Universidad de las Ciencias Informáticas

Facultad 6

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

“Componente para el análisis de proximidad, utilizando el diagrama de Voronoi, para la Plataforma GeneSIG”

Autor:

Alfredo Trujillo Díaz

Tutores:

MSc. Lidisy Hernández Montero

MSc. Romanuel Ramón Antunez

La Habana, julio de 2016

“Año 58 de la Revolución”

DECLARACIÓN DE AUTORÍA

Declaro ser el único autor del trabajo **“Componente para el análisis de proximidad, utilizando el diagrama de Voronoi, para la plataforma GeneSIG”** y se autoriza a la Universidad de las Ciencias Informáticas hacer el uso que estimen pertinente con el mismo.

Para que así conste firmo la presente a los 7 días del mes de julio del año 2016.

_____

Firma del autor

Alfredo Trujillo Díaz

_____

Firma del Tutor

MSc. Romanuel Ramón Antunez

_____

Firma del Tutor

MSc. Lidisy Hernández Montero

Síntesis del tutor

Nombre y apellidos: MSc. Romanuel Ramón Antunez

Correo electrónico: rramon@uci.cu

Año de graduación: 2008

Profesión: Ingeniero en Ciencias Informáticas

Máster en Informática Aplicada: 2011

Síntesis del tutor

Nombre y apellidos: MSc. Lidisy Hernández Montero

Correo electrónico: lhernandez@uci.cu

Año de graduación: 2008

Profesión: Ingeniero en Ciencias Informáticas

Máster en Informática Aplicada: 2015

Síntesis del autor

Nombre y apellidos: Alfredo Trujillo Díaz

Correo electrónico: adias@estudiantes.uci.cu

“Sueña y serás libre de espíritu,

lucha y serás libre en la vida.”

Ernesto Guevara de la Serna

Agradecimientos

Quiero agradecer:

A Dios, a los muertos y a los santos por esta vida.

A mi madre por su ejemplo y por su sacrificio para convertirme en un hombre de bien.

A mi padre, a mis hermanos Jessy y Kathy, a mi segunda madre mi tía Raiza, a mi tía Leonor y mi tío Toca y a mi tío Alcides, que siempre nos acompaña y ayuda.

A todas las personas que han tenido que ver con mi actuar en la universidad: Eddy, Yoel, Iván, Odiel, Manuel de Jesús Luis Díaz, Vilma Orozco, Nadilson, Edson, Kielam, Elizabeth, el Táta, el ruso, Luis Enrique, Yoilán, Yordanis, Ariel, Quintero, Liudmila, Celia, Giselle y Laura.

A mis tutores Romanuel y Lidisy por su comprensión y confianza.

Dedicatoria

A mi madre querida.

Al que lucha y no se rinde.

Resumen

Los Sistemas de Información Geográfica (SIG) son herramientas que pueden ser empleadas en cualquier actividad que requiera información georreferenciada, de aquí parte su gran importancia para la sociedad. GeneSIG es una plataforma utilizada para el desarrollo de Sistemas de Información Geográfica en ambientes web, que permite realizar múltiples análisis entre los que se encuentra el análisis de proximidad. La presente investigación tiene como objetivo dar a conocer la relevancia del “Componente para el análisis de proximidad, utilizando el diagrama de Voronoi, para la Plataforma GeneSIG”. Dicho componente surge de la necesidad de ampliar el análisis de proximidad que se realiza en la plataforma incorporando una funcionalidad que permita la partición del espacio en zonas más próximas a entidades. El diagrama de Voronoi, calculado mediante el algoritmo de Fortune, es el seleccionado para obtener dicha partición del espacio. Para el desarrollo del componente se seleccionaron las herramientas y tecnologías teniendo en cuenta las empleadas en el desarrollo de GeneSIG. El proceso de pruebas arrojó no conformidades que fueron corregidas, asegurándose la correcta ejecución de todas las funcionalidades deseadas.

Palabras clave: Análisis de proximidad, capa vectorial, diagrama de Voronoi, GeneSIG, componente.

Abstract

Geographic Information Systems (GIS) are tools that can be used in any activity that requires geo-referenced information, hence part of its great importance for society. GeneSIG is a platform used for the development of GIS in web environments, which allows multiple analysis including the proximity analysis is. This research aims to publicize the relevance of "Component for proximity analysis, using the Voronoi diagram for the Platform GeneSIG". Said component arises from the need to expand the proximity analysis is performed on that platform incorporating functionality to allow the partition space areas closer to entities. Voronoi diagram calculated by Fortune algorithm is selected for the partition of space. Component development tools and technologies used taking into account in developing GeneSIG selected. The testing process yielded nonconformities that were corrected, ensuring the proper implementation of all desired functionality.

Keywords: Proximity analysis, vectorial layer, Voronoi diagram, GeneSIG, plugin.

Índice general

Índice de tablas

Índice de figuras

Introducción

El desarrollo de las Tecnologías de la Información y las Comunicaciones (TICs) ha permitido hacer frente a las crecientes necesidades del ser humano de gestionar y representar la información, cada vez más abundante. Entre las áreas de impacto se encuentra la información geográfica, donde el empleo de los Sistemas de Información Geográfica (SIG) ha provocado un notable avance en el estudio, gestión y análisis territorial. Según (Olaya, 2014), "(…) *un SIG es un sistema que integra tecnología informática, personas e información geográfica, y cuya principal función es capturar, analizar, almacenar, editar y representar datos georreferenciados* (…)".

La información geográfica es representada generalmente empleando los modelos vectorial y *raster*. Según (Olaya, 2014), "(…) *en el modelo raster, la zona de estudio se divide de forma sistemática en una serie de unidades mínimas (denominadas habitualmente celdas) y para cada una de estas se recoge la información pertinente que la describe* (…)". Además, define el modelo vectorial como aquel que "(…) *modeliza el espacio geográfico mediante una serie de primitivas geométricas que contienen los elementos más destacados de dicho espacio. Estas primitivas son de tres tipos: puntos, líneas y polígonos* (…)". En el caso del modelo vectorial la geometría asociada a las primitivas es empleada para realizar numerosos análisis basados en su transformación para obtener capas cuyas nuevas geometrías pueden ser utilizadas en otros análisis u operaciones.

El análisis de proximidad se encuentra entre los procesos que se ejecutan utilizando los SIG. Este análisis permite interpretar los datos geográficos de una forma distinta a la obtenida visualizando los datos en su forma original, apoyándose en conocimientos de áreas como las matemáticas y la geometría.

Según (ESRI[1], 2016), "(…) *el análisis de proximidad es un tipo de análisis en el que los elementos geográficos (puntos, líneas, polígonos, o celdas raster) se seleccionan en función de su distancia a otros elementos o celdas* (…)". El resultado de este proceso puede ser la tabulación de distancias entre entidades o capas nuevas cuyas geometrías aportan información adicional a las geometrías originales, este último apoya el proceso de toma de decisiones mediante la visualización de una zona de proximidad.

Antecedentes

Entre los centros a la vanguardia en el empleo de las TICs para el desarrollo de la sociedad cubana se encuentra la Universidad de las Ciencias Informáticas (UCI). En esta se desarrolla la plataforma GeneSIG,

[1] ESRI: empresa desarrolladora de software que provee aplicaciones y sistemas de información geográfica.

soportada y desarrollada en cooperación con el centro de desarrollo de software Geoinformática y Señales Digitales (GEySED), el grupo empresarial GeoCuba[2] y la Empresa de Tecnologías de Información para la Defensa (XETID), con el objetivo de desarrollar SIG para entornos web con tecnologías libres.

La plataforma GeneSIG es una aplicación informática utilizada como base cartográfica para la construcción de SIG con perfiles específicos; cuenta con un conjunto de *plugins*[3] en los que se localizan la mayoría de las funcionalidades de los SIG convencionales. Su arquitectura está basada en componentes, lo que permite adaptar sus funcionalidades a cualquier negocio mediante su reutilización, además trabaja sobre los modelos de representación de la información geográfica *raster* y vectorial.

La plataforma GeneSIG permite realizar análisis de proximidad sobre capas vectoriales a través de la generación de zonas de influencia (utilizando buffers). No obstante, el análisis de proximidad en la plataforma GeneSIG es limitado, dado que no permite representar zonas que dividan totalmente el plano y lo asignen a la entidad más cercana.

Esta forma de partición del plano es utilizada en los estudios de composición del suelo (agricultura, minería) para determinar hasta dónde se puede asegurar la validez de un dato de muestra, en el geomarketing para a través de la comparación del tamaño de las zonas deducir dónde la concentración de los elementos de interés (a los que se les calcularon las zonas más próximas) sea mayor y por ende exista una mayor competencia, en el estudio de la abundancia de larvas de mosquitos u otro vector, en la creación de modelos de oferta y demanda y estudios de accesibilidad.

Debido a los elementos anteriormente expuestos se arriba al siguiente **problema a resolver**: ¿Cómo mejorar el análisis de proximidad sobre capas vectoriales en los Sistemas de Información Geográfica que se desarrollan utilizando la plataforma GeneSIG? El **objeto de estudio** de la investigación es el análisis de proximidad sobre capas vectoriales en los Sistemas de Información Geográfica enmarcado en el **campo de acción**, identificación de zonas próximas a entidades en capas vectoriales en Sistemas de Información Geográfica desarrollados sobre la base de Mapserver. Se define como **objetivo general**: Desarrollar el componente para el cálculo de la zona del espacio más próxima a una entidad que mejore el análisis de

[2] GeoCuba es un grupo empresarial que se dedica a la elaboración, producción y venta de planos, mapas y cartas náuticas con diversos fines, así como a la realización de estudios geográficos, de impacto ambiental, e investigaciones científicas en ramas del campo de las geociencias (Hernández, 2011).

[3] *Plugin*, del inglés *plug-in*, es un programa que puede anexarse a otro para aumentar sus funcionalidades (generalmente sin afectar otras funciones ni afectar la aplicación principal).

proximidad sobre capas vectoriales en los Sistemas de Información Geográfica que se desarrollan utilizando la plataforma GeneSIG.

El objetivo propuesto indujo a formular, como guías para el desarrollo de la investigación, las siguientes **preguntas científicas**:

1. ¿Cuál es la fundamentación teórica existente sobre el análisis de proximidad sobre capas vectoriales en Sistemas de Información Geográfica?
2. ¿Cuáles algoritmos permiten calcular la zona del espacio más próxima a una entidad?
3. ¿Cómo integrar un componente a la plataforma GeneSIG?
4. ¿El componente garantiza el cálculo de la zona del espacio más próxima a una entidad en la plataforma GeneSIG?

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

1. Elaboración de un estado del arte de las herramientas que se utilizan para el análisis de proximidad en capas vectoriales con el objetivo de definir características y tendencias de este proceso.
2. Análisis y caracterización de los algoritmos para el cálculo de la zona del espacio más próxima a una entidad.
3. Fundamentación de la metodología de software, herramientas y tecnologías a utilizar en el proceso de desarrollo.
4. Especificación de los requisitos funcionales y no funcionales de la propuesta de solución para definir sus características.
5. Elaboración del diseño de la propuesta de solución para guiar el proceso de implementación.
6. Implementación de la propuesta de solución para dar cumplimiento al objetivo planteado.
7. Validación de la propuesta de solución a través de pruebas para demostrar su correcto funcionamiento.

Se utilizan **métodos investigación,** tanto teóricos como empíricos, los cuales se describen a continuación.

Métodos teóricos

Se basan en la utilización del pensamiento en sus funciones de educación, análisis y síntesis. En la presente investigación se utilizan los siguientes métodos teóricos:

- **Histórico-Lógico:** Se utiliza para estudiar la evolución de los conceptos asociados al análisis de proximidad en Sistemas de Información Geográfica.
- **Analítico-Sintético:** Se emplea para identificar elementos claves que contribuyan a la solución del problema de investigación, durante el proceso de revisión de la bibliografía, permite además sintetizar conceptos que ayudan a comprender la solución del problema.
- **Modelación:** Se utiliza para la elaboración de los artefactos relacionados con el modelo de dominio y modelo de datos.

Métodos empíricos

Se aproximan al conocimiento del objeto mediante sus conocimientos directos y el uso de la experiencia. En la presente investigación se utiliza el siguiente método empírico.

- **Análisis documental:** Consiste en la recopilación adecuada de datos de varias fuentes permitiendo el planteamiento de nuevas interrogantes. Permite el conocimiento de distintos enfoques sobre un mismo tema y ampliar el universo de alternativas de solución y conceptos. Posibilita conocer el estado del arte sobre el análisis de proximidad en SIG y de los algoritmos para el cálculo del diagrama de Voronoi.

Estructura del documento

El presente documento consta de tres capítulos:

Capítulo 1: Fundamentos teóricos sobre el análisis de proximidad en Sistemas de Información Geográfica. Contiene el marco teórico de la investigación, un análisis de las soluciones existentes y se definen las herramientas y lenguajes de programación a utilizar, así como la metodología de desarrollo de software.

Capítulo 2: Análisis y diseño. Contiene el modelado del dominio del problema, los requisitos funcionales y no funcionales del sistema y los patrones de diseño aplicados en la propuesta de solución.

Capítulo 3: Implementación y pruebas. Se presentan el modelo de datos, el de implementación y el de despliegue y se explica el proceso de validación aplicado a la propuesta de solución.

Capítulo 1.Fundamentos teóricos sobre el análisis de proximidad en Sistemas de Información Geográfica

En el presente capítulo es definido el marco conceptual de la investigación y se analizan soluciones existentes asociadas al dominio del problema de la investigación que puedan dar respuesta al mismo. Además, se definen y caracterizan la metodología, las herramientas y los lenguajes a utilizar en el desarrollo de la propuesta de solución.

1.1 Conceptos asociados al dominio del problema

A continuación se relacionan los principales conceptos o temáticas que están asociados al desarrollo de la investigación.

1.1.1 Componente

Un componente es un paquete de software que ofrece servicios a través de sus interfaces, que puede estar compuesto por otros componentes. Estos aportan a programas más grandes una funcionalidad específica. Se utilizan como forma de expansión, permitiendo incorporar una nueva función sin complicar el desarrollo del programa principal ni afectar a las ya existentes.

"(…) Un componente de software (CS) es una unidad de composición con interfaces especificadas en forma de contrato y con dependencias de contexto explicitas (…)" (Bonilla, 2009).

Ventajas de los componentes:

- Son útiles para los *softwares* que permiten crear personalizaciones, puesto que cada una incorpora solo los componentes que necesita y esto reduce el tamaño del programa principal.
- Permiten añadir funcionalidades al programa principal de forma rápida y sencilla.
- Los componentes tienen un funcionamiento independiente con respecto a los otros y al sistema principal, esto no quiere decir que no exista comunicación entre ellos y el sistema principal.

Desventajas de los componentes:

- No funcionan por su cuenta. Siempre necesitan de otro programa al cual son incorporados.
- No acostumbran ser multiplataforma, por lo que generalmente son creados para ser incorporados a un programa principal en específico, y por tanto pueden ser incompatibles por arquitectura o tecnología con otras plataformas.

Las empresas fabricantes de SIG comercializan distintos tipos de componentes software orientados a las necesidades de los desarrolladores SIG. La complejidad de estos componentes es diversa, encontrando desde componentes que solo permiten realizar una función específica, hasta otros que constituyen aplicaciones completas. La plataforma GeneSIG sigue esta línea, quedando definido que se creará un componente que agrupará las funcionalidades necesarias para resolver el problema de investigación.

1.1.2 Sistema de Información Geográfica

Los SIG permiten el uso de información geográfica digital, a través de capas superpuestas que conforman los mapas digitales. " (…) Así, un SIG es fundamentalmente una herramienta para trabajar con información georreferenciada, una definición en la que pueden entrar un gran número de tecnologías y de otros elementos no tecnológicos (…)" (Olaya, 2014). Esta definición es la asumida para la investigación dado que generaliza la definición de SIG, tomando como característica definitoria que la información con la que trabaja el sistema está georreferenciada.

Según (Olaya, 2014), una forma de entender el sistema SIG por una serie de subsistemas, entre los cuales son fundamentales:

- Subsistemas de datos. Se encarga de las operaciones de entrada y salida de datos, y la gestión de estos dentro del SIG. Permite a los otros subsistemas tener acceso a los datos y realizar sus funciones en base a ellos.
- Subsistema de visualización y creación cartográfica. Crea representaciones a partir de los datos (mapas, leyendas), permitiendo así la interacción con ellos. Entre otras, incorpora también las funcionalidades de edición.
- Subsistema de análisis. Contiene métodos y procesos para el análisis de los datos geográficos.

La información geográfica se registra en capas. Existen diversas formas de representar la misma información geográfica en distintas capas, la forma de representación se selecciona en dependencia del tipo de información que se desee representar y del análisis que se le desee hacer. Existen dos formas principales de materialización de la realidad en el modelo geográfico, estos son el modelo de representación vectorial y el *raster*.

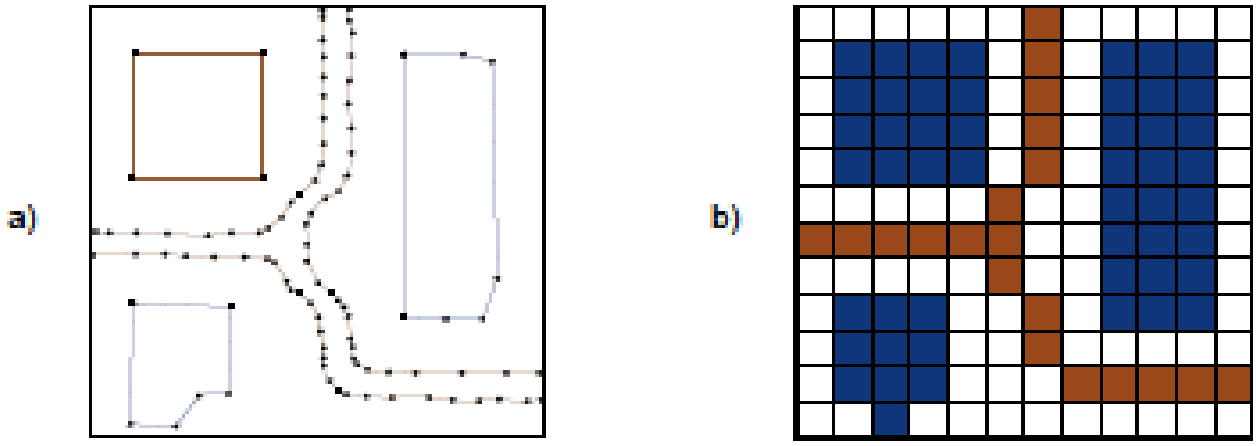


Figura 1: Modelos de representación vectorial (a) y *raster* (b)

Tomado de (Olaya, 2014)

En el modelo vectorial de las entidades se recoge la forma, posición y otras propiedades espaciales. Este modelo permite realizar gran variedad de análisis con gran precisión, entre los que se encuentra el análisis de proximidad.

1.2 Objeto de estudio

Sobre los SIG se realizan variados análisis con el objetivo de encontrar información que revele patrones y relaciones entre los elementos, entre los que se encuentra el análisis de proximidad. Este es definido por algunos autores como de dos tipos, ejemplo " (…) *Algunos procedimientos de análisis espacial comúnmente incluidos en las rutinas de los programas SIG son el análisis de proximidad (que implica el cálculo de distancias entre objetos georreferenciados con valores, rangos o clases específicos de una variable); el análisis de proximidad (que permite identificar áreas cuya característica de análisis tiene el mismo valor o categoría y que mantienen continuidad, eliminando las áreas no significativas según un límite de inclusión)* (…)" (Reyna, 1996).

Según (Santos Preciado, y otros, 2005), el análisis de proximidad es una "*(…) función propia de los SIG que trata de diferenciar las áreas que envuelven una localización determinada (…)*". El análisis de proximidad se realiza cuando una estructura topológica[4] permite investigar las relaciones y distancias entre entidades

[4] Topología: "(…) estudia las propiedades espaciales de las figuras geométricas que subsisten aún si estas se someten a deformaciones tan radicales que las hagan perder todas sus propiedades métricas y de proyección en la geometría euclidiana. (…) '' (Backhoff, 2005).

en el espacio. Este análisis consiste en procedimientos que utilizan algoritmos y funciones especiales que permiten definir áreas de influencia[5] alrededor de un objeto y cálculos de distancias entre objetos.

Tomado en cuenta que la situación problemática se refiere a la identificación de áreas, y no al cálculo de distancia, se analizan las herramientas que se utilizan con este fin en los SIG actuales.

1.2.1 Análisis de soluciones existentes

Para el estudio sobre el análisis de proximidad sobre capas vectoriales se seleccionan diferentes SIGs, tanto libres como privativos. Principalmente aquellos que han sido desarrollados por empresas líderes en el trabajo con información geográfica. Se toman en cuenta las herramientas que soportan el análisis de proximidad en estos sistemas y las características que estas presentan, esto último facilita la identificación de los requisitos funcionales del componente que se espera obtener.

Se revisan los sistemas ArcGIS 10.0, MapInfo Professional 12.0.2, Quantum GIS 2.2, gvSIG 2.2, los dos primeros son privativos y los restantes libres. ArcGIS posee la aplicación web ArcIMS para distribuir servicios referentes a información geográfica, el resto de los SIG revisados son desktop. Las herramientas para el análisis de proximidad en dichos sistemas no contemplan grandes diferencias.

Dentro de las principales herramientas para el análisis de proximidad en capas vectoriales se identifican:

- Áreas de proximidad alrededor de un objeto (*buffer*).
- Áreas de proximidad mediante diagrama de Voronoi/Thiessen.

Tabla 1 Empleo de buffer y diagrama de Voronoi en los SIG analizados

SIG analizados	*Buffer*	Voronoi
ArcGIS 10.0	x	x
MapInfo Professional 12.0.2	x	x
Quantum GIS 2.2	x	x
gvSIG 2.2	x	x
GeneSIG 2.0.5	x	

[5] Área de influencia: "(…) aquellas que a partir de una entidad espacial y de acuerdo a una variable o conjunto de variables define una nueva entidad en el espacio. Estas nuevas entidades suelen ser del estilo de corredores (buffers), círculos o coronas (donuts) o figuras irregulares o regulares en función del polígono de origen. (…)'' (Sastre, 2010).

A partir de la observación de la Tabla 1 se deduce que de todos los SIG analizados poseen la identificación de zonas próximas a partir del cálculo de *buffers;* y en el caso del diagrama de Voronoi, GeneSIG es el único que carece de dicha funcionalidad.

1.2.2 *Buffer*

"(…) *Una de las transformaciones más importantes con capas vectoriales es la creación de zonas de influencia, también conocidas como buffers. Esta transformación puede llevarse a cabo con entidades de tipo punto, línea o polígono, y su resultado siempre es una nueva capa de polígonos. Las áreas cubiertas por estos polígonos reflejan las zonas de influencia de cada entidad, influencia que se considera la ejerce hasta una distancia dada. Pueden verse también de forma inversa, como una influencia recibida, de tal modo que todos los elementos dentro de la zona de influencia afectan a la entidad que la genera* (…)" (Olaya, 2014).

La creación de áreas de influencia (*buffering*) es la generación de una zona alrededor de una primitiva (punto, línea o polígono), de un radio especificado. Como resultado siempre se obtiene un polígono, que se puede utilizar para el análisis de interrogantes como la de identificar qué entidades se encuentran dentro de un área de influencia dada.

Existen algunas desventajas en el uso de *buffers* en SIG. Por ejemplo, cuando una zona *buffer* se extiende más allá de los límites geográficos del conjunto de datos que se está usando. En la mayoría de los casos es improbable que sea significante, pero esto puede provocar que los resultados que se reporten sean inexactos. Otra desventaja es la determinación del valor del radio. Puede algunas veces estar justificada en el campo legal o científico, pero generalmente es una decisión arbitraria. Por otro lado, en ocasiones existe la superposición inadmisible de zonas *buffer*, restando consistencia y coherencia a los resultados del análisis.

La generación de áreas de influencia (*buffering*) se utiliza en el modelo de representación de datos geográficos vectorial y se puede considerar el equivalente vectorial del análisis de distancias en el modelo *raster*. En este caso las zonas de influencia que se obtienen no son delimitadas necesariamente por fronteras equidistantes de los puntos generadores. Existe la variante del empleo de zonas de influencia de anillos múltiples, consiste en la creación de varias zonas de influencia a distancias definidas por el usuario. El resultado de este proceso son polígonos que representan dichas zonas de influencia, este análisis presenta los siguientes inconvenientes (ver Figura 2):

- Las zonas de influencia pueden dejar espacios en el mapa sin cubrir, o sea, que no se abarca todo el espacio.

- Las zonas de influencia pueden quedar solapadas, esto impide delimitar cual parte del espacio está más cerca de cada entidad en particular.

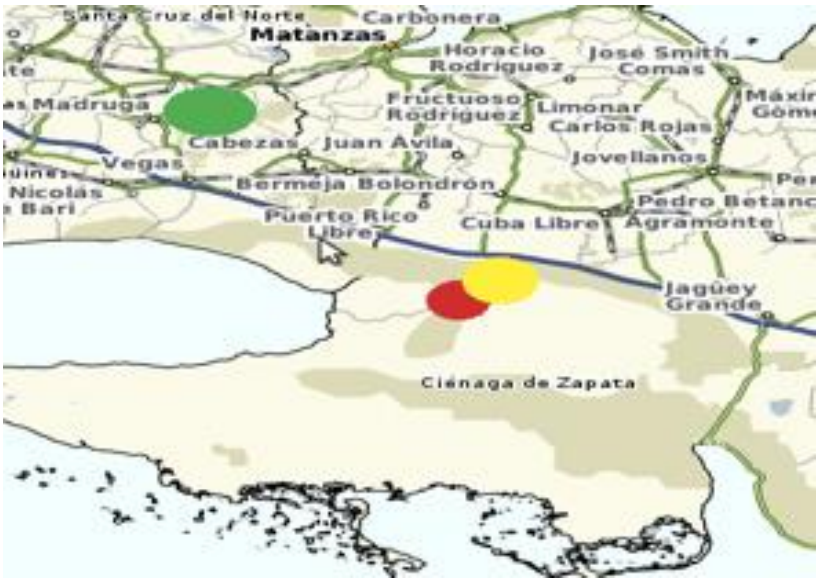


Figura 2: Aplicación de *buffer* sobre primitivas tipo punto.
Fuente: fragmento de imagen de (Mestre, 2014)

Las limitantes anteriormente mencionadas descartan el buffer como posible solución al problema de investigación.

1.3 Diagrama de Voronoi

Un Diagrama de Voronoi para un conjunto S= {p1, ..., pn} de puntos del plano es una partición del plano en n regiones poligonales convexas V1, ..., Vn, tal que para cada i todos los puntos de la región Vi están más cercanos a pi que a cualquier otro punto de S. Todos los puntos del plano quedan asociados a algún pi y existirán puntos que tengan la misma distancia a dos elementos de S y formarán la frontera de V.

(…) *se puede concluir que un punto x del plano pertenece al área de influencia del punto generador pi, denominada polígono de Voronoi, Vi, si y solo si el punto x está más cerca de pi que de cualquier otro punto generador* (Rivero, 2006).

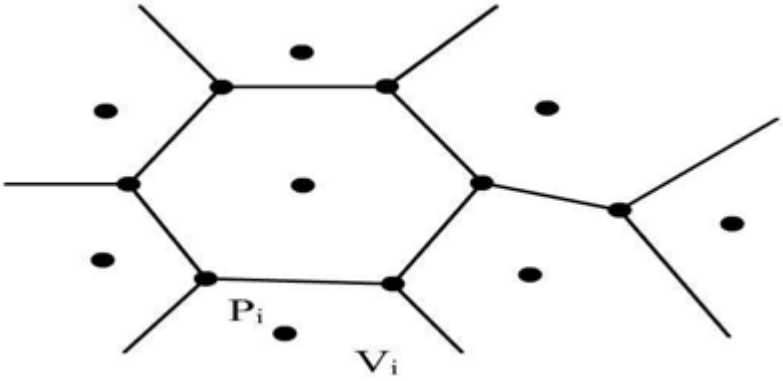


Figura 3: Diagrama de Voronoi.
Fuente (Rivero Mendoza, 2006)

El polígono convexo Vi que contiene al punto pi se llama Polígono de Voronoi del punto pi. Los vértices del diagrama se llaman Vértices de Voronoi y los segmentos de recta del diagrama se llaman los Lados de Voronoi.

Una primera observación al Diagrama de Voronoi permite llegar a las siguientes conclusiones:

- Si se tiene un punto pi en S, entonces su vecino más próximo se halla en alguno de los polígonos de Voronoi adyacentes a Vi.
- Si se ordena en una lista cada punto pi con su vecino más cercano, entonces se puede buscar en dicha lista el par de elementos de S más cercanos.
- El máximo círculo que no contenga puntos pi tiene como centro un vértice de Voronoi o bien está centrado en la intersección de uno de los ejes del Diagrama con la envolvente convexa.

1.3.1 Aplicaciones del Diagrama de Voronoi en Sistemas de Información Geográfica

Los polígonos de Thiessen se aplican, entre otros, en el estudio de áreas de mercado o centros de influencia de servicios (zonas sanitarias o escolares, bibliotecas.), conocidos los centros que prestan el servicio de que se trate. Las áreas resultantes pueden ser interpretadas como las áreas de influencia de cada centro, siempre que se considere idéntico el peso de atracción de los mismos. En el caso de disponer, en otra capa, de información sobre la distribución de la población, se puede calcular la demanda potencial de cada centro (Santos, 2005).

En (Uva, 2006) se interpretan los polígonos del diagrama de Voronoi como las zonas de validez de muestras del suelo tomadas in situ, previas a la agricultura de precisión. Esto permite determinar cuál es la distancia

máxima en que es posible alejarse del punto de extracción de una muestra del rendimiento del suelo y seguir asegurando que tiene validez, previo a la siembra de una campaña.

En (de Almeida, 2006) el diagrama es utilizado para dividir un barrio en zonas que son más próximas a las paradas de los ómnibus, ver Fig. 5.

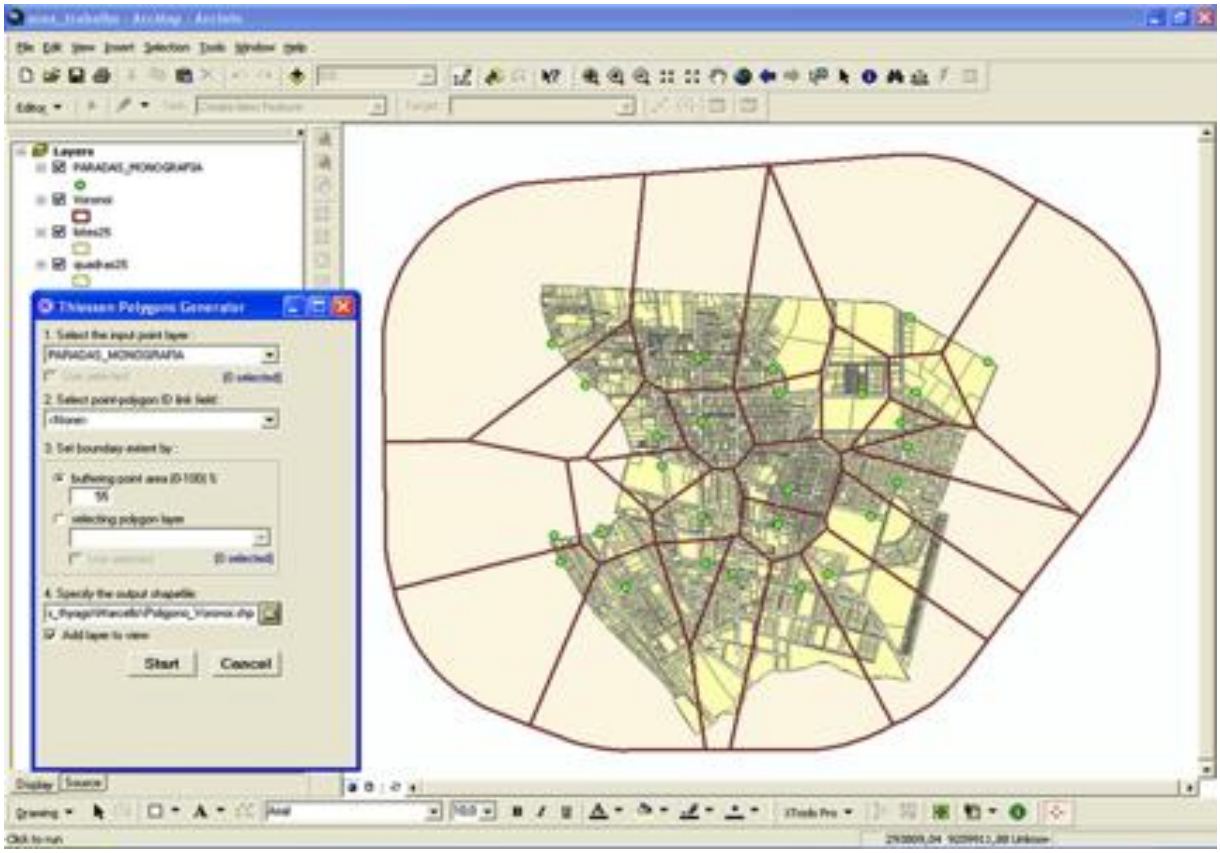


Figura 4: Creación del diagrama de Voronoi.
Fuente (de Almeida, 2006)

En (de Pietri, 2013) y (Roos, 1993) se utiliza el diagrama de Voronoi para generar las zonas de influencia de las infraestructuras de salud.

Entre las aplicaciones más frecuentes del diagrama de Voronoi se encuentran el estudio de áreas de mercado, los problemas de ubicación de servicios y otros problemas en que sea necesario dividir el plano en regiones donde cada región contenga los elementos más cercanos a un punto dado y la cercanía esté dada por la distancia Euclidiana. Las áreas resultantes pueden ser interpretadas como las áreas de influencia de cada punto generador, asumiendo idéntico el peso de atracción de los mismos. Dadas las

características del diagrama de Voronoi para la partición del plano en zonas que se asignan a la entidad más cercana en el mapa, se decide utilizar esta herramienta para dar solución al problema planteado al inicio de esta investigación.

1.3.2 Algoritmos para el cálculo del diagrama de Voronoi de una nube de puntos. Algoritmo de Fortune

Existen diferentes tipos de algoritmos para la construcción de diagramas de Voronoi, basados en la técnica divide y vencerás (Rivero, 2006), la técnica línea de barrido o *sweep-line* (Fortune, 1987), o transformaciones geométricas descritas por (Edelsbrunner, 1986).

En (Martínez, 2015) aparece un análisis de los principales algoritmos para la construcción del diagrama de Voronoi. A partir de este análisis se decide utilizar el algoritmo de línea de barrido descrito por Steven Fortune (Fortune, 1987), este algoritmo tiene una complejidad computacional O(nlogn) que es óptima según (Boots, 2000) para el cálculo de este diagrama y además se cuenta con basta documentación acerca de su uso e implementación.

Algoritmo de Fortune

Este algoritmo tiene como estrategia el barrido del plano utilizando una línea horizontal, la cual desciende sobre cada punto que pasa. Esta línea recoge a su paso arcos parabólicos que son las fronteras de conos que se elevan a partir de los puntos generadores del diagrama. Estos arcos son luego corregidos y se obtienen los segmentos de línea recta o arcos de Voronoi, ya que las dos estructuras son topológicamente idénticas.

Algoritmo Fortune(P)

Entrada: P: = (p1, …, p2) un conjunto de n puntos en el plano

Salida: Diagrama de Voronoi V(P)

Inicializar la cola de eventos ColaEventos Q con todos los EventosSitio
While ColaEventos Q no está vacía **do**
    Considerar el evento con mayor coordenada Y de Q
    Tratar_Evento_Sitio(pi)
    Else
        Tratar_Evento_Circulo(pc), donde pc es el punto más bajo del círculo que causa el evento
    End if
    Eliminar el evento de Q
End while

Generar un rectángulo que contenga todos los vértices del diagrama de Voronoi en su interior, uniéndole los medios-lados infinitos.
Return Diagrama de Voronoi obtenido

Los procedimientos Tratar_Evento_Sitio(pi) y Tratar_Evento_Circulo(pc) son descritos en detalle en (Marbate, 2013).

1.4 Descripción actual del dominio del problema

1.4.1 La plataforma GeneSIG

"(…) *La plataforma GeneSIG 2.0 es desarrollada por la Universidad de las Ciencias Informáticas (UCI), la empresa GeoCuba y la Empresa de Tecnologías de Información para la Defensa (XETID). Esta aplicación SIG Web, es capaz de soportar una amplia gama de funcionalidades relacionadas con la gestión de datos espaciales, por parte de usuarios especializados y nuevos consumidores de servicios de datos geográficos. GeneSIG, tiene una dualidad funcional, que la habilita para su utilización como una aplicación SIG, a través de la cual los usuarios pueden consumir, manipular y consultar bases cartográficas digitales, de diversos formatos y orígenes, como por ejemplo: datos vectoriales, datos raster, datos GPS (global positioning system), bases de datos espaciales en formato Postgis, Servicios de mapas Web y Servicios de Geometrías* (…)" (Salas, 2015).

Esta herramienta fue creada con el objetivo de brindar a usuarios y desarrolladores soluciones geoespaciales, sobre una plataforma tecnológica libre, de código abierto y orientada a la web, para solventar demandas básicas de información espacial, en cuanto a su almacenamiento, análisis y representación.

"(…) La plataforma GeneSIG posee una estructura basada en plugins, lo que la convierte en una plataforma con un alto grado de interoperabilidad debido a que permite agregar o quitar componentes de manera sencilla. De igual modo permite separar los servidores de Bases de Datos y web en dos estaciones de trabajo diferentes, balanceando la carga del sistema, aumentando su disponibilidad y disminuyendo la posibilidad de fallas (…)" (Filgueiras, 2015). La modularidad de la plataforma está dada por los *plugins* que se le incorporan y que le adicionan funcionalidades.

Como parte de su organización interna, el sistema cuanta con tres capas lógicas: la capa de interfaz, la capa de negocio y la capa de datos (Filgueiras, 2015).

- Interfaz: En esta capa están implementadas todas las interfaces gráficas con las que interactúa el usuario y las interfaces de interacción con otros sistemas. Estas interfaces se relacionan directamente con los módulos que se encuentran implementados en la capa de negocio.

- Negocio: En esta capa están incluidas todas las tareas funcionalidades que realiza la plataforma e incluye al servidor de mapas MapServer.
- Datos: En esta capa se encuentran las bases de datos con las que trabaja la plataforma (una para datos cartográficos y otra para la información socio-económica, de configuración y los usuarios).

GeneSIG es portable, permitiendo que el cliente opere en distintos Sistemas Operativos sin necesidad de modificar el código en la parte del servidor.

Entre los módulos (agrupan funcionalidades) que componen GeneSIG se encuentran:

- Módulo de Consulta Espacial.
- Módulo de Catálogo.
- Módulo de Configuración del Mapa.
- Módulo de Análisis.

El módulo de análisis reúne las funcionalidades de análisis de proximidad en la plataforma. Dado que actualmente la plataforma solo permite calcular zonas próximas utilizando buffers; siendo esto una limitante en el proceso de toma de decisiones de situaciones en las que se desee que no queden espacios sin asignar a su entidad más cercana determinada o no se desee que dos entidades (punto) compartan parte del espacio, excepto la línea que representa la frontera entre entidades adyacentes.

1.5 Metodología de desarrollo de software. Prodesoft 1.5

El desarrollo de software tiene gran importancia en la actividad creadora de la sociedad actual. En la búsqueda de mayor eficiencia se han desarrollado tecnologías y técnicas para disminuir el costo y la dificultad de la producción de software.

Para la obtención de un software exitoso y con la calidad requerida es necesaria la selección de una metodología de desarrollo, la cual provee procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a producir a crear un nuevo software. Las metodologías de software se clasifican en tres grupos: ágiles, tradicionales o híbridas.

En aras de garantizar que los especialistas comprendan fácilmente los artefactos generados y los pasos seguidos en el proceso de desarrollo se decide utilizar como metodología de desarrollo el Proceso de Desarrollo de Software (Prodesoft) en su versión 1.5. La línea base del proyecto GeneSIG define este como el proceso de desarrollo de software a utilizar para el desarrollo de funcionalidades para dicha plataforma.

Prodesoft es resultado de la combinación de varias metodologías de software, ágiles y tradicionales, por lo que se clasifica como híbrido. Según la especificación (UCID[6], 2012) el ciclo de vida de un proyecto está compuesto de 5 fases: inicio, modelación, construcción, explotación experimental y despliegue.

Figura 5: Etapas del ciclo de vida del proyecto.
Fuente: (Unidad de compatibilización, integración y desarrollo de software para la defensa (UCID), 2012)

Durante la fase de **inicio** se describen los objetivos y el alcance del proyecto, los involucrados y los ejecutores (entidades involucradas), se establece la estrategia a seguir para la modelación y captura de los requisitos.

Durante la fase de **modelación** se identifican los procesos de negocio fundamentales y se aceptan los requisitos funcionales, obteniéndose la línea base de la arquitectura y una estrategia de construcción de la aplicación. Se libera la arquitectura de sistema, datos y despliegue.

Durante la fase de **construcción** todas las características, componentes, y requisitos deben ser integrados, implementados, y probados en su totalidad, obteniendo una versión liberada del producto.

En la fase de **explotación experimental** se convierte la solución liberada del producto en una solución estable.

En la fase de **despliegue** se instala y configura el sistema para un ambiente de producción real, se capacita al personal que usará la aplicación y se continúa dando soporte durante la explotación del sistema, culminando de ser preciso con transferencias tecnológicas (UCID, 2012).

[6] UCID: Unidad de compatibilización, integración y desarrollo de software para la defensa.

1.6 Herramienta CASE de modelado Visual Paradigm for UML 8.0

Para la creación de la solución se hizo uso de la herramienta de Ingeniería de Software Asistida por Computadoras (CASE, por sus siglas en inglés) Visual Paradigm for UML 8.0. Las herramientas CASE se emplean para facilitar el proceso de modelación del software y estandarizar la documentación durante el ciclo de vida del desarrollo de software.

Visual Paradigm es una herramienta que soporta el ciclo de vida del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Se utiliza debido a que se tiene un conocimiento previo de su utilización y permite generar diagramas a partir de otro que guarde relación con el mismo. Para la modelación esta herramienta utiliza el Lenguaje Unificado de Modelado (UML) que es el mismo que se emplea en la documentación de GeneSIG, lo que permite mantener la homogeneidad en la documentación.

1.7 Lenguaje de Modelado Unificado (UML)

UML es un lenguaje que permite modelar lo que el software debe hacer; posibilitando un mayor entendimiento y comprensión del sistema por parte del equipo de desarrollo. Sus objetivos son varios, pero se pueden resumir en sus funciones (Orallo, 2003).

- Visualizar: Posibilita expresar de una forma gráfica un sistema, de manera que pueda ser entendido fácilmente.
- Especificar: UML permite especificar cuáles son las características de un sistema antes de su construcción.
- Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado, los que pueden ser empleados en su futura revisión.

Este es el lenguaje de modelado utilizado para el desarrollo de la solución; lo que posibilita a los desarrolladores tener un vocabulario y reglas que permiten una comunicación estándar. Su empleo en la documentación del ciclo de desarrollo, con vista a facilitar el mantenimiento y actualización del producto, posibilitando su fácil entendimiento en caso de revisiones futuras.

1.8 Entorno de Desarrollo Integrado PhpStorm 9.0

Un Entorno de Desarrollo Integrado (IDE por sus siglas en inglés) es un programa informático compuesto por un conjunto de herramientas de programación, díganse un editor de código, un compilador, un depurador y un constructor de Interfaz Gráfica de Usuario (GUI por sus siglas en inglés).

Se decide usar el IDE PhpStorm porque provee de un completamiento inteligente de código, detecta código duplicado, mezcla lenguajes (JavaScript, *Structured Query Language*(SQL)), presenta navegación rápida y chequeo de errores al momento, que facilitan el desarrollo de la aplicación. Además es un editor de JavaScript avanzado, que posee navegación de código y facilita la rápida detección de errores en el código.

1.9 Biblioteca ExtJS 3.2

ExtJS es una biblioteca de JavaScript para el desarrollo de aplicaciones web interactivas. Permite flexibilizar el manejo de componentes de la página y crear interfaces de usuario bastantes funcionales. Además, posibilita crear aplicaciones complejas utilizando componentes predefinidos.

Flexibiliza el manejo de componentes de la página web. Entre sus características están:

- Código reutilizable.
- Orientada a la programación de interfaces tipos desktop en la web.
- Independiente o adaptable a marcos de trabajo (*frameworks,* en inglés) diferentes.

1.10 Openlayers 2.0

Es una biblioteca JavaScript para operar con mapas temáticos en el entorno web. Permite crear mapas interactivos, editar información espacial, visualizar información geográfica y la representación de elementos vectoriales. Se utiliza en la solución propuesta para la edición de la capa en la que se muestra el diagrama de Voronoi generado, de esta forma la capa se mantiene temporal facilitando su uso sin necesidad de almacenar la información en la base de datos.

1.11 Lenguajes de programación

Un lenguaje de programación es cualquier lenguaje artificial, el cual, se utiliza para definir adecuadamente una secuencia de instrucciones que puedan ser interpretadas y ejecutadas en una computadora (Achour, y otros, 2006). Dado que la solución se integrará a la plataforma GeneSIG, se recomienda emplear en el desarrollo los mismos lenguajes de programación que utiliza esta plataforma.

El uso del lenguaje **PHP 5.3** (*Hypertext Preprocessor*) es un lenguaje de código abierto para el desarrollo web en el lado del servidor. Existe abundante documentación y permite lograr sistemas lo más libre de licencias posible, cubriendo todas las necesidades del desarrollo; cuenta con una biblioteca, PHP *Mapscript*, que permite trabajar con *MapServer*.

El uso del lenguaje **JavaScript** es obligatorio dado que se emplea en la solución la biblioteca ExtJS. Este lenguaje tiene funciones que permiten la creación de páginas web dinámicas, con interfaces de usuario amigables, menús desplegables y manipulación de datos. Permite el uso de una gran variedad de etiquetas que son las encargadas de mantener la estructura e indicar la semántica de un documento, convirtiéndose en un lenguaje para presentación con soporte de aspectos visuales.

1.12 Sistema Gestor de Base de Datos. PostgreSQL 9.3

Un Sistema Gestor de Base de Datos es un conjunto de programas que sirven para construir y manipular una Base de Datos, de forma transparente al usuario y ordenada. Sirve de interfaz entre las Bases de Datos y las aplicaciones que las utilizan.

PostgreSQL es un Sistema Gestor de Base de Datos relacional y orientado a objetos. Es de código abierto, brinda un control de concurrencia multi-versión (del inglés *multi version concurrent control*. MVCC) que permite trabajar con grandes volúmenes de datos; soporta gran parte de la sintaxis SQL y cuenta con un extenso grupo de enlaces con lenguajes de programación (Morales, 2012).

Postgis 2.1.2 es la extensión que permite el manejo de datos espaciales en PostgreSQL, para lo que contiene cientos de funciones espaciales. Realiza la administración de la base de datos a través de pgAdmin. Posee soporte para datos *raster* y vectoriales. Es compatible con el servidor de mapas MapServer.

pgAdmin 1.18.1 es una plataforma de desarrollo para PostgreSQL. Diseñado para responder a las necesidades de los usuarios, desde escribir consultas SQL (Structured Query Language) simples hasta desarrollar bases de datos complejas. Incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor y un agente para lanzar scripts programados. La conexión al servidor puede hacerse mediante TCP/IP[7].

1.13 Servidor de aplicaciones web Apache 2.0

Apache es un servidor de aplicaciones web, se le considera uno de los mayores triunfos del *software* libre por su amplio uso. Este es el servidor utilizado para el desarrollo de la plataforma GeneSIG, por tanto se decide emplearlo el desarrollo de la solución.

1.14 Servidor de mapas. MapServer 6.4.1

[7] TCP/IP: Para de protocolos, TCP (Protocolo de Control de Transferencia) e IP (Protocolo de Internet), utilizados para comunicar computadoras

Es una plataforma de código abierto para la publicación de los datos espaciales y aplicaciones de mapas interactivos para la web. Es multiplataforma, soporta formatos de datos *raster* y vectorial, presenta una producción cartográfica avanzada y posibilita la automatización del mapa (barra de escala, mapa de referencia, y la leyenda). Se emplea **MapServer** además, porque es el servidor de mapas que utiliza la plataforma GeneSIG.**1.15 Herramienta de pruebas JMeter 2.8**

Herramienta que permite realizar pruebas de rendimiento sobre aplicaciones web. Es gratuita y muestra los resultados de las pruebas de diversas formas (informes, gráficas). Se utiliza para comprobar el tiempo que demora en responder el servidor a varias peticiones concurrentes ya que **JMeter** permite realizar las pruebas con varios hilos de peticiones.

Conclusiones parciales

El estudio de los principales conceptos relacionados al análisis de proximidad en Sistemas de Información Geográfica permite seleccionar la herramienta diagrama de Voronoi para dar solución al problema de investigación.

Se decide desarrollar un componente para la creación del diagrama de Voronoi para la plataforma GeneSIG, sin utilizar las soluciones existentes en SIG, dadas las condiciones de licencia en el caso de **ArcGIS** y **MapInfo Professional** y para evitar tener que adaptar los *plugins* de **QuantumGIS** y **gvSIG** que tienen arquitecturas diferentes dado que son *desktop*.

Un análisis de los algoritmos para la creación del diagrama permite seleccionar el algoritmo de barrido del plano de Fortune para el desarrollo de la solución, dado que tiene la complejidad computacional óptima para el cálculo del diagrama.

Capítulo 2. Análisis y diseño

En este capítulo se presentan y especifican brevemente los requisitos funcionales y no funcionales con los que debe cumplir la aplicación, estos representan las funcionalidades y cualidades del sistema a implementar. Se incluye el modelo de dominio, junto a la descripción de los conceptos que lo componen. También se describen los conceptos relacionados con las definiciones de estilo y patrones de diseño de *software* empleados. Además se muestran los diagramas de clases del diseño y el diseño de la Base de Datos propuestos.

2.1 Modelo de dominio

El Modelo de Dominio o Modelo Conceptual, según (Torres Reyes, y otros, 2014), es el artefacto más importante que se crea durante el análisis orientado a objetos, permite la representación visual de las clases conceptuales u objetos significativos de un dominio de interés. En la realización de este modelo se deben capturar las abstracciones e informaciones necesarias para entender el dominio en el contexto de los requisitos actuales, permitiendo a las personas comprender el negocio, sus conceptos, terminología y relaciones. Para la presente investigación se decidió crear el modelo de dominio debido a que se cuenta con pocos especialistas en el tema y gran parte de la información utilizada ha sido obtenida mediante el estudio de sistemas similares.

2.1.1 Descripción general del Modelo de Dominio

La plataforma GeneSIG se utiliza para desarrollar personalizaciones de SIG en dependencia de las necesidades del cliente. Entre los análisis que se pueden realizar sobre las capas se encuentran los de proximidad. Un usuario puede trabajar con uno a varios análisis de proximidad y aplicarlos sobre una o varias capas, las capas pueden ser *raster* o vectorial. Las vectoriales están compuestas por primitivas que pueden ser: puntos, líneas, polígonos.

2.1.2 Descripción de las clases del Modelo de Dominio

Como el modelo de dominio o modelo conceptual contribuye posteriormente en el proceso de desarrollo del *software* a identificar las clases que se utilizan para modelar el sistema, a continuación se identifican los conceptos fundamentales que se emplean en el modelo a través de un glosario de términos:

Usuario: Son los usuarios autenticados en la Plataforma GeneSIG.

Análisis_Proximidad: Se refiere a operaciones para obtener información referente a la distancia entre las entidades y su relación topológica.

Capa: Se utilizan para controlar la visibilidad de la geometría, y representar la información en los mapas.

***Raster*, Vectorial:** Modelos de representación de la información geográfica.

Punto, Línea y Polígono: Objetos geométricos contenidos en las capas vectoriales.

***Buffer*:** Herramienta para el análisis de proximidad.

2.1.3 Diagrama de clases del Modelo de Dominio

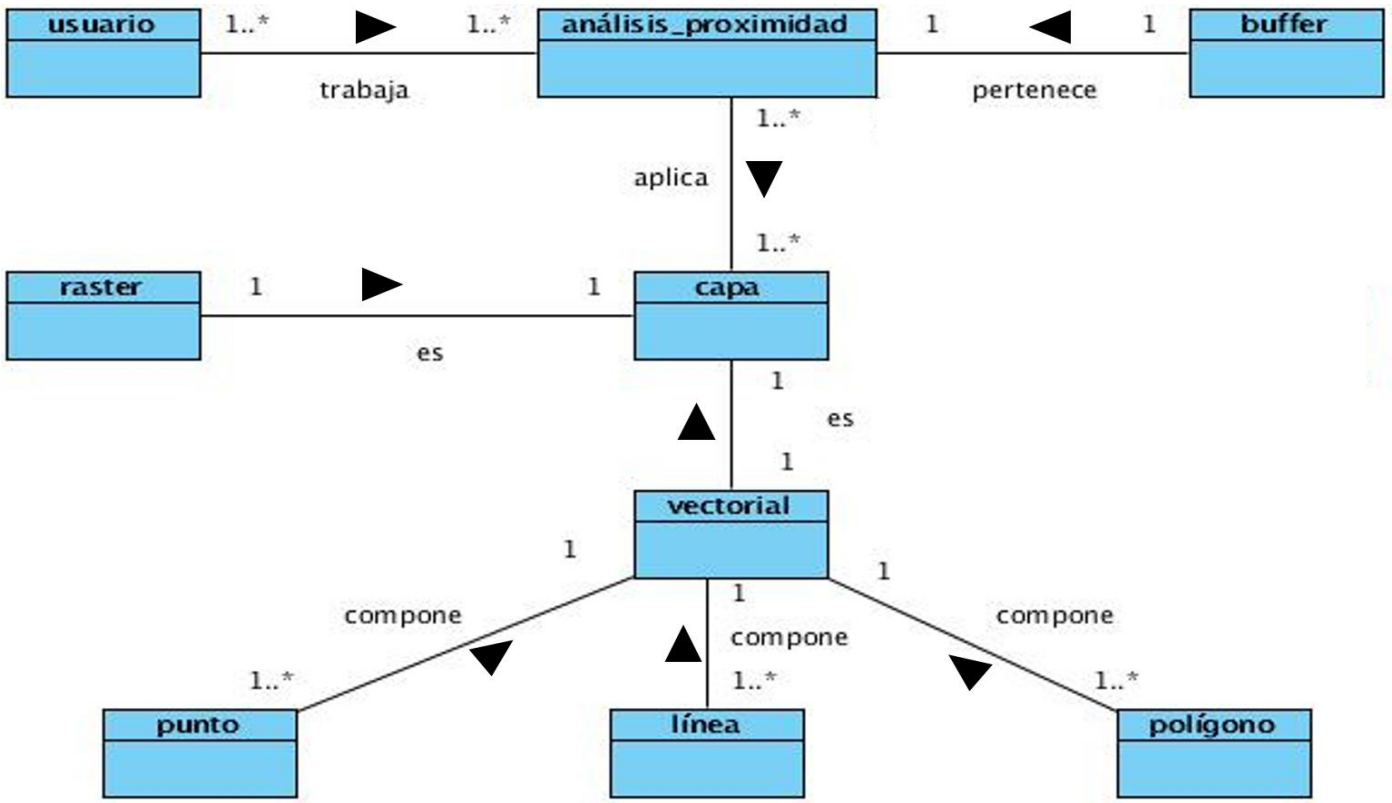


Figura 6: Representación del modelo de dominio

2.2 Técnicas de captura de requisitos

Existen varias técnicas que posibilitan capturar los requisitos de *software* de forma eficiente y segura, como la entrevista, introspección, cuestionarios, listas de verificación, tormenta de ideas y análisis de la documentación (Baños, 2012).

La tormenta de ideas permite "*(…) la realización de reuniones en grupo cuyo objetivo es la generación de ideas en un ambiente libre de críticas o juicios. Puede ayudar a generar una gran variedad de puntos de vista del problema y a formularlo de diferentes formas, sobre todo al comienzo del proceso de captura, cuando los requisitos son todavía muy difusos (…)*" (Ramos,2013).

Para satisfacer las necesidades del cliente se realizó la extracción de los requisitos del *software* mediante la utilización de la técnica tormenta de ideas. Esta se realizó entre un grupo de profesionales que hacen uso de la plataforma GeneSIG con el objetivo de facilitarles el análisis de proximidad.

2.3 Especificación de requisitos

Los requisitos de *software* son las necesidades de los clientes, las funcionalidades que los usuarios desean que proporcione el sistema y las condiciones de restricción en las que debe operar. Los requisitos del *software* son una descripción general de cómo debe funcionar el sistema a implementar y pueden ser clasificados en funcionales y no funcionales.

2.3.1 Requisitos funcionales

Los requisitos funcionales (RF) son capacidades o condiciones que el sistema debe cumplir. Son establecidos entre el cliente y los desarrolladores sobre lo que debe o no hacer el sistema (Kruchten, 2003). Los requisitos, identificados por procesos, para el desarrollo del componente son los siguientes:

Proceso: Generar diagrama de Voronoi de un conjunto de puntos y visualizarlo en el mapa de la plataforma GeneSIG.

RF1. Seleccionar capas de tipo punto que se encuentren en el mapa de la plataforma GeneSIG.

La aplicación debe permitir seleccionar una capa de tipo punto, que se encuentre en el árbol de capas de la plataforma GeneSIG, al usuario.

RF2. Adicionar punto generador.

La aplicación debe permitir seleccionar un punto en el mapa de la plataforma GeneSIG realizando un clic sobre el mapa para incorporarlo al grupo de puntos generadores del diagrama.

RF3. Eliminar punto generador.

La aplicación debe permitir eliminar un punto que haya sido seleccionado anteriormente por el usuario sobre el mapa de la plataforma GeneSIG.

RF4. Generar el diagrama de Voronoi para un conjunto de puntos.

La aplicación debe permitir generar el diagrama de Voronoi correspondiente a un conjunto de puntos, adicionarlo al árbol de capas de la plataforma GeneSIG y visualizarlo en el mapa.

RF5. Generar el diagrama de Voronoi para un conjunto de puntos.

La aplicación debe permitir generar el diagrama de Voronoi correspondiente a un conjunto de puntos, que se encuentren dentro de los límites del *extent* de la interfaz de la plataforma GeneSIG.

Proceso: Guardar diagrama de Voronoi de un conjunto de puntos como capa vectorial.

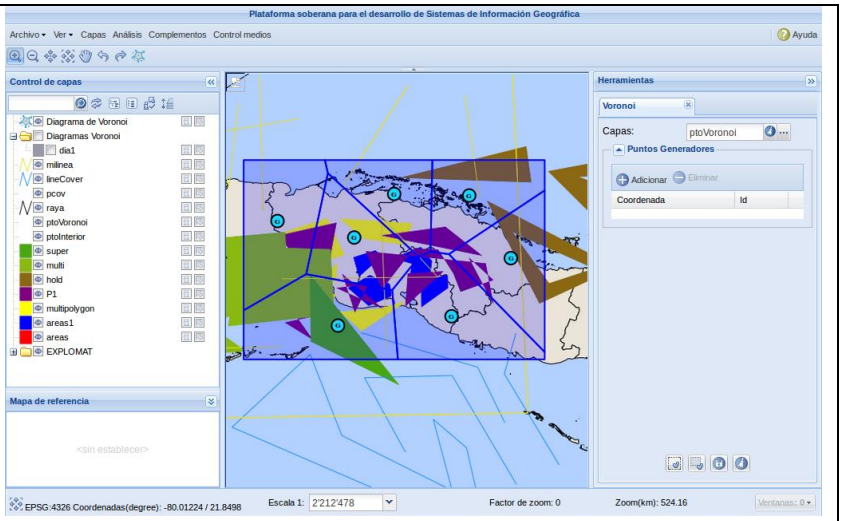
RF6. Guardar el diagrama de Voronoi.

La aplicación debe guardar el diagrama generado como una capa vectorial en la base de datos y adicionar dicha capa al árbol de capas de la plataforma GeneSIG.

A continuación se muestra la especificación del requisito funcional crítico: generar el diagrama de Voronoi para un conjunto de puntos.

Tabla 2 Especificación del requisito funcional: Generar el diagrama de Voronoi para un conjunto de puntos

Conceptos tratados	**Conceptos**	**Atributos**
	Capa, diagrama de Voronoi	-
Precondiciones	**Precondiciones**	**Pre-requisitos**
	Se debe haber cargado una capa de tipo punto y/o seleccionado al menos un punto sobre el mapa.	RF 1, RF 2.
Descripción	1. Se da clic en el botón **Generar diagrama**. 2. Aparece una máscara avisando al usuario que debe esperar a que se genere el diagrama. 3. Se muestra el diagrama en el mapa y se adiciona como una capa en el árbol de capas de la plataforma GeneSIG.	
Validaciones	Solo se activa el botón **Generar diagrama** si se encuentra seleccionada una capa y/o seleccionado al menos un punto sobre el mapa de la plataforma GeneSIG.	
Post-condiciones	Se adiciona la capa con el diagrama al mapa y se visualiza esta sobre el mapa.	
Post-requisitos	RF 5.	

Prototipo de interfaz	

2.3.2 Requisitos no funcionales

Según (Clements, y otros, 2003) existen varias clasificaciones para los requisitos no funcionales (RNF), estas fueron empleadas para determinar los RNF de la propuesta de solución. Los requisitos no funcionales (RNF) son cualidades que el producto debe tener y especifican propiedades y restricciones de rendimiento, fiabilidad, usabilidad y seguridad.

RNF 1 Interfaz externa:

El componente debe poseer una interfaz externa de fácil navegación por el usuario y que mantenga los estándares y características de la plataforma GeneSIG como son: el estilo, los colores, la estructura de los botones y el tamaño de letra. Se tiene en cuenta para lograr una interfaz de usuario amigable y funcional los siguientes principios de diseño:

- Las funcionalidades deben estar al alcance de un clic y nombradas con palabras asociadas a la acción que realizan.
- Los mensajes deben ser en idioma español.
- Garantizar la legibilidad de manera que exista contraste de los colores de los textos con el fondo y el tamaño de la fuente sea adecuado a la vista del usuario.

RNF 2 Usabilidad:

El componente puede ser usado por personas con conocimientos del uso de SIG.

La aplicación debe mostrar al usuario un mensaje, durante el proceso de generación y dibujo del diagrama, que le indique al usuario que debe esperar.

En los mensajes y campos de entrada se deben utilizar palabras de fácil comprensión para el usuario.

RNF 3 *Software*:

Para las estaciones clientes:

- Un navegador web.

Para las estaciones servidor:

- Sistema operativo GNU/Linux Ubuntu 14.04.
- Servidor Web Apache 2.0 o superior, con módulo PHP 5.
- PostgreSQL 9.3 como Sistema Gestor de Base de Datos.
- PostGIS 2.1.2 como extensión de PostgreSQL para el soporte de datos espaciales.
- MapServer 6.4.1 con extensión PHP MapScript.

RNF 4 Hardware:

Para las estaciones clientes:

- Deben tener una tarjeta de red.
- Como mínimo 1 GB de memoria RAM.
- Procesador de 512 MHz como mínimo.

Para las estaciones servidor:

- Deben tener una tarjeta de red.
- El servidor de mapas debe tener como mínimo 2 GB de memoria RAM y 40 GB de disco duro.
- El servidor de base de datos debe tener como mínimo 2 GB de memoria RAM y 40 GB de disco duro.
- Debe tener un procesador de 3 GHz como mínimo.

2.4 Patrón arquitectónico

Según (Clements, y otros, 2003), 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. Existen soluciones probadas dentro de las arquitecturas de *software*, estas son llamadas patrones arquitectónicos.

Para la creación de la solución se toma en cuenta la arquitectura de la plataforma GeneSIG, cuya estructura está basada en el *framework* CartoWeb, el cual tiene como característica poseer una arquitectura bastante modular y escalable, esto implica que lo patrones arquitectónicos seleccionados para regir el desarrollo del *software* son:

Arquitectura basada en componentes: Es independiente de la tecnología que se emplee en la implementación dado que solo cubre aspectos lógicos. El diseño y construcción del sistemas se realiza a través de componentes de *software* reutilizables que se comunican a través de interfaces.

Arquitectura orientada a objetos: Los componentes del estilo se basan en principios Orientados a Objetos: encapsulamiento, herencia y polimorfismo. Son asimismo las unidades de modelado, diseño e implementación, y los objetos y sus interacciones son el centro de las incumbencias en el diseño de la arquitectura y en la estructura de la aplicación (Reynoso, y otros, 2004).

2.5 Estilo arquitectónico

Los estilos arquitectónicos describen un tipo particular de estructura fundamental para un sistema de *software*, conjuntamente con un método asociado que especifica cómo construirlo. Incluye información acerca de cuándo usar la arquitectura que describe, sus invariantes y especializaciones, así como las consecuencias de su aplicación (Larman, 1999).

Para el desarrollo de la solución se seleccionó el estilo arquitectónico: Llamada y Retorno.

Llamada y Retorno: El sistema está compuesto por un programa principal y varios subprogramas que se comunican mediante interfaces. Facilita la modularidad y la escalabilidad.

2.6 Patrones de diseño de *software*

Un patrón de diseño de *software* describe una estructura de diseño que resuelve un problema de diseño particular dentro de un contexto específico. Permite al diseñador obtener una descripción, la cual facilita determinar si el patrón es aplicable al trabajo actual, si se puede reutilizar y así ahorrar tiempo de diseño, y

si el patrón puede servir como guía para desarrollar un patrón similar pero con diferente estructura y funcionalidad (Pressman, 2010).

2.6.1 Patrones para la asignación de responsabilidades (GRASP)

Los GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones (Larman, 2003). Para la implementación de la aplicación se hizo uso de los patrones siguientes:

El patrón Creador, ayuda a identificar quién debe ser el responsable de la creación (o instancia) de nuevos objetos o clases. Una de las consecuencias de usar este patrón es la visibilidad entre la clase creada y la clase creadora, lo que supone bajo acoplamiento y redunda en facilidad de mantenimiento y reutilización. El empleo de este patrón se evidencia, por ejemplo, en la clase ClientVoronoi, encargada de crear las instancias de la clase voronoiRequest.

Se utiliza el patrón Experto el cual indica que cada clase cuenta con la información necesaria para cumplir sus responsabilidades, y es la responsable de manejar la información. Su ventaja es que mantiene el encapsulamiento de la información (bajo acoplamiento), los objetos utilizan su propia información para llevar a cabo sus tareas, las clases son más fáciles de entender. Por ejemplo, la clase ServerVoronoi es la encargada de la interacción con el servidor de mapas y el servidor de bases de datos, por tanto tiene la información correspondiente para realizar todas las funcionalidades del sistema.

El patrón Bajo Acoplamiento que indica una menor dependencia entre clases se emplea para fomentar la reutilización de código. El acoplamiento mide el grado en que una clase está conectada a otra, tiene conocimiento de otra o, de alguna manera, depende de otra. Es utilizado, además, el patrón Alta Cohesión debido a que los conceptos de cohesión y acoplamiento están íntimamente relacionados. Un mayor grado de cohesión implica un menor grado de acoplamiento. La cohesión mide el grado en que están relacionadas las responsabilidades de una clase.

2.7 Diagrama de clases del diseño

Los diagramas de clases del diseño describen gráficamente la estructura de una aplicación, estos.se emplean en el modelado de las vistas del diseño del sistema para describir las especificaciones de las interfaces y las clases del *software*, lo que facilita el trabajo de los implementadores (Quesada, 2015).

Una clase del diseño es una abstracción de una clase real o construcción similar en la implementación del sistema. El lenguaje que se utiliza en dichas clases es el mismo que se emplea para la implementación del sistema. Los atributos y los métodos tienen correspondencia con las operaciones que fueron utilizadas en

la implementación. A continuación se propone el diagrama de clases del diseño para el Proceso 1, Generar diagrama de Voronoi para un conjunto de puntos. El diagrama del Proceso 2 puede ser consultado en el documento “Componente Voronoi. Modelo de diseño” del expediente de proyecto.

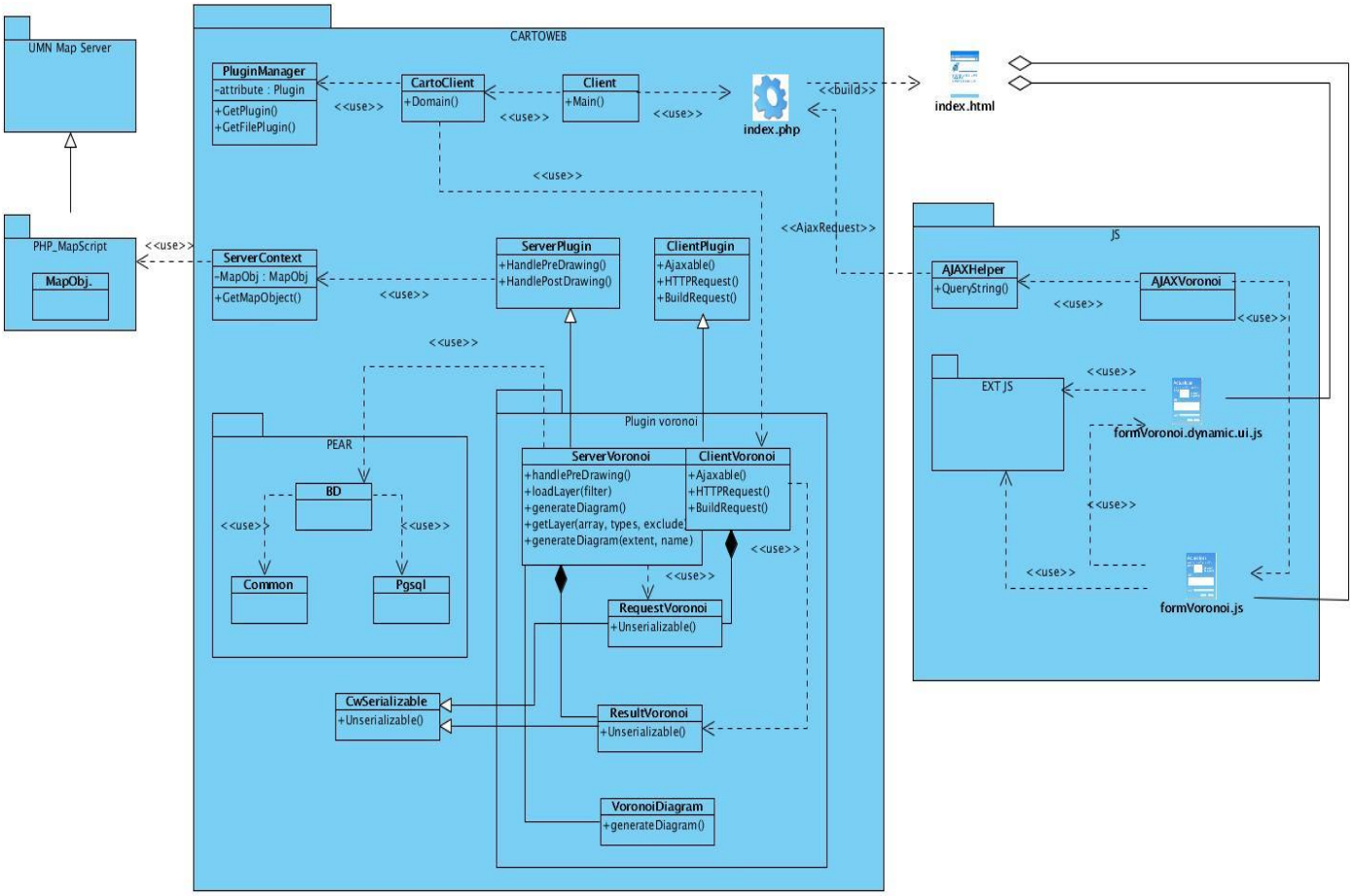


Figura 7: Diagrama de clases del diseño.

2.7.1 Descripción del modelo de diseño

Para la mejor comprensión del diagrama de clases del diseño se describen a continuación los elementos que en el aparecen.

- **VoronoiResult**: Clase encargada de transportar los datos generados en **ServerVoronoi** a la clase **ClientVoronoi**.
- **VoronoiRequest**: Clase encargada de transportar los datos recogidos en **ClientVoronoi** desde la interfaz y transportarlos a la clase **ServerVoronoi**.

- **ClientVoronoi**: Clase encargada de recoger y seleccionar de la interfaz los datos necesarios para operar en el servidor.
- **ServerVoronoi**: Clase que contiene las funciones principales del negocio y la conexión con la base de datos. Envía las respuestas que necesita el **ClientVoronoi** para mostrarlas a través de la interfaz al usuario.
- **Voronoi**: Clase encargada de generar el diagrama de Voronoi de un conjunto de puntos.
- **Ajax_Voronoi**: Encargada de gestionar el pedido y la respuesta a las peticiones del usuario a través de Ajax.
- **Index.html**: Página principal encargada de mostrar en el mapa la región localizada.
- **Ajax_Helper**: Tiene como propósito enviar las respuestas de los *plugins* ¨AJAX¨, para alimentar a los *plugins* que responden a las peticiones del usuario.
- **CwSerializable**: Clase encargada de serializar todas aquellas clases que pueden ser serializadas, permitiendo la comunicación entre el *Client* y el *Server* del *plugin*.
- **Pgsql**: Clase encargada de gestionar desde PHP las funciones de PostgreSQL.
- **Common**: Clase encargada de administrar las conexiones a la base de datos para ejecutar las consultas a la misma satisfactoriamente, esto incluye tratamiento de los datos.
- **BD**: Clase encargada de establecer la conexión con el servidor de base de datos para procesar los objetos a editar.
- **ServerContext**: Clase contenedora de la información común que ha de ser utilizada por la parte cliente y la servidora, empleando la información seleccionada como un objeto para un fácil manejo de los datos.
- **ServerPlugin**: Clase que proporciona la base de herramientas para el desarrollo de *plugins*.
- **ClientPlugin**: Clase que contiene las interfaces necesarias para los *plugins* del lado del cliente.
- **PluginManager**: Clase que se utiliza para gestionar la base de *plugins*.
- **CartoClient**: Clase que obtiene e integra todos los datos y operaciones realizadas en la interfaz y que intervienen en los RF y se definen una serie de variables globales que se utilizan en la aplicación.

- **Client**: Contiene todos los archivos específicos de PHP del lado del **CartoClient** y permite la interacción entre la **index.php** y la **CartoClient**.
- **Index.php**: Controla la realización de los RF, recibe las peticiones solicitadas por el cliente, ejecuta las operaciones necesarias y manda a construir la **ClientPage**.

2.8 Diseño de la base de datos

El diseño de la base datos tiene como propósito asegurarse de que los datos persistentes, estos son los datos que se necesita se conserven en el tiempo, son almacenados y consistentes. Además, se encarga de definir el comportamiento que debe ser implementado en la base de datos.

Los conceptos identificados en el dominio, que iban a persistir en el tiempo, se tuvieron en cuenta para crear el diagrama de clases persistentes. La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. Las clases persistentes identificadas son capa_origen, la cual almacena el nombre de la capa de la cual se tomaron los puntos para generar el diagrama, y diagrama_voronoi, que contiene los polígonos que forman el diagrama y los puntos generadores de cada uno de estos polígonos.

Figura 8: Diagrama de clases persistentes.

A partir del diagrama de clases persistentes se generó el siguiente modelo físico de la base de datos. En este se incluye una tabla que almacena la relación entre la capa origen y los diagramas que se crean a partir de dicha capa.

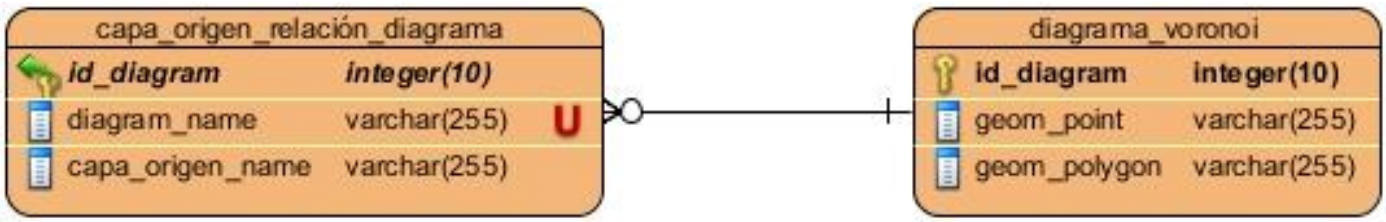


Figura 9: Diagrama de Entidad-Relación.

Conclusiones parciales

Los conceptos identificados en el dominio, que persisten en el tiempo, se tienen en cuenta para crear el diagrama de clases persistentes. Realizada la presentación de la propuesta de solución, ya están creadas las bases para dar paso a la fase de construcción.

La aplicación de la técnica de captura de requisitos tormenta de ideas permitió una mejor comprensión del problema, arrojando como resultado los requisitos funcionales y no funcionales del componente a desarrollar.

Capítulo 3. Implementación y pruebas

En el presente capítulo se tienen en cuenta todos los aspectos del diseño del sistema con el fin de llevar a cabo el desarrollo de los flujos de trabajo de implementación y prueba. Se muestra la organización del sistema mediante el modelo de componentes el cual representa la vista estática del sistema y la situación física de los distintos componentes lógicos desarrollados a través del modelo de despliegue. Se describen los estilos de programación y los estándares de codificación empleados. Y por último se define el proceso de pruebas en el cual se verifica que todos los requisitos hayan sido implementados y funcionen correctamente.

3.1 Diagrama de componente

Los diagramas de componentes permiten modelar la visión física del sistema a desarrollar, mostrando la organización del mismo y las relaciones que existen entre sus componentes (Torres Reyes, y otros, 2014). Facilita la comprensión de los cambios en los requisitos al establecer el sistema como una colección de componentes con interfaces bien definidas. A continuación se muestra el diagrama de componentes resultante (ver Figura 11).

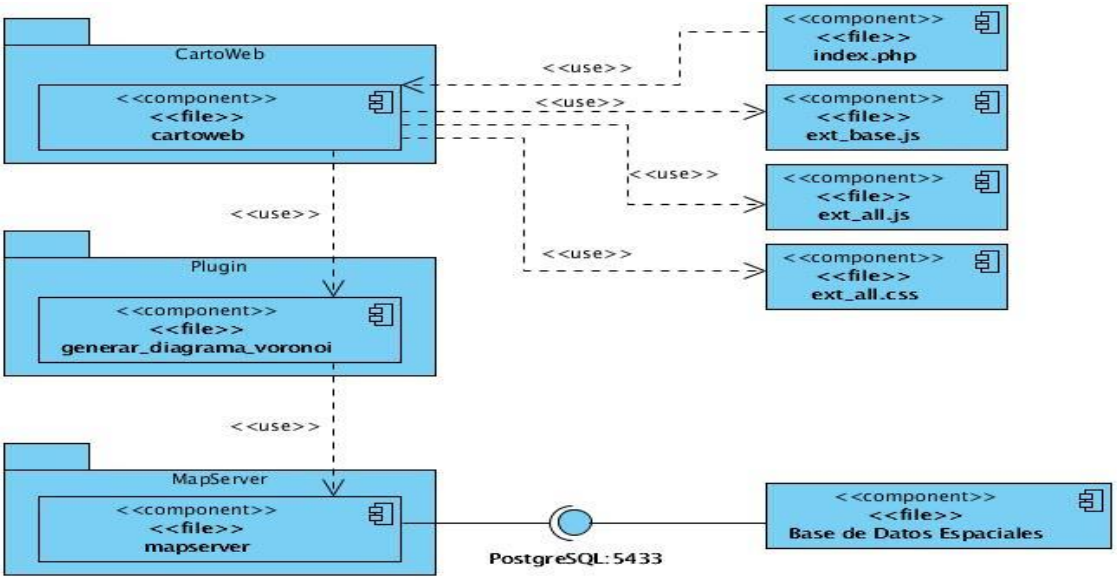


Figura 10: Diagrama de componentes

3.2 Estándares de codificación

Los estándares de codificación benefician la comunicación entre los desarrolladores, favoreciendo el mantenimiento de los sistemas. Para implementar la solución propuesta se utiliza el estándar *CamelCase*, particularmente el tipo *lowerCamelCase*. Este estándar se conoce como Mayúsculas/Minúsculas y se puede

aplicar a palabras o frases unidas en las declaraciones de variable y funciones. El tipo *lowerCamelCase* consiste en que la primera letra de la primera palabra es con minúscula y el resto de las primeras letras son con mayúscula. Se colocan comentarios encima de los métodos para explicar su utilidad.

```
/**
 * Mapserver 5.
 * Funcion que devuelve las features (puntos) de las capas postgis y shape
 * @param $name
 */
public function getFeaturesLayerByName($name)
{
    $db = $this->getPluginConnection();
    $qX="SELECT ST_X(ST_GeomFromText(?))";
    $qY="SELECT ST_Y(ST_GeomFromText(?))";
```

Figura 11: Representación de los estándares de código utilizados.

3.3 Modelo de despliegue

El modelo de despliegue muestra un modelo de objetos y describe las relaciones físicas entre los objetos y como se distribuyen las funcionalidades entre las estaciones de trabajo. En la Figura 13 aparece el modelo de despliegue de la solución propuesta.

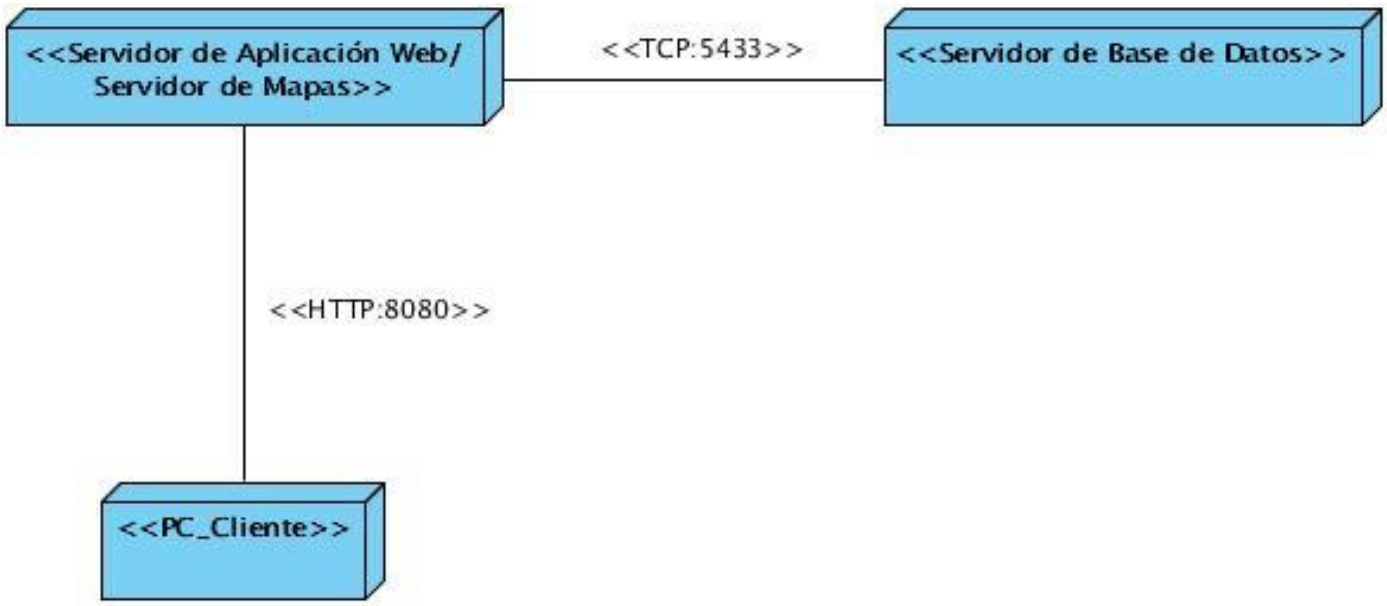


Figura 12: Diagrama de despliegue.

3.4 Modelo de pruebas

Las pruebas de *software* son la actividad más común de control de la calidad realizada en los proyectos para asegurar el correcto funcionamiento del *software*. Tienen como objetivos la verificación de la correcta implementación de los requisitos explícitamente establecidos, la adecuada integración de los componentes que conforman el sistema y la ejecución de casos de prueba que permitan detectar el mayor número de no conformidades y corregirlas antes de la entrega del software al cliente (UCID), 2012).

Las pruebas no aseguran la ausencia de defectos en el *software*. El objetivo de estas es encontrar los errores, por tanto, una buena prueba es aquella que tiene la mayor probabilidad de encontrar un error que no haya sido descubierto aún. Según la metodología de desarrollo empleada la ejecución de pruebas debe ser sistemática, desde la fase de construcción y hasta obtener una versión liberada del producto, para reducir el número de errores no detectados.

Tomando en cuenta que el componente tiene una interfaz de usuario a través de la cual el usuario podrá hacer uso de sus funcionalidades y las indicaciones de la metodología empleada se seleccionaron los siguientes métodos y técnicas de pruebas.

3.4.1 Tipo de prueba aplicada

Para la validación del funcionamiento de los Requisitos Funcionales del *software*, reflejados en su interfaz sin tener en cuenta el funcionamiento interno de la aplicación, se emplea el método de pruebas Caja Negra. Este método no considera la codificación dentro de los parámetros a evaluar. Se basa en que las entradas sean aceptadas de forma adecuada y se reciba una salida correcta demostrando que cada función es completamente operativa. Mediante la descripción de Casos de Prueba se comprueba que cada acción provoque la respuesta correcta del sistema. Se toman en cuenta las características especiales de la solución propuesta, donde la mayoría de las entradas del usuario son eventos del ratón.

Para identificar problemas en el desempeño del componente se emplean las **pruebas de carga**, para examinar el tiempo de respuesta del servidor con diferentes cantidades de peticiones (generar diagrama de Voronoi) concurrentes, y **pruebas de estrés** para determinar la cantidad de peticiones (generar diagrama de Voronoi) que consigue manejar el servidor. Estas pruebas se realizaron con el componente ya integrado a la plataforma GeneSIG.

3.4.2 Diseño de Casos de Prueba

Los diseños de caso de prueba (DCP) permiten comprobar las funcionalidades de la solución. Los DCP están compuestos por campos que describen las entradas y la reacción del sistema en el escenario descrito.

Las variables pueden tomar valores como: V (válido), N/A (en este caso no es necesario conocer el valor de la variable) o I (inválido). A continuación, se muestra un ejemplo de DCP empleado para verificar el correcto funcionamiento de la propuesta de solución.

Tabla 3 Variables para el caso de prueba del RF Generar diagrama de Voronoi para un conjunto de puntos

No.	Nombre del Campo	Valor Nulo	Descripción
1	Seleccionar capa de puntos	No	Es un *searchfield* que permite mostrar una ventana con las capas de tipo punto.
2	Generar diagrama	No	Es un botón que permite generar el diagrama y representarlo en el mapa.

Tabla 4 Descripción del caso de prueba del RF Generar diagrama de Voronoi para un conjunto de puntos

Escenario	Descripción	Seleccionar capa de puntos	Generar diagrama	Respuesta del sistema	Flujo central
EC1.1 Seleccionar capa de tipo punto	Se intenta permitir al usuario seleccionar una capa de tipo punto, de la lista desplegable.	V Seleccionar el *searchfield*.	N/A	Muestra una ventana con los nombres de las capas de tipo punto que se encuentran en el árbol de capas de GeneSIG	1. Se selecciona desde de la ventana principal el *searchfield* “Seleccionar capa”. 2. Se muestra una ventana con los nombres de las capas de tipo punto que se encuentran en el árbol de capas de GeneSIG. 3. Se selecciona la capa a la que se le desea

					generar el diagrama. 4. Se presionar el botón “Aceptar”.
EC1.2 Generar diagrama y mostrarlo en el mapa.	Se intenta permitir al usuario generar el diagrama de la capa seleccionada y visualizarlo en el mapa.	V Seleccionar el *searchfield*.	V Dar clic en el botón “Generar diagrama”.	Genera el diagrama y lo muestra en el mapa.	1. Se selecciona desde la ventana principal la opción “Generar diagrama”. 2. Se muestra el diagrama en el mapa.
EC1.3 Cancelar generación de diagrama.	Se intenta permitir al usuario anular cualquier acción que conlleve generar el diagrama de una capa de puntos.	N/A	N/A	Cierra la ventana principal.	1. Se selecciona la opción del botón “Cancelar” de la ventana principal.

3.5 Aplicación y resultado de las pruebas

Fueron probados todos los requisitos funcionales del sistema. Durante la primera iteración de las pruebas funcionales se encontraron 3 no conformidades referidas a la validación de campos que no pueden estar vacíos, se solucionaron y se aplicaron nuevamente las pruebas con el objetivo de comprobar que las no conformidades antes detectadas estuvieran resueltas y no generaran otras. Durante la segunda iteración se encontraron 2 no conformidades referentes a errores ortográficos. En la tercera iteración no se encontraron errores en la aplicación.

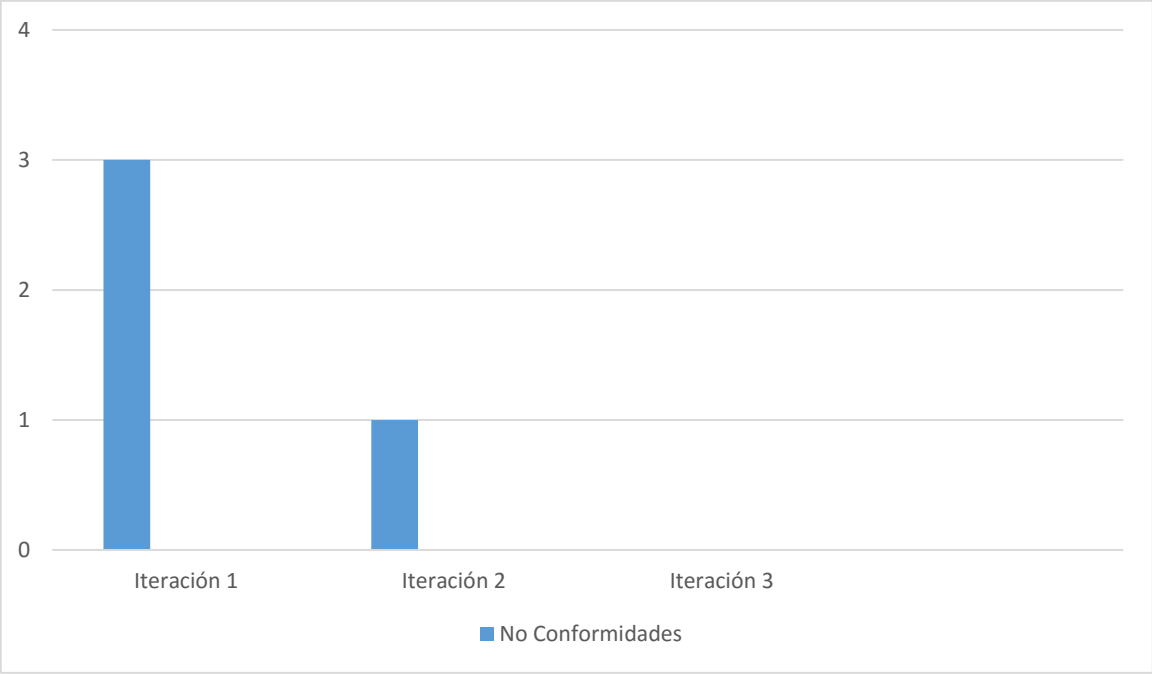


Figura 13: Resultados de las pruebas de caja negra.

Las pruebas de rendimiento fueron realizadas con la herramienta Apache JMeter versión 2.8. Las pruebas fueron realizadas con 5, 50 y 100 muestras de datos, cuyas muestras serán el resultado de haber realizado 40, 50 y 100 peticiones HTTP respectivamente a la aplicación, estas peticiones representan peticiones (generar diagrama de Voronoi) simultáneas.

Luego de observar los resultados de las pruebas obtenidas.se interpreta que: para un total de 40 y 50 peticiones simultáneas, el tiempo medio que demoró el sistema en responder todas las peticiones fue de 1.2 segundos y 1.25 segundos respectivamente y sin la ocurrencia de errores. Finalmente se comprobó con 100 peticiones simultáneas como el caso más crítico, para estresar el sistema; el tiempo arrojado fue de 1.75 segundos, que se considera satisfactorio al no sobrepasar los 10 segundos que plantea como máximos el Manual de usuario de Apache JMeter para pruebas de rendimiento a sistemas web y no ocurrieron errores en las respuestas a las peticiones.

Conclusiones parciales

El empleo de los patrones arquitectónicos, junto un correcto diseño de la base de datos permite realizar una satisfactoria implementación. La creación del diagrama de componentes permitió lograr una representación de la distribución de las clases dentro de los distintos componentes.

La aplicación de las pruebas de caja negra permite verificar el correcto funcionamiento de la implementación de los requisitos funcionales del componente desarrollado, luego de tres iteraciones dejaron de aparecer no conformidades en como resultado de la ejecución de las pruebas.

Conclusiones generales

Una vez culminada la investigación se puede afirmar que se les dio cumplimiento a los objetivos planteados, arribando a las siguientes conclusiones:

- El estudio de los principales conceptos relacionados al análisis de proximidad en capas vectoriales permitió identificar el cálculo del diagrama de Voronoi, utilizando el algoritmo de Fortune, como la herramienta correcta para dar solución al problema de la investigación.
- Las herramientas y tecnologías propuestas para la construcción del componente, se corresponden con los utilizados para el desarrollo de la plataforma GeneSIG, lo que facilita la integración con la plataforma.
- Se desarrolló el componente para el cálculo de la zona del espacio más próxima a una entidad, para mejorar el análisis de proximidad sobre capas vectoriales en los Sistemas de Información Geográfica que se desarrollan utilizando la plataforma GeneSIG.
- A través de la aplicación de las pruebas funcionales se verifica que el componente funciona de acuerdo con lo especificado en las descripciones de los requisitos funcionales.

Recomendaciones

Terminada la investigación y tomando en cuenta las experiencias obtenidas se recomiendan las siguientes acciones.

- Incorporar el cálculo del diagrama para las primitivas línea y polígono.
- Ampliar al componente desarrollado de forma que permita señalizar el punto más cercano a una ubicación determinada, a partir de diagrama de Voronoi obtenido.
- Definir un esquema de almacenamiento para almacenar el diagrama de Voronoi obtenido haciendo uso de la estructura de datos DCEL.

Referencias bibliográficas

Achour, M. y Betz*. PHP Manual Documentation Group*. [En línea] 2006. [Citado el: 2 de febrero de 2016.] http://php.net/manual/en/index.php].

Baños, O. *Análisis del módulo de administración de la plataforma Atlas.* La Habana, 2012.

Bonilla, M. *Arquitectura de software*. [En línea] 12 de diciembre de 2009. [Citado el: 29 de enero de 2016.] https://www.fing.edu.uy/cpap/cursos/arquitectura-de-*software*.

Boots, B., y otros. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.* John Wiley & Sons Ltd, 2000. 83 p.

Camacho, E., Cardeso, F. y Núñez, G. *Arquitecturas de software, Guía de estudio.* 2004. 35 p.

Clements, J., Bass, L. y Kazman, R. *Software Architecture in Practice. London*: Addison Wiley & Sons Ltd, 2003. 105 p.

Reyna, Á. *El uso de Sistemas de Información Geográfica (SIG) en el análisis demográfico de situaciones de desastre.* En Comisión para América Latina y el Caribe (CEPAL) Notas de población. Santiago de Chile: Organización de Naciones Unidas (ONU), 1996. 137 p.

De Almeida, T. *Geoprocessamento aplicado ao planejamiento dos transportes urbanos.* Monografía presentada como requisito para obtener el título de técnico en geoprocesamiento, Centro Federal de Educación Tecnológica de Paraíba, Paraíba, 2006.

ESRI. GIS Dictionary. [En línea] [Citado el: 25 de enero de 2016.] http://support.esri.com/other-resources/gis-dictionary.

Filgueiras, N. y Torres, A. *Módulo de estadística descriptiva para la LPS Aplicativos SIG.* Tesis para optar por el título de ingeniero en ciencias informáticas, Universidad de las Ciencias Informáticas, La Habana, 2015.

Fortune, S. *A fast algorithm for Voronoi Diagrams.* John Wiley & Sons Ltd, 1987. 6p.

Hernández, C. *Módulos de administración y gestión de información del SIG INRH.* Tesis para optar por el título de ingeniero en ciencias informáticas, Universidad de las Ciencias Informáticas, La Habana, 2011. p. 35.

Marbate, P. y Gupta, R. *Fortune's Method: An efficient method for Voronoi Diagram construction.* International Journal of Advanced Research in Computer and Communication Engineering, 2013, Vol. 2.: p. 1-6.

De Pietri, D., y otros. *Indicadores de accesibilidad geográfica a los centros de atención primaria para la gestión de inequidades.* Revista Panamericana de Salud Pública, 2013, Vol. 34.: p. 5-8.

Kruchten, P. *The Rational Unified Process: An Introduction.* Addison Wesley, 2003. p. 19-28.

Larman, C. *UML y patrones.* Prentice Hall Hispanoamérica, 1999. P. 23.

Martínez, U. *Aplicación de la Geometría Computacional en la Reconstrucción 3D basada en Diagramas de Voronoi.* Puebla, México, 2015.

Referencias bibliográficas

Mestre, L. *Módulo de operaciones geométricas en capas vectoriales para GeneSIG.* Tesis para optar por el título de ingeniero en ciencias informáticas, Universidad de las Ciencias Informáticas, La Habana, 2014.

Quesada, A. *Módulo raster para la plataforma GeneSIG.* Tesis para optar por el título de ingeniero en ciencias informáticas, Universidad de las Ciencias Informáticas, La Habana, 2015.

Morales, A. *¿Por qué utilizar PostGIS?* [En línea] 2012. [Citado el: 2 de febrero de 2016.] http://mapserver.org/about.html#about.

Olaya, V. *Sistemas de Información Geográfica.* 2014. P. 8-300.

Orallo, E. *El lenguaje unificado de modelado (UML).* [En línea] 2003. [Citado el: 1 de febrero de 2016.] http://es.scribd.com/doc/54817695/Act-a-Uml.

Guerrero, C. y otros. *Patrones de diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web.* SciELO, 2013, vol. 24: p. 1-8.

Pressman, R. *Ingeniería de software un enfoque práctico.* McGraw-Hill Interamericana Editores, S.A. de C.V., 2010. p. 25-135.

Reynoso, C. y Kicillof, N. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft*. Buenos Aires, 2004. p. 2-16.

Rivero, F. *Geometría Computacional.* Mérida, Universidad de los Andes, 2006. p. 25-38.

Roos, N. *Linking patients to hospitals service populations. Defining urban hospital service populations.* Medical Care. 1993.

Santos, J. y Cocero, D. *Los SIG raster: Herramienta de análisis medioambiental y territorial.* Universidad Nacional de Educación a Distancia, 2005.

Torres, C. y Silvera, A. *Módulo de análisis basado en métodos de interpolación para GeneSIG.* Tesis para optar por el título de ingeniero en ciencias informáticas, Universidad de las Ciencias Informáticas, La Habana, 2014.

Unidad de compatibilización, integración y desarrollo de *software* para la defensa (UCID). 2012. Proceso de desarrollo y gestión de proyectos de *software*. 2012. Vol. 1.5.

Uva, M. y Campanella, O. *AP-SIG: un SIG con funciones específicas para Agricultura de Precisión.* Barcelona, 2006.

Edelsbrunner, H. y Seidel, R. *Voronoi diagrams and arrangements.* Discrete & Computational Geometry, 1986, vol. 1: p 25-44.

Zayas, C. *Metodología de la Investigación Científica.* Santiago de Cuba: Centro de estudios de Educación Superior Manuel F. Gran, 1995.