Universidad de las ciencias Informáticas
Facultad de Ciencias y Tecnologías Computacionales

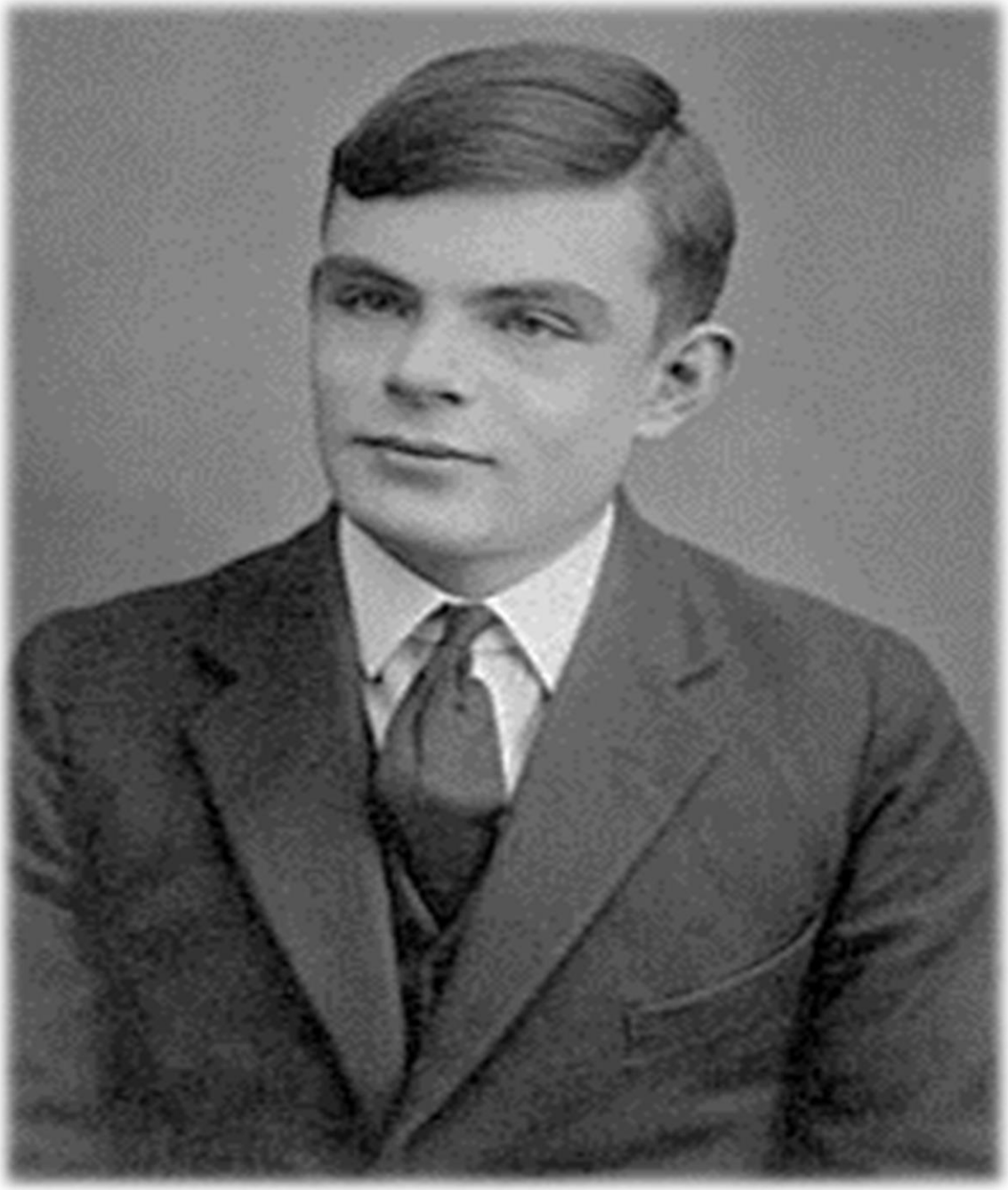
Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Título: “Módulo de edición de capas vectoriales para la plataforma ULTRON v. 2.0 del Centro de Representación y Análisis de Datos”

Autor(es):
Wilson Félix Cassuque Cambila

Tutor(es):
DrC. Odiel Estrada Molina

La Habana, junio del 2020
“Año 63 de la Revolución”



"Sólo podemos ver poco del futuro, pero lo suficiente para darnos cuenta de que hay mucho que hacer"

Alan Mathison Turing

DECLARACIONES

Declaro por este medio que yo: Wilson Félix Cassuque Cambila, ser el único autor de la presente tesis que tiene por título: “Módulo de edición de capas vectoriales para la plataforma ULTRON v 2.0” y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____	_____
Firma del Autor	Firma del Tutor
Wilson Félix Cassuque Cambila	DrC. Odiel Estrada Molina

RESUMEN

El proceso de trabajo para los sistemas de Información Geográfica (SIG), los módulos de edición son componentes sumamente importantes para entidades que aplican su negocio al desarrollo de software para lograr una mejor visualización de los mapas y análisis de datos espaciales. La presente investigación tiene como objetivo desarrollar el Módulo de edición de capas vectoriales para la plataforma ULTRON del Centro de Representación y Análisis de Datos (CREAD), que permita a los usuarios/clientes realizar la edición de las geometrías y sus atributos. Se realizó un estudio a un conjunto de sistemas homólogos que permitió determinar varios aspectos importantes a tener en cuenta en el desarrollo del módulo. Para la implementación de la solución el autor optó por la utilización de las tecnologías y lenguajes de programación: HTML5, CSS3, JavaScript (tanto del lado del cliente como del lado del servidor) y como framework de desarrollo SEAN.JS. Además, para regir el proceso de desarrollo se utilizó la metodología AUP-UCI escenario 4, adaptada a los procesos productivos de la universidad. Como resultado se obtuvo el desarrollo de la aplicación. Su funcionamiento fue evaluado mediante la aplicación de pruebas, proceso que culminó con la aceptación de los usuarios finales.

Palabras Claves: datos geográficos, edición de capas vectoriales, mapas, sistemas de información geográfica.

ABSTRACT

The work process for Geographic Information Systems (GIS), editing modules are extremely important components for entities that apply their business to software development to achieve better visualization of maps and analysis of spatial data. This research aims to develop the Vector layer editing module for the ULTRON platform of the Center for Data Representation and Analysis (CREAD), which allows trained users / clients to carry out a process of editing the geometries and their attributes. A study was carried o ut on a set of homologous systems that allowed determining several important aspects to take into account in the development of the module. For the implementation of the solution, the authors opted for the use of technologies and programming languages: HTML5, CSS3, JavaScript (both client-side and server-side) and as a development framework SEAN.JS. Furthermore, to govern the development process, the AUP-UCI scenario 4 methodology was used, adapted to the university's production processes. As a result, the development of the application was obtained. Its performance was evaluated through the application of tests, a process that culminated in the acceptance of end users.

Keywords: geographical data, vector layer editing, maps, geographic information systems.

ÍNDICE GENERAL

DECLARACIONES II

RESUMEN 1

ABSTRACT 2

INTRODUCCIÓN 6

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS ASOCIADOS A LA EDICIÓN DE CAPAS VECTORIALES EN LOS SISTEMAS DE INFORMACIÓN GEOGRÁFICAS 11

1.1 Introducción..... 11

1.2 Sistema de información Geográfica..... 11

1.3 Módulo de edición de capas vectoriales en sistema de información geográfica..... 15

Definición de que son Capas en los SIG: 15

1.4 Análisis de soluciones informáticas asociados en edición de capas vectoriales 20

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL MÓDULO DE EDICIÓN DE CAPAS VECTORIALES 35

2.1. Introducción. 35

2.2. Requisitos de Software 35

2.2.1. Requisitos Funcionales 35

2.2.2. Requisitos no funcionales..... 37

2.3. Descripción del módulo de edición de capas vectoriales 40

2.3.1. Descripción de los actores que interactúan con el sistema 40

2.3.2. Descripción de los requisitos funcionales 40

2.4. Patrón arquitectónico..... 46

2.5 Diagrama de clases del diseño 52

2.6. Patrones de diseño de software empleados en el desarrollo del módulo. 54

2.6.1 Patrones para la asignación de responsabilidades (GRASP). 54

2.6.2 Patrones GOF. 55
2.7. Modelo físico de la Base de Datos. 56
Conclusiones parciales. 57
CAPÍTULO 3: DISEÑO DE LA ESTRATEGIA DE PRUEBAS A REALIZAR PARA EL MÓDULO DE EDICIÓN DE CAPAS VECTORIALES PARA LA PLATAFORMA ULTRON V 2.0 59
3.1. Introducción 59
3.2 Estándares de codificación 59
3.3 Diagrama de despliegue. 60
3.4 Estrategia de pruebas. 62
3.4.1. Niveles de pruebas. 62
3.4.2 Tipos de pruebas. 63
3.4.3. Métodos de pruebas. 64
3.4.4. Técnicas de pruebas 64
3.5 Aplicación y resultados de las pruebas. 65
Visibilidad del sistema: 68
Diseño estético y minimalista: 68
3.6. Resultado obtenido 76
Conclusiones parciales. 77
CONCLUSIONES GENERALES 78
RECOMENDACIONES 79
REFERENCIAS BIBLIOGRÁFICAS 80
GLOSARIO DE TÉRMINOS 86
ANEXOS 88
Anexos 1: Entrevista. 88
Anexos 2: Descripciones de las Historias de Usuarios del módulo 88

INTRODUCCIÓN

Desde los años 1987 a 2019 la evolución de los Sistemas de Información Geográfica (SIG) se han desarrollado y adaptado de forma muy rápida y variada, adaptándose a una realidad, la de la propia información geográfica, también en constante evolución en todas sus vertientes.

"*Un Sistema de Información Geográfica (SIG) es un conjunto de componentes específicos que permiten a los usuarios finales crear consultas, integrar, analizar y representar de una forma eficiente cualquier tipo de información geográfica referenciada asociada a un territorio*." (Olaya, 2014)

La información geográfica es la "información que tiene algún componente espacial, es decir, una ubicación, y, además, una información atributiva que nos detalle más sobre ese elemento en cuestión". (Anasanz, 2016, p.5). En la edición de capas vectoriales se logra representar el mundo real mediante una proyección vertical sobre un plano bidimensional, simplificando las diferentes entidades para constituir lo que conocemos en la actualidad como mapa. Sin embargo, con la aparición de los primeros SIG estas proyecciones del mundo real se comienzan a almacenar como datos ráster o vectoriales en grandes bases de datos georreferenciadas para luego ser consultadas, editadas y analizadas permitiendo la digitalización de los mapas.

Las operaciones de edición pueden emplearse, por ejemplo, para la actualización de cartografía y la corrección de errores topológicos en la representación geográfica. A lo largo del desarrollo de un proyecto SIG, es muy probable que sea necesario editar de un modo u otro algún dato espacial, bien sea para corregirlo, ampliarlo, mejorarlo o sencillamente adaptarlo a las necesidades del propio proyecto.

Así, si una entidad en una capa vectorial modifica su geometría, no es necesario rehacer todo un mapa, sino simplemente editar ese elemento. En el libro Sistemas de Información Geográfica, Víctor Olaya expone que además de la modificación de una capa ya existente, las herramientas de edición de un SIG se emplean igualmente para la creación de capas nuevas, que pueden crearse a partir de la digitalización de imágenes o bien en base a cualquier otra capa de la que dispongamos. Aunque las tareas de edición más habituales son las relacionadas con la edición de geometrías, en este sentido se pueden distinguir las siguientes formas de edición:

- Edición de geometrías de una capa vectorial.
- Edición de atributos de una capa vectorial.

Sin ellas, los datos espaciales pierden gran parte de su utilidad dentro de un SIG, ya que se limitan las posibilidades de trabajo sobre estos. Las funcionalidades de edición son, por tanto, básicas en un SIG. Entre las funciones de edición están las que permiten facilitar algunas tareas, tales como la división de un polígono en dos simplemente trazando una línea divisoria. Otras funcionalidades similares incluyen la eliminación automática de polígonos o el ajuste automático entre entidades.

Durante el proceso de digitalización manual pueden cometerse errores con facilidad y su corrección, luego de una comparación detenida de los mapas originales y los digitalizados, sigue siendo un componente necesario del proceso de conversión de datos.

En la Universidad de las Ciencias Informáticas (UCI), en el Centro de Representación y Análisis de Datos (CREAD) se desarrolló la plataforma web para el desarrollo de Sistemas de Información Geográfica ULTRON (Ultimate Run Of Nodejs), la cual está desarrollada sobre tecnologías libres, haciendo uso del JavaScript. Integra las librerías de AngularJS, Express, corriendo sobre el servidor NodeJS.

Hasta la fecha dicha plataforma no cuenta con opciones o funcionalidades de edición de capas vectoriales, lo cual imposibilita la corrección de errores topológicos, resultantes de la digitalización manual, que pueden variar desde límites imprecisos, solapamiento de elementos, líneas no cerradas, entre otros. La corrección de errores topológicos favorece la exactitud y calidad de las capas vectoriales, ayudando en gran medida la capacidad de análisis y visualización en un SIG. Como consecuencia se pudieran generar resultados de análisis incorrectos y provocar atrasos en la producción que influirían en el desarrollo de un proyecto.

Dada la problemática expuesta anteriormente surge como **problema de investigación**: ¿Cómo garantizar que la plataforma ULTRON v.2.0 edite capas vectoriales?
El **objeto de estudio** de la investigación estará centrado en la edición de capas vectoriales en los SIG, con el **campo de acción** enfocado en la edición de capas vectoriales en la plataforma ULTRON.
Para darle solución al problema se tiene como **objetivo general**: Desarrollar un módulo de edición para la plataforma Ultron v.2.0.

Para guiar la investigación se elaboró la siguiente **hipótesis:** “Si se implementa un módulo empleando la plataforma SEAN-JS y la API OpenLayers entonces se posibilitará la edición de capas vectoriales en la plataforma ULTRON v 2.0”.

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

1. Elaboración del marco teórico–metodológico de la investigación referente a los procesos fundamentales en la edición de capas vectoriales.
2. Caracterización de las principales soluciones existentes en el ámbito nacional e internacional.
3. Selección de las tecnologías y herramientas necesarias para la implementación del módulo de capas vectoriales de la Plataforma Ultron 2.0
4. Elaboración de los artefactos asociados a la metodología de desarrollo de software escogida.
5. Implementación del módulo de edición de capas vectoriales para la plataforma Ultron 2.0.
6. Valoración de los resultados obtenidos en la aplicación de una estrategia de prueba al módulo de edición de capas implementado.
7. Validación y pruebas de la solución implementada.

Al culminar la realización de las tareas de la investigación definidas se esperan los **posibles resultados** siguientes:

- Módulo de Edición de capa vectorial para la plataforma ULTRON.
- La documentación técnica ingenieril asociada al desarrollo del módulo.

Para la investigación se utilizaron los siguientes **métodos de investigación**:

Métodos Teóricos

Permiten estudiar las características del objeto de investigación y facilitan la construcción de la hipótesis de investigación, contribuyendo al desarrollo de las teorías científicas y para su ejecución se apoyan en el proceso de análisis y síntesis. Dentro de estos se encuentran presente en esta investigación:

- **Histórico-Lógico:** Se utilizó para estudiar la trayectoria histórica y evolución de los productos de software existentes para la edición de capas vectoriales en los SIG.

- **Analítico-Sintético:** Posibilitó determinar la definición y las características de la edición de capas vectorial y de las funciones básicas de los SIG. Se empleó en la selección de las herramientas y tecnologías a utilizar durante el desarrollo del módulo. Además, contribuyó al estudio de las soluciones existentes asociadas al dominio del problema, generándose nuevos conocimientos respecto a las cualidades y funcionalidades que el módulo debe cumplir.
- **Modelación:** Se utilizó en la realización de los artefactos generados por la metodología de desarrollo.

Métodos Empíricos

Describen y explican las características fenomenológicas del objeto, representan un nivel de la investigación cuyo contenido procede de la experiencia y es sometido a cierta elaboración racional. Dentro de estos se encuentra:

- **Entrevista:** Este método se utiliza en la realización de entrevistas a los desarrolladores del proyecto ULTRON y Aplicativos SIG. Se aplica con el fin de obtener las principales deficiencias que dan origen al problema planteado.

Población: los 7 trabajadores involucrados en el desarrollo de la plataforma ULTRON.
Unidad de estudio: trabajador de la plataforma ULTRON.
Muestra: 4 trabajadores que representan el 60% (jefe de proyecto, Analista y 2 Programadores).
Técnica de muestreo: No probabilística.
Dentro de esta técnica el **Muestreo intencional**: debido a que el investigador escoge según su buen juicio a los integrantes de la muestra que son representativos o con probabilidad de dar mayor información.

Estructura del Documento

Capítulo.1 Fundamentos de la investigación, la solución informática asociados en edición de capas vectorial, metodología y tecnologías empleadas

Contiene la fundamentación teórica del tema a desarrollar. Se describen los conceptos fundamentales sobre la edición de capas vectorial en los SIG. Se exponen las herramientas, metodologías, lenguajes y plataforma a utilizar para la implementación del módulo.

Capítulo.2 Descripción de la solución propuesta

Se realiza la presentación de la solución propuesta del módulo y su funcionamiento, así como los requisitos tanto funcionales como no funcionales con los que debe cumplir el sistema y los artefactos generados por la metodología seleccionada.

Capítulo.3 Evaluación de la solución n propuesta

Se aborda todo el proceso de implementación de la solución propuesta, en función de los diagramas de componentes y despliegue y se valida la solución a partir de pruebas de caja negra realizadas al módulo de edición cartográfica.

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS ASOCIADOS A LA EDICIÓN DE CAPAS VECTORIALES EN LOS SISTEMAS DE INFORMACIÓN GEOGRÁFICAS

1.1 Introducción

En este capítulo se realiza un análisis teórico de los conceptos sobre la edición de capas vectoriales en los SIG. Se expone las herramientas, metodologías lenguajes y tecnología que se utilizarán para la implementación del módulo de edición de capas vectoriales en la plataforma Ultrón v.2.0 Se realiza una descripción del estado de arte de las tecnologías SIG.

1.2 Sistema de información geográfica

“un Sistema de Información Geográfica (SIG o GIS por las siglas inglesas – Geographic Information Systems), define “un conjunto de procedimientos con capacidad de construir modelos o representaciones del mundo real, a partir de datos geográficos de localización cierta y mensurable” (Chang, 2017).

Estos sistemas (SIG), utilizan herramientas de gran capacidad de administración de datos y procesamiento gráfico que logran capturar, almacenar, visualizar y analizar información georreferenciada, dando respuesta a las siguientes preguntas:

- ¿Qué hay en determinada localización?
- ¿Cuáles son los atributos de la misma (frecuencia, perímetro, área, volumen)?
- ¿Dónde se ubica A con relación a B?
- ¿Cuál es el camino más corto (menor resistencia o menor costo) sobre el terreno desde el punto A hasta el punto B?
- ¿Cuántas ocurrencias de un fenómeno determinado hay en el área de influencia del punto A?
- ¿Dónde se repite el fenómeno detectado en la localización mencionada?

¿Cuál es la distribución espacial de este fenómeno?

A su vez, (Olaya, 2014) expresa que: “un SIG es un sistema compuesto por cinco piezas fundamentales: “datos, tecnología, análisis, visualización y factor organizativo. Cada una de ellas

cumple un papel determinado dentro del sistema SIG, el cual se caracteriza fundamentalmente por su naturaleza integradora.”

El uso de este tipo de sistemas facilita la visualización de los datos obtenidos en un mapa con el fin de reflejar y relacionar fenómenos geográficos de cualquier tipo, desde mapas de carreteras hasta sistemas de identificación de parcelas agrícolas o de densidad de población. Además, permiten realizar las consultas y representar los resultados en entornos web y dispositivos móviles de un modo ágil e intuitivo, con el fin de resolver “problemas complejos de planificación y gestión, conformándose como un valioso apoyo en la toma de decisiones” (Sánchez, 2017).

Se puede concluir que SIG es un software específico que permite a los usuarios crear consultas interactivas, integrar, analizar y representar de una forma eficiente cualquier tipo de información geográfica referenciada asociada a un territorio.

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. La palabra estructura viene del latín struere, que significa construir, coincidir, y contiene la idea de una cosa organizada. Existen estructuras espaciales porque el espacio geográfico no está constituido por un conjunto de lugares únicos, que ocupan localizaciones aleatorias.

Se puede decir que una estructura espacial en SIG se define de cómo se representan los datos adaptando una mejor realidad del mundo real y pueden ser representados por línea, puntos o polígonos y por fotos.
“Los SIG a su vez son constituido por cinco **componentes** fundamentales que estos representan las **arquitecturas** de un Sistema de información geográfico, que son:

- Software
- Hardware
- Datos geográficos
- Personas (Recursos Humanos)
- Procesos”. (Anasanz, 2016)

Según este autor, estos componentes presentan diversas características que lo tipifican, entre ellas se encuentra:

- **Software** “Las aplicaciones proporcionan experiencias de usuario focalizadas para trabajar y hacer que los SIG estén disponibles para todos. Funcionan en cualquier dispositivo: en teléfonos móviles, tabletas, buscadores web y computadoras de escritorio”. (Espáriz, 2014)

  Para el correcto análisis e interpretación de la información geográfica es necesaria la participación de un software SIG que tenga la potencia y funcionalidad de trabajar con información de este tipo.

- **hardware** “permite la entrada y salida de información geográfico en diversos medios y formas”.
  Como es lógico, para poder utilizar algunos de los softwares anteriormente mencionados es necesario un ordenador o hardware. Dependiendo de las características de esta máquina, obtendremos un mayor o menor rendimiento a la hora de realizar nuestros análisis.

- **datos geográficos** “SIG integra diversos tipos de capas de datos que utilizan la ubicación espacial. La mayoría de los datos tienen un componente geográfico. Los datos SIG incluyen imágenes, atributos y mapas base vinculados a hojas de cálculo y tablas.”. (Cebrián, 1988)

  Esta información geográfica será el inicio de partida para empezar a trabajar con los SIG, los cuales nos permitirán analizarla y extraer toda la información posible para plasmarla en un mapa que nos ayude a la interpretación de esa información.

- **Personas** “Y es que el profesional SIG es un perfil muy cuestionado (y demandado) en los últimos años, ya que existen muchas tareas dentro de un análisis SIG, las cuales necesitan de uno o varios profesionales, incluso profesionales temáticos”.(Kim, 2017)

  Dentro de los perfiles SIG se encuentran dos perfiles fundamentales:

Técnico/Analista SIG. Profesional que se encarga de realizar análisis geográficos y obtener resultados acordes con la investigación o proyecto que se esté llevando a cabo".

- **Procesos** "conjunto o encadenamiento de fenómenos**,** asociados al ser humano o a la naturaleza, que se desarrollan en un periodo de tiempo finito o infinito y cuyas fases sucesivas suelen conducir hacia un fin específico" (Naharudin, 2016)

  "El entorno de la **arquitectura** (CAD) se sustenta en su base de capas, para representar las entidades. El ambiente de los sistemas de información geográfica (SIG) utilizan el lenguaje de marcado geográfico para modelar las suyas y tienden a evolucionar hacia modelos tridimensionales". (Ramírez, 2005. Page 30). Ejemplo de ello se observa en la Fig.1

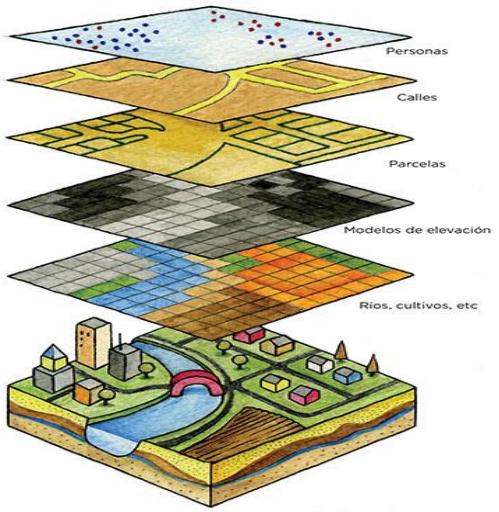


Fig. 1: capa de una arquitectura
Fuente: (Benítez, 2012)

Los SIG por tendencia están compuesto por cuatro módulos, asociados a edición de capas vectoriales; módulo de edición cartográfica; módulos de edición de capas de impresión, módulo de edición de errores topológico, selección de áreas de interfaz y módulo de edición de capas vectoriales.

Uno de los más necesarios en los SIG es el de edición de capas vectoriales, pues este modelo puede brindar funcionalidades importantes para la edición de capas, ya que los mapas son representados por capas: puntos, líneas y polígonos.

Se puede concluir que el uso de este tipo de sistemas (SIG) facilita la visualización de los datos obtenidos en un mapa con el fin de reflejar y relacionar fenómenos geográficos de cualquier tipo, desde mapas de carreteras hasta sistemas de identificación de parcelas agrícolas o de densidad de población.

1.3 Módulo de edición de capas vectoriales en sistema de información geográfica

En este epígrafe se estará abordando de manera general conceptos sobre el módulo de edición de capas vectoriales, como están representados los objetos espaciales, tipos de capas que suelen ser representados en los SIG, funcionalidades más comunes que llevan en los modelos de capas vectoriales y funcionalidades que estarán implementadas en este módulo que se estará desarrollando (modelo de edición de capas vectoriales).

Definición de que son Capas en los SIG:

"*Las capas u los datos geográficos de los SIG representan colecciones lógicas de entidades individuales con sus formas y ubicaciones geográficas, además de información descriptiva sobre cada entidad almacenada como atributos*. ". (Sernanp, 2015). Al igual que un mapa, un Sistema de información geográfica es un sistema fundamentalmente basado en capas.

En los Sistemas de Informaciones Geográficos las capas son elementos primordiales pues los datos geográficos son representados por capas, y estos datos pueden ser representada por los "objetos espaciales" que nos brindan una mejor información de lo que viene ser el mundo real en los mapas dentro de los SIG. Y a su vez estas capas pueden clasificarse en dos tipos principales de formatos de datos con los que se trabaja en los SIG la primera clasificación está asociada a capas ráster y la segunda está asociada a capas vectoriales. Los datos ráster son cuadrículas de celdas o píxeles. Los datos vectoriales son polígonos que usan puntos

(llamados nodos) y líneas. Los formatos vectoriales son útiles para almacenar datos con fronteras firmes, como distritos escolares o calles. (Evers, 2017)

Sobre el modelo vectorial (Moreno López, 2015) enunció:

La información se distribuye en tablas dentro de la Base de Datos correspondiéndose con una geometría vectorial que definen a los datos geográficos mediante punto, línea y polígono. Estas contienen una serie de propiedades que la definen, denominados atributos. Los atributos del modelo y sus propiedades se distribuyen en tablas diferenciadas para poder establecer las relaciones entre ellas, constituyendo el modelo propio de la base de datos (Fig. 2).

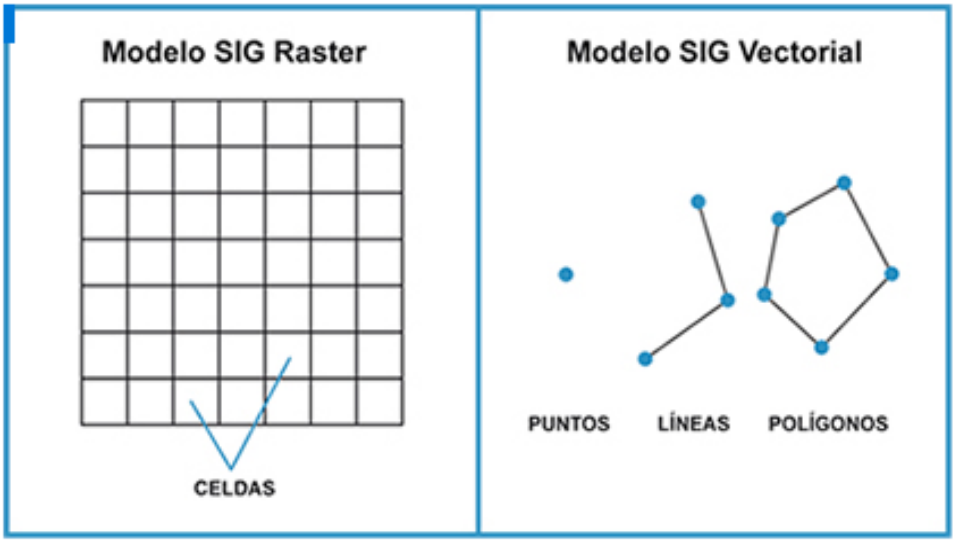


Fig. 2: Modelo Ráster y Modelo Vectorial
Fuente: (Mapa Satélite, 2010)

"La elección de un modelo u otro dependerá de si las propiedades topológicas son importantes para el análisis. Sí es así, el modelo de datos vectorial es la mejor opción, pero su estructura de datos, aunque muy precisa, es mucho más compleja y esto puede ralentizar el proceso. Por ello, si el análisis que nos interesa no requiere acudir a las propiedades topológicas, es mucho más rápido, sencillo y eficaz el uso del formato ráster". (Athan, et al., 2014)

Para la implementación u desarrollo de este módulo se estará usando el dato Vectorial (Capa vectoriales). En las capas de datos vectoriales se visualizan atributos tales como:

- **Objeto espacial**

  “proporciona una manera de representar del mundo real dentro de un ambiente SIG. Un objeto espacial es algo que puede ver en el paisaje. Imagine que está en lo alto de una colina. Mirando abajo, puede ver casas, carreteras, arboles, ríos etc.”. (Olaya, 2014)

  Objeto espacial es una variable asociada a una localización del espacio. Normalmente se utilizan datos vectoriales, los cuales pueden ser expresados mediante tres tipos de objetos espaciales: punto, línea y polígonos.

  Cada una de estas cosas sería un objeto espacial cuando los representamos en una aplicación SIG. Los objetos espaciales vectoriales tienen atributos, que consiste en texto o información numérica que describe los objetos espaciales.

- **Punto**

  “Se encuentran determinados por las coordenadas terrestres medidas por latitud y longitud. Por ejemplo, ciudades, accidentes geográficos puntuales, hitos” (Olaya, 2014). Lo primero que tenemos que darnos cuenta cuando hablamos de objetos espaciales puntuales es que lo que describimos como un punto en el SIG es una cuestión de opinión, y a menudo depende de la escala.

  Elegir utilizar los puntos para representar un objeto espacial es sobre todo una cuestión de escala (a qué distancia se encuentran del objeto), comodidad (se tarda menos tiempo y esfuerzo para crear entidades de puntos que las entidades poligonales), y el tipo de objeto espacial (algunas cosas como postes de teléfono simplemente no tienen sentido almacenarlas como polígonos).

- **Línea/Polilínea**

  Cuando un elemento punto es un sólo vértice, una polilínea tiene dos o más vértices

"La polilínea/línea es una ruta continúa trazada a través de cada vértice, cuando dos vértices están unidos, una línea es creada, cuando más de dos están unidos, forman una 'línea de líneas' o polilínea" (Olaya, 2014)

A veces tenemos reglas especiales para polilíneas, además de su geometría básica. Por ejemplo, las curvas de nivel pueden tocar (por ejemplo, en un acantilado), pero nunca deben cruzar entre sí.

Del mismo modo, polilíneas utilizadas para almacenar una red de carreteras deben conectarse en las intersecciones. En algunas aplicaciones SIG se pueden establecer estas reglas especiales para un tipo de entidad (por ejemplo, carreteras) y el SIG se asegurará de que estas polilíneas siempre cumplan con estas normas.

Si una polilínea curva tiene distancias muy grandes entre vértices, puede parecer angular o irregular, dependiendo de la escala.

Debido a esto, es importante que las polilíneas se digitalicen (capturado en la computadora) con distancias entre vértices que sean lo suficientemente pequeña para la escala en la que desea utilizar los datos.

- **Polígonos**

  "*Las entidades poligonales son zonas cerradas como presas, islas, límites del país, etc., como las entidades de polilínea, los polígonos se crean de una serie de vértices que se conectan con una línea continua*." (Whitacre, 2017). Sin embargo, debido a que un polígono siempre describe un área cerrada, el primer y último vértice deberán siempre estar en el mismo lugar.

  Los polígonos regularmente tienen geometría compartida limites que tienen en común con un polígono vecino. Muchas aplicaciones SIG tienen la capacidad para asegurar que los límites de los polígonos vecinos coinciden exactamente. De entre las gran principales ventajas de los modelos vectoriales, (Olaya, 2018) define que:

- “Gran capacidad de compactar información utilizando el menor volumen de datos posible del SIG.

- En cuanto a la precisión de ambos ráster y vectoriales modelos, los archivos vectoriales son más precisos que los ráster cuando se calculan superficies y distancias.

- Los modelos vectoriales permiten tener límites más precisos al tratarse de líneas y puntos de fácil definición y distribución, favoreciendo las relaciones de vecindad entre elementos y haciendo a estos archivos los más óptimos cuando se quiere realizar un análisis entre unidades espaciales.

- Por el contrario, los modelos ráster, presentan límites basados en el propio tamaño de píxel y tienen ciertas dificultades para desarrollar análisis espaciales”.

Varios artículos, documentos y revistas publicadas brindan informaciones importantes relacionadas al tema de módulo de edición de capas vectoriales en los SIG de aplicaciones ya existente. Las misma ofrecen ciertas funcionalidades asociadas al módulo de edición de capas vectoriales. Las principales funcionalidades asociada a la edición de capas vectoriales son:

- crear vértice
- mover vértice
- borrar vértice
- Crear objetos espaciales
- Dividir objetos espaciales
- Modificar relleno del vértice

Algunas funcionalidades principales asociadas a módulo de edición de capas vectoriales son:

- Añadir objetos espaciales
- Copiar Objetos espaciales
- Cortar objetos espaciales
- Borrar objetos espaciales

- Pegar objetos espaciales
- Guardar objetos espaciales
- Rotar objetos espaciales
- Estructura de vértices
- Selección de vértices
- Dividir objetos espaciales
- Rotar símbolos de ponto
- Añadir vértices
- Borrar vértices
- Mover vértices
- Guardar vértices
- Dibujar polígono
- Mostrar los vértices
- Dividir polígono en dos partes

1.4 Análisis de soluciones informáticas asociados en edición de capas vectoriales

GENESIG: "*es una aplicación informática desarrollada en conjunto con las empresas GEOCUBA y XETID orientada a la creación y personalización de Sistemas de Información Geográfica (SIG) para la toma de decisiones a partir de la representación y el análisis de información sobre mapas. Soportada en estándares internacionales, la plataforma soberana GENESIG puede ser personalizada a cualquier entorno de la sociedad a partir de las necesidades más exigentes en un corto período de tiempo*". (Lastres Romero, et al., 2013)

Esta plataforma cuenta con un módulo de edición el cual tiene el propósito de realizar la edición de geometrías en la plataforma GeneSIG. Incluye mecanismos para la edición de entidades gráficas basándose en el cambio de color, posición, escala, dibujo de nuevas entidades gráficas, entre otros. Se incluyen además mecanismos para la edición de datos descriptivos basándose en la modificación de atributos, actualización de datos, generación de nuevos datos, entre otros. Este permite realizar la selección, movimiento, dibujo, localización, combinación y eliminación de geometrías u objetos espaciales

GENESIG es una plataforma de SIG y su módulo de edición de capas vectoriales brinda ciertas funcionalidades: dibujar un polígono, autos amblado, mostrar los vértices, dividir polígono en dos partes, combinar geometría. (Cervera, 2015)

ArcGIS: es un completo sistema que permite recopilar, organizar, administrar, analizar, compartir y distribuir información geográfica. Es la plataforma líder mundial para crear y utilizar Sistemas de Información Geográfica (SIG), ArcGIS es utilizada por personas de todo el mundo para poner el conocimiento geográfico al servicio de los sectores del gobierno, la empresa, la ciencia, la educación y los medios.

"*ArcGIS se debe concebir como una plataforma completa en la que cualquier persona puede trabajar con información geográfica y aplicarla. La mayoría de las personas utilizan mapas para trabajar con información geográfica, pero no solo mapas impresos, sino mapas en línea interactivos que permiten comprender la información de su organización, las herramientas de análisis, tareas y flujos de trabajo, las cuales las personas de su organización utilicen para trabajar en forma más eficiente*". (Arenas Quiñones, et al., 2017)

La familia de aplicaciones de ArcGIS de escritorio facilita un conjunto amplio de funcionalidades de edición de capas vectoriales que pueden ser de importante uso para el desarrollo del módulo tales como: añadir vértices, crear polígonos, edición de puntos, edición de vértices, y otras opciones de edición.

gvSIG Desktop: "Es el primer programa informático desarrollado dentro del proyecto de desarrollo de software para Sistemas de Información Geográfica basado en software libre (gvSIG). Este proyecto fue inicialmente impulsado por el gobierno regional de la Comunidad Valenciana de España, dentro de un proceso de migración a software libre de todos los sistemas informáticos de la organización; precisamente la sigla gvSIG abrevia la denominación **G**eneralitat **V**alenciana **S**istema de **I**nformación **G**eográfica. Se utiliza para el manejo de información geográfica con precisión de edición de capas vectoriales que se distribuye bajo licencia GNU GPL v3. Permite acceder a información vectorial y ráster, así como a servidores de mapas que cumplan las especificaciones del OGC". (Álvaro, 2016)

La aplicación gvSIG Desktop brinda funcionalidades de vital importancia para el desarrollo del módulo tales como: añadir vértices, crear polígonos, edición de puntos, edición de vértices

Sin embargo, es una aplicación desarrollada para entorno de escritorio y además aplicaciones como esta y otros programas de SIG escritos en Java se promocionan como “alternativas libres” y no obstante requieren de bibliotecas Java privativas como JAI (Java Advanced Imaging API) para funcionar debido a que por muy libre que sea cada una de las líneas de código que forman ese software en sí.

Mapinfo: MAPINFO. Profesional es una potente herramienta de Sistemas de Información Geográfica que le permite realizar diversos y complejos análisis geográficos ideales para facilitar la toma de decisiones: Captura, Consulta, Edición, Análisis y Reportes de Información Geográfica dinámicamente relacionada con Bases de Datos. Es la solución fundamental de MapInfo para gestión de mapas desde puesto de trabajo, aclamada como el más amigable y potente software de gestión de mapas basado en PC. Es la elección de analistas de negocio y profesionales de GIS en todo el mundo para visualizar y analizar las relaciones entre datos y geografía. Algunas características clave serían perfecta conectividad con bases de datos relacionales, representación de mapas 3D y herramientas de listado, un creador de informes integrado, ricas funciones de representación temática y múltiples opciones de publicación. (Alvaro, 2016)

Mapinfo brinda funcionalidades importantes para la edición de capas vectoriales de entre ellas las más comunes son: autos amblado, mostrar los vértices, dividir polígono en dos partes.

QGIS: “QGIS (anteriormente llamado también Quantum GIS) es un Sistema de Información Geográfica (SIG) de software libre para plataformas GNU/Linux, Unix, Mac OS, Microsoft Windows y Android.2 Era uno de los primeros ocho proyectos de la Fundación OSGeo y en 2008 oficialmente graduó de la fase de incubación. Permite manejar formatos ráster y vectoriales a través de la biblioteca GDAL (GADL/OGR), así como bases de datos. (3.10.0 3.4.13 LTR). QGIS brinda funcionalidades de vital importancia a lo que di respeto la edición de capas vectoriales. Las capas editadas se pueden guardar,

imprimir y exportar. También permite visualizar bases de datos, informes, gráficos e imágenes distribuido sobre un eje de coordenadas geográficas. El amplio número de herramientas y los diferentes procesos de trabajo con que cuenta permiten generar y mantener la información geográfica de manera sencilla y eficiente" (Sernanp, 2015)

QGIS es una plataforma muy usada y potente su módulo de edición de capas vectoriales brinda las siguientes funcionalidades:

- Selección de vértices
- Dividir objetos espaciales
- Rotar símbolos de punto
- Añadir vértices
- Borrar vértices
- Mover vértices
- Guardar vértices
- deshacer/rehacer
- copiar
- rotar
- escalar
- desplazar
- editar vértice
- polígono interno
- autocompletar polígono
- insertar punto
- polígono

Al realizar un análisis de SIG existentes se reafirma lo expuesto en el epígrafe 1.2 defendiéndose que las funcionalidades más empleadas asociadas a la edición de capas vectoriales son:

- Añadir vértices
- Borrar vértice
- Mover vértices
- Selección de vértices

- Crear objetos espaciales
- Listar atributos de los objetos espaciales
- Modificar atributos de los objetos espaciales
- Eliminar objetos espaciales
- Dibujar punto
- Dibujar línea
- Dibujar polígono
- Dibujar polígono regular
- Seleccionar geometría
- Rotar geometría
- Trasladar geometría
- Escalar geometría
- Transformar múltiples geometrías
- Compensar geometría
- Duplicar geometría
- Dividir línea
- Dibujar agujero en un polígono
- Copiar geometría
- Cortar geometría
- Pegar geometría

Por tal motivo se asume que en esta investigación se tendrán estas para sus implementaciones, para cumplir el objetivo de alcanzar un mejor trabajo investigativo de módulo de edición de capas vectoriales para los SIG.

Realizado el estudio y el análisis a las soluciones existentes abordadas anteriormente se concluye que las mismas no posibilitan una solución factible para el problema planteado, debido que estas están desarrolladas para entorno de escritorio y poseen un sistema público (gvSIG y ArcGIS), por lo que presentan imposibilidad de copias, de modificación y redistribución. Además de forma general las soluciones analizadas al estar desarrolladas con otras tecnologías su adquisición

consumiría tiempo y recursos monetarios al ser necesario su modificación para adaptarlas a las necesidades que se desean en el desarrollo del módulo.

1.5 Herramientas, Tecnologías y Metodología de Ingeniaría de software para implementar el módulo de edición de capa en la plataforma ULTRON v.2.0

Metodología de desarrollo AUP-UCI.

"*Una metodología de desarrollo de software, es aquella que hace posible la planificación, organización y construcción de un sistema o proyecto, con independencia de su temática o complejidad. Actualmente estas metodologías son una guía en el proceso de desarrollo de las aplicaciones informáticas, permitiendo que se obtengan resultados con la mayor calidad, rapidez y eficiencia posible, para evitar cometer errores futuros*". (Pressman, 2017)

En el desarrollo del módulo se decide emplear como metodología AUP-UCI escenario 4 (Proceso Unificado Ágil, por sus siglas en inglés) en su variante UCI ya que es la utilizada en la Línea de productos de software Aplicativos SIG, del Centro de Representación y Análisis de Datos (CREAD) en el desarrollo de la plataforma web ULTRON 2.0. En este sentido, se da continuidad a la organización ingenieril del proyecto productivo.

"Actualmente estas metodologías se basan en una combinación de los modelos de proceso genéricos que definen artefactos, roles y actividades, junto con prácticas y técnicas recomendadas para guiar el proceso de desarrollo de un software. También se define una metodología de desarrollo de software como un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito" (Maida, et al., 2015)

Herramientas:

De entre las herramientas a ser usado para el desarrollo del sistema, se han utilizado las siguientes herramientas:

PostgreSQL v.9.4 y PostGIS v.2.1: "*PostgreSQL es un sistema gestor de bases de datos relacionales, está orientado a objetos, es multiplataforma y open source. Está desarrollado desde*

1996 por la comunidad partir del SGBD POSGRES, que surgió a partir de un proyecto de investigación militar estadounidense con participación civil". (Zea Ordóñez, et al., 2017).

PostgreSQL, llamado Sistema Gestor de Bases de Datos Relacionales Orientadas a Objetos, es usado para manejar grandes cantidades de información, y está basado en el modelo relacional, aunque incorpora conceptos del modelado orientado a objetos. Se realizó la selección de PostgreSQL como SGBD ya que el mismo es el que se utiliza en el centro de desarrollo y además presenta ventajas tales como:

- Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.
- Fácil de administración, estable y de gran flexibilidad.

Y **PostGIS** es un software compatible con Open Geoespacial Consortium (OGC) utilizado como una extensión para PostgreSQL, que es una forma de base de datos objeto-relacional PostGIS amplía las capacidades de PostgreSQL a fin de aumentar sus capacidades de gestión mediante la adición de tipos y funciones geoespaciales para mejorar los datos espaciales manejados dentro de una estructura de base de datos relacional. El lenguaje de PostGIS es similar al SQL y permite realizar análisis espaciales y consultas típicas sobre datos espaciales con relativa facilidad. (LLasac Huaca, et al., 2015).

Teniendo en cuenta las descripciones anteriormente mencionadas el motivo por la cual se decidió llevar a cabo el uso de estas herramientas en el módulo son uno de muchos factores positivo que favorecen, como por ejemplo su seguridad que nos brinda estos SGBD. Aunque PostGIS es libre y de código abierto, se utiliza tanto en software comercial (por ejemplo, ArcGIS) como en software de código abierto (por ejemplo, QGIS). Además, la plataforma ULTRON es gestionada en base estos SGBD.

PgAdimin v.1.20.0: Es una aplicación de diseño y manejo de bases de datos para su uso con PostgreSQL. La aplicación se puede utilizar para manejar PostgreSQL 7.3 y superiores y funciona sobre casi todas las plataformas. Este software fue diseñado para responder a las necesidades de todos los usuarios, desde la escritura de simples consultas SQL a la elaboración de bases de datos complejas. (Crestaz, et al., 2015).

Se opta por la utilización de esta herramienta ya que posee una interfaz gráfica que soporta todas las características del SGBD PostgreSQL y hace simple la administración de dicho gestor. Además, es multiplataforma y su licencia es fácil de adquirir. También es esta herramienta usada por el proyecto para el trabajo con el sistema gestor de base de datos.

Visual Paradign v.8.0: Traducción del inglés-Visual Paradign es una herramienta UML CASE que admite UML 2, SysML y notación de modelado de procesos empresariales del grupo de gestión de objetos. Además del soporte de modelado, proporciona capacidades de generación de informes e ingeniería de código, incluida la generación de código.

Visual Paradigm es una herramienta CASE, que utiliza el lenguaje de modelado UML; la misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software y fue diseñada para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos

Se optó por seleccionar como herramienta CASE Visual Paradigm ya que esta es la herramienta que se utiliza en el centro de desarrollo para realizar el modelado y presenta varias ventajas que facilitan el trabajo para los desarrolladores las cuales se describen a continuación:

- Entorno de creación de diagramas para UML 2.1.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Soporta patrones de diseño.
- Soporta las tres formas del Diagrama Entidad-Relación: conceptual, lógica y física.
- Mejora la trazabilidad de elementos utilizando el historial de revisiones.

- Soporta líneas anidadas.
- Muestra como elemento estereotipado del modelo un ícono de imagen.
- Muestra y oculta elementos del diagrama según se desee.

UML v.2.0: UML son las siglas de “Unified Modeling Language” o “Lenguaje Unificado de Modelado”. Se trata de un estándar que se ha adoptado a nivel internacional por numerosos organismos y empresas para crear esquemas, diagramas y documentación relativa a los desarrollos de software (programas informáticos).

“*UML es una herramienta propia de personas que tienen conocimientos relativamente avanzados de programación y es frecuentemente usada por analistas funcionales (aquellos que definen qué debe hacer un programa sin entrar a escribir el código) y analistas-programadores (aquellos que, dado un problema, lo estudian y escriben el código informático para resolverlo en un lenguaje como Java, C#, Python o cualquier otro)”.* (Visual Paradigm, 2015)

Lenguaje de Modelado Unificado (UML por sus siglas en ingles) se define como un “lenguaje cuyo vocabulario y reglas se centran en la representación conceptual y física de un sistema” (UML, 2016). El mismo permite visualizar, especificar, construir, documentar y modelar los artefactos del módulo que el proyecto necesita. Se opta por usar UML ya que la metodología utilizada lo exige y porque ofrece una serie de ventajas según (Meliá, 2008) las cuales se enuncian a continuación:

- Dispone de una semántica y sintaxis precisa, permitiendo conocer de forma no ambigua lo que el diagrama indica.
- Tiene una alta capacidad expresiva lo que posibilita representar todos los aspectos de la arquitectura web.
- Los modelos pueden ser intercambiables entre diferentes herramientas que tengan soporte de UML, así los modelos pueden ser reutilizados.

Open Layers v.4.6.4: OpenLayers es una biblioteca de JavaScript de código abierto bajo una derivación de la licencia BSD para mostrar mapas interactivos en los navegadores web.

OpenLayers ofrece un API para acceder a diferentes fuentes de información cartográfica en la red: Web Map Services, Mapas comerciales (tipo Google Maps, Bing, Yahoo), Web Features Services, distintos formatos vectoriales, mapas de OpenStreetMap, etc. (LLasac Huaca, et al., 2015)

Lo que se quiere alcanzar con esta herramienta es una mejor manera de representar en los navegadores una mejor visión y ajuste de mapa, para la edición de capas y que tenga una mejor adaptación a lo se utiliza actualmente en los SIG.

Tecnología:

De entre las tecnologías a ser usado para el desarrollo del sistema, se han utilizado las siguientes tecnologías:

Tecnología lado para el cliente (la interfaz):

Bootstrap: Bootstrap es una excelente herramienta para crear interfaces de usuario limpias y totalmente adaptables a todo tipo de dispositivos y pantallas, sea cual sea su tamaño. Además, Bootstrap ofrece las herramientas necesarias para crear cualquier tipo de sitio web utilizando los estilos y elementos de sus librerías.

Basándose en las descripciones anteriormente mencionadas se decidió llevar a cabo el uso de esta tecnología en el módulo pues es fácil utilizar. Se trata de una herramienta muy sencilla de utilizar a cualquier programador y que permite crear grandes sitios web en poco tiempo.

AngularJS v 8.0: "*AngularJS es JavaScript. Es un proyecto de código abierto, realizado en JavaScript que contiene un conjunto de librerías útiles para el desarrollo de aplicaciones web y propone una serie de patrones de diseño para llevarlas a cabo. En pocas palabras, es lo que se conoce como un framework para el desarrollo, en este caso sobre el lenguaje JavaScript con programación del lado del cliente*". (Dorta, 2016)

"Angular es un framework de JavaScript de código abierto y totalmente libre, que permite el desarrollo de aplicaciones web en el lado del cliente y utiliza un patrón Modelo-Vista-Controlador

(MVC), aunque no lo hace de una forma tradicional: más bien es un patrón de desarrollo del tipo Modelo-Vista-Modelo de Vista.”. (Dorta, 2016)

Basándose en las descripciones anteriormente mencionadas se decidió llevar a cabo el uso de esta tecnología en el módulo pues AngularJS permite extender el vocabulario HTML con directivas y atributos, manteniendo la semántica y sin necesidad de emplear librerías externas como jQuery o Underscore.js para que funcione.

HTML 5: “*HTML es el lenguaje con el que se define el contenido de las páginas web. Básicamente se trata de un conjunto de etiquetas que sirven para definir el texto y otros elementos que compondrán una página web, como imágenes, listas, vídeos, etc*.”. (Garro, 2015)

HTML5 (Hipertexto Markup Language, versión 5) es la quinta versión del lenguaje HTML. Esta nueva versión (aún en desarrollo), y en conjunto con CSS3, define los nuevos estándares de desarrollo web, rediseñando el código para resolver problemas y actualizándolo así a nuevas necesidades También incorpora etiquetas para manejar grandes conjuntos de datos: Datagrid, Details, Menu y Command, las que permiten generar tablas dinámicas que pueden filtrar, ordenar y ocultar contenido en cliente.

“Además, realiza mejoras en los formularios, con nuevos tipos de datos (email, number, url, datetime) y facilidades para validar el contenido sin JavaScript. También posee Drag & Drop, que no es más que una nueva funcionalidad para arrastrar objetos como imágenes; y los visores: MathML (fórmulas matemáticas) y SVG (gráficos vectoriales), que de forma general se deja abierto a poder interpretar otros lenguajes XML. Esta versión establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos; por lo que en la presente investigación se hace uso del mismo”. (Garro, 2015)
Características de este lenguaje.

- No es necesario ningún programa especial para crear una página Web. Gracias a ello se ha conseguido que se puedan crear páginas con cualquier ordenador y sistema operativo.
- Es un lenguaje descriptivo.
- Describe hipertexto, texto de forma estructurada y agradable.

- Permite inserciones multimedia.

CSS 3:

“Es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML. Permite hacer aplicaciones webs con mayor separación entre estilos y contenidos. Brinda soporte a muchas necesidades de las webs actuales, sin tener que recurrir a trucos de diseñadores o lenguajes de programación. Trae consigo muchas novedades altamente esperadas, como las esquinas redondeadas, sombras, gradientes, transiciones o animaciones, y nuevos layouts como multi-columnas, cajas flexibles o maquetas de diseño en cuadrícula (grid layouts)”. (Centeno Defas , y otros, 2016).

Jscript: “*JScript es la implementación de Microsoft de la especificación de lenguaje ECMA 262 (ECMAScript Edition 3). Aparte de algunas excepciones de poca importancia (para mantener la compatibilidad con versiones anteriores), JScript es una implementación completa del estándar ECMA. Esta introducción pretende ayudarle a comenzar a trabajar con JScript*”. (Centeno Defas , y otros, 2016)

“Es un lenguaje de programación muy extendido y que cada vez cobra más importancia en el ámbito de la informática. Es un lenguaje de programación de propósito general, concurrente, basado en clases y orientado a objetos” (Lastres Romero, y otros, 2013).

Se seleccionó JavaScript como lenguaje de programación tanto del lado del cliente como del lado del servidor ya que es un lenguaje bastante utilizado en centro de desarrollo y permite el desarrollo de aplicaciones bajo el esquema de Cliente-Servidor; además es sencillo, fácil de aprender, rápido y potente, ideal para implementar pequeñas funciones dentro de la página web.

Para el Servidor:

NodeJS v.10.15.0: “*Node.js es un entorno de tiempo de ejecución de JavaScript (de ahí su terminación en .js haciendo alusión al lenguaje JavaScript)*. *También aporta muchos beneficios y soluciona muchísimos problemas, por lo que sería más que interesante realizar nuestro curso de*

Node.js para obtener las bases, conceptos y habilidades necesarias que nos motiven a profundizar en sus opciones e iniciar la programación". (Kiessling, 2015).

Se puede afirmar por tanto que NodeJS encaja de manera excelente si se quiere:

- Interfaces ligeros REST/JSON: el modelo de Entrada/Salida no bloqueante, para atender las peticiones REST, en combinación con JavaScript, para el soporte nativo JSON, lo hacen óptimo como capa superior de fuentes de datos como bases de datos u otros servicios web.
- Aplicaciones mono página: las aplicaciones mono página son aquellas que se presentan en una única página web, emulando a las aplicaciones de escritorio.
- Reutilizar herramientas de Unix: La capacidad de Node de lanzar miles de procesos hijos que ejecuten comandos y tratar su salida como streams permiten utilizar la plataforma para reutilizar el software existente en lugar de reinventarlo para ella.
- Datos por streaming: al tratar las conexiones HTTP como streams, se pueden procesar ficheros al vuelo, conforme se envían o reciben.
- Comunicación: por las características comentadas, aplicaciones de mensajería instantánea o web en tiempo real, e incluso, juegos multijugador.

Basándose en las descripciones anteriormente mencionadas se decidió llevar a cabo el uso de esta tecnología en el módulo pues este en entorno de tiempo de ejecución en tiempo real incluye todo lo que se necesita para ejecutar un programa escrito en JavaScript.

Express v.4.16.3: "Express.js es un framework para Node.js que sirve para ayudarnos a crear aplicaciones web en menos tiempo ya que nos proporciona funcionalidades como el enrutamiento, opciones para gestionar sesiones y cookies, y un largo". (Nallu, 2015). Express.js está basado en Connect, que a su vez es un framework basado en http para Node.js. Podemos decir que Connect. A su vez, Express hace lo mismo con Connect, con lo que tenemos un framwork ligero, rápido y muy útil

Lo que se quiere alcanzar con esta herramienta es que nos brindara crear aplicaciones en menos tiempo, tiene todas las opciones del módulo http que viene por defecto con Node y le suma funcionalidades.

Jscript: ***"****JScript es la implementación de Microsoft de la especificación de lenguaje ECMA 262 (ECMAScript Edition 3). Aparte de algunas excepciones de poca importancia (para mantener la compatibilidad con versiones anteriores), JScript es una implementación completa del estándar ECMA. Esta introducción pretende ayudarle a comenzar a trabajar con JScript".* (2001 Microsoft Corporation)

Se seleccionó JavaScript como lenguaje de programación tanto del lado del cliente como del lado del servidor ya que es un lenguaje bastante utilizado en centro de desarrollo y permite el desarrollo de aplicaciones bajo el esquema de Cliente-Servidor; además es sencillo, fácil de aprender, rápido y potente, ideal para implementar pequeñas funciones dentro de la página web.

Entorno de Desarrollo Integrado10 Visual Studio Code v1.7.2: "Visual Studio Code es un editor de código fuente, gratuito y de código abierto, desarrollado por Microsoft para Windows, Linux y MacOS. Incluye soporte para la depuración, finalización inteligente y refactorización de código. Viene con soporte integrado para JavaScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes y tiempos de ejecución". (Microsoft Corp., 2017)

Conclusiones del capítulo:

- Los SIG son conjunto de procedimientos con capacidad de construir modelos o representaciones del mundo real, a partir de datos geográficos de localización cierta y mensurable y están compuesto fundamentalmente por los siguientes módulos: edición cartográfica; edición de capas de impresión; módulo de edición de errores topológico; selección de áreas de interfaz y edición de capas vectoriales.

- El módulo de edición de capas vectoriales este compuesto por 20 funcionales destacado se: Copiar Objetos espaciales, cortar objetos espaciales, borrar objetos espaciales, pegar objetos espaciales, guardar objetos espaciales, rotar objetos espaciales, estructura de vértices, selección de vértices, dividir objetos espaciales, Rotar símbolos de ponto, Añadir vértices, Borrar vértices, Mover vértices, Guardar vértices, Dibujar polígono, Mostrar los vértices. Dividir polígono en dos partes, crear vértice, Crear objetos espaciales, Modificar relleno del vértice, Mover vértices, Selección de vértices, Rotar símbolos de ponto.

- El análisis de diversos módulos de edición de capas vectoriales en distintos SIG nacionales e internacionales permitió identificar que las funcionales :(rotar objetos espaciales, estructura de vértices, selección de vértices, dividir objetos espaciales, Rotar símbolos de ponto, Añadir vértices, Borrar vértices, Mover vértices, Guardar vértices, Dibujar polígono, Mostrar los vértices. Dividir polígono en dos partes, crear vértice, Crear objetos espaciales,) son las principales.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL MÓDULO DE EDICIÓN DE CAPAS VECTORIALES

2.1. Introducción.

En el presente capítulo se realiza la solución propuesta mediante la identificación de los requisitos funcionales y no funcionales con los que debe cumplir el módulo. Para su demostración se hace necesario seguir los pasos de la metodología propuesta en el capítulo anterior (AUP-UCI) en su escenario 4, y análisis de los artefactos fundamentales del proceso de desarrollo. Se expone el diseño arquitectónico y ejemplificando el empleo de los patrones del diseño.

2.2. Requisitos de Software

"*Requisitos de software consiste en la especificación y validación de las funcionalidades que el sistema a desarrollar debe proporcionar, así como de las restricciones que el sistema debe cumplir*". (Pressman, et al., 2015).

Y según (García Velázquez, 2016). define que: "*La ingeniería de requisitos del software es un proceso de descubrimiento, refinamiento, modelado y especificación. Se refinan en detalle los requisitos del sistema y el papel asignado al software*".

La ingeniería de requerimientos permite determinar las necesidades que un cliente tiene en la implementación o desarrollo de software, mediante el proceso de recopilar, analizar y verificar la especificación de los requisitos del software de una manera completa y correcta.

2.2.1. Requisitos Funcionales

"*Los requisitos funcionales describen la interacción entre el sistema y a su ambiente independientemente de su implementación. El ambiente incluye al usuario y cualquier otro sistema externo que interactúa con el sistema*" (Pressman, 2017)

Los requisitos definidos por el cliente se listan a continuación:

RF1: Listar atributos de los objetos espaciales:

El módulo debe permitir crear objetos espaciales (punto, línea, polígono)., determinando para ello sus atributos (punto, línea, polígono).

RF2: Modificar atributos de los objetos espaciales:

El módulo debe permitir al usuario modificar los atributos de los objetos espaciales ().

RF3: Eliminar objetos espaciales:

El módulo debe permitir eliminar objetos espaciales a partir de su selección previa por el usuario

.

RF4: Dibujar Geometría:

El módulo debe permitir al usuario escoger una de las funcionalidades presente en el menú y dibujarlo en el mapa.

RF5: Dibujar punto:

El módulo debe permitir al usuario escoger un punto, funcionalidad presente en el menú, y dibujarlo en el mapa.

RF6: Dibujar línea :

El módulo debe permitir al usuario dibujar una línea, para ello el usuario selecciona la opción línea presente en el menú y con el mouse, dibuja una línea en el mapa cuya longitud y dirección varia en correspondencia a la ubicación del puntero.

RF7: Dibujar polígono:

El módulo debe permitir al usuario dibujar un polígono, para ello el usuario selecciona la opción línea presente en el menú y con el mouse, dibuja una línea en el mapa cuya longitud y dirección varia en correspondencia a la ubicación del puntero.

RF8: Dibujar polígono regular:

El módulo debe permitir al usuario dibujar un polígono regular, para ello el usuario selecciona la opción línea presente en el menú y con el mouse, dibuja una línea en el mapa cuya longitud y dirección varia en correspondencia a la ubicación del puntero.

RF9: Seleccionar geometría:
El módulo debe permitir que el usuario seleccione una geometría en el mapa.

RF10: Rotar geometría:
El módulo debe permitir que el usuario seleccione una geometría en el mapa y una vez seleccionado pueda rotarlo cada 45º

RF11: Trasladar geometría:
El módulo debe permitir que el usuario seleccione una geometría en el mapa y posteriormente trasladarlo para cualquier región del mapa.

RF12: Escalar geometría:
El módulo debe permitir hacer escala geométrica para una mejor visibilidad, pequeño o grande

RF13: Transformar múltiples geometrías:
Debe el módulo transformar múltiples geometrías, en cualquier lado de la geometría, poner un mejor ajuste

RF14: Compensar geometría:
El módulo le debe permitir al usuario que una vez dibujada una geometría en el mapa poderlo redimensionar en su misma estructura.

RF15: Dibujar agujero en un polígono:
Debe el sistema permitir dibujar agujero en un polígono señalado por el usuario previamente,

2.2.2. Requisitos no funcionales

"*En este epígrafe se estará mencionado los principales requisitos no funcionales que el sistema presenta. "Los requisitos no funcionales son restricciones en los servicios o funciones ofrecido por el sistema, los cuales incluyen restricciones de tiempo, restricciones en el proceso de desarrollo y restricciones impuestas por los estándares*". (Sommerville, 2016)

Estos a menudo se aplican al sistema como un todo en lugar de características o servicios individuales del sistema. Suelen ser más críticos que los requisitos funcionales ya que los usuarios de generalmente pueden encontrar formas de evitar una función del sistema que realmente no satisface sus necesidades.

Para el desarrollo del módulo de edición de capas vectoriales fueron definidos los requisitos no funcionales que se muestran a continuación:

Usabilidad:

RNF1. El módulo estará conformado por un conjunto de íconos que reflejan en sí la acción que realizará el usuario dando respuesta a cada una de las funcionalidades del módulo. Estos tendrán una forma rectangular.

Rendimiento:

RNF2. El módulo debe responder en un máximo de dos segundos las solicitudes de los usuarios.

Interfaz:

- **RNF3.** El módulo debe poseer una interfaz acorde a los principios del diseño que son: sin saturación de colores ni imágenes.

  Y debe poseer la siguiente gama de colores HTML en su diseño:

  - #00CCFF
  - #FFFFFF
  - #33CCC

```
html,
body {
    height: 100%;
}

.tooltip {
    color: red;
    z-index: 1010;
}
ultron-control {
 color: rgb(240, 73, 73) !important;
    font-size: 12px;
    /* text-align: right; */
```

Requerimientos de Hardware:

- **RNF5.** La PC cliente es recomendable que cumpla con los siguientes requisitos mínimos que se especifican a continuación:
  - ✓ Memoria RAM 512 MB o superior.
  - ✓ Procesador 512 MHz o superior
  - ✓ Capacidad de almacenamiento mínimo de 40GB
- **RNF6.** La PC servidor de Mapas y el servidor de aplicaciones están incluidas en la misma PC por lo cual debe tener como mínimo 2 GB de RAM, 40 GB de disco duro y un procesador de 2 GHz como mínimo.
- **RNF7.** La PC servidor de base de datos debe tener como mínimo 2 GB de RAM, 40 GB de disco duro y un procesador de 2 GHz como mínimo.

Requerimientos de Software:

- **RNF 8**. La PC Cliente debe poseer un Navegador Web que cumpla con los estándares de la Word Wide Web Consorcio (W3C).
- **RNF 9.** La PC Servidor debe tener instalado el Node-JS v10.15.0, Angular v 8.0 y Mapserver v2.0.

- **RNF 10**. La PC servidor de la Base de Datos debe tener instalado el PostgreSQL 9.4 o superior y PostGIS 2.0 o superior como extensión de PostgreSQL para el soporte de datos espaciales.

2.3. Descripción del módulo de edición de capas vectoriales

2.3.1. Descripción de los actores que interactúan con el sistema

Según (Cevallos, 2015) define que “Los actores son similares a las entidades externas; existen fuera del sistema e interactúan con este. El término actor se refiere a un rol específico de un usuario del sistema. Un actor puede ser un empleado, pero también puede ser un cliente en la tienda de la empresa. En ocasiones siendo la misma persona en el mundo real, se representa como dos símbolos distintos en un diagrama de caso de uso, ya que la persona interactúa con el sistema en distintos roles” (p. 15).

“Un actor es una entidad externa al sistema y que puede interactuar con él. Puede ser una persona o un grupo de personas homogéneas, otro sistema, o una máquina. El actor representa un papel, no a un usuario individual del sistema. El conjunto de funcionalidades a las que un actor tiene acceso define un rol en el sistema y el alcance de su acción”. (Cevallos, 2015, p. 10).

A continuación (Tabla 1), se menciona el actor que estará interactuando con el módulo a desarrollar, definiendo el rol que le ocupa dentro del mismo y su descripción.

Tabla 1: Descripción de los actores del sistema

Actor	**Descripción**
Especialista	Es el encargado de realizar la gestión de edición de capas vectoriales.

2.3.2. Descripción de los requisitos funcionales

Las historias de usuarios son parte del desarrollo ágil de software que ayuda a enfocarse en hablar de los requerimientos en lugar de escribir acerca de ellos. Estas son cortas

descripciones de una funcionalidad desde la perspectiva de la persona que la desea, usualmente un usuario o cliente

A continuación, se presenta la descripción de los requisitos del Módulo de edición de capas vectoriales a partir del modelo propuesto por la Historia de usuarios (Tabla 2, Tabla 3 y Tabla 4).

Tabla 2: Historia de usuario “Listar atributos de los objetos espaciales”.

Número: 01	**Nombre del Requisito: Listar atributos de los objetos espaciales**
Programador: **Wilson Félix Cassuque cambila**	**Iteración Asignada:** 1
Prioridad: **Alta**	**Tiempo estimado: 7** días
Riesgo en desarrollo: • **Planificación irreal.** • **Interrupción del servicio eléctrico.** • **Afectaciones externas al personal de trabajo.**	**Tiempo Real:** 9 días
Descripción: **Se deben mostrar los campos para dar entrada de los datos del tipo de objeto, posibilitando la edición de los siguientes campos:** • **Nombre**: campo que permite nombrar el tipo de capa que se necesita editar, el nombre debe ser una región territorial, un punto de referencia o una calle u otros tipos de regiones geográficos. • **Tipo**: Campo que permite seleccionar el formato, o sea seleccionar el tipo de capa a editar, punto, lineo o polígono. • **Descripción**: Campo que permite mostrar las descripción o informaciones que bebe tener el tipo de capa. • **Capa**: Campo que permite mostrar el nombre de la capa a la que pertenece el objeto espacial.	

Observaciones:

- El nombre es un campo obligatorio, no debe exceder los 30 caracteres y un mínimo de 3 caracteres. No permite caracteres especiales.
- La descripción no debe exceder los 120 caracteres.

Prototipo de Interfaz:

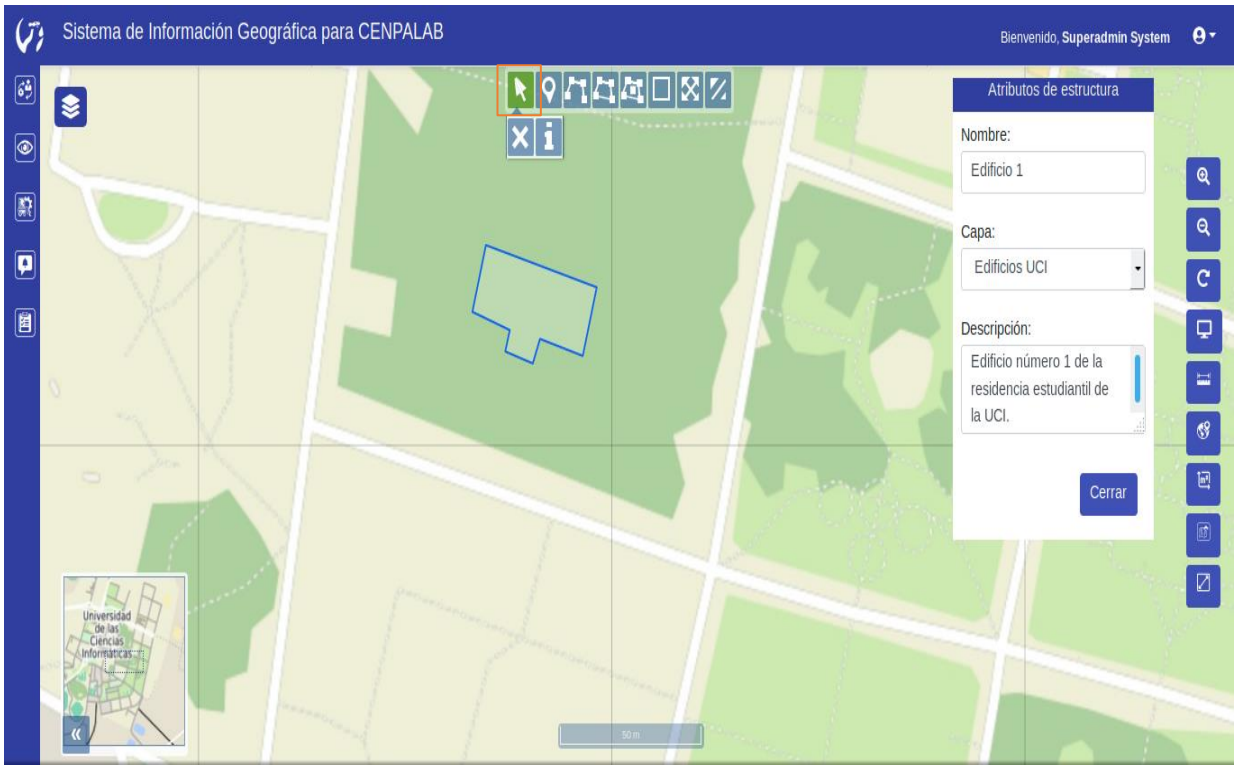


Tabla 3: Historia de usuario “Modificar atributos de los objetos espaciales”.

Número: 02	**Nombre del Requisito:** Modificar atributos de los objetos espaciales
Programador: Wilson Félix Cassuque cambila	**Iteración Asignada:** 1
Prioridad: **Alta**	**Tiempo estimado: 5** días
Riesgo en desarrollo: • Planificación irreal. • Interrupción del servicio eléctrico. • Afectaciones externas al personal de trabajo.	**Tiempo Real: 8** días
Descripción: **Se deben mostrar una opción que permita modificar los atributos del objeto espacial:** • Nombre: • Tipo: • Descripción: • Capa:	
Observaciones: • **Nombre**: es un campo obligatorio, no debe exceder los 30 caracteres y un mínimo de 3 caracteres. No permite caracteres especiales ni números. • **Descripción**: no debe exceder los 120 caracteres. • **Tipo**: debe mostrar un menú donde el usuario escoja uno de las siguientes opciones punto, línea o polígono.	

- **Capa**: Campo que permite mostrar el nombre de la capa a la que pertenece el objeto espacial.

Prototipo de Interfaz:

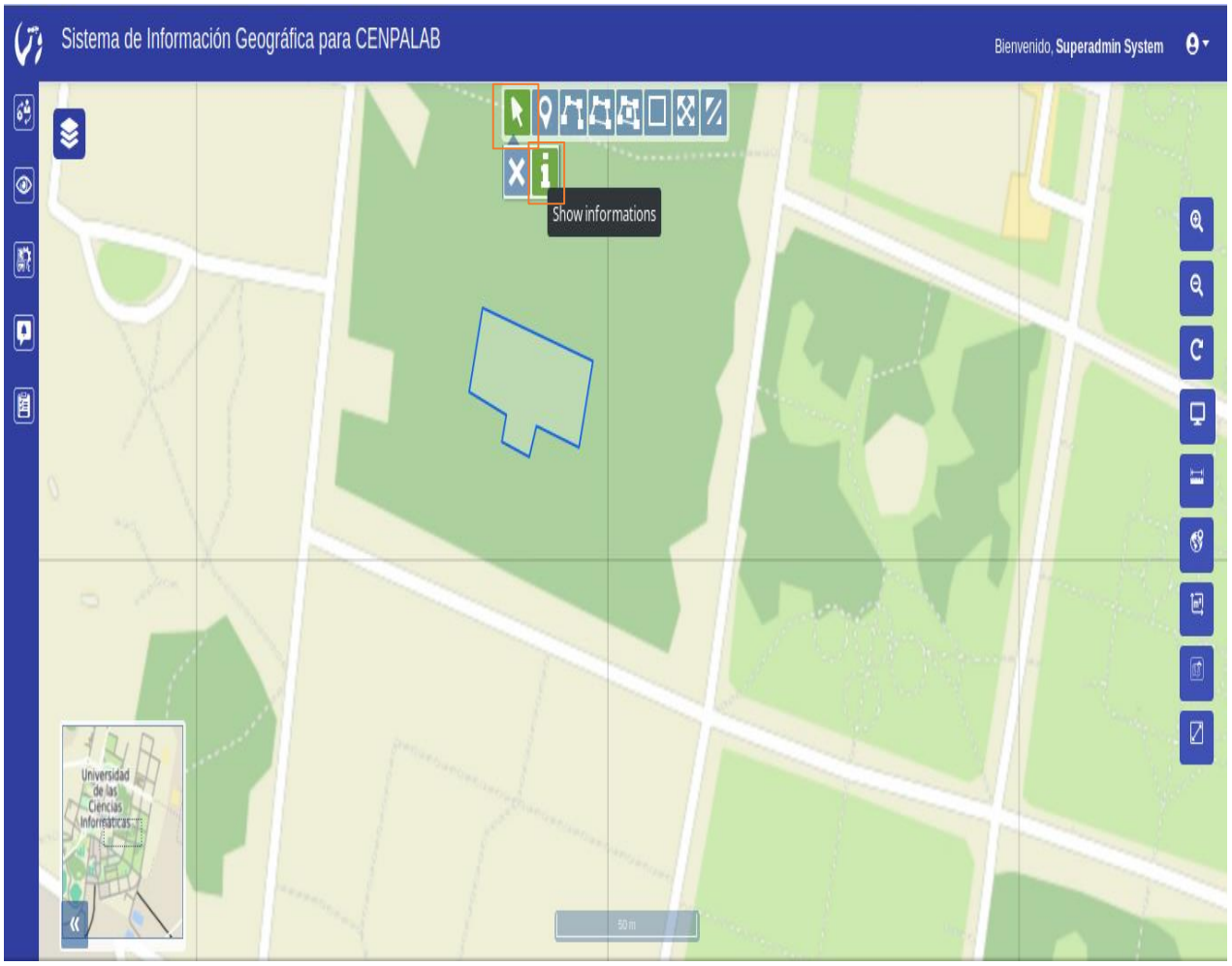


Tabla 4: Historia de usuario “Eliminar objetos espaciales”.

Número: 03 Nombre del Requisito: **Eliminar objetos espaciales**	
Programador: Wilson Félix Cassuque Cambila	**Iteración Asignada:** 1
Prioridad: **Alta**	**Tiempo estimado: 6** días
Riesgo en desarrollo: • Planificación irreal. • Interrupción del servicio eléctrico. • Afectaciones externas al personal de trabajo.	**Tiempo Real: 4** días
Descripción: El usuario selecciona en el mapa un objeto espacial y sistema le debe permitir eliminarlo.	
Observaciones: • Cuando se borra el objeto espacial el sistema debe garantizar no recuperar los datos borrados.	

Prototipo de Interfaz:

2.4. Patrón arquitectónico

"*La arquitectura del software de un programa o sistema de cómputo es la estructura o estructuras del sistema, lo que comprende a los componentes del software, sus propiedades externas visibles y las relaciones entre ellos. Además, se refiere a las estructuras de un sistema, compuestas de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos*". (Valdés Fernández, et al., 2017)

La arquitectura de software se refiere a la estructuración del sistema que, idealmente, se crea en etapas tempranas del desarrollo. Representa un diseño de alto nivel que tiene dos propósitos primarios: satisfacer los atributos de calidad y servir como guía en el desarrollo.

Según (Medina, 2016) “un patrón arquitectónico es una solución general y reutilizable a un problema común en la arquitectura de software dentro de un contexto dado. Son similares al patrón de diseño de software, pero tienen un alcance más amplio. Además, constituye una colección de decisiones de diseño arquitectónico, que tiene un nombre específico y son parametrizadas para tener en cuenta diferentes situaciones durante el desarrollo de software”.

Para el desarrollo del módulo se decide hacer uso del Patrón Arquitectónico Modelo - Vista – Controlador (MVC). Esta arquitectura es empleada en el sistema ULTRON y es la que sigue el framework de JavaScript Angular.JS.

Además, este patrón arquitectónico tiene las ventajas que a continuación se describen según (Medina, 2018):

- Vistas múltiples del mismo modelo. La separación estricta del modelo de los componentes de la interfaz de usuario, permite que múltiples vistas sean implementadas usando un único modelo.

- Vistas sincronizadas. El mecanismo de propagación del modelo asegura que todos los observadores conectados son notificados de cambios a los datos de la aplicación en el momento correcto.

- Vistas y controladores conectables. La separación conceptual de MVC permite intercambiar los objetos vista y controlador de un modelo. Los objetos de interfaz de usuario pueden ser sustituidos en tiempo de ejecución.

- Intercambiabilidad de “look and feel”. Como el modelo es independiente de todo el código de la interfaz de usuario, el portar la aplicación a una nueva plataforma no afecta el núcleo funcional de la aplicación.

- La separación del Modelo y la Vista, lo cual logra separar los datos, de su representación visual.
- Facilita el manejo de errores.
- Permite que el sistema sea escalable si es requerido.

- Es posible agregar múltiples representaciones de los datos.

"La arquitectura Modelo Vista Controlador (Model-View-Controller) es un patrón de diseño de software en torno a la interconexión de los tres tipos de componentes principales en un lenguaje de programación. A menudo con un fuerte enfoque en la programación orientada a objetos (POO). Estos tres tipos de componentes son llamados modelos, vistas y controladores". (Gómez, 2015)

El patrón MVC (Fig. 1) según (Medina, 2018) "divide una aplicación, subsistema o módulo en tres áreas: procesamiento, entrada y salida:

- El Modelo es el que encapsula los datos y la funcionalidad central; es independiente de las representaciones específicas de la salida o el comportamiento de la entrada.

- La Vista se encarga de presenta la información al usuario obteniendo los datos del modelo; puede haber múltiples vistas del modelo y cada una de ellas tiene asociado un componente Controlador.

- El Controlador es el que recibe y responde a las entradas o eventos (usualmente acciones del usuario), estos eventos son trasladados a solicitudes de servicio al modelo o a la vista

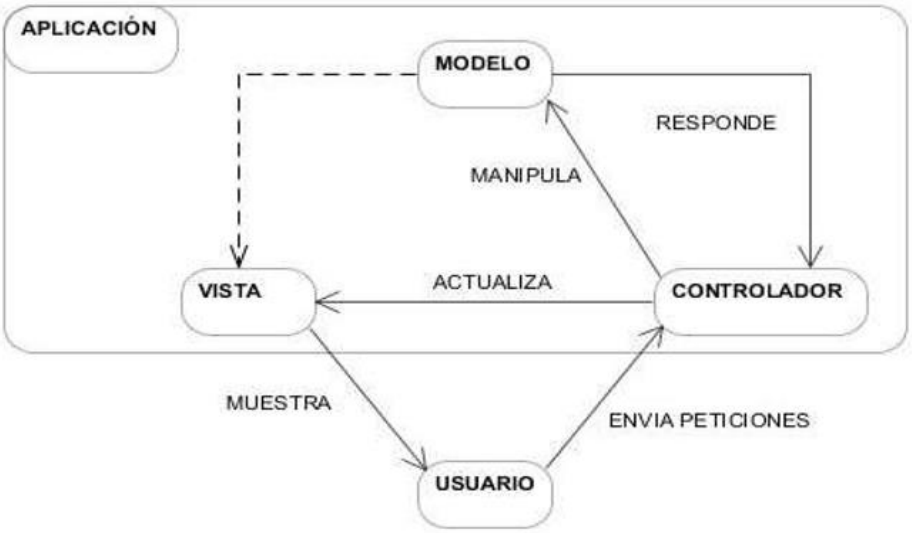


Fig. 3: Patrón arquitectónico.

Fuente: (Suarez, 2018)

Teniendo en cuenta este patrón se diseñó el Diagrama de componentes del módulo de edición de capas vectoriales. Los diagramas de **componentes** describen los elementos físicos del sistema y su relación, mostrando las dependencias lógicas entre los componentes de software. El diagrama de componentes hace parte de la vista física de un sistema, la cual modela la estructura de implementación de la aplicación por sí misma, su organización en componentes y su despliegue en nodos de ejecución. La vista de implementación se representa con los diagramas de componentes.

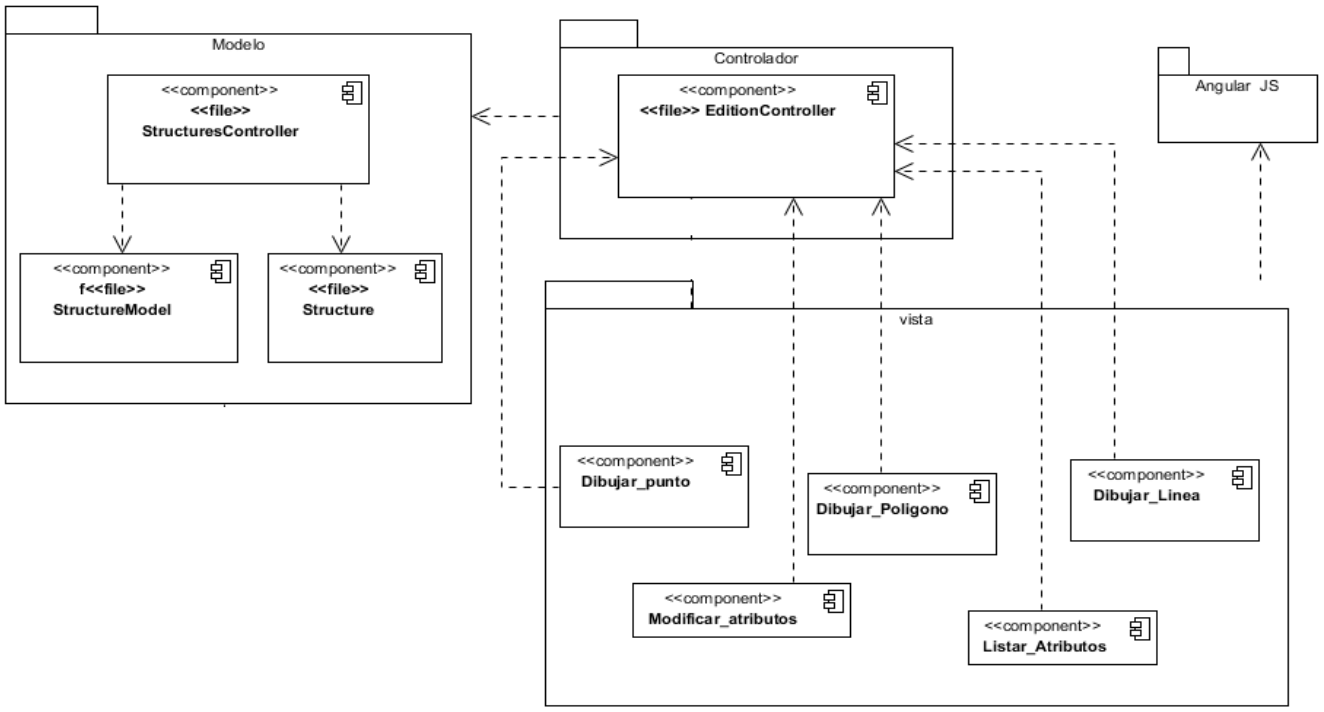


Fig. 4: Diagrama de componentes

"*Un diagrama de componentes muestra cómo un software es dividido en componentes, los artefactos por los que está compuesto como los archivos de código fuente, las librerías o las tablas de una base de datos y representa las dependencias entre estos componentes. Uno de los usos principales es que puede servir para ver como los componentes pueden compartirse entre sistemas o bien entre diferentes partes de un sistema*". (Delgado, 2015)

El subsistema de la solución para el trabajo está dividido en tres subsistemas de implementación fundamentales: el subsistema de las Vistas, el subsistema de las Controladoras y el subsistema de los Modelos, estructurados de forma tal que se agrupan los scripts de acuerdo con el papel que desempeñan dentro del patrón arquitectónico Modelo - Vista - Controlador

- Las **Vistas** contiene los componentes necesarios para la interacción del usuario con el sistema, los cuales son manejados por el subsistema de las Controladoras.

- El paquete de las **Controladoras**, es el rector de las actividades de la aplicación, este contiene los ficheros de código fuente, los cuales interactúan con los demás subsistemas coordinando las acciones del software.

- El paquete de los **Modelos** es el encargado de la interacción con la base de datos, para de esta forma gestionar la información con la que trabaja el sistema.

Teniendo en cuenta la descripción anteriormente mencionado, para cada componente en cada paquete se clasifican de la siguiente manera:

El paquete Modelo:

componente **Strutures_Controller** es la encargada de construir y controlar los datos con lo que se trabaja en el sistema, y su vez este componente interactúa con dos componentes:

- **Struture_model:** este componente es el encargado de representar la estructuración de las informaciones en la base de datos dentro del sistema.
- **Struture:** es un *framework* para construir aplicaciones web JavaScript basada en la filosofía MVC. Con el uso de este frameworck estaremos seguros de que se dispondremos de una gran cantidad de recurso: documentaciones (tutoriales, artículos, libros,)

El paquete Controlador y Vista:

contiene el componente **Edition_Controller,** garantiza que el flujo de datos que llegan, cumple con las condiciones y restricciones definidas y a su vez este componente interacciona con el paquete vista que incluyen 6 componentes:

- **CP_Principal:** En este componente es el que interactúa con usuario, recibe y valida los datos entrado por el usuario.
- **Dibujar Punto, Dibujar_Linea, Dibujar_Poligono:** estos componentes controlan los datos que se llevan a cabo en la vista con lo que el usuario interactúa. Las ediciones establecidas son comprobadas a partir del código fuente implementado en el módulo.
- **Listar Atributos:** este componente, asegura que cualquier dato u información ingresado por el usuario estos bienes cumplan con las restricciones y principios implementado en el código fuente.

2.5 Diagrama de clases del diseño

El diagrama de clases del diseño muestra los bloques de construcción de cualquier sistema orientado a objetos. *“El diagrama de clases del diseño permite representar gráficamente y de manera estática la estructura general de un sistema, mostrando cada una de las clases y sus interacciones (como herencias y asociaciones), representadas en forma de bloques, los cuales son unidos mediante líneas y arcos*”. (Arango Isaza, et al., 2016)

A continuación, se representa el diagrama de clases del diseño correspondiente al módulo de edición de capas vectoriales para la plataforma ULTRON v.2.0. Entre los diagramas de UML los diagramas de clases del diseño ocupa un lugar destacado ya que son un tipo de diagrama estático que describen y estructuran gráficamente los datos asociados con el área de aplicación y el software.

Para el desarrollo del módulo se utilizan los diagramas de clases del diseño, ya que los mismos brindan a los desarrolladores una aproximación de lo que se desea implementar. A continuación, se muestra en la Fig.2 cómo están relacionados los paquetes del modelo visto controlador a partir del diagrama de clases.

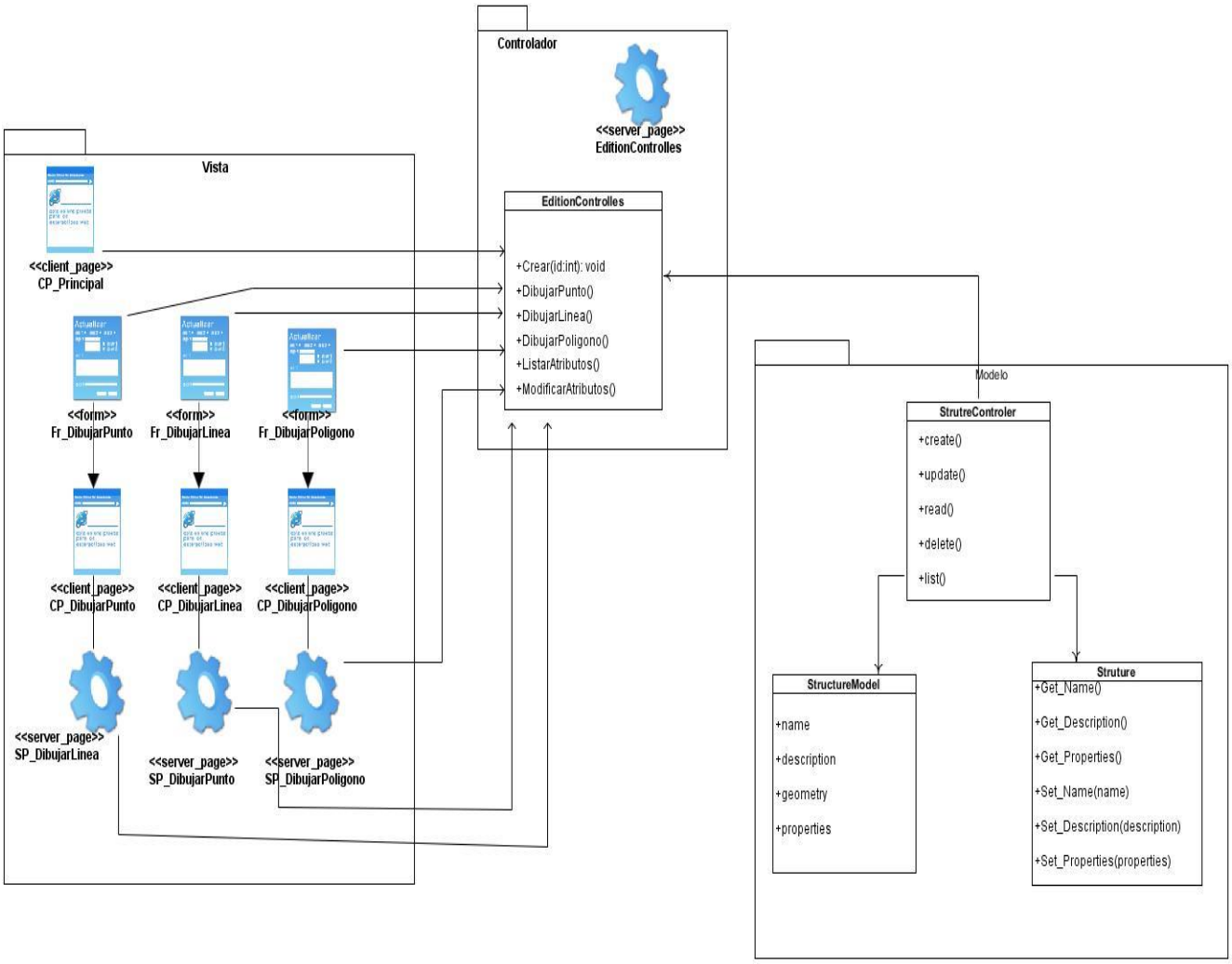


Fig. 5: Diagrama de clases del diseño del modulo

Teniendo en cuenta las descripciones del diagrama de clases Fig.2. los paquetes son evidenciados en el modelo de la siguiente manera:

- **Paquete vista:** Su responsabilidad es mostrar los formularios para realizar funcionalidades: dibujar geometría, seleccionar estructura, eliminar estructura, obtener, modificar e introducir información. Procesan las peticiones del usuario y envían las necesidades al paquete Controlador. Este paquete interacciona con Angular JS, quién es el que le permite la implementación de las páginas clientes.

- **Paquete Controlador:** Tiene función principal dar respuesta a las peticiones realizadas por el usuario a través de las páginas clientes presentes en el Paquete Vista. Para ello emplea la clase servidora EditionController, con sus métodos creadoras, manejando el flujo de eventos e información del módulo.
- **Paquete Modelo:** Tiene como función interaccionar con la Base de Datos y en ella dibujar y almacenar los datos del objeto espacial. Para ello emplea la clase StructuresController que tiene la responsabilidad de responder las peticiones que le llegan del controlador ya sea a él o directamente a la vista. Esta contiene todos los métodos que van a ser necesarios para crear o eliminar geometrías y modificar o listar atributos de las geometrías. Este paquete tiene la clase Estructure, la cual es la encargada de contener toda la información referente a una estructura espacial, mientras que la clase StructureModel, posee los atributos necesarios que brindan al controlador un modelo para administrar esta información.

2.6. Patrones de diseño de software empleados en el desarrollo del módulo.

"Los patrones de diseño son una descripción de problemas que ocurren en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces, así como la base para la búsqueda de soluciones a estos problemas, de tal modo que se pueda aplicar esta solución un millón de veces, sin repetir lo mismo dos veces". (González, 2016)

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. Se utilizan para crear propuestas de soluciones reusables y con documentación ya probada, partiendo de técnicas que se han aplicado en problemas similares. Estableciendo un lenguaje estándar entre todos los miembros de un equipo".

2.6.1 Patrones para la asignación de responsabilidades (GRASP).

Los patrones GRASP son una colección de principios de diseño orientados a objetos que guían la asignación de responsabilidades sobre los objetos y un intento de documentar lo que los diseñadores expertos probablemente conozcan intuitivamente. (Trellini, 2015).

Para el desarrollo del módulo se hizo uso de los siguientes patrones:

El patrón **Experto (Expert)** es el encargado de asignar responsabilidades a las clases que tiene la información necesaria para cumplir dicha responsabilidad. Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada. Hay que tener en cuenta que esto es aplicable mientras estemos considerando los mismos aspectos del sistema. La utilización de dicho patrón se ve aplicada en clases como: LayerModel y Layer, debido que cada una de ellas es responsable de manejar su información.

Este padrón de diseño Experto en el módulo es evidenciado en paquete Modelo a partir de la tabla StrutureController, pues contiene todas las información y datos que son procesado dentro del sistema.

El patrón **Creador (Creator)** en el módulo es evidenciado por dar la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Está aplicado en la clase LayersController, la cual utiliza la información de la clase LayerModel para crear objetos de ella.

El patrón **Controlador (Controller**) ofrece una guía para tomar decisiones apropiadas que generalmente se aceptan. La misma clase controlador debería utilizarse con todos los eventos sistémicos, de modo que se pueda conservar la información referente al estado del caso. El mismo se refleja en la clase DesignerController, la cual controla el flujo de eventos del módulo.

2.6.2 Patrones GOF.

"*los patrones GoF describe soluciones simples y elegantes a problemas específicos en el diseño de software orientado a objetos, permiten enfrentarse a la programación de software propiciando reutilización y extensibilidad de soluciones que han funcionado en el pasado para garantizar la calidad de los mismo, estos patrones son conocidos como los patrones de la pandilla de los cuatro (GoF, gang of four). Se clasificarían según el propósito para el que han sido definidos en*:" (Valdés, 2014)

Patrones creacionales: Se encargan de crear instancias de los objetos, abstraer y ocultar cómo son creados e inicializados los objetos. Separan la forma en que se crean los objetos, para tratar las clases

a crear de forma genérica. Se ocultan los métodos y clases concretas de tal forma que al variar su implementación no se vea afectado el resto de la aplicación.

- Instancia única (Singleton): Este patrón garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Este tipo de patrón es muy utilizado en frameworks como AngularJS. Se evidencia en la clase EditionController, la cual define varias variables globales de instancia única que permiten la comunicación con el modelo.

Patrones estructurales: Se encargan de combinar las clases y objetos para formar nuevas estructuras y proporcionar nuevas funcionalidades. Lo más relevante son las relaciones de uso entre los objetos, determinadas por las interfaces que soportan los objetos.

- **Decorador (Decorator):** Añade dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender la funcionalidad. La aplicación de dicho patrón se ve reflejado en el diseño de las clases cliente de la vista al heredar el código global HTML que es usual en todas las páginas del sistema, para no tener que repetirlo en cada una de estas.

2.7. Modelo físico de la Base de Datos.

Para la implementación del módulo se empleó el modelo físico de la base de datos perteneciente al sistema ULTRON. El módulo Edición de capas Vectoriales trabaja específicamente con las entidades Layer, Layer_type, Structure, Attribute, Data_type. A continuación, se muestra en la fig. 3, como las tablas del modelo físico están relacionados:

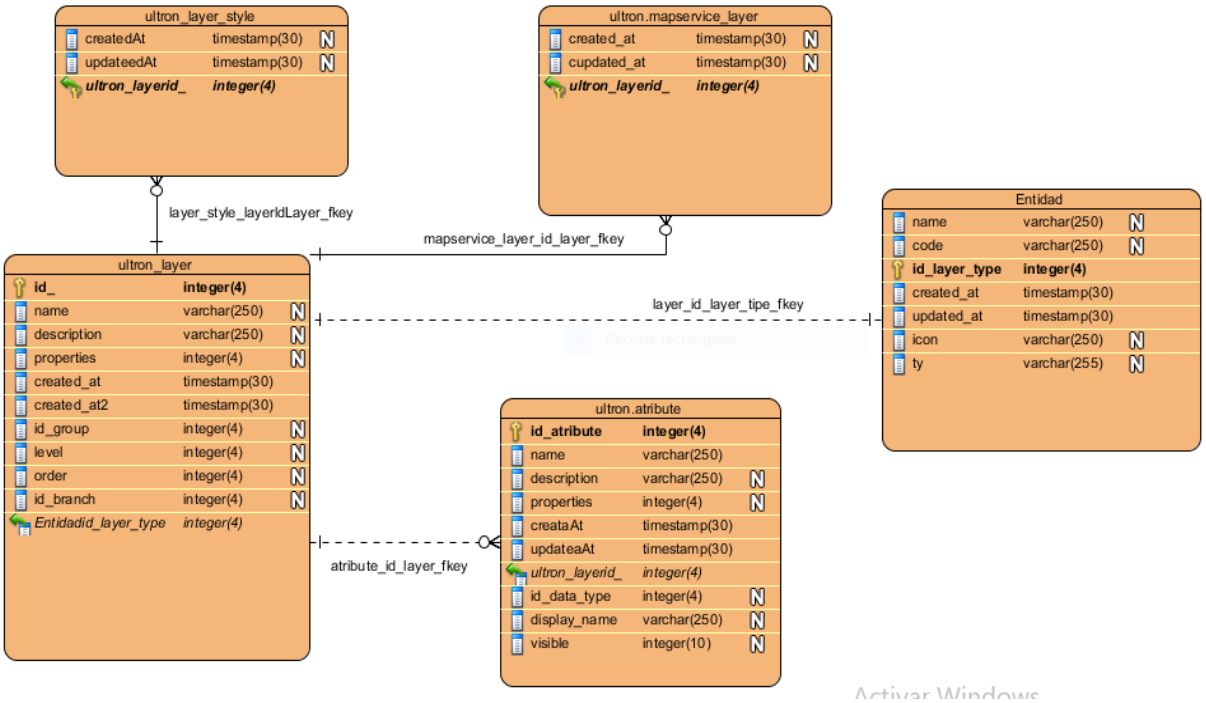


Fig. 3: Modelo físico de la base de datos.

Teniendo en cuenta las descripciones de del modelo físico anteriormente mencionado, se puede decir que el modelo consta de cinco tablas, la tabla **Layer_Styles** es el que define el estilo de cada capa que son visualizado en el sistema. Y a su vez interactúa con la tabla **Ultron_Layer** lo cual contiene todos los tipos de capas a ser usados dentro de la plataforma Ultron según como está definida e implementado. **ultron.mapservice.layer** consta todos los servicios de mapas que son usados para las capas según como está definido en la plataforma Ultron. La tabla **Entidad** su principal funcionalidad es de crear todos los atributos necesarios que son creado en la tabla Ultron_Layer. **ultron_atribute** almacena todos los atributos de cada capa creado dentro del sistema.

Conclusiones parciales.

- La descripción y el diseño del modelo de dominio permitieron lograr una mejor abstracción del proceso de negocio facilitando relacionar los conceptos y objetos significativos del mismo.

- La identificación de los requisitos funcionales y su modelado, propició agruparlos en 12 historias de usuarios, y para su posterior implementación se seleccionaron patrones de diseño (GRASP, GOF) donde el patrones de diseño GRASP son colecciones de principios de diseño orientados a objetos, que guían la asignación de responsabilidades sobre los objetos y un intento de documentar lo que los diseñadores expertos probablemente conozcan intuitivamente, y los patrones de diseño GOF describe soluciones simples y elegantes a problemas específicos en el diseño de software orientado a objetos. Y estos dos patrones se estará aplicar en su estructuración en la clase del diseño contribuyendo a su correcta trazabilidad.

- El empleo de la arquitectura modelo – vista - controlador y de los patrones de diseño GRASP y GOF permitieron establecer las bases para fomentar la reutilización y las buenas prácticas de programación, así como disminuir el impacto de los cambios futuros en el código fuente y también contribuyeron al diseño de la solución, evidenciando la utilización de buenas prácticas durante el desarrollo de la solución propuesta.

- La identificación de los requisitos del módulo, su modelado a partir del diagrama de casos de uso del sistema y la descripción de los casos de uso propició un mejor entendimiento de la solución propuesta al equipo de desarrollo en cuanto a las funcionalidades y características que debe poseer.

CAPÍTULO 3: DISEÑO DE LA ESTRATEGIA DE PRUEBAS A REALIZAR PARA EL MÓDULO DE EDICIÓN DE CAPAS VECTORIALES PARA LA PLATAFORMA ULTRON V 2.0

3.1. Introducción

En el presente capítulo se analizan los principales artefactos que están relacionados a la implementación y validación de la solución propuesta para el desarrollo del módulo de edición de capas vectoriales. Se presentan los componentes y los estándares de codificación que sustentan la implementación del módulo de edición de capas vectoriales. Y también se presenta la estructuración del mismo a través del diagrama de componentes, y distribución física mediante el diagrama de despliegue

3.2 Estándares de codificación

"*Los estándares de codificación son parte de las llamadas buenas prácticas o mejores prácticas, estas son un conjunto no formal de reglas, que han ido surgiendo en las distintas comunidades de desarrolladores con el paso de tiempo y las cuales, bien aplicadas pueden incrementar la calidad de tu código, notablemente*". (Domínguez Vera, et al., 2015). Y también entendemos que estándar de código a un conjunto de convenciones establecidas de ante mano (denominaciones, formatos, etc.) para la escritura de código. Estos estándares varían dependiendo del lenguaje de programación elegido y además varían de cobertura.

Para la implementación del módulo edición de capas vectoriales se hacen uso de los siguientes estándares de codificación definidos por los desarrolladores de la plataforma ULTRON:

- Añadir comentarios para explicar determinadas partes de tu código.
- Al elegir nombres de las variables, se recomienda el utilizar variables descriptivas.
- Todas las variables de un bloque de código se declararán al inicio del método en cuestión.

- Las variables se declaran una por línea y empleando la palabra reservada <var> que nunca podrá omitirse, aunque el lenguaje lo permita.
- Añadir comentarios descriptivos en casos necesarios que expongan el objetivo del código.
- Las funciones se escriben en minúsculas y separados con guiones de suelo, y los argumentos que reciben son descriptivos.
- Se dejará una línea en blanco inmediatamente después del inicio de un bloque de código, dígase bloques de funciones, bloques condicionales o bucles.
- Las declaraciones de atributos, clases y funciones creadas por el usuario deben escribirse en minúscula, separadas por guion bajo (_).
- Añadir una línea de comentario con al menos 60 asteriscos para separar las funciones creadas, lo cual ayudará a visualizar donde empieza y dónde termina cada función.
- Escribir espaciado, con el fin de mejorar la apariencia del código.
- Evitar el uso de más de una instrucción por línea de código.
- La llave de apertura de un bloque de código se coloca después de la instrucción anterior y no en una línea en blanco independiente.
- La llave de clausura de un bloque de código tendrá el mismo nivel de indotación que la línea de la llave de apertura.
- Cada bloque de código anidado tendrá un nivel de dentición más que el bloque padre exceptuando las llaves de apertura y cierre.
- Todas las variables de un bloque de código se declararán al inicio del método en cuestión.
- Las variables se declaran una por línea y empleando la palabra reservada <var> que nunca podrá omitirse, aunque el lenguaje lo permita.
- Añadir comentarios descriptivos en casos necesarios que expongan el objetivo del código.
- Los nombres de cada elemento del programa deben ser significativos; es decir, que con el puro nombre se pueda o ayude a deducir a qué se refiere o cuál es su funcionalidad.
- Las declaraciones de atributos, clases y funciones creadas por el usuario deben escribirse en minúscula, separadas por guion bajo (_).

3.3 Diagrama de despliegue.

“Los diagramas de despliegue muestran la configuración en funcionamiento del sistema incluyendo su software. Para cada componente de un diagrama es necesario que se deba documentar las características técnicas requeridas, el tráfico de la red, el tiempo de respuesta”. (Delgado, 2015)

Diagramas de despliegue se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. un tipo de diagrama del lenguaje unificado de modelado que muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. En la siguiente figura se muestra el diagrama de despliegue del módulo desarrollado.

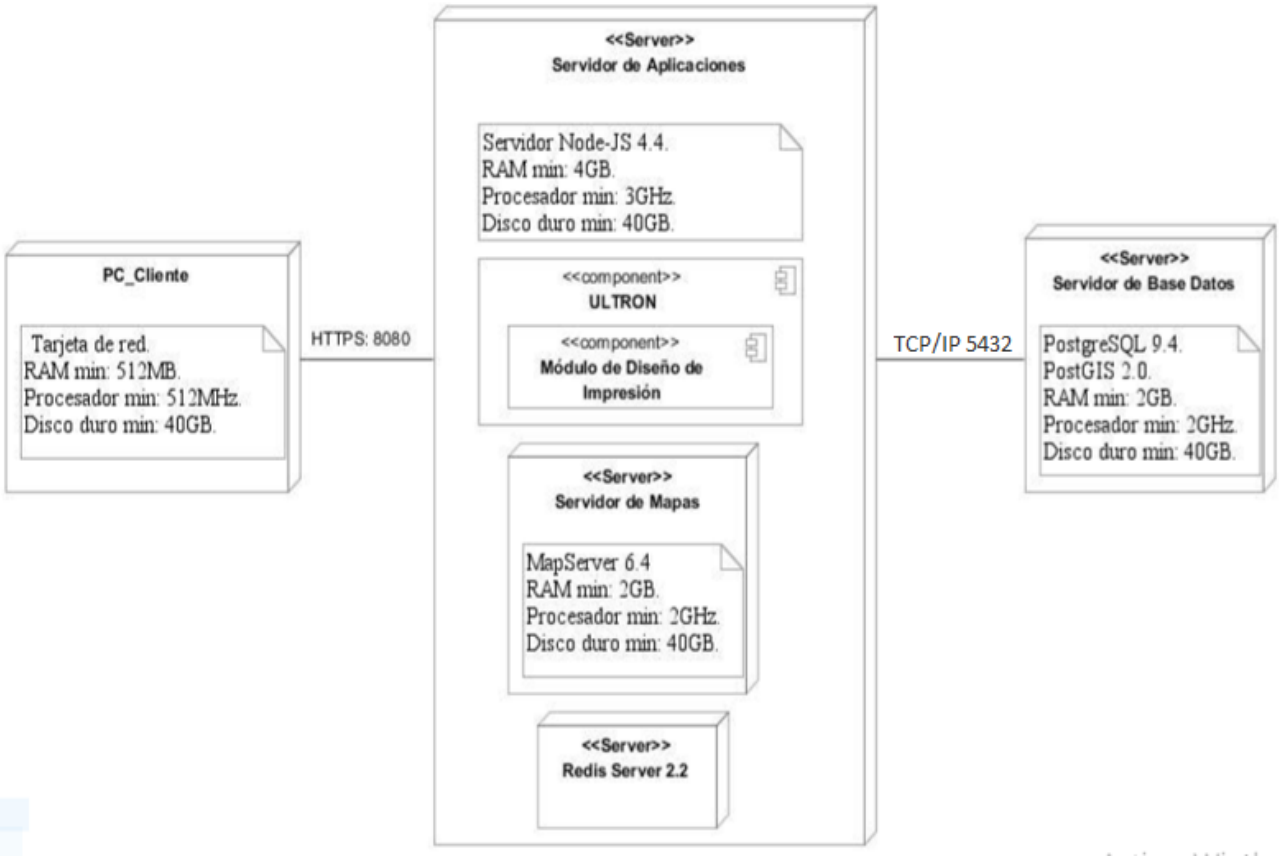


Fig. 5: Diagrama de Despliegue

Teniendo en cuenta a la Fig. 5, el **nodo PC cliente** se comunica con el nodo servidor de la aplicación a través del canal puerto 8080, protocolo de transferencia segura de datos de hipertexto. Dicho nodo PC cliente está equipado por una tarjeta de red, RAM min:512 MB, procesador min: 513 MHz y Disco duro min:40 GB. Del otro lado el **nodo Servidor de base de datos** se comunica con el servidor de aplicación por el canal puerto 5432, modelo protocolo TCP que permite comunicarse dentro de una rede, y dicho nodo está equipado por un conjunto de servidores de base de datos, que de entre ellas son: PostgreSQL V 9.4, PostGIS v 2.0, RAM min: 2GB, Procesador min: 2GHz, Disco duro min: 40 GB. Y el **nodo servidor de aplicación** está equipado por un conjunto de elementos, que de entre ellas uno es el **nodo servidor de mapa** equipado por las siguientes características: Mapa Server v 6.4, RAM min: 2GB, Procesador min: 2GHz, Disco duro min: 40 GB.

3.4 Estrategia de pruebas.

> "Una estrategia de prueba integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que llevan a la construcción correcta del software. Es una parte fundamental del proceso de validación y verificación del software. La verificación es una actividad la cual nos aseguramos que las distintas partes del software cumple con la función para la cual fueron diseñadas, en este sentido la verificación se encarga de revisar el funcionamiento de los módulos del software, mientras que la validación se encarga de comprobar que los módulos verificados cumplen con los requisitos que el cliente ha expresado". (Pressman, et al., 2015)

> También detalla los módulos, componentes y funcionalidades; así como la cobertura de pruebas a aplicar en cada uno de ellos. Además, logra indicar los posibles riesgos del proyecto y las acciones para mitigarlos.

3.4.1. Niveles de pruebas.

> "*Para la realización de las pruebas se definieron los siguientes niveles de pruebas que a continuación se describen según*" (Báez, 2016)

- **Pruebas Unitarias:** Corresponden a la validación de una pieza, componente, módulo o subprograma especifico de un sistema para comprobar que cada una de esas partes funcionan correctamente por separado.

- **Pruebas de Sistema**: Las pruebas del sistema tienen como objetivo ejercitar profundamente el sistema. Estas verifican el funcionamiento correcto de las interfaces entre los distintos subsistemas o módulos que lo componen y con el resto de sistemas de información con los que se comunica. Permiten probar el sistema en su conjunto y con otros sistemas con los que se relaciona para verificar que las especificaciones funcionales y técnicas se cumplen.
- **Pruebas de Integración:** Verifican la correcta interacción entre dos o más componentes que se desarrollan aisladamente en un sistema para comprobar que funcionan correctamente juntos, es decir, se prueba la interacción entre las distintas partes del software.

3.4.2 Tipos de pruebas.

Pruebas de Funcionalidad: “Pruebas funcionales son diseñadas tomando como referencia las especificaciones funcionales de un componente o sistema (lo que se va a testear, el software o una parte de él). Las características no funcionales del software se pueden medir de diversas maneras, por ejemplo, por medio de tiempos de respuesta en el caso de pruebas de rendimiento o por número máximo de sesiones en pruebas de estrés”. (Pressman,2017).

Pruebas no Funcionales: Se enfocan en validar un sistema o aplicación por medio de sus requerimientos no funcionales, la forma en que el sistema funciona y no por medio de comportamientos específicos”. (Maceo, 2015). Para la realización de las pruebas no funcionales al módulo de edición los desarrolladores optaron por aplicar la siguiente prueba que a continuación se describe:

- **Pruebas de Usabilidad:** “se basan principalmente en determinar cuán bien el usuario podrá usar y entender la aplicación además de identificar las áreas de diseño que hacen al sistema de difícil uso para el usuario”. (Maceo, 2015)

- **Pruebas de Carga:** “consisten en la medición del comportamiento del sistema para aumentar la carga del mismo, ya sea mediante el número de peticiones que se realizan a una WEB al mismo tiempo o el número de usuarios que trabajan simultáneamente”. (Peño, 2015)

3.4.3. Métodos de pruebas.

Método de Caja Negra: El método de caja negra pone a prueba la funcionalidad y el rendimiento del sistema para el cumplimiento de los requisitos funcionales documentados en la especificación del producto.

“Implica adicionalmente diseñar casos de prueba en los que el programa no actúe como está esperado que lo haga si se desea utilizar este método para encontrar todos los errores en el programa, el factor determinante es que las entradas sean aceptadas de forma adecuada y que se produzca una salida correcta; así como que la integridad de la información externa se mantenga”. (López, 2015).
Para la realización de este método se utiliza la técnica de Partición de Equivalencia.

Método de Usuarios Expertos: Los usuarios expertos contribuyen a las pruebas de usabilidad detectando errores del sistema, basando sus opiniones en su propia experiencia. (Ruíz Eguino, et al., 2013). Para la realización de este método se optó por la utilización de la técnica de Evaluaciones Heurísticas.

3.4.4. Técnicas de pruebas

Para realización de las pruebas al módulo de edición se hizo uso de la siguiente técnica de prueba que a continuación se describen según (Valdés Fernández, et al., 2017):

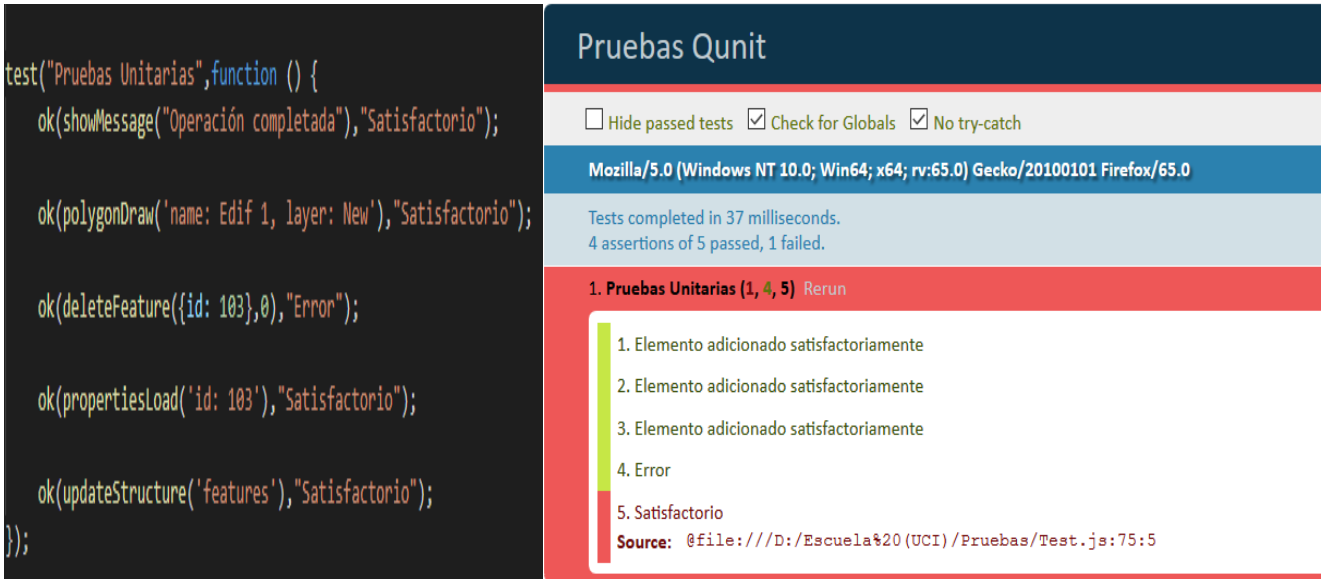
- **Partición de Equivalencia**: “*Esta técnica consiste en clasificar las entradas de datos del sistema en grupos que presentan un comportamiento similar. Las particiones también pueden definirse en función de las salidas de datos, valores internos, valores relacionados antes o*

después de ciertos eventos, y también para los valores que reciben las interfaces". (Terrera, 2017)

El objetivo fundamental de esta técnica es dividir los valores válidos y no válidos para entradas y salidas en un número de reducido de particiones, de forma tal que el comportamiento del software sea el mismo para cualquier valor contenido en una partición particular. El propósito principal de una partición es reducir la cantidad de casos de prueba generados en el proceso.

3.5 Aplicación y resultados de las pruebas.

Para llevar a cabo las pruebas unitarias se utilizó la herramienta QUnit v1.11.0 la cual basa su funcionamiento en una clase test que posibilita probar el código de aplicaciones web desarrolladas en JavaScript. Estas fueron desarrolladas constantemente a la par de la implementación de las funcionalidades, lo que posibilitó una correcta implementación de todos los requisitos del módulo. A continuación, se muestran los resultados de las pruebas unitarias asociadas a los métodos de los requisitos: "Listar los atributos del objeto espacial" y "Modificar los atributos de los objetos espaciales"



Luego de ter finalizada dicha prueba, los errores identificados fueron solucionados, y repetido este proceso de manera cíclica hasta no encontrar no conformidades en el código de la solución propuesta. Posteriormente, se hace necesaria la realización de pruebas de integración. Estas se realizaron con la finalidad de validar la compatibilidad y el funcionamiento de las interfaces que comunican las diferentes partes que componen la solución informática. Para su realización se probó el módulo en la plataforma ULTRON, El desarrollo de las pruebas garantizó que el módulo diseño de mapas de impresión presentaba un correcto funcionamiento una vez integrado al sistema.

Para comprobar la usabilidad del módulo de edición de capas vectoriales se empleó las pruebas con usuarios expertos basada en la técnica evaluaciones heurísticas. Se tuvo que seleccionar 3 especialistas del proyecto: jefe de proyecto, arquitecto de software y administrador de la calidad.

"*Estos especialistas interactuaron con el sistema y ejecutaron las funcionalidades para lo cual fue diseñado, con el fin de evaluar la interfaz a través de las heurísticas de diseño descritas por Jakob Nielsen, en su libro Usability Engineering*" (Rojas Rodríguez, et al., 2017).

El equipo de desarrollo fue tomando nota de los errores que tenía el sistema en función de sus funcionalidades. Estas heurísticas estuvieron definidas por 9 criterios dados por dicho autor, las cuales se muestran a continuación:

ID	Principio	Definición
1	Visibilidad del sistema	**El sistema debe mantener siempre informado al usuario sobre lo que está ocurriendo, a través de retroalimentación dentro de un tiempo razonable**
2	**Coincidencia entre el sistema y el mundo rea**	El sistema debería “hablar” el idioma del usuario, con palabras, frases y conceptos familiares, más que términos orientados al sistema. Seguir convenciones, de modo que la información parezca lógica y natural
3	**Control y libertad del usuario**	Los usuarios eligen a veces funciones del sistema por error, y necesitarán una salida de emergencia, con opciones de deshacer y rehacer.
4	**Prevención de errores**	Mucho más adecuado que mostrar mensajes de error entendibles, es un diseño cuidadoso que evite la ocurrencia de problemas. Se deben eliminar estas situaciones presentando una opción de confirmación a los usuarios antes de que realicen la acción.
5	**Consistencia y estándares**	Los usuarios no deberían preguntarse si diferentes palabras, situaciones o acciones significan lo mismo. Se deben seguir las convenciones de la plataforma
6	**Minimizar la carga de memoria**	Reconocimientos, más que recordatorios. Minimizar la carga de memoria de los usuarios mediante el uso de objetivos, acciones y opciones visibles. El usuario no debería recordar información de una parte del sistema a otra
7	**Flexibilidad y eficacia del uso**	Aceleradores que pasan desapercibidos para los usuarios novatos, pero que deben agilizar la interacción con el sistema a los usuarios expertos. Debe facilitar a los usuarios la ejecución de acciones frecuentes
8	**Diseño estético y minimalista**	minimalista El sistema no debe mostrar información que sea poco relevante o que rara vez sea utilizada por el usuario.
9	**Ayuda al usuario para reconocer, diagnosticar y recuperarse de errores**	Los mensajes de error deben estar expresados en un lenguaje natural entendible a los usuarios (no en código o lenguaje máquina). Estos deben indicar de manera precisa el problema y sugerir una solución de forma constructiva.

Para una mejor representación se lleva a cabo dicha prueba, estableciendo una lista de chequeo con un conjunto de requerimientos formulados en las 9 categorías. A continuación, se muestran los ejemplos para 2 de ellas:

Forma de uso “Evaluación”: El mismo se evalúa de 0 en caso de mal (cuando la respuesta al indicador sea “No”) y 1 en caso que elemento revisado no presente errores (cuando la respuesta al indicador sea “Sí”).

Visibilidad del sistema:

- ❖ Mantiene la homogeneidad de estilo con el resto de elementos del sitio web.
- ❖ Se ubica en los lugares preestablecidos, sin romper con la composición estándar.
- ❖ Las etiquetas de los menús son descriptivas de cada una de las opciones.
- ❖ No incluye más de ocho opciones, o si lo hace, existen subcategorías.
- ❖ Favorecen la previsibilidad, usando términos que anticipen al usuario lo que se encontrará detrás

- ❖ Su longitud se adecua a la disponible en cada caso (según se trate de un menú o una cabecera).
- ❖ Tienen un tratamiento gráfico coherente según el tipo de etiqueta: color, tamaño y tipografía adecuados.
- ❖ Mantiene la alineación entre los campos para asegurar la armonía visual.

Diseño estético y minimalista:

- Los elementos (bloques de texto, imágenes, tablas) quedan alineados con las líneas que crean los elementos estructurales de la página: líneas de cabecera, logotipo o menús.

- La composición de los elementos responde a formas rectangulares apaisadas, pues son las que más armonía generan y más se adaptan a las líneas maestras del monitor.
- El orden de los elementos se ajusta a una línea visual que va de izquierda a derecha y de arriba a abajo.

Los elementos se sitúan en la línea visual de forma jerárquica (cuanto más abarcan, antes se encuentran).

- Los elementos visuales (texto, tablas o gráficos) se adecuan a la paleta de colores establecida para el sitio web.
- Hay suficiente contraste entre el color del texto y el color de fondo, tanto en textos convencionales como dentro de tablas o diagramas.
- Los íconos utilizados evocan lo representado sin lugar a equívocos ni lecturas ambiguas.
- Guardan coherencia y homogeneidad con el resto de iconos del sitio web.
- Se usan fondos de color blanco o tonos tenues (gris, crema, azul pastel) y nunca de colores vivos.
- Si se usan colores para representar datos u otras informaciones en diagramas, no se contrastan verdes sobre rojos o marrones, pues es la forma de daltonismo más habitual.

Una vez ter realizada la prueba, a partir de las descripciones anteriormente descrito se puede decir que, de un total de 18 indicadores de usabilidad, el módulo implementado cumple con 19 indicadores, incumpliendo en la composición de los elementos en formas rectangulares apaisadas, cifra que representa el 95% de usabilidad para las funciones presentes. Para una mejor comprensión de los resultados obtenidos se han representado los mismos en la siguiente gráfica.

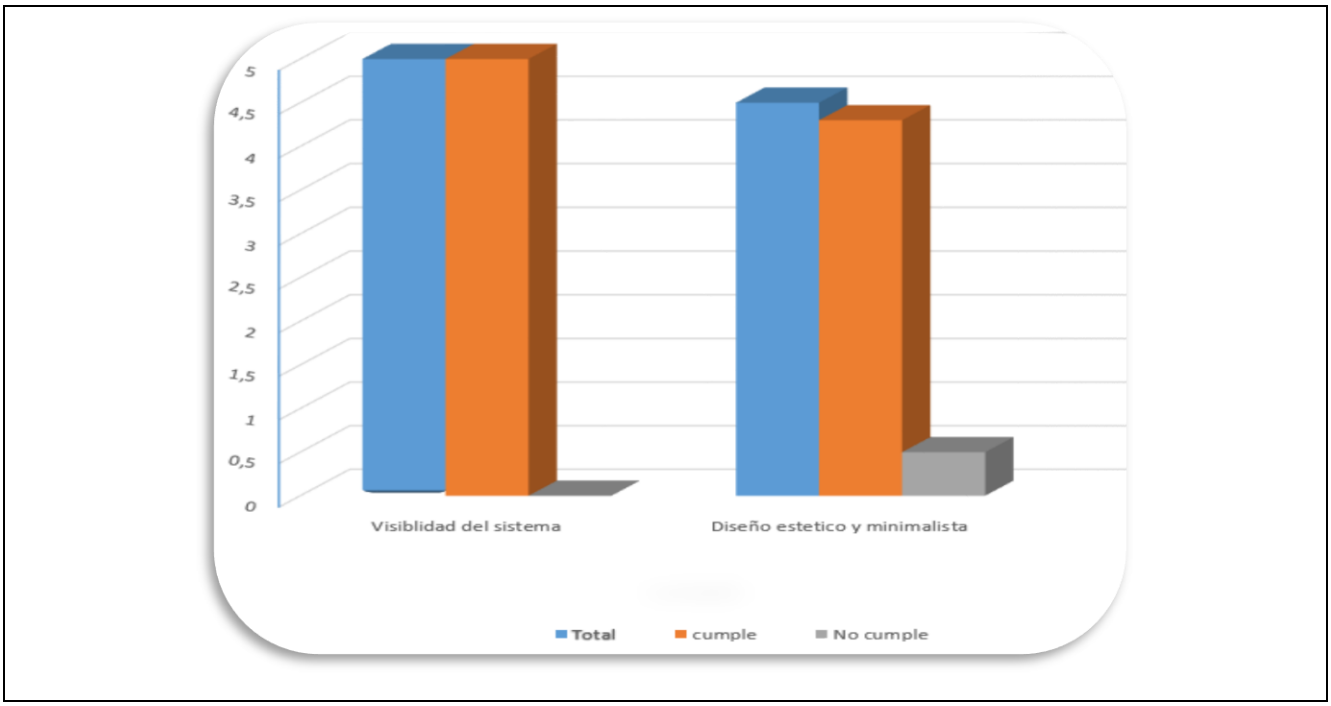


Fig. 7: Cumplimiento de los indicadores de las categorías de usabilidad.

Los resultados obtenidos fueron satisfactorios ya que los especialistas manifestaron que el módulo presentaba una interfaz sencilla e intuitiva. Por último, los expertos involucrados en la prueba expresaron el estar satisfechos con el diseño del módulo.

Para la realización de las pruebas de carga se empleó la herramienta Apache JMeter. El ambiente de prueba estuvo conformado por:

- Sistema Operativo: Windows 10
- Microprocesador: AMD E-300 @1.30 GHz
- Memoria RAM: 12 GB
- Disco Duro: 1T

Los resultados de las pruebas de carga se consideran satisfactorios. Esto es debido a que los tiempos de respuesta del servidor ante la interacción de 20, 60 y 100 usuarios concurrentes se encontraron en

el rango de tiempo de 1 a 6 segundos. Para ello fueron utilizados las principales funciones del módulo con un volumen de carga marcado por geometrías de hasta los 1000 vértices. Con ello queda demostrado que la propuesta de solución es estable, ya que se mantuvo prestando servicios todo el tiempo, sin incurrir en fallos.

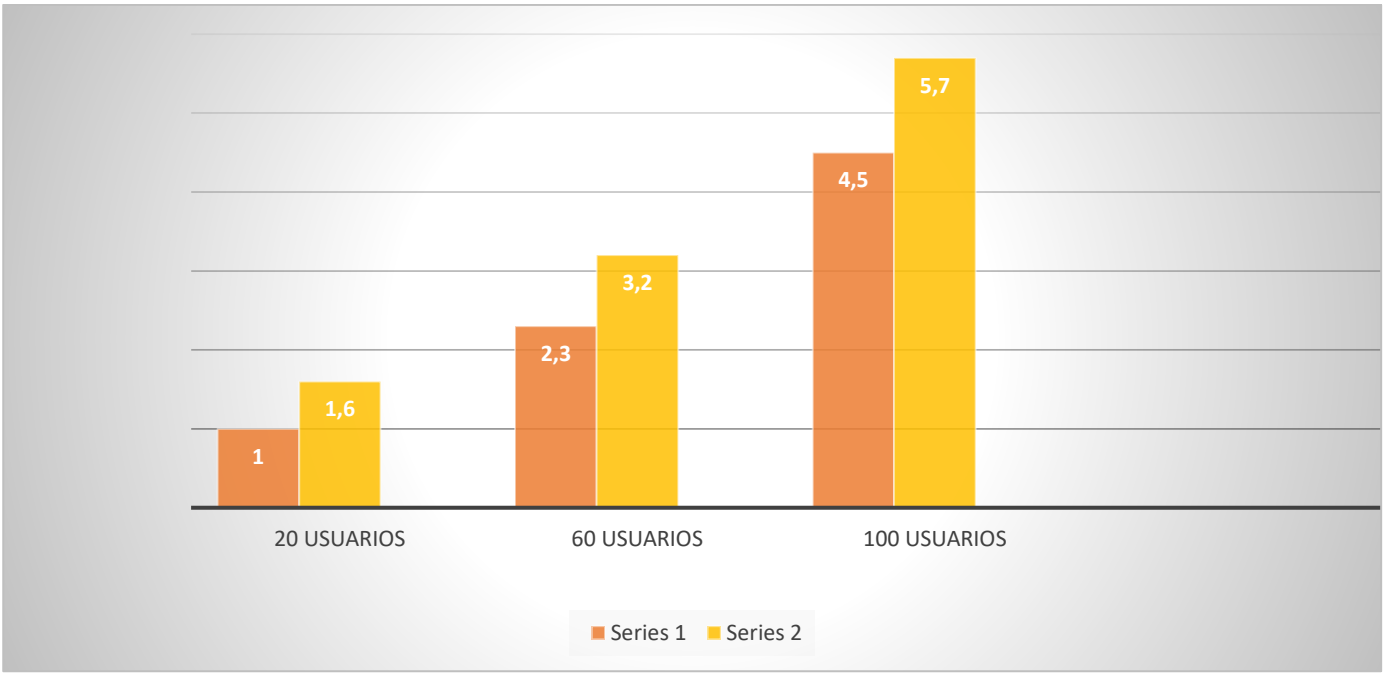


Fig. 8: Resultados de la prueba de carga para usuarios concurrentes

Las pruebas de caja negra o funcionales son realizadas a la interfaz del software y aplicadas a las funcionalidades del mismo. Como parte del proceso de dichas pruebas al módulo se diseñaron los casos de prueba en correspondencia con las descripciones de las historias de usuario. A continuación, se presentan la Descripción del Caso de Prueba correspondiente a los requisitos Editar propiedades del documento y Adicionar etiqueta de texto.

Tabla 6: Descripción de variables del caso de prueba correspondiente a los requisitos listar atributos de los objetos espaciales y modificar los atributos del objecto espaciales

NO.	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	**Nombre**	Campo de texto	NO	

				Debe introducirse cualquier combinación alfanumérica menor de 30 caracteres.
2	**tipo**	Lista desplegable	NO	Campo que permite seleccionar el formato, o sea seleccionar el tipo de capa a editar, punto, lineo o polígono
3	**Descripción**	Campo de texto	SI	Campo que permite mostrar las descripción o informaciones que bebe tener el tipo de capa.
4	**capa**	Lista desplegable	NO	Campo que permite mostrar el nombre de la capa a la que pertenece el objeto espacial.

Tabla 7: Descripción del Caso de Prueba para los RF: listar atributos de los objetos espaciales y modificar los atributos del objecto espaciales.

Escenario	Descripción	Nombre	Tipo	Descripción	Capa	Respuesta Del Sistema	Flujo Central

EC 1 Listar atributos del objetos espaciales correctamente	El usuario introduce todos los parámetros correctamente para editar propiedades del documento	V (Objecto 1)	V (Polígono)	V (dibujar un barrio)	V (are: Objecto 1)	Modifica las propiedades de objecto espaciales	Se selecciona la opción "edición de capas vectoriales". Se introducen los datos de los objetos espaciales. Se dibuja el área de edición. Se presiona el botón "Guardar".
EC 2 Listar los atributos del objetos espaciales con campos obligatorios vacíos	El usuario intenta llenar el listado con campos obligatorios vacíos	vacío	V (Polígono)	V (dibujar un barrio)	V (are: Objecto 1)	El sistema muestra el mensaje: "El campo nombre es de carácter obligatorio.	Se selecciona la opción "edición de capas vectoriales". Se introducen los datos de los objetos espaciales. Se dibuja el área de edición. Se presiona el botón "Guardar".
EC 3 listar atributos del objetos espaciales con datos incorrectos.	El usuario intenta Listar los atributos del objecto espaciales con datos incorrectos.	Módulo se diseñaron los casos de prueba en 74 fase Correspondencia	V (Polígono)	V Carga marcada por geometrías de hasta los 1000 vértices. Con ello queda	V (are: Objecto 1)	Para todos los casos el sistema mantiene la interfaz y muestra un mensaje de error. En el primer caso "El campo	Se selecciona la opción "edición de capas vectoriales". Se introducen los datos de los

		con las descripciones		demostrado que la propuesta de solución es estable, ya que se mantuvo prestando servicios todo el tiempo, sin incurrir en fallos.		nombre admite hasta 30 caracteres". En el segundo caso "El campo descripción admite hasta 120 caracteres".		objetos espaciales. Se dibuja el área de edición. Se presiona el botón "Guardar". .	

En el proceso de aplicación de las pruebas se realizaron tres iteraciones utilizando el caso de prueba diseñado para detectar la mayor cantidad de errores en el funcionamiento del módulo.

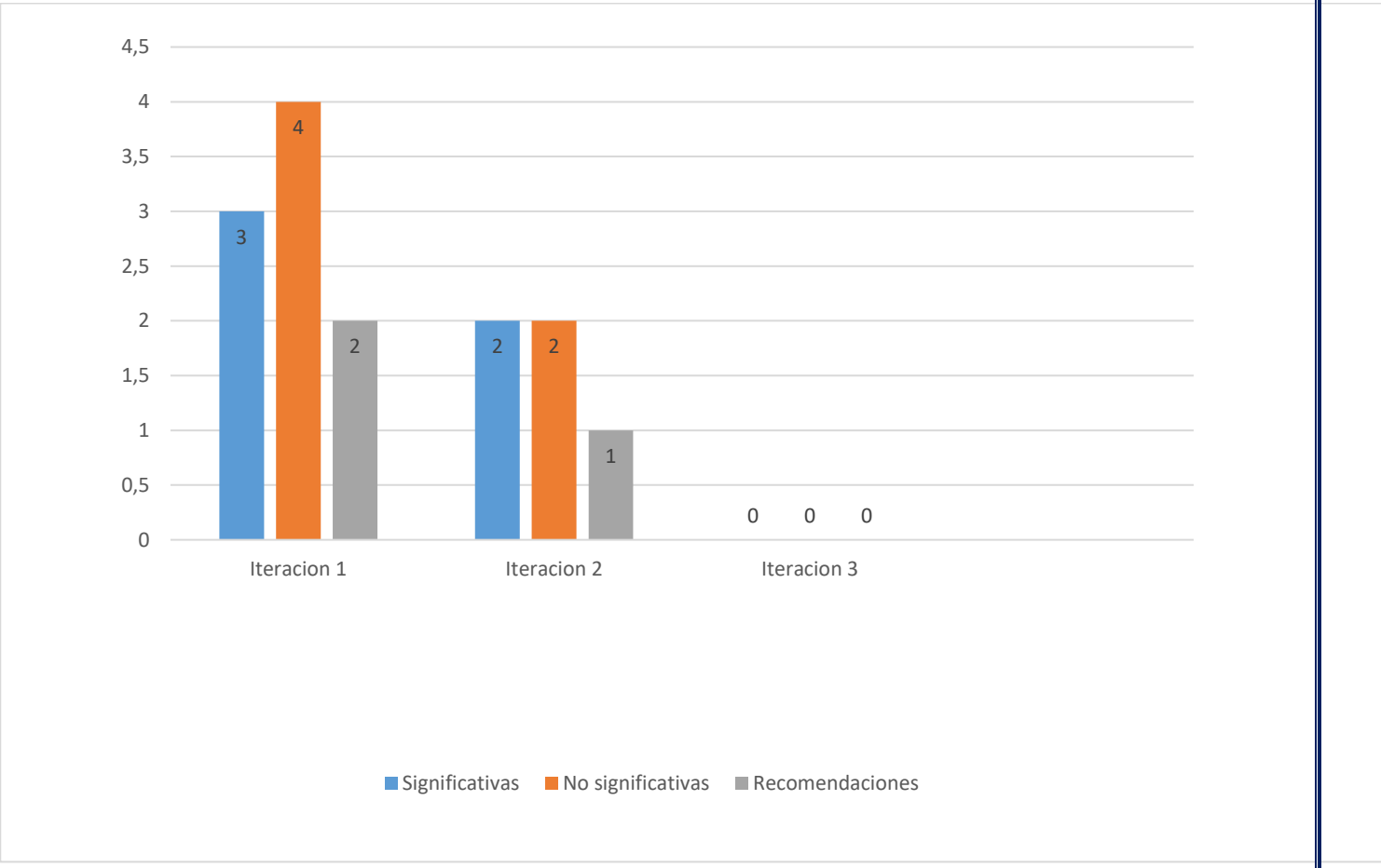


Fig. 9: Gráfico de no conformidades detectadas al realizar las pruebas.

En la Fig. 9 se evidencian las no conformidades detectadas al realizar las pruebas, estas fueron clasificadas atendiendo al nivel de afectación en el funcionamiento del módulo, en significativas, no significativas y recomendaciones. Al realizarse la primera iteración se detectaron 9 no conformidades de las cuales 3 fueron significativas al afectar el funcionamiento del módulo; como no significativas se detectó 1 error ortográfico, 3 errores de mensajes y las 2 restante fueron recomendaciones para mejorar el diseño del módulo. Una vez concluida la corrección de los errores detectados, se inicializó la segunda iteración donde se identificaron 2 no conformidades clasificadas en 2 significativa, 1 errores ortográficos en el caso de las no significativas y la restante fue 1 recomendación para solucionar los errores. Al terminar de solucionar los errores detectados se finalizó la segunda iteración y se dio inicio a una tercera iteración para continuar comprobando si existían errores. Al culminar la misma no se detectaron no conformidades.

3.6. Resultado obtenido

Como resultado del desarrollo del módulo de edición de capas vectoriales vectorial para la plataforma ULTRON v 2.0 del centro CREAD se obtuvo una aplicación que brinda a los usuarios la posibilidad de crear y modificar objetos vectoriales, así como los datos asociados a estos. La solución posibilita la corrección de los errores topológicos resultantes de la digitalización manual de los mapas a partir de la implementación de funcionalidades como la adición, eliminación, rotación y traslación de las geometrías (puntos, líneas, polígono).

El módulo ofrece una solución novedosa que minimiza el consumo de los recursos de la PC Cliente y el tiempo de espera resultante de realizarla edición de capas vectoriales, haciendo uso de un tipo de dato espacial denominado Image_vector (imágenes vectoriales). Este permite cargar las geometrías en la capa a editar edición de capas vectoriales.

Fig. 10: Interfaz del módulo de diseño modelo edición de capas vectoriales

Conclusiones parciales.

- La aplicación de patrones arquitectónicos y de diseños permitieron implementar el módulo de edición de capas vectoriales con una adecuada estructura ingenieril.
- La estrategia de pruebas aplicada permitió valorar el nivel de calidad que tiene el módulo de edición de capas vectoriales. Según el criterio de expertos aplicado, el módulo es considerado como adecuado por el 95 % de los especialistas consultados.

CONCLUSIONES GENERALES

Una vez finalizada la investigación y el desarrollo del presente trabajo se concluye que:

- Los estudios teóricos analizados permitieron determinar que los requisitos funcionales y no funcionales básicos para la edición de capas vectoriales.
- La aplicación del escenario historia de usuario de la metodología de desarrollo de software AUP - UCI posibilitó guiar el proceso de desarrollo del módulo acorde a las normas vigentes en la UCI.
- El diseño de una arquitectura modelo – vista - controlador y de los patrones de diseño GRASP y GOF permitieron obtener un módulo para la educación de capas vectoriales.
- Se desarrolló un módulo para la edición de capas vectoriales al cual se le aplicó una estrategia de pruebas comprobando sus funcionalidades y constatándose cumple lo estipulado en sus requisitos funcionales.

RECOMENDACIONES

Una vez implementada la solución los autores recomiendan:

- se recomienda la inclusión de nuevas funcionalidades a la aplicación que permitan la adición de capas vectoriales. Tales como: Duplicar geometría, cortar geometría, pegar geometría.

- Incorporar nuevas funcionalidades al módulo propuesto que permitan realizar la edición de los tipos de datos espaciales multilíneas y multipolígonos.

- Desarrollar nuevas capacidades en el módulo para posibilitar la edición de capas vectoriales de tipo ráster.

REFERENCIAS BIBLIOGRÁFICAS

1. **Alvaro. 2016.** gvSIG blog. [En línea] 14 de Octubre de 2016. [Citado el: 11 de Diciembre de 2017.] https://blog.gvsig.org/2016/10/14/gvsig-desktop-2-3-manual-en-espanol-disponible/.

2. **Arenas Quiñones, Carmen Liliana, Gómez Santamaría, Patricia y Isaza Rengifo, Julián Yesid. 2017**. Módulo en ambiente web para la gestión del recurso hídrico en concesiones de aguas superficiales, haciendo uso de SIG. Manizales: s.n., 2017.

3. **Arias, Ángel y Durango, Alicia. 2016.** Ingeniería y Arquitectura del Software: 2ª Edición. s.l.: IT Campus Academy, 2016. ISBN: 978-1523365487

4. **Zea Ordóñez, Mariuxi Paola, Molina Ríos, Jimmy Rolando y Redrován Castillo, Fausto Fabían. 2017.** Administración de bases de datos con PostgresSQL. Alicante: Área de Innovación y Desarrollo, S.L., 2017. ISSN: 978-84-946684-6-3.

5. **Arango Isaza, Fernando, Álvarez Eraso, Danny Alejandro y Moreno Arboleda, Francisco Javier. 2016.** Una propuesta para la formalización del diagrama de clases en el lenguaje Maude. 2016. ISBN: 2256-5353.

6. **Anasanz, 2016.** Sisitemas de Información Geográfica. [En línea] 10 de Diciembre de 2014. [Citado el: 16 de septembre de 2017.] http://tecamb-sig.blogspot.com/.

7. **Arias, Ángel y Durango, Alicia. 2016.** Ingeniería y Arquitectura del Software: 2ª Edición. s.l.: IT Campus Academy, 2016. ISBN: 978-152336548

8. **Athan, Tara, Blazek, Radim y Ersts, Peter. 2014.** QGIS: Un Sistema de Información Geográfica libre y de Código Abierto. [En línea] 2014. [Citado el: 28 de Noviembre de 2017.]
9. https://docs.qgis.org/2.8/en/docs/gentle_gis_introduction/vector_data.html

10. **Báez, Olga Lucia Cárdenas. 2016.** Automarización de casos de prueba para mejorar el proceso de calidad de software. Bogotá : s.n., 2016.

11. **Benítez, Yari Guillermo. 2012.** Sisitemas de Información Geográfica. [En línea] 10 de Diciembre de 2012. [Citado el: 15 de Junio de 2018.] http://tecamb-sig.blogspot.com/.

12. **Burrough, P. A. 1989**. Principies of Geographical Information Systems for Land Resource Assesment. Oxfordshires.n.,1989

13. **Burrough, Peter A., MCDONNELL, Rachael A. y LLOYD, Christopher D. 2015**. Principles of geographical information systems. s.l. : Oxford University Press, 2015. ISBN: 978-0-19-874284-5.

14. **Calzado Maceo, Rafael. 2015**. Sistema de Administración para el Sistema de Información Geográfica de la Universidad de las Ciencias Informáticas. La Habana : s.n., 2015.

15. **Cebrián, J. A. 1988.** Sistemas de Información Geográfica. Madrid : Síntesis, 1988

16. **Cebrián, Juan Antonio. 1994**. GIS concepts. 1994. 84-604-8313-4.

17. **Centeno Defas , Elvis Paúl y Cordonez Suntasig , Sergio Paúl. 2016.** Implementación de un sistema de gestión documental administrativa. Latacunga, Ecuador. : s.n., 2016

18. **Centro de Investigación y Desarrollo en Información Geográfica. 2016**. Cartografía - GISTeledelección. [En línea] 15 de Noviembre de 2016. [Citado el: 14 de Marzo de 2018.] http://cartografiagis-teledeteccion.blogspot.com/2016/

19. **Cervera, Leiny Ruiz. 2015**. Especificación de los requisitos funcionales del sistema 2.0. GeneSIG. La Habana : s.n., 2015

20. **Cevallos, Karla. 2015.** UML: Casos de Uso. [En línea] 04 de Junio de 2015. [Citado el: 02 de Marzo de 2018.] https://ingsotfwarekarlacevallos.wordpress.com/2015/06/04/uml-casos-de-uso/.

21. **Chang, Kang-Tsung. 2017**. Geographic Information System. The International Encyclopedia of Geography. 2017

22. **Chang, Kang-Tsung. 2015**. Introduction to geographic information systems. s.l. : McGraw-Hill Science/Engineering/Math, 2015.

23. **Crestaz, E., y otros. 2015** . Advancements in concurrent native spatial database technology for groundwater monitoring and modeling applications. A case study aimed at PostgreSQL-PostGIS coupling with GIS and Feflow. 2015 .

24. **Delgado, Dayana H. Bailón. 2015**. Lenguaje Unificado de Modelado - UML. 2015

25. **Domínguez Vera, Edgar Danilo , y otros. 2015.** Reglas de calidad para la codificación estandarizada en lenguaje C: Una propuesta para la enseñanza a nivel superior. 2015. ISBN: 2395-9029

26. **Dorta, Maikel José Rivero. 2016**. AngularJs Paso a Paso: Segunda Edición. 2016

27. **Espáriz, Germán Gómez. 2014** . Normas de edición de MTN25. Versión 1.3. s.l.: Instituto Geográfico Nacional, 2014 .

28. **Evers, Jeannie. 2017**. National Geographic. [En línea] 21 de Junio de 2017. [Citado el: 08 de Marzo de 2018.] https://www.nationalgeographic.org/encyclopedia/geographic-information-system-gis/.

29. **Fuentes Hernández, Julio César y Galeana Piedra, Selene Nallely. 2013**. Sistema en línea para la evaluación docente en una institución educativa. México D.F : s.n., 2013.

30. **García Velázquez, Luis Ambrosio. 2016**. RECI Revista Iberoamericana de las Ciencias Computacionales e Informática. [En línea] Julio de 2016. [Citado el: 07 de Marzo de 2018.] http://reci.org.mx/index.php/reci/article/view/49/221.

31. **Garro, Arkaitz. 2015**. HTML5. 2015

32. **Gil, Luis Alexander Aldazabal. 2015**. Code Design – Principios GRASP parte 1. [En línea] 02 de Octubre de 2015. [Citado el: 05 de Abril de 2018.] https://code2read.com/2015/10/02/code-design-csharp-oodprincipios-grasp-parte1/.

33. **González, Sergio Fresneda. 2016**. Aplicación de patrones de diseño para la resolución de problemas de software en el desarrollo de una aplicación móvil iOS. 2016

34. **Goodchild, M. F. 1985**. Geographic Information Systems in Undergraduate Geography: A Contemporary Dilemma. 1985.

35. **Gutiérrez Puebla, Javier y Gould, Michael. 1994**. SIG, sistema de información geográfica. Madrid : Síntesis, 1994. 84-7738-246-8

36. **Hernández Sampieri, Roberto, Fernández Collado, Carlos y Baptista Lucio, María del Pilar. 2010**. Metodología de la investigación. Quinta Edición. México : McGraw-Hill / Interamericana Editores, S.A., 2010. ISBN: 978-607-15-0291-9.

37. **Kellman, Peter y Hansen, Michael S. 2014.** Mapping in the heart: accuracy and precision. Austin, Texas : BioMed Central Ltd., 2014. ISSN: 1532-429

38. **Kiessling, Manuel. 2015**. Libro para principiantes en Node js. 2015.

39. **Kim, Changjoo. 2017**. Overlay, Topological. The International Encyclopedia of Geography. 1-4. s.l. : Wiley Online Library, 2017.
40. **Larrudet, Daniel R. Suárez. 2016**. Módulo para la gestión del catálogo anual de superación profesional y servicios académicos. La Habana : s.n., 2016.

41. **Lastres Romero, Blanca Mayra y Rico Blanco, Luis Orlando . 2013**. Módulo para la edición de una ontología del dominio geográfico en la plataforma GeneSIG. La Habana : s.n., 2013.

42. **Leempoel, Kevin , y otros. 2017**. Simple Rules for an Efficient Use of Geographic Information Systems in Molecular Ecology. Frontiers in Ecology and Evolution. 2017

43. **LLasac Huaca, Juan Carlos y Alvarado Sánchez, Daniel Fernando. 2015**. BistStrema. [En línea] 05 de Marzo de 2015. [Citado el: 09 de Marzo de 2018.] https://dspace.ups.edu.ec/bitstream/123456789/10112/1/UPS%20-%20ST001647.pdf

44. **López, Oscar Andrés Peña. 2015**. Formulación de propuesta para ejecución de pruebas funcionales en un sistema basado en PMI. 2015.

45. **Maceo, Rafael Calzado. 2015**. Sistema de Administración para el Sistema de Información Geográfica de la Universidad de las Ciencias Informáticas. La Habana : s.n., 2015

46. **Maida, Esteban Gabriel y Pacienzia, Julián. 2015**. Metodologías de desarrollo de software: Tesis de Licenciatura en Sistemas y Computación. [En línea] 2015. [Citado el: 08 de Marzo de 2018.] http://bibliotecadigital.uca.edu.ar/repositorio/tesis/metodologias-desarrollo-software.pdf.

47. **Medina, Cesar Julio Bustacara. 2016**. Model-View-Controller Pattern (POSA1).[En línea] 2016.[Citado el: 20 de Marzode2018.] https://sophia.javeriana.edu.co/~cbustaca/docencia/DSBP-201603/presentaciones/Model_View_Controller_Pattern.pdf

48. **Microsoft Corp. 2017**. Microsoft Docs. [En línea] 27 de Septiembre de 2017. [Citado el: 13 de Junio de 2018.] https://code.visualstudio.com/en-us/dotnet/core/tutorials/with-visual-studio-code

49. **Moreno López, Almudena. 2015**. Detección, Edición y Resolución de conflictos de edición de capas vectoriales en la representación a escala. Madrid : s.n., 2015

50. **Muños de la Torre, Arturo. 2013**. Introducción a NODE.JS a través de KOANS. 2013.

51. **Naharudin, N., Ahamad, M. S. S., & Sadullah, A. F. M. 2016**. GIS Data Collection for Pedestrian Facilities and Furniture Using MAPinr for Android. Kuala Lumpur : The International Archives of Photogrammetry, 2016

52. **Nallu, Surya. 2015.** Introduction to Express.js. [En línea] 06 de Febrero de 2015. [Citado el: 06 deDiciembre de 2017.] http://www.teach.cs.toronto.edu/~csc309h/winter/posted_tutorials/Introduction_to_Express.pdf.
53. **Olaya, Víctor. 2014**. Sistemas de Información Geográfica. 2014

54. **Onishi, Hirofumi. 2002**. Geographic information system. s.l. : U.S. Patent, 2002. ISBN: 6-389-356.

55. **Olaya, Víctor. 2014**. Sistemas de Información Geográfica. 2014

56. **Open Source Geospatial Foundation. 2018**. MapServer: Open source web mapping. [En línea] 14 de Junio de 2018. [Citado el: 19 de Junio de 2018.] http://mapserver.org/about.html

57. **Pavón, Santiago. 2017**. Desarrollo del Servidor Quiz Sequelize. [En línea] 2017. [Citado el: 06 de Diciembre de 2017.]

58. **Peño, José Manuel Sánchez. 2015**. Pruebas de Software. Fundamentos y Técnicas. Madrid: s.n., 2015.

59. **Prashanth, M., RAO, K. Nageswara y FAZAL, Shahab. 2017**. Fundamentals of Geographic Information system. 2017

60. **Pressman, Roger S. y Maxim, Bruce R. 2015**. Software engineering: a practitioner's approach, Eighth Edition. New York: McGraw-Hill Education, 2015. ISBN 978-0-07-802212-8

61. **Pressman, Roger S. 2017**. Software engineering: a practitioner's approach. New York: McGraw-Hill Education, 2017. ISBN 978-0-07-803917-9 https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjnlvOam_rpAhXoTTABHedfAvAQFjAAegQIBRAB&url=http%3A%2F%2Fcotana.informatica.edu.bo%2Fdownloads%2Fld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF&usg=AOvVaw3AEXe_YeKugMBXlttMfA3o

62. **Sabarí, Yidian Yosbel Castellanos. 2015**. GEGMA:Aplicación para la gestión y apoyo al control del proceso de enseñanza-aprendizaje de la Matemática Discreta 1. 2015.

63. **Sánchez, Luis Martín de Eugenio. 2017**. Los Sistemas de Información Geográfica (SIG) inducen a apreciaciones, valoraciones y conclusiones erróneas. Madrid : s.n., 2017. 1131-9100

64. **SEAN.JS. 2017. Sitio Oficial de SEAN.JS. [En línea] 2017**. [Citado el: 10 de Noviembre de 2017.] https://seanjs.org/

65. **Sernanp. 2015.** Manejo Básico de QGIS para el uso en Sistemas de Información Geográfica en ANP. Lima : s.n., 2015. 2015-16303

66. **Sommerville, Ian. 2016**. Software engineering: Tenth Edition. s.l. : Pearson Education, 2016. 978-0-13394303-0

67. **Trelles Labairu, Hilton y Pina Calafi, Alfredo. 2016**. Desarrolllo de un sitio web para clubs de lectura. Pamplona : s.n., 2016

68. **Trellini, Ariel. 2015**. Arquitectura y Diseño de Sistemas. 2015.

69. **UML. 2016. UML. [En línea] 2016**. [Citado el: 16 de Noviembre de 2017.] http://www.uml.org/.

70. **Valdés Fernández, Yudeisy y García Prieto, Osmín. 2017**. Subsistema de planificación y gestión de canales virtuales para el sistema de gestión y transmisión de calales virtuales. La Habana : s.n., 2017

71. **Valdés, Wilmer Eduardo Parra. 2014**. Integración de patrones de seguridad y patrones de diseño. Madrid : s.n., 2014

72. **Vega Prieto, Roexcy, Rodríguez Luis, Zaylí y Justo Morell, Yaneisi Ofelia. 2015**. Informática Jurídica. [En línea] 01 de Enero de 2015. [Citado el: 18 de Mayo de 2018.] http://www.informaticajuridica.com/trabajos/procedimiento-para-realizar-pruebas-de-usabilidad

73. **Visual Paradigm. 2015**. Visual Paradigm. What VP - UML Provides. [En línea] 2015. [Citado el: 20 de Octubre de 2015.] http://www.visual-paradigm.com/product/vpuml/provides

74. **Whitacre, James. 2017**. Geographic Information Systems 101: Understanding GIS. 2017.

75. **Younes, Georges, y otros. 2016**. Integration Challenges of Pure Operation-based CRDTs in Redis. Lisboa : s.n., 2016. ISBN: 978-1-4503-4775-4

76. **Zanin, Christine. 2014. HyperGEO - Diseño cartográfico. [En línea] 17 de Abril de 2014**. [Citado el: 21 de Noviembre de 2018.] http://www.hypergeo.eu/spip.php?article444#.

77. **Zsolt, MAGYARI-SASKA. 2015**. Developing and implementing multiuser, fully relational gis database for desktop systems using open source technologies. Geographia Technica. 2015.

GLOSARIO DE TÉRMINOS

Web Map Service (WMS): Definido por el OGC, produce mapas de datos referenciados espacialmente, de forma dinámica a partir de información geográfica. Este estándar internacional define un "mapa" como una representación de la información geográfica en forma de un archivo de imagen digital conveniente para la exhibición en una pantalla de ordenador

CRUD: Es el acrónimo de "Crear, Leer, Actualizar y Borrar" (del original en inglés: Create, Read, Update and Delete), que se usa para referirse a las funciones básicas en bases de datos o la capa de persistencia en un software

Módulo: Es una porción de un programa de ordenador. De las varias tareas que debe realizar un sistema para cumplir con su función u objetivos.

QT: Es un framework multiplataforma orientado a objetos. Usado para desarrollar programas que utilicen interfaz gráfica de usuario, así como también diferentes tipos de herramientas para la línea de comandos y consolas para servidores que no necesitan una interfaz gráfica de usuario.

C++: Es un lenguaje de programación orientado a objetos que toma la base del lenguaje C y le agrega la capacidad de abstraer tipos como en Smalltalk.

Píxeles: Es la unidad más pequeña y diminuta de una imagen digital. Cada píxel es una unidad homogénea de color, con una importante variación de colores dan como resultado una imagen más o menos compleja.

BSD: Se define como “Berkeley Software Distribution”. Es el nombre de las distribuciones de código fuente de la Universidad de California, Berkeley, que originalmente eran extensiones del sistema operativo UNIX® de AT&T Research.

SQL: Es un lenguaje específico del dominio utilizado en programación, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales.

Plugins: Es aquella aplicación que, en un programa informático, añade una funcionalidad adicional o una nueva característica al software. Por lo tanto, puede nombrarse al plugin como un complemento.
ECMAScript: Es una especificación de lenguaje de programación publicada por ECMA International. Es el estándar que sigue JavaScript desde junio de 2015. Actualmente está aceptado como el estándar ISO 16262.

API: Son un conjunto de comandos, funciones y protocolos informáticos que permiten a los desarrolladores crear programas específicos para ciertos sistemas operativos. Estas permiten al informático usar funciones predefinidas para interactuar con el sistema operativo o con otro programa.

ANEXOS

Anexos 1: Entrevista.

La entrevista fue realizada a los especialistas del proyecto ULTRON del centro CREAD, específicamente 4 trabajadores de este equipo de desarrollo (jefe de proyecto, Analista y 2 Programadores).
Objetivo de la entrevista: “Obtención de los requisitos y la base tecnológica para la implementación de la solución”.
Preguntas:
¿Qué es la plataforma ULTRON?

¿Qué sistema gestor para el tratamiento de datos utiliza la plataforma ULTRON?

¿Por qué se necesita en la plataforma ULTRON funcionalidades de edición de capas vectoriales?

¿Qué requerimientos considera que deba poseer la solución a implementar?

¿Qué tipo de modelo de datos se gestionarían?

¿Qué herramientas se utilizan en el desarrollo de la plataforma y cuáles recomienda, además, para el desarrollo de dicho módulo?

¿Qué funcionalidades y características considera que sean importantes para la implementación del módulo?

Anexos 2: Descripciones de las Historias de Usuarios del módulo

Tabla 8: Descripción de la Historia de Usuario: “Dibujar línea”

Número: 06	**Nombre del Requisito: “Dibujar línea”**
Programador: Wilson Félix Cassuque cambila	Iteración Asignada: 1

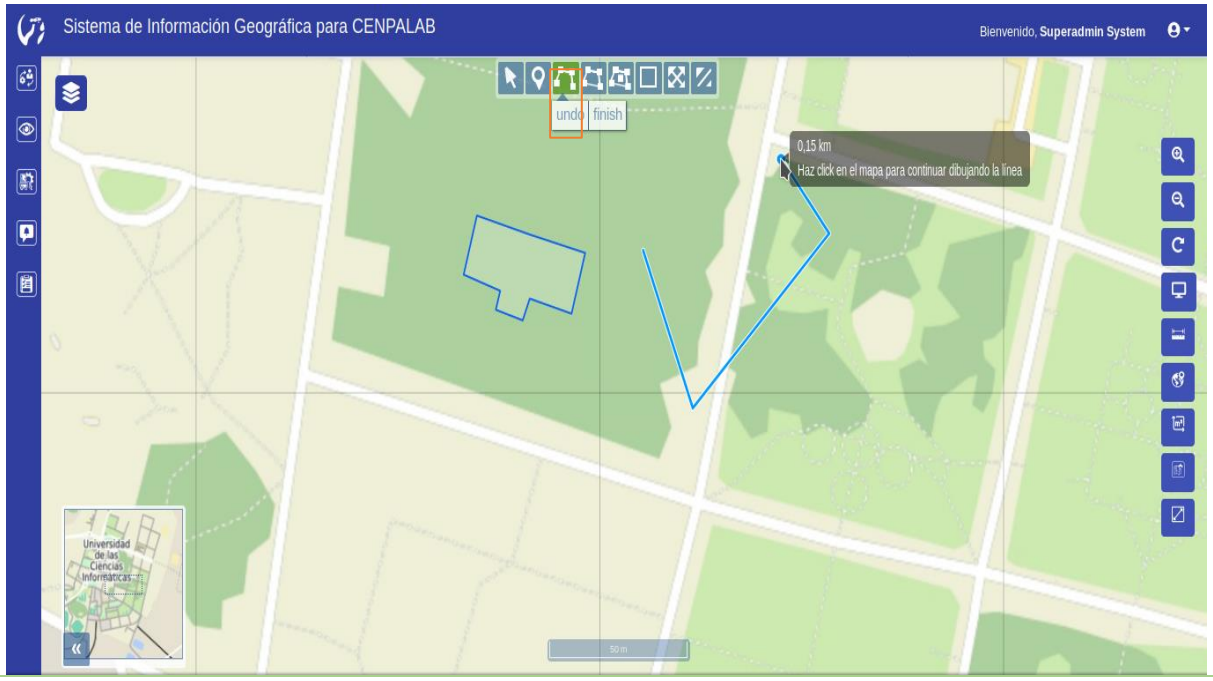
Prioridad: **Alta**	Tiempo estimado: 7 días
Riesgo en desarrollo: • Planificación irreal. • Interrupción del servicio eléctrico. • Afectaciones externas al personal de trabajo.	Tiempo Real: 4 días
Descripción: El módulo debe permitir al usuario dibujar una línea, para ello el usuario selecciona la opción línea presente en el menú y con el mouse, dibuja una línea en el mapa cuya longitud y dirección varia en correspondencia a la ubicación del puntero	
Observaciones:	
Prototipo de Interfaz: 	

Tabla 9: Descripción de la Historia de Usuario: “Dibujar polígono”

Número: 07	Nombre del Requisito: “Dibujar polígono”
Programador: Wilson Félix Cassuque cambila	Iteración Asignada: 1
Prioridad: **Alta**	Tiempo estimado: 4 días
Riesgo en desarrollo: • Planificación irreal. • Interrupción del servicio eléctrico. • Afectaciones externas al personal de trabajo.	Tiempo Real: 6 días

Descripción: El módulo debe permitir al usuario dibujar un polígono, para ello el usuario selecciona la opción línea presente en el menú y con el mouse, dibuja una línea en el mapa cuya longitud y dirección varia en correspondencia a la ubicación del puntero

Observaciones:

Prototipo de Interfaz:

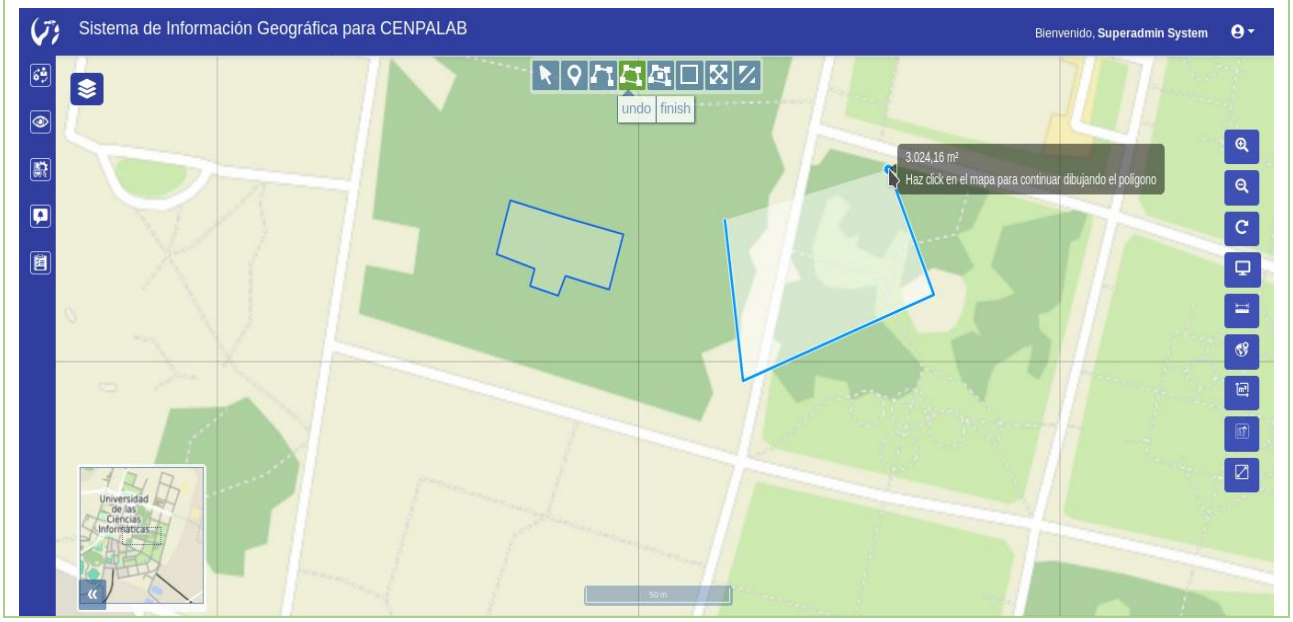


Tabla 10: Descripción de la Historia de Usuario: “Rotar geometría”

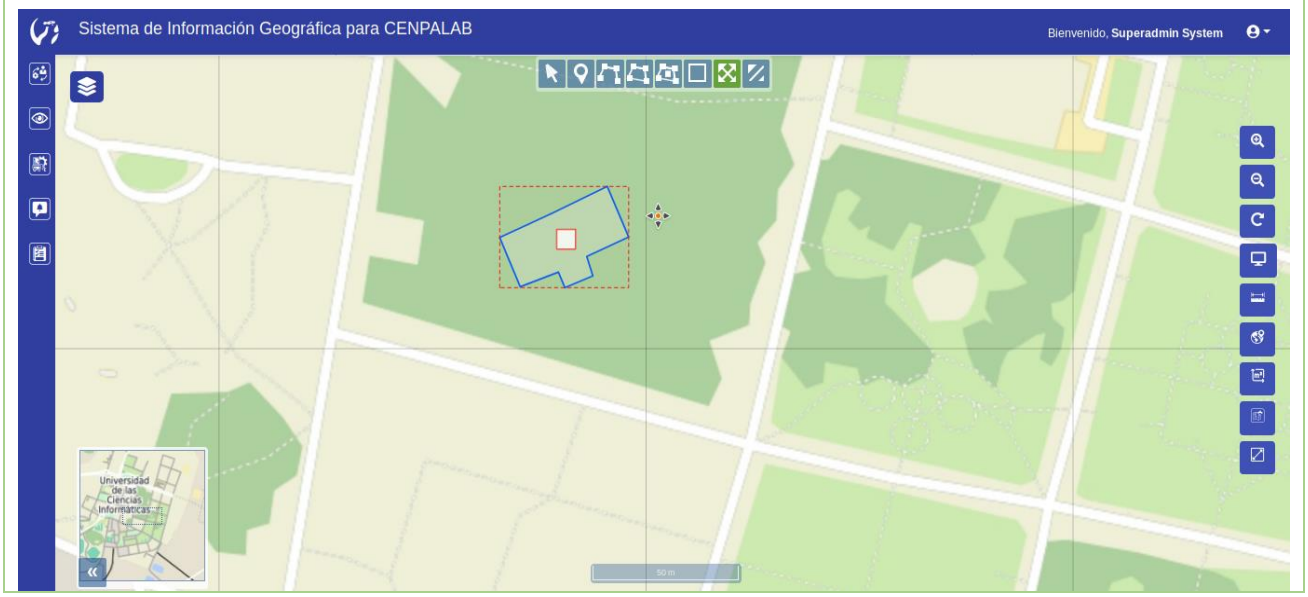
Número: 08	**Nombre del Requisito: “Rotar geometría”**
Programador: Wilson Félix Cassuque cambila	Iteración Asignada: 1
Prioridad: **Alta**	Tiempo estimado: 9 días
Riesgo en desarrollo: • Planificación irreal. • Interrupción del servicio eléctrico. • Afectaciones externas al personal de trabajo.	Tiempo Real: 9 días
Descripción: El módulo debe permitir que el usuario seleccione una geometría en el mapa y una vez seleccionado pueda rotarlo cada 45º	
Observaciones:	
Prototipo de Interfaz: 	

Tabla 11: Descripción de la Historia de Usuario: “Trasladar geometría

Número: 09	**Nombre del Requisito: “Trasladar geometría”**
Programador: Wilson Félix Cassuque cambila	Iteración Asignada: 1
Prioridad: **Alta**	Tiempo estimado: 8 días
Riesgo en desarrollo: • Planificación irreal. • Interrupción del servicio eléctrico. • Afectaciones externas al personal de trabajo.	Tiempo Real: 7 días
Descripción: Debe el módulo transformar múltiples geometrías, en cualquier lado de la geometría, poner un mejor ajuste	
Observaciones:	
Prototipo de Interfaz: 	

Tabla 12: Descripción de la Historia de Usuario: “Compensar geometría”

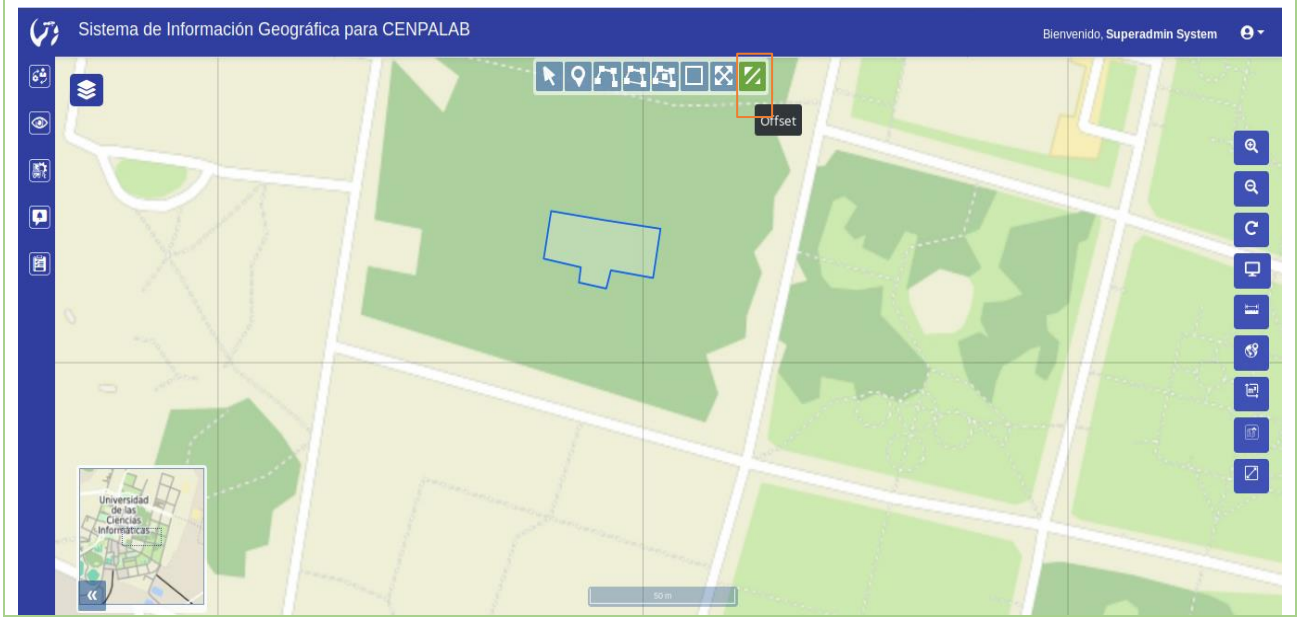
Número: 10	**Nombre del Requisito: “Compensar geometría”**
Programador: Wilson Félix Cassuque cambila	Iteración Asignada: 1
Prioridad: **Alta**	Tiempo estimado: 9 días
Riesgo en desarrollo: • Planificación irreal. • Interrupción del servicio eléctrico. • Afectaciones externas al personal de trabajo.	Tiempo Real: 5 días
Descripción: El módulo le debe permitir al usuario que una vez dibujada una geometría en el mapa poderlo redimensionar en su misma estructura	
Observaciones:	
Prototipo de Interfaz: 	

Tabla 13: Descripción de la Historia de Usuario: “Dibujar agujero en un polígono”

Número: 11	**Nombre del Requisito: “Dibujar agujero en un polígono”**
Programador: Wilson Félix Cassuque cambila	Iteración Asignada: 1
Prioridad: **Alta**	Tiempo estimado: 5 días
Riesgo en desarrollo: • **Planificación irreal.** • **Interrupción del servicio eléctrico.** • **Afectaciones externas al personal de trabajo.**	Tiempo Real: 3 días
Descripción: Debe el sistema permitir dibujar agujero en un polígono señalado por el usuario previamente	
Observaciones:	

Prototipo de Interfaz:

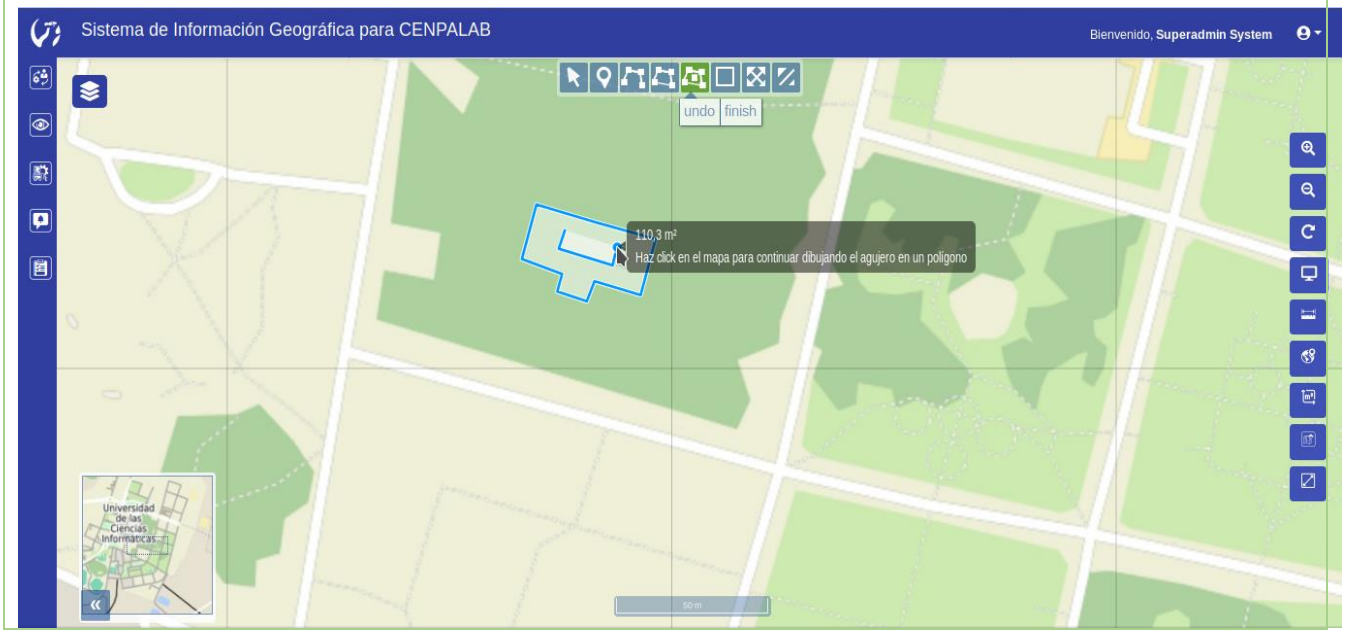


Tabla 13: Descripción de la Historia de Usuario: “Transformar múltiples geometrías”

Número: 12	**Nombre del Requisito: “Transformar múltiples geometrías”**
Programador: Wilson Félix Cassuque cambila	Iteración Asignada: 1
Prioridad: **Alta**	Tiempo estimado: 6 días
Riesgo en desarrollo: • **Planificación irreal.** • **Interrupción del servicio eléctrico.** • **Afectaciones externas al personal de trabajo.**	Tiempo Real: 7 días
Descripción: El módulo debe permitir Transformar múltiples geometrías una geometría seleccionada por el usuario. Dicha geometría mantiene las mismas propiedades de la geometría base.	
Observaciones:	

Prototipo de Interfaz: