas de software que comparten características comunes o similitudes, pero al mismo tiempo mantienen características propias” (RABISER *et al.* 2011).

### 1.2.3 Sistemas de Información Geográfica

Según el *National Center for Geographic Information and* Analysis un SIG se define como un sistema de hardware, software y procedimientos elaborados para facilitar la obtención, gestión, manipulación, análisis, modelado, representación y salida de datos espacialmente referenciados, para resolver problemas complejos de planificación y gestión (ANALYSIS 2010).

También se define a un SIG como un sistema de hardware, software y procedimientos diseñados para soportar la captura, administración, manipulación, análisis, modelamiento y graficación de datos u objetos referenciados espacialmente, para resolver problemas complejos de planeación y administración (BRIONES *et al.* 2009).

Otros conceptos definen un SIG como un sistema que integra hardware, software y datos para capturar, gestionar, analizar y mostrar todas las formas de información geográficamente, permiten comprender, cuestionar, interpretar y visualizar los datos de múltiples formas que revelan las relaciones, los patrones y tendencias en forma de mapas, globos terráqueos, informes y gráficos (ESRI 2011; GIS.COM 2011).

En el marco de esta investigación, se entiende por SIG a un sistema que integra hardware, software y procedimientos diseñados para capturar, gestionar, analizar, modelar y representar datos u objetos referenciados espacialmentepara la generación de información aplicable a proyectos o cuestiones específicas.

## 1.3 Desarrollo de Sistemas de Información Geográfica

Durante los últimos 30 años las empresas, organismos, instituciones académicas, incluso los gobiernos, han implementado programas basados en SIG para aprovechar los beneficios que estos brindan (ESRI 2011). Los SIG son herramientas multipropósito con aplicaciones en campos tan diversos y dispares como la planificación urbana, la gestión catastral, la ordenación del territorio, el medio ambiente, la planificación del transporte, el mantenimiento y la gestión de redes públicas, el análisis de mercados, desastres naturales, información poblacional, zonas arqueológicas, epidémicas, turísticas, entre otras (LARA *et al.* 2003). Sin embargo dichas herramientas pueden ser muy diferentes en su utilidad para el negocio pero todas se caracterizan por poseer componentes y funcionalidades comunes.

### 1.3.1 Componentes fundamentales de un SIG

Según los autores (OLAYA 2010) y (VALERIO 2005) los SIG están integrados por cinco componentes fundamentales: hardware, software, datos, personal y métodos.

**El hardware:** Se refiere a la computadora en la cual operará el SIG. Actualmente, estos sistemas pueden ser ejecutados en una amplia variedad de hardware, desde servidores de computadoras centralizados, hasta computadoras de escritorio utilizadas en configuraciones individuales o conectadas en red.

**El software:** Proporciona las funciones y herramientas necesarias para almacenar, analizar y desplegar información geográfica. Los componentes claves del software son: un sistema de manejo de Base de Datos (SMBD), las herramientas para la entrada y manipulación de información geográfica; las herramientas de soporte para consultas, análisis y visualización geográfica, y una interface gráfica de usuario (GUI, por sus siglas en inglés) para un fácil acceso a las herramientas.

**Los datos:** Los datos en los SIG son en su mayoría espaciales y descriptivos, estos permiten analizar su interacción dentro de los mapas y obtener uno nuevo con características geográficas propias. Dentro de la estructura de un SIG los datos son la parte mediante la cual se representa la realidad, a la vez que permiten enlazarla a situaciones y aplicaciones específicas Las Bases de Datos constituyen una presentación simplificada del mercado real, los datos tienen que ser en formato digital. La mayoría de los SIG emplean un Sistema Gestor de Bases de Datos con el objetivo de crear y mantener una Base de Datos y así ayudar a organizar y manejar dichos datos.

**El personal:** La tecnología de los SIG es de valor limitado sin que el personal maneje el sistema y desarrolle planes que se apliquen a los problemas del mundo real. Frecuentemente subestimados, sin personas, los datos no se actualizan o se manejan equivocadamente; además, el hardware no se utiliza en todo su potencial. Sin embargo, los usuarios de SIG varían y van desde especialistas técnicos, que diseñan y mantienen los sistemas, hasta aquellos que lo utilizan para ayudar a realizar sus tareas diarias.

**Los métodos:** El éxito en la operación de los SIG debe estar acorde con un buen diseño en la planeación y con las reglas de operación de la organización, pues son los modelos prácticos de operaciones únicas para cada organización.

### 1.3.2 Funcionalidades comunes que forman parte de los SIG

Cualquier SIG cuenta con un conjunto de funcionalidades denominadas "básicas" que independientemente de la forma o negocio (que pueden variar de una aplicación a otra) siempre están presentes en un SIG. A continuación se especifican:

**Visualización de datos**

La visualización de datos constituye una de las funcionalidades principales de los SIG. Esta herramienta ayuda a la comprensión y análisis de grandes conjuntos de datos con baja densidad espacial y a su vez permite encontrar similitudes entre las variables del tiempo y el espacio (RODRÍGUEZ *et al.* 2011).

Esta estructura se compone fundamentalmente de un lienzo sobre el que se sitúan distintas capas de información geográfica, que el usuario conforma añadiendo nuevas capas y editando la forma en la que estas se representan. Dichas capas se posicionan en un orden dado dentro del lienzo, lo que permite construir una jerarquía de representación y lograr así el aspecto deseado. Junto a este lienzo existen herramientas de navegación que permiten ampliar o reducir la escala, o bien modificar el encuadre.

**Entrada y salida de datos**

Todos los SIG implementan capacidades para leer datos, y, en algunas ocasiones, se conservan dichos datos. Esta conservación es necesaria en el caso en que el SIG deba generar nuevos datos geográficos, pero no en aquellos sistemas sin capacidades de análisis o edición, donde su empleo no ha de crear nuevos datos.

Pese a ser de tal importancia, la implementación de las capacidades de entrada y salida es muy variable de un SIG a otro. Una razón por la que esto sucede es el gran número de formatos de fichero distintos, así, cada uno es capaz de abrir unos u otros formatos de archivo, mientras que algunos tratan a todos ellos por igual (OLAYA 2010).

## 1.4 El desarrollo de SIG en la LPS Aplicativos SIG

### 1.4.1 ¿Qué es Aplicativos SIG?

Aplicativos SIG es una LPS que tiene como objetivo general desarrollar SIG que permitan la consulta, el análisis y la visualización de la información referente a objetivos socioeconómicos de cualquier sector de la sociedad, teniendo en cuenta su representación y análisis espacial.

Esta LPS está compuesta por 14 profesionales de las Ciencias Informáticas y Geográficas, y funciona desde el año 2010 como parte de la estructura de GEYSED. Para el desarrollo de los aplicativos se utiliza la metodología AUP (del inglés, Agile Unified Process), además, los requisitos, las tecnologías y los componentes reutilizables con los que se cuenta son bien conocidos por el equipo y comunes para la mayoría de los productos de software desarrollados (CASTELL 2012).

### 1.4.2 Estructura de la LPS

Según (ZALDÍVAR 2012), la LPS Aplicativos SIG se estructura en cinco grupos principales y un grupo externo los cuales se especifican a continuación:

- **Equipo de Integración y Componentes:** Encargado del desarrollo de nuevos activos de software y la integración de éstos a los nuevos productos.
- **Equipo de Interfaz:** Encargado del desarrollo de las interfaces de usuario de los nuevos productos siguiendo la arquitectura de información definida.
- **Equipo de Bases de Datos Espaciales:** Encargado de la creación, gestión, administración y mantenimiento de la información espacial en la base de datos de la línea.
- **Equipo de Ingeniería:** Encargado de la modelación de los artefactos ingenieriles y la documentación técnica de toda la línea.
- **Equipo de Gestión de Proyectos:** Encargado de todos los procesos de gestión de proyectos asociados al desarrollo, como la planificación, los riesgos, los recursos humanos y materiales.

- **Equipo de Calidad:** Es un grupo externo de la LPS, su trabajo actúa de manera horizontal para todos los equipos de desarrollo y se encarga del aseguramiento de la calidad de todo el proceso de desarrollo.

### 1.4.3 El proceso de desarrollo de un SIG en la LPS

La plataforma GeneSIG, un producto concebido y elaborado en el Centro Geoinformática y Señales Digitales de la UCI, de conjunto con las Fuerzas Armadas Revolucionarias (FAR) y el Grupo Empresarial GEOCUBA es la plataforma utilizada para el desarrollo de SIG en la LPS Aplicativos SIG, su implementación comenzó en el año 2008 y desde su inicio fue desarrollada utilizando herramientas y tecnologías libres.

GeneSIG cumple técnicamente con las especificaciones OpenGIS que establece el Open Geospatial Consortium (OGC) que garantizan la interoperabilidad global entre los SIG y en consecuencia con la política de migración a software libre y de soberanía tecnológica que es hoy impulsada por el país (CABALLERO 2010).

GeneSIG organiza sus funcionalidades por plugins o componentes ensamblables, permitiendo que se puedan construir aplicativos personalizados con una complejidad menor. Están definidos los módulos o subsistemas que son comunes en cada desarrollo sobre GeneSIG y funcionalidades generales que se disponen y configuran en dependencia del negocio a modelar con la herramienta final. A día de hoy, este importante producto de software cuenta con los siguientes módulos:

- **Módulo de Navegación**: se encarga de gestionar toda la interacción del usuario con la interfaz visual donde se encuentra el mapa y garantiza que éste pueda realizar las operaciones de movimiento, acercamiento (Zoom in) y alejamiento (Zoom out) en sus diferentes variantes.
- **Módulo de Selección**: ofrece la posibilidad de selección de objetos geográficos por el usuario dentro de las capas seleccionables definidas y permite realizar operaciones de consulta o persistencia de selección.
- **Módulo de Consulta Espacial:** ofrece la posibilidad de consultar espacialmente objetos puntuales o los determinados por un área que defina el usuario, rectangular, circular o poligonal irregular.
- **Módulo de Configuración del Mapa:** permite la configuración de la aplicación para el manejo de los datos y el mapa, unidades de medidas, tipo de coordenadas, proyección, entre otras variables de trabajo.

- **Módulo de Impresión**: ofrece la posibilidad de impresión del área que defina el usuario en el formato de papel que corresponda.
- **Módulo de Catálogo**: ofrece la posibilidad de configurar la representación del mapa en cuanto a estilos y simbología desde una interfaz amigable, generando un archivo de configuración que es utilizado por un servidor de mapas para su representación, y haciendo persistente esta configuración en una Base de Datos (ZALDÍVAR 2012).

El esquema de trabajo implantado en el equipo de proyecto de la LPS Aplicativos SIG basa su funcionamiento en cinco actividades principales llevadas a cabo por los mismos miembros del proyecto en cada iteración como se muestra en la Tabla 1.

**Tabla 1: Actividades llevadas a cabo en la LPS Aplicativos SIG para la realización de un SIG.**

| Actividad | Descripción | Roles | Tareas | Entradas | Salidas | Tecnología |
|---|---|---|---|---|---|---|
| **Solicitar Aplicativo** | Se solicita a la LPS el desarrollo de un Aplicativo | Cliente, Jefe de Proyecto | Realizar reunión de inicio | Solicitud de desarrollo de aplicativo | Notificación de aceptación | |
| **Realizar levantamiento de funcionalidades** | Se realiza un levantamiento de las funcionalidades del SIG | Desarrolladores | Realizar procesos asociados al levantamiento de requisitos | | Cronograma | GESPRO |
| | | Jefe del equipo de gestión, Planificador | Confeccionar el cronograma | | | |
| **Ejecutar acciones de desarrollo** | Se desarrolla el Aplicativo SIG | Desarrolladores | Ejecutar tareas de desarrollo según cronograma | Requisitos | Aplicativo SIG | Herramienta de desarrollo |
| **Liberar producto** | Se libera el producto por calidad | Equipo de Ingeniería | Revisar artefactos | Doc. Técnica Aplicación | Acta de Liberación | Herramienta de pruebas |
| **Desplegar producto** | Se despliega el producto | Equipo de Integración y componentes | Desplegar el producto | Aplicativo SIG, RNF | Acta de Aceptación | Instaladores |

## 1.5 Herramientas y tecnologías a utilizar

Para obtener un sistema con calidad es necesario utilizar herramientas y tecnologías que agilicen el proceso de desarrollo de software y que estén enfocadas en la obtención de un producto final. El "Generador automático de CRUD" que se propone, una vez obtenido, debe formar parte de las funcionalidades de la plataforma GeneSIG, concibiéndose como un módulo de esta. A propósito de lo anterior, las herramientas, y tecnologías que se emplean en esta investigación coinciden con las herramientas y tecnologías que se utilizan en el desarrollo y mantenimiento de GeneSIG.

### 1.5.1 Metodología de desarrollo de software

Una metodología de desarrollo tiene como objetivo aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo; definiendo Quién debe hacer Qué, Cuándo y Cómo debe hacerlo. Estas metodologías guían a los desarrolladores de un software en el diseño e implementación del mismo.

Hoy día existen disímiles propuestas metodológicas que incurren en las distintas dimensiones del proceso de desarrollo de software. Uno de los principales problemas a los que se enfrentan los equipos de desarrollo en la actualidad, es seleccionar la metodología adecuada que permita obtener resultados óptimos en el proceso de desarrollo de software.

La LPS de referencia utiliza en su proceso de desarrollo la metodología ágil *Agile Unified Process* (AUP), por tanto, los autores de la investigación actual desarrollan su propuesta a partir de ella.

**Proceso Unificado Ágil**

El proceso unificado ágil es un enfoque de modelado híbrido creado por Scott Ambler al combinar el Proceso Unificado de Rational (del inglés *Rational Unified Process*, RUP) con los métodos ágiles (EDEKI 2013). Es una versión simplificada de RUP que describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP (BECERRA and CRUZ 2015).

AUP, de acuerdo con la Figura 3, está constituida por cuatro fases de trabajo comienzo, elaboración, construcción y transición; abarca siete flujos de trabajos, cuatro ingenieriles y tres de apoyo: modelado, implementación, prueba, despliegue, gestión de configuración, gestión de proyectos y entorno.

Las cuatro fases o etapas de la metodología AUP son:

- Comienzo: Tiene como objetivo obtener una comprensión común entre el cliente y el equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.
- Elaboración: Tiene como objetivo que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.
- Construcción: Durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.
- Transición: el sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación y aceptación y finalmente se despliega en los entornos de producción.

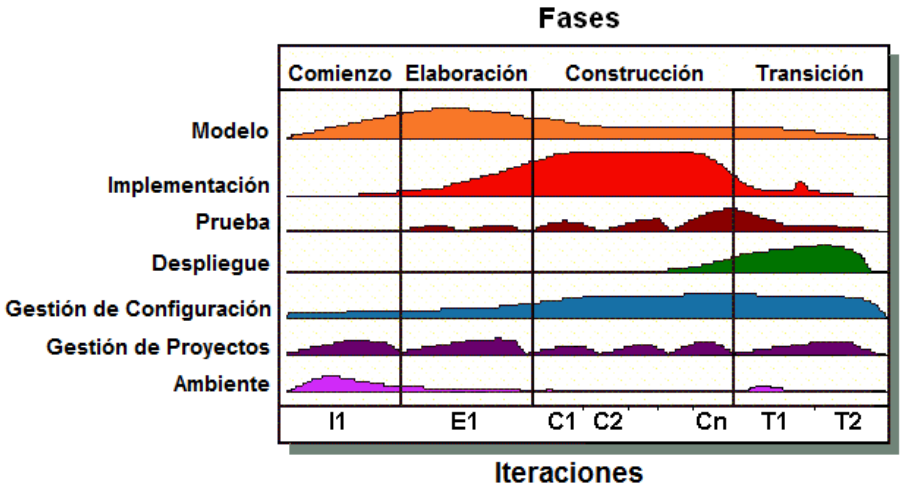



**Figura 3: Fases y flujos de trabajo en AUP.**

Para diseñar los esquemas del software, trabajando con esta metodología, se utiliza el Lenguaje Unificado de Modelado.

### 1.5.2 Lenguaje Unificado del Modelado

El Lenguaje Unificado del Modelado (del inglés *Unified Modeling Language*, UML) se considera “un lenguaje estándar para escribir planos de software”. UML puede utilizarse para visualizar, especificar, construir y documentar los artefactos de un sistema intensivo de software (PRESSMAN 2008).

Según (JACOBSON *et al.* 2003), los rasgos principales que han contribuido a hacer de UML un estándar de la industria actual, son:

- Permite modelar sistemas utilizando técnicas orientadas a objetos.
- Es un lenguaje muy expresivo que cubre las vistas necesarias para desarrollar y luego desplegar los productos.
- Ampliamente utilizado por la industria del software.
- Reemplaza a decenas de notaciones empleadas por otros lenguajes.
- Comportamiento del sistema: casos de usos, diagramas de secuencia, de colaboración, que sirven para evaluar el estado de las máquinas.

UML permite describir los procesos que se realizan en el desarrollo del presente trabajo, así como hacer referencia a detalles de los artefactos que se obtienen. El uso de este lenguaje facilita el entendimiento y asimilación por parte del equipo de trabajo, minimizando el tiempo de implementación del producto.

### 1.5.3 Herramienta CASE

Las herramientas CASE (del inglés *Computer Aided Software Engineering*) se definen como un conjunto de programas y ayudas que asisten a los analistas, ingenieros de software y desarrolladores durante el ciclo de vida del desarrollo de una aplicación informática. La realización de un nuevo software requiere que las tareas sean organizadas y completadas de forma correcta y eficiente. Las herramientas CASE fueron desarrolladas para automatizar esos procesos y facilitar las tareas de coordinación de los eventos que necesitan ser mejorados en el ciclo de desarrollo del software (ALFARO 1999).

**Visual Paradigm**

Visual Paradigm es una herramienta CASE que utiliza UML como lenguaje de modelado. Está diseñada para una amplia gama de usuarios interesados en construir sistemas fiables con el uso del paradigma orientado a objetos, incluyendo actividades como la ingeniería de software, el análisis de sistemas y el análisis de negocios (JACOBSON *et al.* 2003). Presenta características que son factibles a utilizar durante el proceso de desarrollo de software. Es una herramienta desarrollada con tecnologías libres y multiplataforma, lo que facilita su uso en los diferentes sistemas operativos existentes.

Entre las características de esta herramienta se encuentran:

- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Licencia: gratuita y comercial.

- Generación de bases de datos, transformación de diagramas de Entidad - Relación en tablas de Base de Datos.
- Disponibilidad en múltiples plataformas.
- Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.

### 1.5.4 Lenguajes de programación

Un lenguaje de programación es aquel elemento dentro de la informática que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que pone a disposición del programador para que este pueda comunicarse con los dispositivos hardware y software existentes (RODRIGUEZ 2015).

**Lenguaje del lado del cliente**

Para las aplicaciones basadas en tecnologías web el lenguaje del lado del cliente es considerado aquel lenguaje que es interpretado en el navegador del usuario, que es donde este puede interactuar con la aplicación. Resulta de vital importancia prestar atención a la utilización de dichos lenguajes para lograr compatibilidad entre los diferentes estándares que los navegadores utilizan para interpretarlos ya que la selección del navegador a utilizar es independiente de cada usuario.

**Ext.js**

Ext.js es una biblioteca basada en JavaScript para construir aplicaciones web. Esta biblioteca permite que el servidor pueda atender a más clientes ya que, realiza un balance en la relación cliente – servidor, disminuyendo la carga de procesamiento. Además es compatible con la mayoría de los navegadores, por lo cual permite crear páginas e interfaces web dinámicas. Proporciona un aumento en el rendimiento, la coherencia, flexibilidad y mejoras de la interfaz de usuario. Utiliza tecnologías como AJAX, DHTML (del inglés *Dynamic* HTML) y DOM (del inglés *Document Object Model*).

Originalmente construida como una extensión de la biblioteca YUI, en la actualidad puede usarse como extensión para las bibliotecas JQuery y Prototipo. Incluye soporte para peticiones directas, CRUD (Crear, Obtener, Actualizar, Borrar) y REST (Transferencia de Estado Representacional), nuevos ejemplos y componentes (para gráficas), más de 1,000 mejoras y correcciones, API documentada y CSS (del inglés *Cascading Style Sheets*) reautorizado y compatibilidad con versiones anteriores (LEÓN *et al.* 2011):

Otras Características que la distinguen son:

- Código reutilizable.
- Independiente o adaptable a frameworks diferentes.
- Soporte Comercial y Open Source, con una extensa comunidad de usuarios.

JavaScript es un lenguaje de programación interpretado, creado para que se ejecute en el navegador de cada usuario. Es utilizado en la creación de páginas web dinámicas y en la manipulación y personalización de aplicaciones. Que sea un lenguaje interpretado significa que no es necesario compilarlo para poder ejecutarlo. Este lenguaje de programación interpretado, se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámica (EGUILUZ 2011).

Se utiliza Ext JS para el diseño de las interfaces del lado del cliente, ya que con pocas líneas de código se puede realizar una interfaz amigable para el usuario. Esta biblioteca Java Script ofrece varias facilidades para la conexión con el servidor lo cual permite un intercambio de información entre este y el usuario. A su vez se tuvo en cuenta que la misma es compatible con la mayoría de los navegadores y es una de las bibliotecas más utilizadas en la creación de páginas web dinámicas.

**Lenguaje del lado del servidor**

Los lenguajes del lado del servidor son aquellos que son ejecutados en el servidor web. Son los encargados de recibir una petición realizada por el usuario, procesarla y enviar una respuesta.

**Preprocesador de Hipertextos PHP**

PHP es un lenguaje de secuencia de comandos diseñado específicamente para el desarrollo web, puede ser incrustado dentro de una página Web sin necesidad de utilizar archivos externos aunque se pueden definir archivos específicos con código PHP. Generalmente se ejecuta en un servidor web que analiza las sentencias sin necesidad de compilarlas, tomando el código en PHP como su entrada y creando páginas web como salida(WELLING and THOMSON 2005). Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas. Permite el desarrollo de aplicaciones dinámicas que se conectan a servidores de Bases de Datos.

Según (TEAM 2011) PHP al ser un lenguaje libre dispone de una gran cantidad de características que lo convierten en una opción apropiada para la creación de páginas web dinámicas:

- Soporte para una gran cantidad de Bases de Datos: MysSQL, PostgreSQL, Oracle, MS SQL Server, SybasemSQL, Informix, entre otras.
- Integración con varias bibliotecas externas, permite generar documentos en PDF y hasta analizar código XML.
- Ofrece una solución simple y universal para las paginaciones Web dinámicas de fácil programación.

### 1.5.5 Sistema Gestor de Bases de Datos

Un Sistema Gestor de Bases de Datos (en lo adelante SGBD) es aquel software que se encarga de brindar un sistema de almacenamiento de datos de manera ordenada y transparente al usuario, proporcionándole a sus vez los mecanismos necesarios para que interactúen con las Bases de Datos, o sea, son software que sirven de interfaz entre las Bases de Datos y las aplicaciones que la utilizan, contando con un lenguaje de definición, manipulación de datos y un lenguaje de consulta.

A través de los SGBD se crean, almacenan y acceden a los datos de una Base de Datos. El SGBD utilizado en este trabajo es PostgreSQL.

**PostgreSQL**

PostgreSQL es un SGBD objeto-relacional, distribuido bajo licencia BSD, con su código fuente disponible libremente. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando (MARTÍNEZ 2013).

Entre sus principales características se encuentran:

- Creación de sistemas con gran robustez y alto nivel de escalabilidad.
- Soporte completo para claves foráneas, vistas, triggers y procedimientos.
- Permite herencia entre tablas.
- Maneja diversos tipos de datos.
- Soporta objetos y volúmenes de datos de gran tamaño.

PostgreSQL tiene la capacidad de añadir soporte a objetos geográficos para lo cual se utiliza el módulo PostGIS.

**PostGIS**

"Con la finalidad de que la Base de Datos PostgreSQL soporte objetos geográficos se ha desarrollado el módulo PostGIS, convirtiéndola en una base de datos espacial que se puede utilizar en Sistemas de Información Geográfica" (MORETA 2009).

PostGIS es una extensión de PostgreSQL. Permite el uso de objetos SIG e incluye soporte y funciones básicas para el análisis de ellos. Creado por *Refractions Research Inc* como un proyecto de investigación de tecnologías de Bases de Datos espaciales está publicado bajo licencia GNU y es posible con él usar todos los objetos que aparecen en la especificación OpenGIS como puntos, líneas, polígonos, multi-líneas, multi-puntos, y colecciones geométricas (MARTÍN 2009).

Para el trabajo con esta librería las funciones más importantes son los Constructores, los Editores y las Salidas de geometrías, además de los Operadores, las Relaciones espaciales y medidas, las Funciones geométricas de procesamiento y las de manejo de geometrías.

PostgreSQL es un SGBD fácil de administrar y fácil de aprender, permite el trabajo con grandes volúmenes de información como lo demandan los SIG y posee gran escalabilidad siendo capaz de soportar gran cantidad de peticiones simultáneas de forma correcta. En aras de que el SGBD PostgreSQL permita el almacenamiento de objetos GIS en una base de datos se utilizó su extensión PostGIS la cual es habilitada como repositorio de datos espaciales para cualquier SIG.

### 1.5.6 Servidores

El servidor no es más que el equipo informático central que forma parte en una red y provee servicios a otros equipos que actúan como clientes. Existen varios tipos de servidores según los servicios que estos ofrecen, se hará énfasis en los servidores utilizados en la LPS Aplicativos SIG.

**Servidor de mapas: MapServer**

MapServer es una herramienta para la construcción de aplicaciones web interactivas que permita la visualización y consulta de información geográfica en forma de mapas. Es una herramienta de código

abierto. Este puede ser utilizado como una aplicación de Interfaz de Entrada Común (CGI) o a través del acceso a la Interfaz de Programación de Aplicaciones (API) de MapServer que proveen las bibliotecas Mapscript (PACHECO 2012).

MapServer utiliza la información pasada en una petición de usuario y un archivo Mapfile para crear una imagen del mapa requerido, dicha petición puede incluir imágenes para la leyenda, barras de escala, mapas de referencia y valores como variables. Los datos de los archivos se organizan por capas las cuales fueron divididas por diferentes clases, permitiendo que cada una de ellas se observe con diferentes estilos.

Según (GONZALEZ and VALDIVIESO 2007) entre las principales características de MapServer se encuentran:

- Una salida cartográfica avanzada que posibilita ejecutar y dibujar elementos según la escala, así como automatizar los componentes del mapa como son la leyenda, la barra de escala y el mapa de referencia. Brinda soporte a los lenguajes de script y plataformas más reconocidas, también a formatos vectoriales como PostGIS y formatos ráster como JPG, PNG, GIF, entre otros.
- Es compatible con múltiples sistemas operativos.

**Servidor de aplicaciones: Apache**

Apache constituye una tecnología gratuita de código abierto. Es un servidor web flexible, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos. Permite la creación de sitios web dinámicos mediante el uso de lenguajes de scripting como PHP, JavaScript, Python, entre otros. Se ejecuta en varios sistemas operativos. Posee una arquitectura modular que admite ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona. Tiene una alta configurabilidad en la creación y gestión de logs; y soporta personalizar la respuesta ante los posibles errores que se puedan generar en el servidor (FOUNDATION 2010).

## 1.6 Soluciones existentes

De las principales soluciones existentes a nivel mundial y nacional, los autores del presente trabajo de diploma analizaron aquellas que poseían funcionalidades y características que pudiesen ser consideradas en la elaboración de la solución propuesta.

### 1.6.1 CRUD-PG

CRUD-PG es una herramienta que permite estandarizar y obtener los procedimientos almacenados y/o funciones que realicen operaciones CRUD sobre Bases de Datos en PostgreSQL, y así proveer a las aplicaciones que usen dichas Bases de Datos de operaciones CRUD centralizadas en el mismo clúster y lograr mejores tiempos de respuesta entre las aplicaciones y PostgreSQL.

La herramienta CRUD-PG está centrada, por tanto, en PostgreSQL, específicamente en las operaciones básicas que se realizan (escritura, lectura, actualizar y eliminar), generando para ello funciones estándares capaces de realizar dichas operaciones, escritas en lenguaje **pl/pgsql**, lo cual permite implementar parte de la lógica del negocio en la Base de Datos.

Es decir, cada Base de Datos donde se aplique CRUD-PG generará funciones capaces de operar los datos, permitiendo a los programadores hacer uso de las mismas sin tener que preocuparse por implementarlas en sus aplicaciones, además estas funciones son independientes del lenguaje en que están programadas dichas aplicaciones (LEÓN *et al.* 2011).

### 1.6.2 ZRAD

Zrad tiene como principal objetivo automatizar la creación y actualización de procesos base (crear, editar, eliminar) de cada entidad definida en el modelo de datos, permitiendo que las tareas habituales se realicen automáticamente generando archivos con código destinado a resolver un determinado tipo de problemas de aparición recurrente, permitiendo disminuir el 70% del tiempo de desarrollo.

Este además obtiene los metadatos de las tablas definidas en la Base de Datos, basándose en un conjunto de reglas; crea proyectos modulares bajo el estándar Zend Framework. Su base de conocimientos sigue los estándares de Zend Technologies. Puede ser utilizado en distintos sistemas operativos como son Linux, Mac y Windows. Zrad trabaja con PHP 5 y con Bases de Datos MySQL (MINAYA 2013).

### 1.6.3 Generador CRUD

Este generador dinámico de CRUD trabaja con la librería de Java Script Extjs 4.0, el mismo está programado en el lenguaje PHP5.3 y utiliza Bases de Datos MySQL 5.5. Fue creado por Amarildo Días en el año 2013, funciona solo en los navegadores Firefox y Chrome. Dicha aplicación tiene como objetivo

principal generar un CRUD funcional con Bases de Datos MySQL donde las aplicaciones generadas no dependen del generador CRUD para funcionar.

No existe una documentación acertada de dicha herramienta por lo cual, el análisis se realiza superficialmente a partir de la interacción directa con la misma. Una imagen del sistema puede apreciarse en la Figura 4.

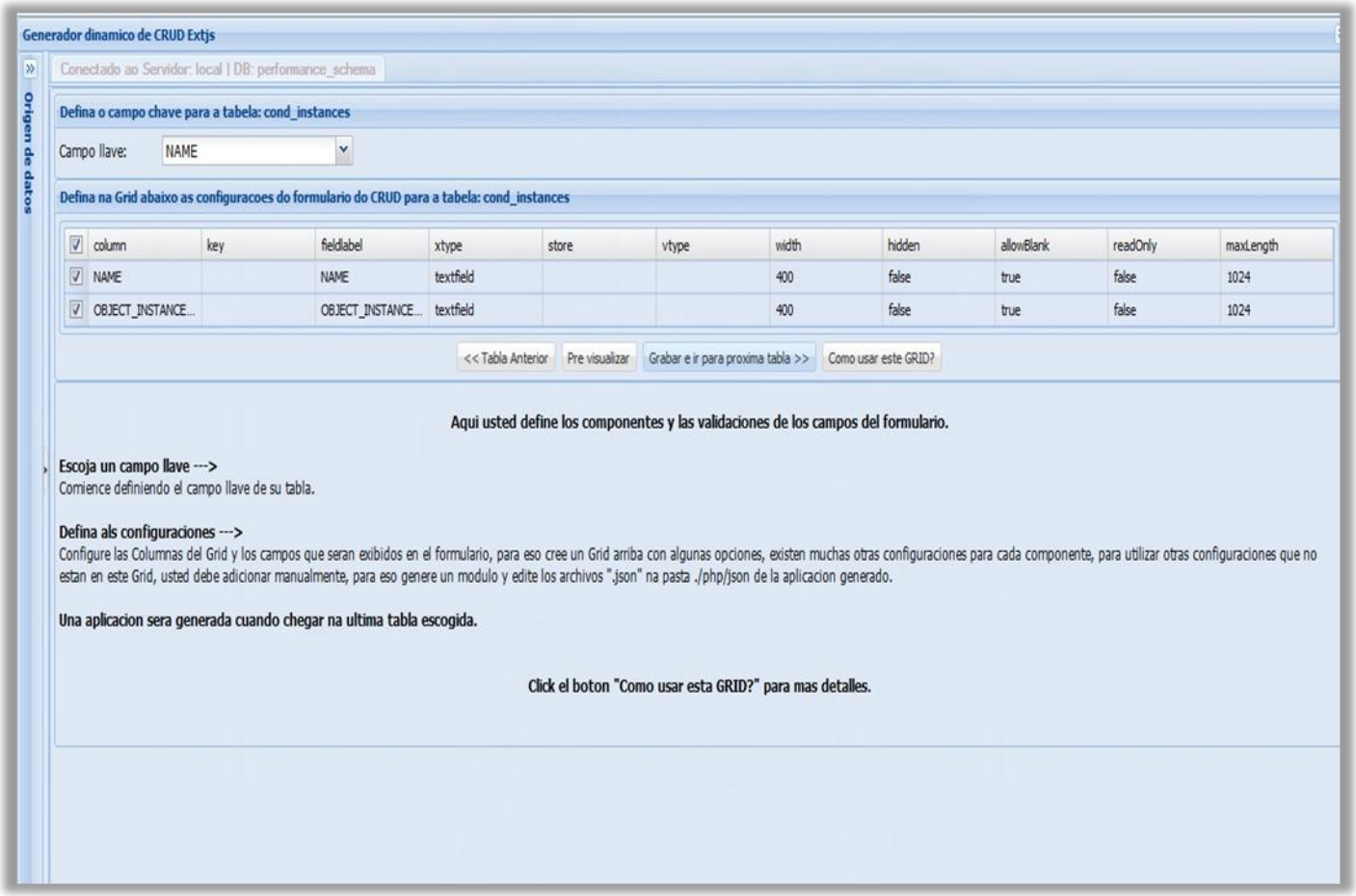



**Figura 4: Generador dinámico de CRUD Extjs.**

### 1.6.4 Generador de CRUD de Symfony

El framework de desarrollo Symfony también integra un módulo de generación de CRUD el cual permite a los desarrolladores trabajar con todos los objetos a través del navegador web, facilitando así el manejo de las operaciones básicas realizadas sobre la información del modelo de datos.

Según el acápite 11.3 del Manual “Formularios de symfony 1.4” en este módulo, el flujo de trabajo del formulario está definido por los métodos “create, edit y update”. Permite realizar todas las operaciones del CRUD en un mismo método y toma de la Base de Datos los campos de la tabla sobre la cual se realizarán las acciones CRUD según necesite el desarrollador. Finalmente, es posible comprobar que las reglas de validación realizadas en la Base de Datos también se aplican directamente al formulario generado por la aplicación (POTENCIER 2015).

Este generador CRUD utiliza como Objeto de Mapeo Relacional (ORM) la herramienta Doctrine la cual ofrece al programador la posibilidad de generar clases a partir de una base de datos existente y después el programador puede especificar relaciones y añadir funcionalidad extra a las clases autogeneradas (DOCTRINE 2015).

### 1.6.5 Valoración general

Después de analizar las soluciones existentes, los autores de la presente investigación consideran que aún cuando varias aplicaciones poseen características similares a las que se pretenden obtener, ninguna se adecua a las características que debe cumplir el generador automático de CRUD para la LPS dado los siguientes elementos:

- En el caso de CRUD – PG el generador de CRUD solo funciona para realizar las acciones sobre la tabla de la Base de Datos, no soporta Bases de Datos espaciales y no permite la generación de código, solo realiza la acción directamente en la tabla.
- La aplicación ZRAD no puede ser utilizada ya que esta no utiliza las mismas herramientas, la misma arquitectura ni el mismo tipo de datos que utiliza GeneSIG aún cuando utiliza PHP5 y genera el código para las acciones de crear, actualizar, leer y eliminar.
- El generador de CRUD es la que más se asemeja a la solución propuesta en esta investigación, pero, al igual que la anterior, no utiliza las mismas herramientas, no posee el mismo tipo de datos ni la misma arquitectura que posee la plataforma GeneSIG.
- El módulo de generador de CRUD de Symfony constituye la guía para obtener un generador de CRUD para GeneSIG, sin embargo, su utilización implicaría incluir el framework dentro de la arquitectura de GeneSIG y, como consecuencia, adecuar todo su funcionamiento (el de GeneSIG) a las especificaciones de Symfony.

## 1.7 Conclusiones del capítulo

Los elementos significativos de las herramientas y tecnologías a utilizar, satisfacen las necesidades existentes para el desarrollo del generador automático de CRUD. La utilización de las herramientas propuestas facilita y fortalece el trabajo, a su vez estas aportan rapidez y eficacia a lo largo del ciclo de vida del desarrollo del módulo y se garantiza una mayor usabilidad del mismo.

Los CRUD disponibles en internet, o desarrollados dentro del territorio nacional, en alguna medida tributan al problema a resolver de esta investigación pero estos no satisfacen los requerimientos solicitados por la LPS Aplicativos SIG, fundamentalmente basados en la diferencia considerable de la estructura de datos que posee la plataforma GeneSIG con respecto a las demás soluciones analizadas.

# CAPÍTULO II: ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE LA SOLUCIÓN

## 2.1 Introducción al capítulo

En este capítulo se muestra una descripción de la solución propuesta en términos de procesos del negocio, así como la especificación de los requisitos funcionales y no funcionales que debe presentar el sistema a construir. Además, se definen los actores del sistema, mostrando el diagrama de casos de uso del sistema así como su respectiva descripción.

## 2.2 Modelo de negocio

El modelado del negocio es una técnica para comprender los procesos del negocio de la organización. Está soportada por dos tipos de modelo UML: modelos de casos de uso y modelos de objetos. Ambos se definen en la extensión UML relativa al negocio. Un modelo de casos de uso del negocio describe los procesos del negocio de una empresa en términos de casos de uso del negocio y actores del negocio y se corresponden con los procesos del negocio y los clientes respectivamente (JACOBSON *et al.* 2003).

Para realizar este tipo de modelo (de negocio) se debe tener en cuenta que los procesos a modelar estén claramente definidos y que en el transcurso de la implementación no ocurrirán cambios, solo así se procede a diseñarlos. En otro caso se realiza un modelo de dominio.

## 2.3 Modelo de dominio

Un modelo del dominio es un modelo conceptual de un sistema, que identifica las relaciones entre todas las entidades importantes dentro del sistema e identifica generalmente sus métodos y cualidades. Puede ser tomado como el punto de partida para el diseño del sistema. El modelo del dominio se crea para documentar los conceptos dominantes del sistema, representa los conceptos u objetos del mundo real, significativos para un problema o área de interés. Ayuda además a los usuarios, clientes, desarrolladores y otros interesados a utilizar un vocabulario común (JACOBSON *et al.* 2003).

### 2.3.1 Diagrama de clases del dominio

**Descripción**

El especialista es el desarrollador que trabaja sobre la plataforma GeneSIG la cual contiene varios módulos como son el de navegación, selección, consulta espacial, configuración del mapa, impresión y el

de catalogo, estos a su vez contienen varios plugins que son los encargados de interactuar con las entidades o tablas de la Base de Datos. Un plugin representa una funcionalidad, entre las que se destacan: identificar, localizar, analizar y representar información sobre un mapa. Un SIG tiene varias funcionalidades, está compuesto por varios plugins y representa una personalización de la plataforma GeneSIG.

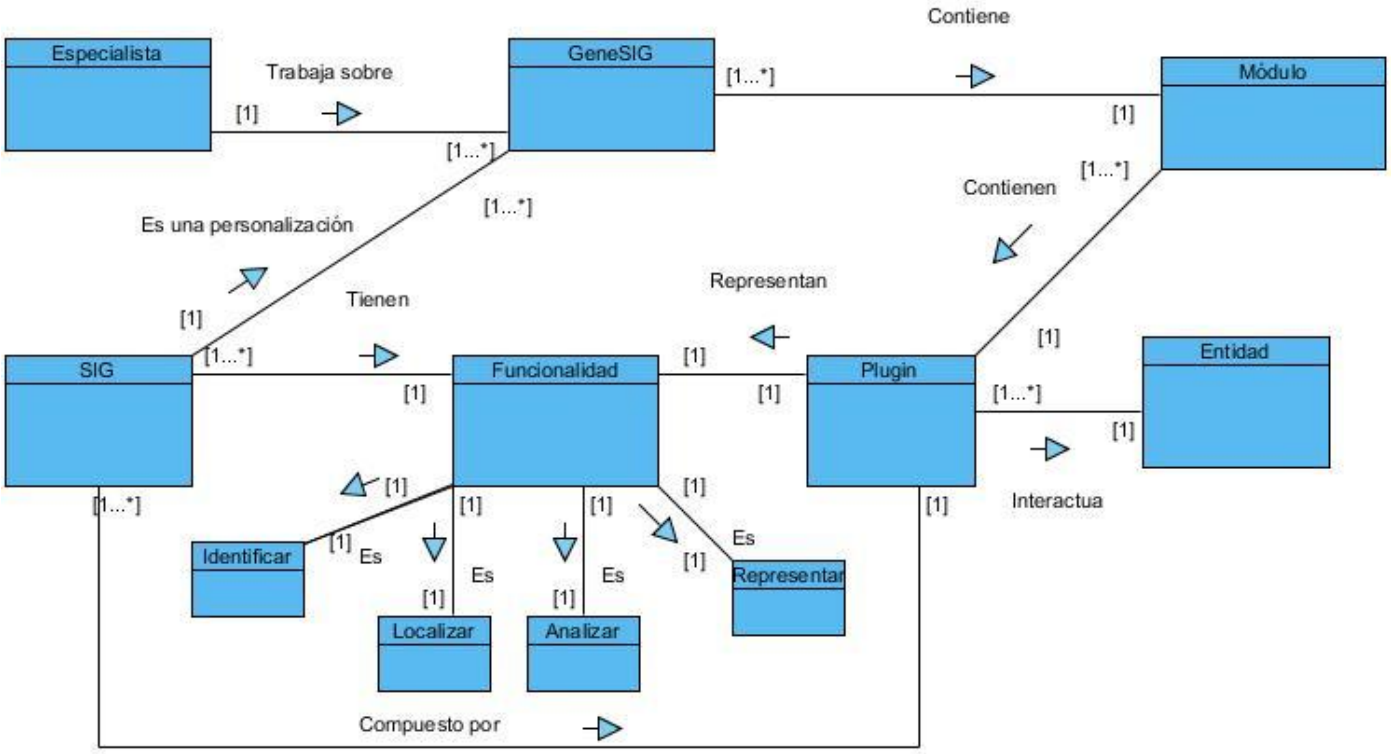



**Figura 5: Modelo de Dominio del sistema.**

### 2.3.2 Definición de clases del diagrama del dominio

**Especialista:** Este es el desarrollador que trabaja sobre la plataforma GeneSIG. Interactúa directamente con el sistema.

**GeneSIG:** Es la plataforma de desarrollo donde el especialista trabaja personalizando los SIG. Se compone de diferentes módulos.

**SIG:** Es el sistema de información geográfica personalizado sobre GeneSIG el cual está compuesto por varios plugins y a su vez tiene distintas funcionalidades.

**Módulos:** GeneSIG está compuesta por varios módulos entre ellos el de navegación, selección, consulta espacial, configuración del mapa, impresión y el de catalogo. Estos módulos están compuestos por plugin.

**Entidad:** Representan las tablas de la Base de Datos.

**Plugin:** Representan las distintas funcionalidades del sistema y utilizan las entidades.

**Funcionalidad:** Constituyen la razón de ser de los SIG, ya que representan las distintas operaciones que se pueden realizar.

## 2.4 Levantamiento de requisitos

Los requisitos de software representan las necesidades, los servicios que los usuarios desean que proporcione el sistema de desarrollo así como las restricciones en las que se debe operar. Los requisitos se dividen en funcionales y no funcionales, muestran las capacidades o condiciones que el sistema debe cumplir y las propiedades o cualidades que el producto debe tener. Dichos requisitos en la fase de construcción deben ser posibles de probar y verificar (PRESSMAN 2005).

### 2.4.1 Requisitos Funcionales

- **RF1.** Listar la Base de Datos existente en la plataforma Genesig: El sistema debe ser capaz de mostrar la Base de Datos ya existente en la plataforma Genesig, así como la agregada por el usuario.
- **RF2.** Habilitar una Base de Datos: El sistema debe permitir seleccionar una Base de Datos ya existente, de una lista de Bases de Datos.
- **RF3.** Deshabilitar una Base de Datos: El sistema debe permitir deseleccionar la Base de Datos previamente seleccionada.
- **RF4.** Conectarse con una Base de Datos externa a la plataforma Genesig: El sistema debe permitir la conexión con una Base de Datos independiente a la plataforma Genesig.
- **RF5.** Adicionar una Base de Datos: El sistema debe permitir al usuario adicionar una Base de Datos a la lista de Bases de Datos existente.
- **RF6.** Mostrar detalles de la Base de Datos seleccionada: El sistema debe permitir visualizar los detalles de conexión de la Base de Datos seleccionada.
- **RF7.** Mostrar las tablas de una Base de Datos previamente seleccionada: El sistema debe ser capaz de mostrar todas las tablas de una Base de Datos previamente seleccionada por el usuario.

- **RF8.** Seleccionar una tabla de la Base de Datos: El sistema debe permitir seleccionar tablas de la Base de Datos previamente seleccionada.
- **RF9.** Deseleccionar una tabla de la Base de Datos: El sistema debe permitir deseleccionar la tabla de la Base de Datos anteriormente seleccionada.
- **RF10.** Listar atributos de una tabla seleccionada: El sistema debe ser capaz de mostrar los atributos que corresponden a la tabla seleccionada.
- **RF11.** Listar el tipo de datos que poseen los atributos de la tabla seleccionada: El sistema debe permitir mostrar el tipo de dato que poseen los atributos de la tabla seleccionada.
- **RF12.** Seleccionar los atributos de una tabla: El sistema debe permitir seleccionar los atributos con los que el usuario interactuará directamente.
- **RF13.** Deseleccionar los atributos de una tabla: El sistema debe permitir deseleccionar los atributos previamente seleccionados por el usuario.
- **RF14.** Registrar el nombre del plugin: El sistema debe permitir al usuario guardar el nombre del plugin ha generar.
- **RF15.** Listar la acción de CRUD en la tabla seleccionada: El sistema debe permitir mostrar las acciones que se van a realizar sobre la tabla previamente seleccionada.
- **RF16.** Seleccionar la acción CRUD sobre la tabla seleccionada: El sistema debe permitir seleccionar la acción que se va a realizar sobre la tabla seleccionada.
- **RF17.** Deseleccionar la acción CRUD sobre la tabla seleccionada: El sistema debe permitir deseleccionar la acción previamente seleccionada.
- **RF18.** Crear plugin según la acción CRUD seleccionada: El sistema debe ser capaz de crear un plugin por cada código generado según la acción CRUD seleccionada.
- **RF19.** Eliminar el plugin generado: El sistema debe permitir eliminar el plugin que previamente se ha generado.
- **RF20.** Visualizar los plugin generados: El sistema debe permitir visualizar los plugin ya generados anteriormente.

### 2.4.2 Requisitos no Funcionales

**Usabilidad**

- **RNF1:** El sistema será utilizado por especialistas con conocimiento de ingeniería de software y programación orientada a objetos, preferentemente especialistas en el área de los SIG. Asimismo, deben poseer experiencia en la utilización y en el funcionamiento de GeneSIG.
- **RNF2:** El módulo tendrá una correcta Arquitectura de la Información, con iconografía relevante a la función realizada y con colores acordes a la plataforma.

**Fiabilidad**

- **RNF3:** La herramienta de implementación a utilizar debe tener soporte para recuperación ante fallos y errores.
- **RNF4:** La información manejada por el módulo estará protegida de acceso no autorizado.
- **RNF5:** La información y las funcionalidades que forman parte del sistema estarán disponibles como parte de la plataforma GeneSIG y el usuario podrá acceder a ellas las 24 horas del día durante los siete días de la semana.
- Los dispositivos o mecanismos utilizados para lograr la seguridad no ocultarán o retrasarán a los usuarios para obtener los datos deseados en un momento dado.

**Soporte**

- **RNF6:** El soporte será realizado por los propios autores de conjunto con especialistas de la LPS Aplicativos SIG.

**Restricciones de diseño e implementación**

- **RNF7:** El diseño debe ser sencillo, con pocas entradas, donde no sea necesario mucho entrenamiento para utilizar el módulo.
- **RNF8:** El modulo debe ser diseñado a partir de las especificaciones que propone la arquitectura basado en componentes a partir de un modelo cliente-servidor.

- **RNF9:** Se deben emplear los estándares establecidos por la LPS Aplicativos SIG y la Universidad de las Ciencias Informáticas, fundamentalmente referidos a diseño de interfaces y Base de Datos.
- **RNF10:** Se utilizará el estándar lowerCamelCase para la codificación de las clases.

**Requisitos de Interfaz**

**Interfaces de Usuario**

- **RNF11:** La Interfaz de usuario debe ajustarse al diseño de la plataforma GeneSIG. El acceso al módulo de Generación de CRUD se realizará a través de la pantalla principal y será accesible por medio de un botón ubicado en el área de las funcionalidades de la plataforma.

**Interfaces de Hardware**

Para las computadoras cliente:

- **RNF12:** 128 MB de RAM como mínimo.
- **RNF13:** Procesador 512 MHz como mínimo.

Para los servidores:

- **RNF14:** 1GB de RAM y 40 GB de disco duro para el servidor de Base de Datos como mínimo.
- **RNF15:** Procesador 3.0 GHz como mínimo.

**Interfaces de Software**

La construcción de la aplicación funcionará bajo los conceptos de arquitectura cliente/servidor. Por tanto, el servidor del usuario final debe tener como requerimientos mínimos de software:

Para las PCs clientes:

- **RNF16:** Un navegador como Mozilla Firefox 16.0 o superior, Zafari 4.0 o superior u otro que cumpla con los estándares W3C.
- **RNF17:** Sistema operativo basado en GNU/Linux, Windows 7 (o superior) o Mac OS.

Para los Servidores:

- **RNF18:** Sistema operativo GNU/Linux Ubuntu Server 11.04.
- **RNF19:** Servidor Web Apache 2.0 o superior, con módulo PHP 5 configurado con la extensión pgsql incluida.
- **RNF20:** PostgreSQL 8.4 o superior como Sistema Gestor de Base de Datos.
- **RNF21:** Extensión PostGIS de PostgreSQL para el manejo de los datos espaciales.
- **RNF22:** MapServer 5.2.2 o superior, con extensión PHP mapscript.

**Interfaces de Comunicación**

- **RNF23:** El módulo Generador de CRUD garantizará mediante su interfaz la configuración del entorno de trabajo mediante funcionalidades propias como ocultar y mostrar paneles.
- **RNF24:** Debe utilizarse 1024 x 768 como resolución de pantalla óptima.

**Requisitos de Licencia**

- **RNF25:** Se permite la utilización, modificación y distribución de la aplicación por terceros sin necesidad de obtener la autorización de sus respectivos titulares.

**RNF11.Requisitos Legales, Derecho del Autor**

- **RNF26:** El módulo debe ajustarse y regirse por la ley, decretos leyes, decretos, resoluciones y manuales (órdenes) establecidos por la República de Cuba y la UCI.
- **RNF27:** Como producto, este módulo se distribuye amparado bajo las normativas legales establecidas en el Centro GEYSED y la Universidad de las Ciencias Informáticas.

## 2.5 Modelo del Sistema

Los requisitos funcionales y no funcionales se estructuran de forma natural mediante Casos de Uso. El modelo de casos de uso permite a los desarrolladores de software y los clientes llegar a un acuerdo sobre las condiciones y capacidades que debe cumplir el sistema. La mayoría de los sistemas tienen muchos tipos de usuarios. Cada usuario se representa mediante un actor. Los actores utilizan el sistema al interactuar con los Casos de Uso (PRESSMAN 2005).

### 2.5.1 Actores del Sistema

Los actores se definen como los roles que puede tener un usuario; pueden ser personas, empresas o cualquier ente que se beneficie con el uso del sistema para de esta forma intercambiar datos. Durante el desarrollo de la aplicación se ha definido un actor del sistema.

**Tabla 2: Descripción de los actores del sistema.**

| Actor | Descripción |
|---|---|
| **Especialista (Desarrollador)** | Es el encargado de realizar todas las funcionalidades del sistema que requieran la intervención humana para la configuración del mismo, la introducción de datos y la confirmación de acciones. |

### 2.5.2 Diagrama de Casos de Uso del Sistema

El Diagrama de Casos de Uso del Sistema se utiliza para mostrar las funciones de un sistema de software desde el punto de vista de sus interacciones con el exterior y sin entrar en la descripción detallada ni en la implementación de las funciones. Un diagrama de casos de uso consta de los siguientes elementos: el actor, el Caso de Uso y las Relaciones entre ellos (FALGUERAS 2003).

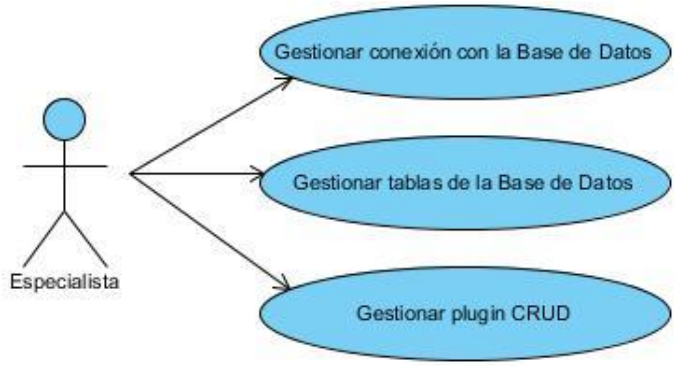



**Figura 6: Diagrama de Casos de Uso del sistema.**

### 2.5.3 Descripción de los Casos de Uso del Sistema

La Descripción del Caso de Uso ayuda a identificar los objetos y operaciones en los sistemas. Cada Caso de Uso se describe utilizando una descripción en lenguaje natural. Esto ayuda a los diseñadores a identificar los objetos en el sistema y les permite comprender el comportamiento que tendrá el sistema (SOMMERVILLE 2005).

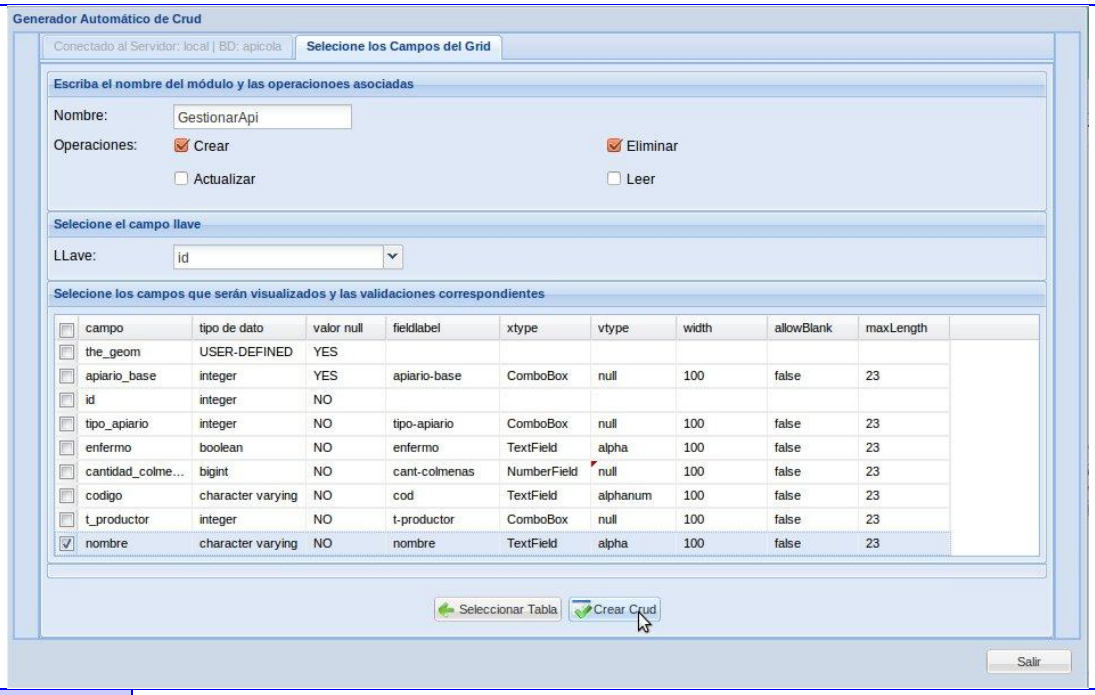
**Tabla 3: Descripción del Caso de Uso Gestionar plugin CRUD.**

| | |
|---|---|
| **Caso de Uso:** | Gestionar plugin CRUD |
| **Actores:** | Especialista |
| **Propósito:** | El Caso de Uso se realiza con el objetivo de generar el plugin de las acciones Crear, Leer, Actualizar y Eliminar una tabla seleccionada. |
| **Resumen:** | El Caso de Uso se inicia cuando el usuario realiza la acción de seleccionar una operación a realizar sobre una tabla previamente seleccionada, se procede a registrar el nombre del plugin a generar, se selecciona la operación a realizar en dicha tabla y se eligen los atributos de la tabla que deberá entrar el usuario obligatoriamente, luego se procede a generar el código. |
| **Precondiciones:** | El especialista tiene que estar autenticado en la plataforma. Tiene que haber seleccionado una Base de Datos y una tabla de esa Base de Datos. |
| **Referencias:** | RF 10, RF 11, RF 12, RF 13, RF 14, RF 15, RF 16, RF 17, RF 18, RF 19 y RF 20. |
| **Prioridad:** | Crítico |

| **Flujo Normal de Eventos** | |
|---|---|
| **Acción del Actor** | **Respuesta del Sistema** |
| **1.** El usuario selecciona la operación (Crear, Visualizar o Eliminar) a realizar sobre la tabla previamente seleccionada. | **2.** El sistema realiza la operación según la opción seleccionada por el usuario.<br>- Si selecciona “Crear”, ver sección “Crear”.<br>- Si selecciona “Visualizar”, ver sección “Leer”.<br>- Si selecciona “Eliminar”, ver sección “Eliminar”.<br>**3.** El sistema construye el plugin. |
| **Sección “Crear”** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1.** El especialista registra el nombre del nuevo plugin.<br>**2.** Selecciona la acción (crear, leer, actualizar y/o eliminar) a realizar sobre la tabla antes seleccionada.<br>**3.** Selecciona la llave primaria de la tabla.<br>**4.** Elige una serie de atributos de la tabla antes seleccionada (los que van a aparecer en el formulario). | **5.** El sistema procesa la información.<br>**6.** El sistema construye el plugin.<br>**7.** El sistema muestra un mensaje ”Se ha creado el plugin: nombre_plugin correctamente”<br>**8.** El sistema actualiza la lista de plugin generados. |

# Generador automático de CRUD para la LPS Aplicativos SIG

| Sección "Visualizar" | |
|---|---|
| **Acción del Actor** | **Respuesta del Sistema** |
| | **1.** El sistema muestra una lista con los plugin generados. |
| **Sección "Eliminar"** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1.** El especialista selecciona un plugin de la lista de plugin generados.<br>**2.** Elige la opción de eliminar plugin. | **3.** El sistema procesa la información.<br>**4.** El sistema elimina el plugin.<br>**5.** El sistema actualiza la lista de plugin generados. |
| **Flujos Alternos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| | **3.1.** El sistema muestra un mensaje de error "no ha seleccionado ningún atributo".<br>**3.2.** El sistema muestra un mensaje de error "no ha registrado el nombre del plugin".<br>**3.3.** El sistema muestra un mensaje de error "no se ha seleccionado ninguna acción a realizar". |
| ***Prototipo de Interfaz*** | |




| **Pos-condiciones:** | El sistema visualiza el nuevo plugin creado por el especialista. |
|---|---|

## 2.6 Elementos fundamentales de la arquitectura de software

En la actualidad, la selección de la arquitectura de software se ha convertido en uno de los factores decisivos para el éxito o no del sistema a implementar. Según lo definido por (PRESSMAN 2005) se tiene que la arquitectura es una vista estructural de alto nivel, que define estilo o combinación de estilos para una solución. Se puede decir que la arquitectura es esencial para el éxito o el fracaso de un proyecto. Además la arquitectura de un software es necesaria para comprender el sistema, organizar el desarrollo del mismo, fomentar la reutilización y controlar la evolución del proyecto.

Según la definición dada por (MACERO and GUAMAN 2009) la Arquitectura de Software es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones.

### 2.6.1 Patrones arquitectónicos

Los patrones arquitectónicos, según (CAMACHO *et al.* 2004) expresan el esquema de organización estructural fundamental para sistemas de software. Proveen un conjunto de subsistemas predefinidos, especifica sus responsabilidades e incluye reglas y pautas para la organización de las relaciones entre ellos. La selección de un patrón arquitectónico es, por lo tanto, una decisión fundamental de diseño en el desarrollo de un sistema de software

El módulo Generador de CRUD, se ha desarrollado bajo el patrón arquitectónico basado en componentes. Este define cómo organizar el modelo en componentes funcionales, exponiendo interfaces de comunicación que contienen métodos, eventos y propiedades. El mismo permite que se pueda representar el módulo como un componente que es incorporado a la plataforma GeneSIG, de manera que no altere los restantes componentes, ni la estructura de la misma; brindando la posibilidad de que el sistema sea flexible y fácil de personalizar.

### 2.6.2 Estilos arquitectónicos

Un estilo arquitectónico expresa componentes y las relaciones entre estos, con las restricciones de su aplicación y la composición asociada, así como también las reglas para su construcción. Se consideran

como un tipo particular de estructura fundamental para un sistema de software, conjuntamente con un método asociado que especifica cómo construirlo incluyendo información acerca de cuándo usar la arquitectura que describe, sus invariantes y especializaciones, así como las consecuencias de su aplicación (CAMACHO *et al.* 2004)

Los estilos establecen un vocabulario común, y brindan soporte a los ingenieros para conseguir una solución que haya sido aplicada con éxito anteriormente, ante ciertas situaciones de diseño. Además, su aplicación en el diseño de la arquitectura del sistema es determinante para la satisfacción de los requerimientos de calidad.

Los estilos arquitectónicos más utilizados son; Flujo de Datos, Centrado de Datos y Llamada y Retorno que es el que se utiliza en la confección del módulo. El estilo llamada y retorno permite que el sistema se constituya de un programa principal que tiene el control del sistema y de varios subprogramas que se comunican con éste mediante el uso de llamadas, permite construir una estructura de programa relativamente fácil de modificar y ajustar. Esta familia de estilos enfatiza la modificabilidad y la escalabilidad. Son los estilos más generalizados en sistemas en gran escala (CAMACHO *et al.* 2004).

### 2.6.3 Patrones de diseño

Los patrones de diseño Pressman los define, de manera general, como soluciones estándar que brindan respuesta a un problema común durante el diseño de un software. Una vez que se ha desarrollado el modelo de análisis, el diseñador puede examinar una representación detallada del problema que debe resolver y las restricciones que impone (PRESSMAN 2005).

En el diseño de la solución propuesta se aplican los patrones generales de asignación de responsabilidades (del inglés General Responsibility Asigment Software Patterns, GRASP) y los patrones del Grupo de los Cuatro (del inglés Group of Four, GOF).

A continuación se mencionan los patrones utilizados, que permiten describir los principios fundamentales del diseño de objetos para la asignación de responsabilidades.

**Patrones GRASP:**

- **Experto:** En este patrón se persigue que la responsabilidad de realizar una labor sea de la clase que tiene o pueda tener los datos involucrados (atributos) en esa labor. En este caso, se tiene la clase

ClientGeneradorCrud, que es experta en procesar los datos que son enviados a través del navegador web, la clase ServerGeneradorCrud, que es la responsable de la interacción con el servidor de mapas y el servidor de bases de datos. El uso de este patrón permite que se conserve el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento en el futuro.

- **Creador:** Se asigna la responsabilidad de que una clase B cree un objeto de la clase A. En el caso de esta investigación, el patrón se implementa en las clases ClientGeneradorCrud y ServerGeneradorCrud, las cuales son las encargadas de crear una instancia de las clases DatosRequest y DatosResult, clases que describen las variables que contienen la información que forman parte de lo valores de entrada de la solicitud que realiza el ClientGeneradorCrud al ServerGeneradorCrud y viceversa.

- **Alta cohesión:** Cada elemento del diseño debe realizar una labor única dentro del sistema. En este caso, se garantiza que cada una de las clases del plugin GeneradorCRUD, posean alta cohesión, de manera que las clases posean la característica de tener las responsabilidades estrechamente relacionadas y que no realicen un trabajo enorme. 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 de funcionalidad.

- **Bajo acoplamiento:** Se utiliza para lograr la mínima dependencia entre las clases del sistema, lo cual disminuye considerablemente el flujo de datos, propiciando rapidez en la ejecución de sus funcionalidades. En este caso, se refleja el bajo acoplamiento, en cada una de las clases del plugin GeneradorCRUD, con el objetivo de que una clase no dependa de muchas clases, de esta forma, no se afectan las clases por cambios de otros componentes, son fáciles de entender por separado y fáciles de reutilizar (MORENO 2012).

**Patrones GOF:**

Se reconocen como patrones GOF, según el libro “Design Patterns: elements of reusable object-oriented software” a un conjunto de patrones (alrededor de 23) dentro del campo del desarrollo de software definidos por un grupo de cuatro personas: Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides (GAMMA *et al.* 1997).

A continuación se describe los patrones GOF tenidos en cuenta en la implementación de la solución:

- **Command**: patrón que permite encapsular las peticiones a través de un objeto, lo que permite realizar operaciones como gestionar las acciones de dicho objeto. 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.

- **Singlenton** (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Se utiliza en la clase ServerContext() que tiene como objetivo crear un objeto MapObj para que este no se cree cada vez que se realiza un envió a la aplicación.

## 2.7 Modelo del diseño

Los modelos de diseño muestran los objetos o clases en un sistema y, donde sea apropiado, los diferentes tipos de relaciones entres estas entidades. Son esencialmente el diseño mismo y el puente entre los requerimientos y la implantación del sistema (SOMMERVILLE 2005).

### 2.7.1 Diagrama de Clases del Diseño

El Diagrama de Clases del Diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación, permite modelar la vista de diseño del sistema y mejorar el modelo conceptual o de dominio. En la Figura 7 se presenta el Diagrama de Clases del Diseño del Caso de Uso crítico Generar plugin CRUD.

## 2.8 Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes, como ficheros de código fuente, ejecutables, entre otros. El modelo de implementación describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros (JACOBSON *et al.* 2003).

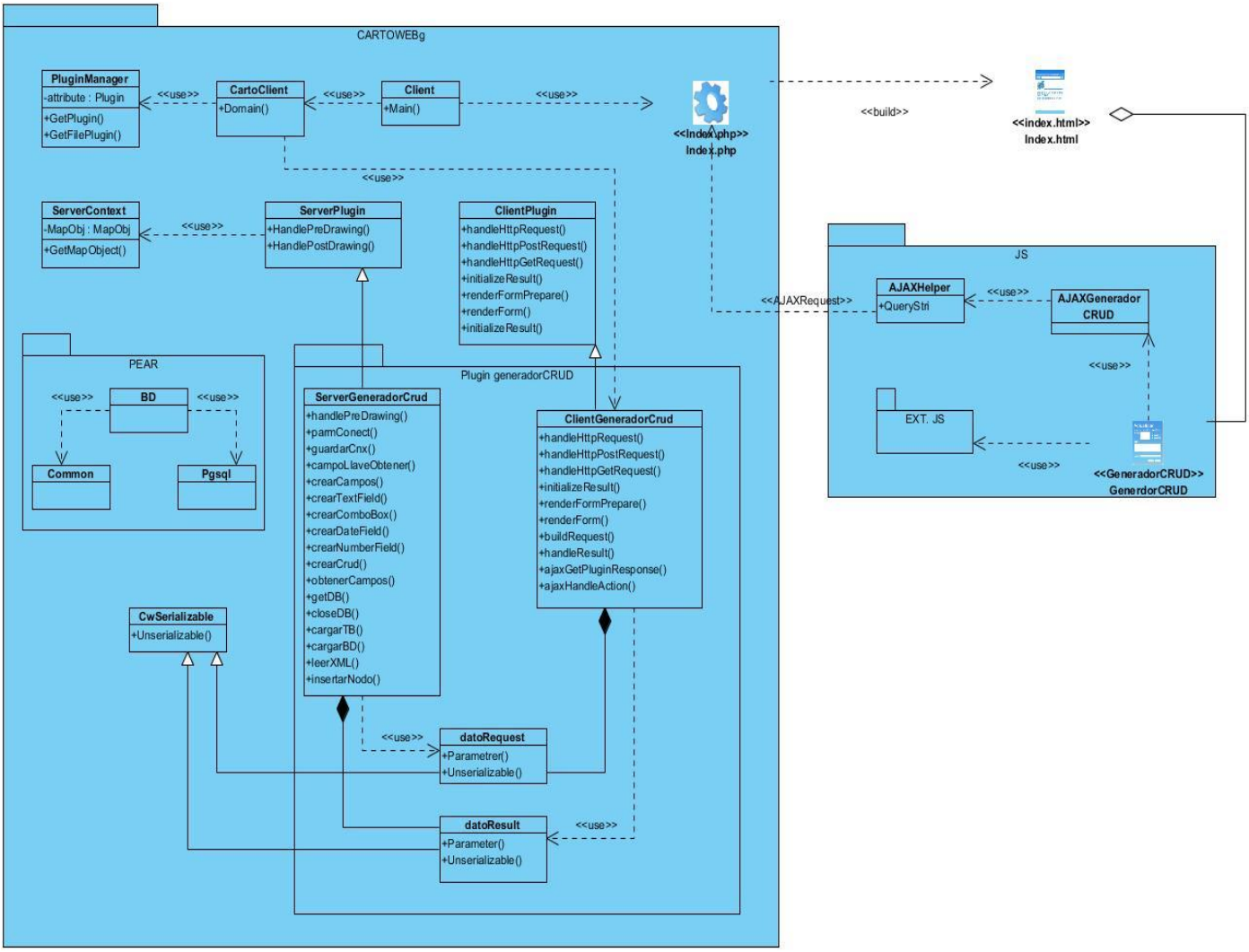



**Figura 7: Diagrama de Clases del Diseño del Caso de Uso Gestionar plugin CRUD.**

### 2.8.1 Diagrama de componentes

Los Diagramas de componentes modelan la vista estática de un sistema. Se representan como un grafo de componentes unidos por medio de relaciones de dependencia (compilación, ejecución), pudiendo mostrarse las interfaces que estos soporten. Estos tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo (PRESSMAN 2005). En la Figura 8 se presenta el Diagrama de Componentes propuesto para la implementación de la solución.

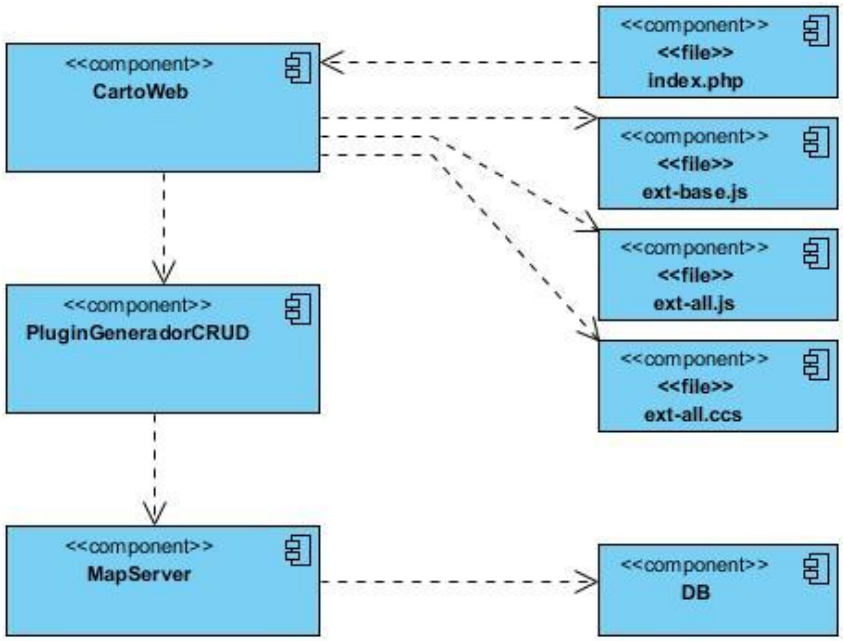



**Figura 8: Diagrama de Componentes.**

## 2.9 Modelo de despliegue

El Modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye el sistema entre los nodos de cómputo. Se utiliza como entrada fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una influencia principal en su diseño (JACOBSON *et al.* 2003). A continuación se presenta el modelo de despliegue propuesto.

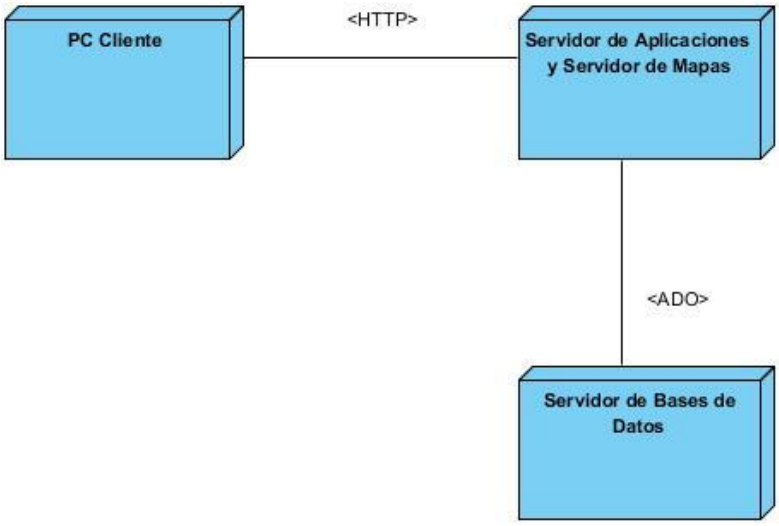



**Figura 9 Diagrama de Despliegue.**

### 2.9.1 Protocolos

**Protocolo HTTP:**del inglés, HyperText Transfer Protocol, usado para acceder a la web. Se encarga de procesar y dar respuestas a las peticiones para visualizar una página web. Además sirve para el envío de información adicional como el envío de formularios con mensajes.

Cuando finaliza una transacción, HTTP no guarda ninguna información sobre la misma, por lo que es considerado un protocolo "sin estado".

Este protocolo está basado en el modelo cliente-servidor, en donde un cliente HTTP (a menudo un navegador) abre una conexión y realiza una solicitud al servidor. Este responde a la petición con un recurso (texto, gráficos u otros) o un mensaje de error, y finalmente se cierra la conexión (ALEGSA 2009b).

**Protocolo ADO**: del inglés Microsoft ActiveX Data Objects, representa un mecanismo que utilizan los programas para comunicarse con las Bases de Datos. Provee una capa entre los lenguajes de programación y las Bases de Datos que permite a los programadores escribir programas que accedan a datos, sin saber cómo está implementada la Base de Datos (ALEGSA 2009a).

## 2.10 Conclusiones Parciales

El modelo de dominio y el modelo del sistema contribuyen al entendimiento del equipo de desarrollo de manera que las etapas sucesivas de la construcción de la solución pueden ser comprendidas y ejecutadas sin dificultad por los demás miembros. Los requisitos funcionales y no funcionales ubican al equipo de desarrollo en los objetivos de la solución proporcionando una guía para su desarrollo y posterior validación. La utilización de patrones de diseño en la implementación de la solución propició mayor organización, comprensión y menor tiempo de desarrollo.

# CAPÍTULO III: EL PROCESO DE PRUEBAS DE SOFTWARE

## 3.1 Introducción al capítulo

El flujo de trabajo de pruebas no es más que el proceso de ejecutar el programa con la finalidad de descubrir los errores del mismo. Las pruebas realizadas al software constituyen un elemento crítico para asegurar la garantía de la calidad del mismo y representan una revisión final de las especificaciones, del diseño, y de la codificación del software (PRESSMAN 2005).

En el presente capítulo se desarrolla el proceso de pruebas de la solución desarrollada, su ejecución y resultados finales.

## 3.2 Las pruebas

Algunas metodologías identifican a la etapa de pruebas como el flujo de trabajo fundamental cuyo propósito general es comprobar el resultado de la implementación mediante las pruebas de cada construcción, incluyendo tanto construcciones internas como intermedias, así como las versiones finales del sistema que van a ser entregadas a terceras partes (JACOBSON *et al.* 2003).

Según (DUHARTE 2008) las pruebas deben ayudar a mejorar la calidad: uno de los principales propósitos de las pruebas es detectar errores en los sistemas. Pero, también puede pensarse que ayudan a evitar errores, ejecutándolas de una forma correcta con el personal indicado y siguiendo un buen procedimiento durante todo el proceso de desarrollo.

## 3.3 Los métodos de prueba

Luego de ser generado el código fuente de un software, este debe ser probado para describir y corregir el máximo de errores antes de ser entregado al cliente. Para esto existen métodos de prueba dirigidos fundamentalmente a probar la lógica interna del programa (que son denominados pruebas de caja blanca) y a comprobar los requisitos del sistema (que son denominados pruebas de caja negra) (PRESSMAN 2005).

El proceso de pruebas propuesto tiene en cuenta estos dos métodos de prueba fundamentalmente. A continuación se describen cada uno de ellos.

### 3.3.1 Pruebas de Caja Negra

Las pruebas de Caja Negra se utilizan debido a que esta se centra en los requisitos funcionales y se lleva a cabo sobre la interfaz del software. Este método tiene como objetivo demostrar que las funciones del software son operativas, que las entradas se acepten de forma adecuada y se produzca un resultado correcto, teniendo en cuenta la integridad de la información externa.

El método mencionado anteriormente puede ser ejecutado a partir de la aplicación de varias técnicas, de las cuales se selecciona la técnica de partición equivalente, con el objetivo de probar cada componente de la interfaz y dividir el campo de entrada en clases de datos de los cuales se pueden derivar casos de prueba. Es un intento por dividir el dominio de entrada de un programa en un número finito de clases de equivalencia. Estas divisiones son denominadas Diseños de Caso de Prueba los cuales son elaborados a partir de la descripción textual de cada Caso de Uso del sistema.

**Diseño de Casos de Prueba**

El diseño de las pruebas se basa en la creación de casos de prueba cuya ejecución permitirá observar posibles síntomas de defectos. Se puede definir un caso de prueba como “el conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para un objetivo particular (por ejemplo, ejercitar un camino concreto de un programa o verificar el cumplimiento de un determinado requisito) (WYERS 2004).

**Diseño del Caso de Prueba del Caso de Uso “Gestionar plugin CRUD”**

- **Descripción General:**

El Caso de Uso se inicia cuando el usuario realiza la acción de seleccionar una operación a realizar en la tabla previamente seleccionada, se procede a registrar el nombre del plugin a generar, además se selecciona la operación a realizar en dicha tabla y se eligen los atributos de la tabla que deberá entrar el usuario obligatoriamente, luego se procede a generar el plugin.

- **Condiciones de ejecución:**

El especialista tiene que estar autenticado en la plataforma. Tiene que haber seleccionado una Base de Datos y seleccionado una tabla de dicha Base de Datos.

### Secciones a probar en el Caso de Uso “Gestionar plugin CRUD”

**Tabla 4: Secciones a probar en el Caso de Uso Gestionar plugin CRUD.**

| Nombre de la sección | Escenas de la sección | Descripción de la funcionalidad | Flujo Central |
|---|---|---|---|
| **SC 1: “Crear Plugin”** | **EC 1.1:** Crear Plugin | El especialista teclea el nombre del Plugin que desea crear y selecciona la acción CRUD que va a realizar. Luego, selecciona los atributos que van a aparecer en el formulario a generar y teclea los datos solicitados.<br>El especialista accede a la opción “Crear Crud”. El sistema muestra un mensaje de información “Se ha creado satisfactoriamente el plugin”. | El especialista introduce los datos requeridos por el sistema y accede a la opción “Crear Crud”.<br>El sistema muestra un mensaje confirmando la creación satisfactoria del plugin. |
| **SC 2: “Visualizar”** | **EC 2.1:** Visualizar | El sistema muestra una lista con los plugin generados.<br>Una vez creado el CRUD, el nombre del mismo pasa a formar parte de la lista de plugin generados. | El sistema muestra la lista de plugin generados.<br>El sistema agrega el nuevo plugin a la lista de plugin generados y visualiza el plugin en la lista. |
| **SC 3: “Eliminar plugin”** | **ES 3.1:** Eliminar plugin | El usuario selecciona el plugin que desea eliminar de la lista de plugin creados y accede a la opción “Eliminar plugin”. El sistema muestra el mensaje “Se ha eliminado con éxito el plugin seleccionado”. | El sistema muestra una lista con los plugins creados. El especialista selecciona un plugin y accede a la opción “Eliminar plugin”.<br>El sistema muestra el mensaje “Se ha eliminado con éxito el plugin seleccionado” |

- **Descripción de variables**

**Tabla 5: Descripción de variables.**

| No. | Nombre de Campo | Clasificación | Valor nulo | Descripción |
|---|---|---|---|---|
| 1 | Nombre del plugin | Campo de Texto | No | El especialista teclea el nombre del plugin a crear. |
| 2 | Acción | Campo de selección | No | El especialista selecciona la acción CRUD a realizar. |
| 3 | fieldLabel | Campo de texto | No | El especialista teclea la descripción del componente en el formulario. |
| 4 | xType | Campo de selección | No | El especialista selecciona el tipo de componente. |
| 5 | vType | Campo de selección | No | El especialista selecciona el tipo de validación. |
| 6 | width | Campo de selección | No | El especialista selecciona el ancho del componente. |
| 7 | allowBlank | Campo de selección | No | El especialista selecciona si el campo puede o no ser nulo. |
| 8 | maxLength | Campo de texto | No | El especialista selecciona el máximo de caracteres a entrar en el componente. |
| 9 | Lista de Plugin | Campo de selección | No | El especialista selecciona un plugin de los ya creados. |

- **Matriz de Datos**

**Tabla 6: Matriz de Datos de la Sección de Pruebas 1.**

| SC 1: “Crear plugin” | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Id. E. | Nombre del Plugin | Acción | FieldLabel | xType | vType | width | AllowBlank | MaxLength | Respuesta del sistema | Resultado de la prueba |
| EC 1.1 | gestionarApiario | Crear, Eliminar, Actualizar, Leer | Apiario | TexField | alpha | 100 | False | 23 | El sistema crea el plugin con éxito. | Satisfactorio |
| EC 1.2 | gestionarFloración | Crear, Eliminar | (Vacío) | (Vacío) | Alpha | 100 | False | 43 | El sistema muestra un mensaje de error. | Satisfactorio |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| EC 1.3 | (Vacío) | Crear, Eliminar, Leer | Floración | TexField | Alpha | (Vacío) | (Vacío) | (Vacío) | El sistema muestra un mensaje de error. | Satisfactorio |
| EC 1.4 | gestionarFloración | (Vacío) | Floración | TexField | Alpha | 100 | false | 43 | El sistema muestra un mensaje de error. | Satisfactorio |
| EC 1.5 | gestionarFloración | Crear, Eliminar | Floración | TexField | (Vacío) | (Vacío) | (Vacío) | (Vacío) | El sistema muestra un mensaje de error. | Satisfactorio |

**Tabla 7: Matriz de la Sección de Pruebas 2.**

| SC 2: "Visualizar" | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Id. E.** | **Nombre del Plugin** | **Acción** | **FieldLabel** | **xType** | **vType** | **width** | **AllowBlank** | **MaxLength** | **Respuesta del sistema** | **Resultado de la prueba** |
| EC 2.1 | gestionarApiario | Crear, Eliminar, Actualizar, Leer | Apiario | TexField | Alpha | 100 | False | 20 | El sistema visualiza el nombre del plugin creado con éxito. | Satisfactorio |
| EC 2.2 | (Vacío) | (Vacío) | (Vacío) | (Vacío) | (Vacío) | (Vacío) | (Vacío) | (Vacío) | El sistema visualiza el nombre de los plugin creados anteriormente | Satisfactorio |

**Tabla 8: Matriz de la Sección de Pruebas 3.**

| SC 3: "Eliminar" | | | |
|---|---|---|---|
| **Id. del escenario** | **Lista de plugin** | **Respuesta del sistema** | **Resultado de la prueba** |
| EC 3.1 | gestionarApiario | El sistema elimina el plugin con éxito. | Satisfactorio |
| EC 3.2 | (Vacío) | El sistema muestra un mensaje de error. | Satisfactorio |

### 3.3.2 Pruebas de Caja Blanca

Las pruebas de Caja Blanca se utilizan debido a que estas se centran mayormente en la parte interna del software, específicamente en el código fuente generado, a diferencia de las pruebas de Caja Negra que se centran en la interfaz del programa. Este método tiene como objetivo verificar que el algoritmo utilizado en cada uno de los procesos del sistema funcione correctamente.

Este método está compuesto por varias técnicas, de las cuales se selecciona la prueba de "camino básico" con el objetivo de garantizar que se ejecute por lo menos una vez cada sentencia del programa. Es un intento para tener la medida de la complejidad lógica de un diseño procedimental y así usarla como guía para definir un conjunto básico de caminos de ejecución.

### Camino básico

A continuación se realiza la prueba de camino básico al método *crearCrud($request)* el cual tiene como función crear el plugin CRUD a partir de los datos que provee el especialista.

***Paso 1:*** *Obtener el Grafo de Flujo, a partir del diseño o del código del módulo.*

El código de este método se encuentra en el **Anexo 1** del presente documento. Para la construcción del grafo se tuvo en cuenta la notación referenciada por (ESPINOSA and CUTIÑO 2008) y presentada en la Figura 10.

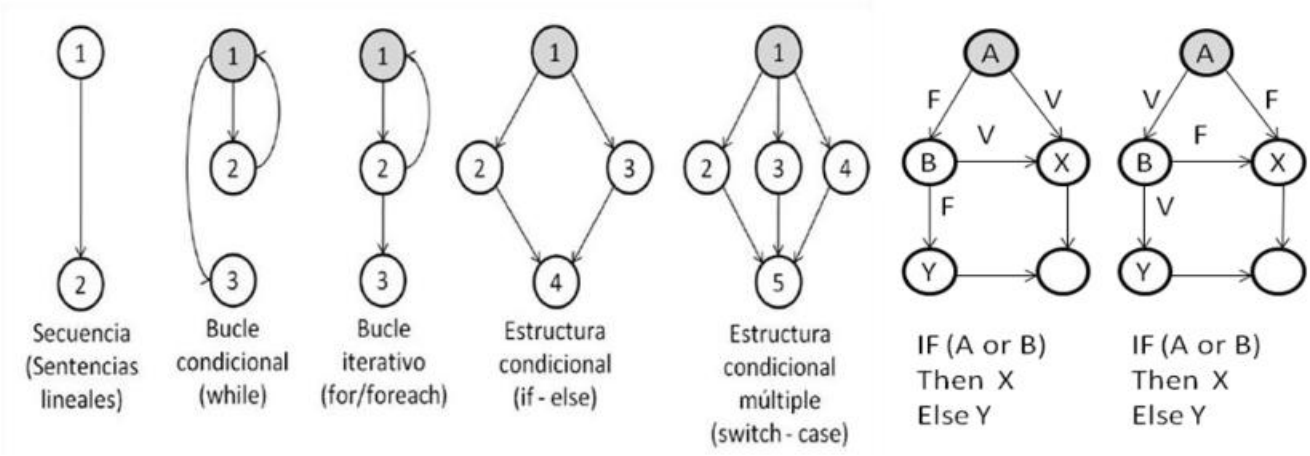



**Figura 10: Notación de grafo de flujo.**

De esta notación se acota lo siguiente:

- Cada circulo denominado Nodo representa 1 o más sentencias.
- Un solo nodo puede corresponder a una secuencia de cuadros de procesos y a un rombo de decisión.
- Cada nodo que contiene una condición se denomina Nodo Predicado y está caracterizado porque 2 o más aristas emergen de él.
- Las flechas denominadas Aristas representan flujo de control y deben terminar en un nodo.
- Las áreas delimitadas por aristas y nodos se denominan Regiones.
- Cuando se contabilizan las regiones se incluye el área exterior del grafo.
- Todo segmento de código de cualquier programa se puede traducir a un Grafo de Flujo.

De acuerdo con lo anterior, y a partir de la codificación de la función *crearCrud($request)*, se obtiene el siguiente grafo:

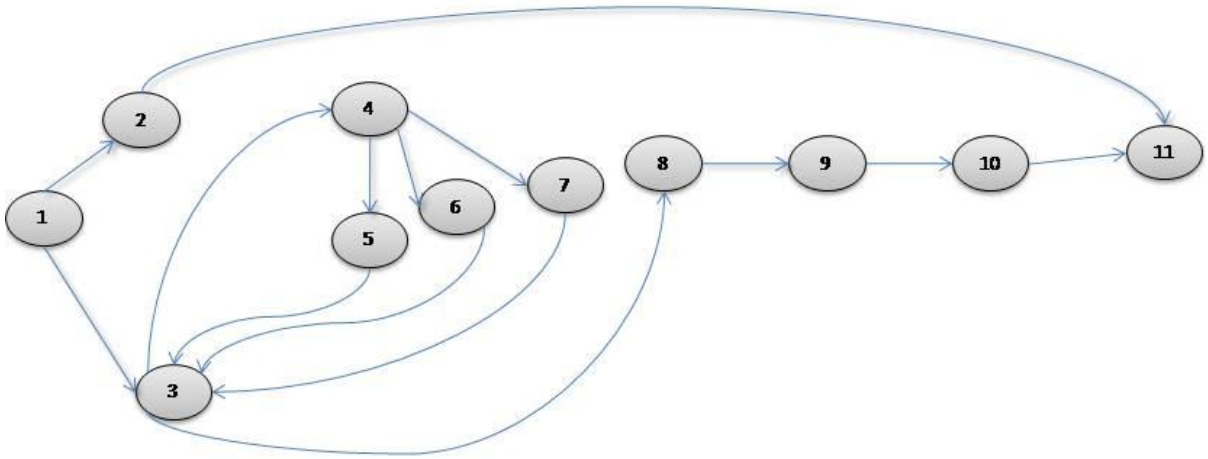



**Figura 11 Grafo de Flujo de la función *crearCrud($request).***

***Paso 2:*** *Obtener la Complejidad Ciclomática del grafo de flujo.*

Para efectuar el cálculo de la Complejidad Ciclomática del código es necesario conocer el valor de varios parámetros como son: la cantidad total de aristas (**A**), la cantidad total de nodos (**N**), la cantidad total de nodos predicados (**P**) y por último la cantidad total de regiones (**R**). Estos parámetros se utilizan en uno o varias de las fórmulas presentadas en la Figura 12 con la finalidad de efectuar el cálculo. Se desarrollan las tres fórmulas propuestas para garantizar la efectividad del resultado.

| | | |
|---|---|---|
| $V(G) = (A - N) + 2$<br>$V(G) = (13 - 11) + 2$<br>$V(G) = 4$ | $V(G) = P + 1$<br>$V(G) = 3 + 1$<br>$V(G) = 4$ | $V(G) = R$<br>$V(G) = 4$ |

**Figura 12: Fórmulas para el cálculo de la complejidad ciclomática.**

***Paso 3:*** *Definir el conjunto básico de caminos independientes.*

Del cálculo de la complejidad ciclomática se obtiene el número de caminos linealmente independientes de la estructura de control del programa. Entiéndase por camino linealmente independiente a cualquier camino del programa que introduce, al menos, una nueva condición. Para este método, la cantidad de caminos independientes es de 4.

- Camino 1 : 1-2-11
- Camino 2 : 1-3-4-5-3-8-9-10-11
- Camino 3 : 1-3-4-6-3-8-9-10-11
- Camino 4 : 1-3-4-7-3-8-9-10-11

***Paso 4:*** *Determinar los casos de prueba que permitan la ejecución de cada uno de los caminos anteriores.*

Para determinar los casos de prueba se deben seleccionar los datos de manera que las condiciones de los nodos predicados estén adecuadamente establecidas, con el fin de comprobar cada camino.

**Tabla 9: Caso de prueba para el método crearCrud().**

| Camino | Condición de ejecución | Entrada | Resultado esperado |
|---|---|---|---|
| **1** | Para esta prueba se hace necesario que el valor de la variable $nombrePlugin coincida con el nombre de algún plugin presente en la plataforma. | $request = (Es un Objeto)<br>DSN = geia<br>Tabla = servicio<br>Llave = id<br>Acciones = Adicionar<br>Campos = nombre, id<br>NombrePlugin = Servicio | Teniendo en cuenta la condición de ejecución se espera que el módulo muestre un mensaje “Ya existe el Plugin Servicio” y termine la ejecución del método.<br><br>*El resultado fue satisfactorio.* |

| | | | |
|---|---|---|---|
| **2** | Para esta prueba se hace necesario que el valor del campo destinado a la acción Crear se encuentre seleccionado. | $request = (Es un Objeto)<br>DSN = geia<br>Tabla = servicio<br>Llave = id<br>Acciones = Crear<br>Campos = nombre, id<br>NombrePlugin = CrearServicio. | Teniendo en cuenta la condición de ejecución se espera que el módulo muestre un mensaje “Se ha creado el plugin CrearServicio satisfactoriamente”.<br>*El resultado fue satisfactorio.* |
| **3** | Para esta prueba se hace necesario que el valor del campo destinado a la acción Modificar se encuentre seleccionado. | $request = (Es un Objeto)<br>DSN = geia<br>Tabla = servicio<br>Llave = id<br>Acciones = Crear<br>Campos = nombre, id<br>NombrePlugin = ModificarServicio. | Teniendo en cuenta la condición de ejecución se espera que el módulo muestre un mensaje “Se ha creado el plugin ModificarServicio satisfactoriamente”.<br>*El resultado fue satisfactorio.* |
| **4** | Para esta prueba se hace necesario que el valor del campo destinado a la acción Eliminar se encuentre seleccionado. | $request = (Es un Objeto)<br>DSN = geia<br>Tabla = servicio<br>Llave = id<br>Acciones = Crear<br>Campos = nombre, id<br>NombrePlugin = EliminarServicio. | Teniendo en cuenta la condición de ejecución se espera que el módulo muestre un mensaje “Se ha creado el plugin EliminarServicio satisfactoriamente”.<br>*El resultado fue satisfactorio.* |

## 3.4 Resultado de las pruebas

Una vez diseñados todos los casos de prueba a partir de las descripciones textuales de los Casos de Uso del Sistema se desarrollaron tres iteraciones como se muestra en la

**Primera Iteración:** Se obtuvieron cinco No Conformidades distribuidas por Casos de Uso como se muestra en la Figura 13.

**Segunda Iteración:** Una vez corregidas las No Conformidades anteriores, se ejecutaron nuevamente los diseños elaborados, obteniéndose una No Conformidad en el Caso de Uso que modela la creación del plugin CRUD, la cual fue corregida de inmediato. Se decide no realizar la tercera Iteración.

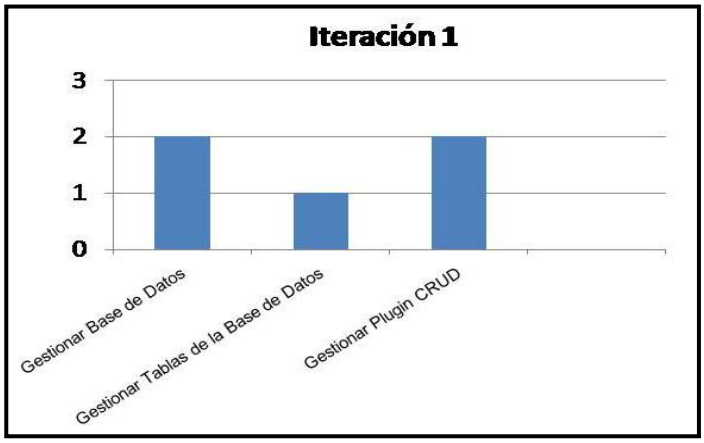



**Figura 13: Primera iteración de la ejecución de las pruebas.**

## 3.5 Conclusiones Parciales

- Las pruebas que se seleccionan se consideran acertadas y adecuadas pues buscan el correcto funcionamiento de la aplicación desde la experiencia del usuario.
- El proceso de pruebas desarrollado permitió identificar y resolver los errores en la implementación aumentando la calidad del producto final obtenido.

# CONCLUSIONES

1. Las herramientas y tecnologías utilizadas para resolver el problema planteado -en su condición de libres y multiplataforma-, permiten al proyecto Aplicativos SIG disponer de un producto informático que contribuye a la soberanía tecnológica que impulsa el país y la universidad.
2. Las soluciones existentes analizadas no se consideran soluciones efectivamente del problema de esta investigación debido fundamentalmente a la incompatibilidad de sus arquitecturas y la imposibilidad de vincularlas a la plataforma GeneSIG. No obstante, fueron tenidas en cuenta para la propuesta de funcionalidades e interfaz.
3. Los patrones de diseño utilizados en la implementación del módulo proporcionan uniformidad, comprensión y escalabilidad a la solución desarrollada, al mismo tiempo que favorecieron la rigurosidad en su diseño.
4. El proceso de pruebas desarrollado permitió identificar y resolver los errores en la implementación, aumentando la calidad del producto final obtenido.
5. La utilización del módulo Generador Automático de CRUD desarrollado para la LPS Aplicativos SIG favorecerá el desarrollo de los SIG en dicha LPS, dotando al equipo de desarrollo de una herramienta que agilizará el proceso de desarrollo y reducirá las oportunidades de error a la hora de realizar un SIG.

# Generador automático de CRUD para la LPS Aplicativos SIG

## RECOMENDACIONES

1. Desarrollar la funcionalidad de modificar el plugin creado, que le permita el desarrollador realizar modificaciones en el plugin una vez que este ha sido creado.
2. Ajustar la implementación y la arquitectura del plugin y desarrollarlo para la versión 2.0 de GeneSIG.

## REFERENCIAS BIBLIOGRÁFICAS

1. ALEGSA, L. *Definición de ADO*, DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA, 2009a. [2015]. Disponible en: http://www.alegsa.com.ar/Dic/ado.php
2. ---. *Definición de HTTP*, DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA, 2009b. [2015]. Disponible en: http://www.alegsa.com.ar/Dic/http.php
3. ALFARO, F. M. *Herramientas CASE*, Instituto Nacional de Estadística e Informática, 1999. [2015]. Disponible en: http://es.scribd.com/doc/43573942/Herramientas-Case#scribd
4. ANALYSIS, N. C. F. G. I. A. *Los Sistemas de Información Geográfica*, 2010.
5. BECERRA, F. H. M. and L. A. G. CRUZ. *Sistema de control y seguridades para el proyecto AVES (Aplicación de Tecnologías de Virtualización para la ESPE) empleando la metodología AUP*, Universidad de las Fuerzas Armadas ESPE. Vicerrectorado de Investigación, Innovación y Transferencia de Tecnología., 2015.
6. BRIONES, K.; C. MALIZA*, et al. Análisis, diseño e implementación de una aplicación multimedia interactiva para mostrar tiempos, distancias y rutas en un sistema de transporte masivo urbano utilizando herramientas de software libre y tecnología de web 2.0*, Escuela Superior Politécnica del Litoral, 2009. [Disponible en: http://www.dspace.espol.edu.ec/handle/123456789/7073
7. CABALLERO, E. V. *Estrategias para la implementación de sistemas de información geográfica del petróleo sobre la base de la Plataforma GeneSIG.*: Geoinformática. La Habana, Universidad de las Ciencias Informáticas, 2010. p.
8. CAMACHO, E.; F. CARDESO*, et al. Arquitecturas de Software, Guía de estudio*, 2004. [Disponible en: http://prof.usb.ve/lmendoza/Documentos/PS-6116/Guia%20Arquitectura%20v.2.pdf
9. CASTELL, L. *Procedimiento para la monitorización y el control de la Línea de Productos de Software Aplicativos SIG.*: Geoinformática. La Habana, Universidad de las Ciencias Informáticas, 2012. p.
10. CUESTA, S. R. *Patrones de Casos de Uso. SG Buzz.*, 2013. [Disponible en: http://sg.com.mx/content/view/510.
11. DOCTRINE. *Doctrine 2 ORM 2 documentation* 2015. [Disponible en: http://doctrine-orm.readthedocs.org/en/latest/tutorials/getting-started.html
12. DUHARTE, F. R. *Pruebas Unitarias de Software en la Plataforma J2EE*. Facultad 4. Ciudad de La Habana, Universidad de las Ciencia Informáticas, 2008. 120. p.
13. EDEKI, C. Agile Unified Process *INTERNATIONAL JOURNAL OF COMPUTER SCIENCE*, 2013, 1(3): 5.
14. EGUILUZ, J. P. Desarrollo web ágil con Symfony2, 2011.

15. ESPINOSA, S. G. and D. D. CUTIÑO. *Estrategia para la aplicación de Pruebas de Caja Blanca y Caja Negra al proyecto Registros y Notarías.* La Habana, Universidad de las Ciencias Informáticas, 2008. 112. p.
16. ESRI. *¿What is GIS?*, 2011.
17. FALGUERAS, B. C. *Ingeniería de Software*. 1. Barcelona, 2003. 323 p. 84-8318-997-6
18. FOUNDATION, A. S. *Overview of new features in Apache 2.2*, 2010.
19. GAMMA, E.; R. HELM*, et al. Design Patterns*. Sidney, 1997. 431 p.
20. GARCÍA, S. H. G. and I. A. SUÁREZ. *IMPLEMENTACIÓN DE LA RESTITUCIÓN FOTOGRAMÉTRICA PARA EL DESARROLLO DE LA CARTOGRAFÍA.*, 2002. [Disponible en: www.bibliociencias.cu/gsdl/collect/eventos/index/assoc/...dir/doc.doc
21. GIS.COM. *What is GIS?,*, http://www.gis.com/content/what-gis, 2011. [Disponible en: http://www.gis.com/content/what-gis
22. GLOSARIOIT. *CRUD - Sección BD/Programación*, 2014. [Disponible en: http://www.glosarioit.com/#!CRUD.
23. GÓMEZ, F. *Sistemas y procedimientos administrativos*, 2000.
24. GONZALEZ, V. H. and F. O. VALDIVIESO. *Infraestructura de datos espaciales (IDE) para el estudio y análisis ambiental: una experiencia en el sur de ecador*, XII Congreso de la Serie Española de Teledetección, 2007. [Disponible en: http://www.fronate.pro.ec/fronate/wp-content/media/2008/01/ide_ambiental_ecuador.PDF
25. JACOBSON, I.; G. BOOCH*, et al. El proceso unificado de desarrollo de software*. 1. Madrid, Félix Varela, 2003. 458 p. 84-7829-036-2
26. KRUEGER, C. W. *Introduction to Software Product Lines. BigLever Software.* , 2006. [Disponible en: http://www.softwareproductlines.com/ introduction/.
27. LARA, P.; L. FERNANDÉZ*, et al. Generación de casos de prueba a partir de especificaciones UML*. *Actas de la VII Jornadas de Innovación y Calidad del Software*, 2003. 11.
28. LEÓN, A. R. S.; M. G. RODRIGUEZ*, et al.* CRUD-PG *Revista Cubana de Ciencias Informáticas*, 2011, 5.
29. MACERO, D. P. M. and D. G. B. GUAMAN. *"ESTRATEGIA PARA IMPLEMENTAR UNA APLICACIÓN WEB A TRAVES DEL ESTUDIO DE PATRONES ARQUITECTONICOS. CASO PRACTICO: DEPARTAMENTO MEDICO ESPOCH"* FACULTAD DE INFORMATICA Y ELECTRONICA. RIOBAMBA – ECUADOR ESCUELA SUPERIOR POLITECNICA DE CHIMBORAZO, 2009. 290. p.
30. MARTÍN, M. M. *Manual de PostGIS*, 2009. 21.
31. MARTÍNEZ, R. *Sobre PostgrSQL*, 2013.

32. MINAYA, J. L. *Zrad*, 2013. [Disponible en: http://www.zend-rad.com/recursos
33. MONTILVA, J. A. Sistemas de Información Geográfica y Diseño de Geodatabases. , 1994.
34. MORENO, M. L. *Propuesta de arquitectura para la Línea de Productos de Software Aplicativos SIG*. Geoinformática. La Habana, Universidad de las Ciencias Informáticas, 2012. 81. p.
35. MORETA, O. R. E. *Diseño e implementación de Mapa Interactivo utilizando Web Mapping y Base de Datos Espacial: Ciudad de Quevedo*. Quito, Universidad de San Francisco de Quito, 2009. p.
36. OLAYA, V. Sistemas de Información Geográfica., 2010: p. 911.
37. P. CLEMENT AND L. NORTHRHOP Software Product Lines: Practices and Patterns Pittsburgh, EE.UU.. Carnegie Mellon University., 2002: 563 p.
38. PACHECO, G. A. A. *"Implementación de un Servidor de Mapas con los datos registrados del Inventario de Bienes Inmuebles y Arqueológicos de la provincia del Azuay, del Instituto Nacional de Patrimonio Cultural Regional 6"*. Extensión Maestría Internacional en SIG. Ecuador, UNIVERSIDAD SAN FRANCISCO DE QUITO, 2012. 176. p.
39. POTENCIER, F. *Los formularios de Symfony*. 2015. p.
40. PRESSMAN, R. Ingeniería del Software: Un enfoque práctico. . en. Sexta Edición. 2005. 927 p.
41. ---. Modelado del Análisis. Ingeniería de Software. en. 8VA. New York, 2008.p.
42. RABISER, R.; M. O.´LEARY*, et al.* Key activities for product derivation in software product lines. *Journal of Systems and Software*, 2011, 84(2): 300.
43. RODRIGUEZ, L. *LENGUAJE DE PROGRAMACION CLASIFICACIÓN DE LENGUAJES DE PROGRAMACIÓN POR NIVELES DE PROGRAMACION*, 2015. [Disponible en: http://www.academia.edu/10523359/LENGUAJE_DE_PROGRAMACION_CLASIFICACI%C3%93N_DE_LENGUAJES_DE_PROGRAMACI%C3%93N_POR_NIVELES_DE_PROGRAMACION
44. RODRÍGUEZ, R. V.; C. P. RISQUET*, et al. Visualización Espacio-Temporal Mediante Técnicas De Visualización De Datos Multiparamétricos En Sig X Congreso Cubano De Informatica Y Geociencias*. *GEOCIENCIAS*. La Habana, Sociedad Cubana de Geología, 2011.
45. SOMMERVILLE, I. *Ingeniería del Software.* . Séptima Madrid, pearson Education S.A., 2005. 712 p. 84-7829-074-5
46. TEAM, T. S. *Características de PHP*, 2011. [2015]. Disponible en: http://es.scribd.com/doc/50288837/Caracteristicas-de-PHP
47. VALERIO, M. S. L. ¿Qué son los sistemas de información geográfica? *Revista de Divulgación Científica y técnológica de la universidad veracruzana,*, 2005: p. XVIII.
48. WELLING, L. and L. THOMSON. *Desarrollo web con PHP y MySQL*. Madrid, 2005. 974 p. 84-415-1818-1

49. WYERS, G. J. *The Art of software Testing* 2da. Canada, John Wiley & Sons, 2004. 151 p. 0-471-46912-2
50. ZALDÍVAR, Y. P. *Modelo de desarrollo basado en líneas de productos de software para Sistemas de Información Geográfica sobre la base de la Plataforma GeneSIG.*: Facultad 6. La Habana, UCI, 2012. p.

# Generador automático de CRUD para la LPS Aplicativos SIG

## ANEXOS

Anexo 1: Código del método public function crearCrud($request).

**Tabla 10 Código del método crearCrud($request).**

| Nodo | Fragmento de código |
|---|---|
| 1 | `$path = CARTOWEB_PROY . "/plugins/";`<br>`if (file_exists($path . lcfirst($request->nombre))) {` |
| 2 | `echo "Ya existe el plugin " . $request->nombre;`<br>`die;`<br>`}` |
| 3 | `else {`<br>`try {`<br>`mkdir($path . lcfirst($request->nombre), 0777);`<br>`mkdir($path . lcfirst($request->nombre) . '/client', 0777);`<br>`/*`<br>`* creando el fichero client.php`<br>`* $PluginName es sustituido por el nombre del Plugin`<br>`* $AtrValorAdd es sustituido por los campos construidos del adicionar`<br>`*/`<br>`$cliente_nombre = ucwords($request->nombre);`<br>`$cliente_codigo = file_get_contents('/var/www/genesig/plugins/generadorCrud/Plantilla/client/ClientGest.php');`<br>`$case_acciones_cliente = $this->crearAccionesCliente($request->acciones, $request->campos);`<br>`$plantilla_cliente = str_replace('$PluginName', $cliente_nombre, $cliente_codigo);`<br>`$plantilla_cliente = str_replace('$Casos', $case_acciones_cliente, $plantilla_cliente);`<br>`file_put_contents($path . lcfirst($request->nombre) . '/client' . '/Client' . ucwords($request->nombre) . '.php', $plantilla_cliente);`<br>`/*`<br>`* creando el fichero php en common`<br>`* $PluginName es sustituido en la plantilla por el nombre del plugin`<br>`*/`<br>`mkdir($path . lcfirst($request->nombre) . '/common', 0777);`<br>`$common_nombre = ucwords($request->nombre);`<br>`$common_codigo = file_get_contents('/var/www/genesig/plugins/generadorCrud/Plantilla/common/Gestionar.php');`<br>`$plantilla_common = str_replace('$PluginName', $common_nombre, $common_codigo);`<br>`file_put_contents($path . lcfirst($request->nombre) . '/common' . '/' . $common_nombre . '.php', $plantilla_common);`<br>`/*` |

```
                    * creando los ficheros js
                    * el fichero htdocs q contiene los ficheros css,gfx,js

                   */
                   mkdir($path . lcfirst($request->nombre) . '/htdocs',
0777);
                   mkdir($path . lcfirst($request->nombre) . '/htdocs/css',
0777);
                   mkdir($path . lcfirst($request->nombre) . '/htdocs/gfx',
0777);
                   mkdir($path . lcfirst($request->nombre) . '/htdocs/js',
0777);
                   /*
                    * Construyendo el plugin ajax.js
                    * $PluginName sera sustituido por el nombre del plugin
                    * $PluginNamMin sera sustituido por el nombre del plugin
en minusculas
                    * */
                   $ajax_cod =
file_get_contents('/var/www/genesig/plugins/generadorCrud/Plantilla/htdocs
/js/plugin.ajax.js');
                   $plantilla_ajax = str_replace('$PluginName',
$common_nombre, $ajax_cod);
                   $plantilla_ajax = str_replace('$PluginNamMin',
strtolower($common_nombre), $plantilla_ajax);

                   file_put_contents($path . lcfirst($request->nombre) .
'/htdocs/js/plugin.ajax.js', $plantilla_ajax);
                   /*
                    * Creando el fichero handlers dentro del fichero js
                    *
                    */
                   mkdir($path . lcfirst($request->nombre) .
'/htdocs/js/handlers', 0777);
                   /*
                    * Creando el formulario.js que tiene la logica del
negocio
                    * $PluginName sera sustituido por el nombre del plugin
                    * $PluginNamMin sera sustituido por el nombre del plugin
en minusculas
                    * recordPlugin crea los valores que van dentro de la
funcion capturarDatos del formulario.js
                    */
                   $formulario_cod =
file_get_contents('/var/www/genesig/plugins/generadorCrud/Plantilla/htdocs
/js/handlers/formulario.js');
                   $ventana =
file_get_contents('/var/www/genesig/plugins/generadorCrud/Plantilla/htdocs
/js/handlers/VentanaGestionarPlug.js');
                   $sust_ventana = str_replace('$Plugin', ucwords($request-
>nombre), $ventana);
                   file_put_contents($path . lcfirst($request->nombre) .
'/htdocs/js/handlers/VentanaGestionar' . ucwords($request->nombre) .
'.js', $sust_ventana);
                   $plantilla_formulario = str_replace('$PluginName',
```

```
        $common_nombre, $formulario_cod);
                        $plantilla_formulario = str_replace('$PluginNamMin',
        strtolower($common nombre), $plantilla formulario);
                        $handler_capturar_datos = $this->handlerCapturar($request-
        >campos, strtolower($common_nombre));
                        $handler_cargar_datos = $this->handlerCargar($request-
        >campos, strtolower($common_nombre));
                        $handler clear datos = $this->handlerClear($request-
        >campos);
                        $plantilla_formulario = str_replace('$Atribute',
        $handler_capturar_datos, $plantilla_formulario);
                        $plantilla_formulario = str_replace('$Value',
        $handler_cargar_datos, $plantilla_formulario);
                        $plantilla_formulario = str_replace('$Clear',
        $handler_clear_datos, $plantilla_formulario);
                        file_put_contents($path . lcfirst($request->nombre) .
        '/htdocs/js/handlers/formulario.js', $plantilla_formulario);
                        /*
                         * Creando el fichero views que contiene la vista del
        sistema
                         */
                        mkdir($path . lcfirst($request->nombre) .
        '/htdocs/js/views', 0777);
                        /*
                         * Construyendo el formulario.ui.js q contiene los
        componentes de la vista
                         * $PluginName es sustituido por el nombre del plugin
                         * crearCampos es la funcion que construye los campos y
        devuelve un string con todos los campos
                         * $componentes es sustituido en la plantilla por los
        campos de la vista
                         * */
                        $ventanaui =
        file_get_contents('/var/www/genesig/plugins/generadorCrud/Plantilla/htdocs
        /js/views/VentanaGestionarPluginUI.js');
                        $sust_ventanaui = str_replace('$Plugin', ucwords($request-
        >nombre), $ventanaui);

                        $acciones_arreglo = (array)$request->acciones;
                        while (list($indice, $valor) = each($acciones_arreglo)) {
4                           $z = ($valor == -1) ? 'true' : 'false';
                            switch ($indice) {
5                               case 'adicionar':
                                    $sust ventanaui = str replace('$add', "$z",
        $sust_ventanaui);
                                    break;
6                               case 'modificar':
                                    $sust_ventanaui = str_replace('$mod', "$z",
        $sust_ventanaui);
                                    break;
7                               case 'eliminar':
                                    $sust_ventanaui = str_replace('$elim', "$z",
        $sust_ventanaui);
                                    break;
```

```
                        }
8               }
9       file_put_contents($path . lcfirst($request->nombre) .
        '/htdocs/js/views/VentanaGestionar' . ucwords($request->nombre) . 'UI.js',
        $sust_ventanaui);
                        $formulario_ui_codigo =
        file_get_contents('/var/www/genesig/plugins/generadorCrud/Plantilla/htdocs
        /js/views/formulario.ui.js');
                        $campos_formulario = $this->crearCampos($request->campos);
                        $plantilla_formularioui = str_replace('$PluginName',
        $common_nombre, $formulario_ui_codigo);
                        $plantilla_formularioui = str_replace('$componentes',
        $campos_formulario, $plantilla_formularioui);
                        file_put_contents($path . lcfirst($request->nombre) .
        '/htdocs/js/views/formulario.ui.js', $plantilla_formularioui);
                        /*
                         * Creando la carpeta server
                         */
                        mkdir($path . lcfirst($request->nombre) . '/server',
        0777);
                        /*
                         *Creando el Server.php del plugin
                         * $PluginName es sustituido por el nombre del plugin
                         */
                        $server_codigo =
        file_get_contents('/var/www/genesig/plugins/generadorCrud/Plantilla/server
        /ServerGestionar.php');
                        $contenido_adicionar_server = $this-
        >crearAdicionarServer($request->campos, $request->tabla, $request-
        >acciones);
                        $case_acciones_server = $this-
        >crearAccionesServer($request->acciones, ucwords($request->nombre),
        $request->acciones);
                        $contenido_modificar_server = $this-
        >crearModificarServer($request->campos, $request->tabla, $request-
        >acciones);
                        $contenido_eliminar_server = $this-
        >crearEliminarServer($request->tabla, $request->acciones);
                        $contenido_listar_server = $this-
        >crearListarServer($request->campos, $request->tabla);
                        $contenido_otras_funciones = $this-
        >crearOtrasFuncionesServer($request);
                        $case_acciones_otras_server = $this-
        >crearAccionesOtrasServer($request->acciones);
                        $plantilla_server = str_replace('$PluginName',
        $common_nombre, $server_codigo);
                        $plantilla_server = str_replace('$Add',
        $contenido_adicionar_server, $plantilla_server);
                        $plantilla_server = str_replace('$CasosPrincipales',
        $case_acciones_server, $plantilla_server);
                        $plantilla_server = str_replace('$CasosOtros',
        $case_acciones_otras_server, $plantilla_server);
```

```
                    $plantilla_server = str_replace('$Mod',
$contenido_modificar_server, $plantilla_server);
                    $plantilla server = str replace('$Elim',
$contenido_eliminar_server, $plantilla_server);
                    $plantilla_server = str_replace('$Listar ',
$contenido_listar_server, $plantilla_server);
                    $plantilla_server = str_replace('$Otras',
$contenido otras funciones, $plantilla server);
                    file_put_contents($path . lcfirst($request->nombre) .
'/server' . '/Server' . ucwords($request->nombre) . '.php',
$plantilla_server);
                    /*
                     *Creando los ficheros .ini del serverConf y del
clientConf
                     *
                     */
                    $ini =
file_get_contents("/var/www/genesig/plugins/generadorCrud/Plantilla/client
e/gestionar.ini");
                    $ini_nuevo = str_replace('$PlginUI', ucwords($request-
>nombre) . 'UI', $ini);
                    $ini_nuevo = str_replace('$Plugin', ucwords($request-
>nombre), $ini_nuevo);
                    file_put_contents(CARTOWEB_PROY . "/client_conf/" .
lcfirst($request->nombre) . ".ini", $ini_nuevo);
                    /*
                     * Cambiando el dsn del ini creado por el correspondiente
del plugin
                     */
                    $ini_server =
file_get_contents("/var/www/genesig/plugins/generadorCrud/Plantilla/servid
or/gestionar.ini");
                    $ini_server_nuevo = str_replace('$Q', $request->dsn,
$ini_server);
                    file_put_contents(CARTOWEB_PROY . "/server_conf/geoweb/" .
lcfirst($request->nombre) . ".ini", $ini_server_nuevo);
                    echo "Se ha creado el plugin " . $request->nombre . "
correctamente";
                    die;
10      } catch (Exception $e) {
                    echo 'ExcepciÃ³n capturada: ', $e->getMessage(), "\n";
                }

            }
11      }
```