UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Herramienta de modelación de superficies de terrenos sobre Geotools

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

Autor: **Felix Alejandro Prieto Carratalá**

Tutor: **Lic. Yusnier Valle Martínez**

Ciudad de la Habana, Cuba
15 de junio de 2009, 9:25am

Agradecimientos

Quiero agradecer, primero a mis maravillosos padres por la dedicación y el amor que me han dado durante toda mi vida, sin importar las dificultades y mis defectos; a mi novia Yeimí, por ser tan especial y cariñosa conmigo, a pesar de la distancia; a mi hermana linda por estar siempre dispuesta a ayudarme; y por último, pero no menos importante, a mi tutor y cuñado Valle, que me ayudó y guió durante esta difícil experiencia, con su paciente colaboración. A todos, mi más sincero agradecimiento.

Dedicatoria

A mi padres (únicos en el mundo), mi hermana y su esposo (mi tutor) y a mi preciosa novia.

DECLARACIÓN JURADA DE AUTORÍA Y AGRADECIMIENTOS

Yo Felix Alejandro Prieto Carratalá, con carné de identidad 85091611623, declaro que soy el autor principal del resultado que expongo en la presente memoria titulada "Herramienta de modelación de superficies de terreno sobre Geotools", para optar por el título de Ingeniero en Ciencias Informáticas.

El presente trabajo fue desarrollado individualmente en el transcurso de los años 2008−09.

En especial deseo agradecer a Yusnier Valle Martínez, quien fungió como tutor de mi formación como ingeniero. A todos ellos, así como a otros colegas y amigos que no he mencionado por razones de espacio, mi más sincero agradecimiento.

Finalmente declaro que todo lo anteriormente expuesto se ajusta a la verdad, y asumo la responsabilidad moral y jurídica que se derive de este juramento profesional.

Y para que así conste, firmo la presente declaración jurada de autoría en Ciudad de la Habana a los ... días del mes de ... del año 2009.

_____

Firma del Titular

Resumen

La modelación de terrenos en 3D es la representación digital de la topografía de la superficie terrestre a partir de conjuntos de datos. Las herramientas de visualización de superficies de terrenos usualmente son parte de un sistema de procesamiento de datos georefenciados o Sistema de Información Geográfica (SIG). Los SIGs se han convertido en una de las tecnologías de más rápido crecimiento en la actualidad y son utilizados en muchos sectores de la sociedad y la economía como herramientas de ayuda en la gestión y toma de decisiones. La Universidad de las Ciencias Informáticas (UCI), en su papel de empresa productora de software, no se ha quedado al margen de las tendencias en el mundo de la Informática y creó el Polo de Geoinformática, organización dedicada a desarrollar productos del ámbito de los SIGs. El propósito de la presente investigación es brindarle una respuesta efectiva a una necesidad del Polo, puesto que ilustra el proceso de desarrollo de una aplicación SIG que sirve de ejemplo de estudio para futuras implementaciones de sistemas de este tipo. La ejemplificación se lleva a cabo mediante el uso de una de las librerías más empleadas actualmente: Geotools, la cual permite obtener soluciones basadas en estándares definidos para describir la información geográfica.

La investigación abarca el estudio de las principales funcionalidades de la librería Geotools, su arquitectura y componentes; incluye el diseño e implementación de un prototipo de aplicación SIG con funciones básicas comunes de estos sistemas, teniendo en cuenta los diversos elementos brindados por la librería, como por ejemplo: lectura de archivos de imágenes, visualización interactiva de mapas, etc; y concluye con la adhesión de una extensión al prototipo: una herramienta de visualización de terrenos en 3D -que tiene como fuente de datos archivos de tipo raster- para demostrar la posibilidad de ampliar el alcance de los productos de software desarrollados con Geotools.

Palabras claves: Sistema de Información Geográfica (SIG), raster, Geotools y modelación de superficies.

Índice general

Índice de figuras

CAPÍTULO 1

Introducción

El análisis y representación de datos geográficos existen posiblemente desde hace 15000 años o tal vez más, cuando el hombre de Cro-Magnon pintaba en las paredes los animales que cazaba, asociando estos dibujos con trazas lineales que, se cree, cuadraban con las rutas de migración de esas especies[33]. Este ejemplo constituye tal vez el primer antecendente de un SIG moderno, pues ya existía una imagen asociada con un atributo de información.

Es difícil establecer la definición de SIG ya que abarca la integración de áreas muy diversas, pero existe una bastante aceptada, redactada por el National Center of Geographic Information and Analysis (NCGIA) que lo enfoca como:

> **Sistema de hardware, software y procedimientos elaborados para facilitar la obtención, gestión, manipulación, análisis, modelado, representación y salida de datos espacialmente referenciados, para resolver problemas complejos de planificación y gestión[11].**

Esta investigación no pretente abarcar todos los aspectos de los SIGs -pues se alejaría de los objetivos-, por tanto, la definición se limitará a la siguiente:

> **Software que usa datos, tanto georeferenciados como datos no espaciales, y que incluye operaciones de análisis de los mismos[11].**

Los SIGs han ido ganando en importancia y aplicaciones en los últimos años, pues convirtiendo mapas y otros tipos de información espacial a un formato digital, permiten la manipulación y observación del conocimiento geográfico en una forma nunca antes vista. A pesar de que el núcleo de las funcionalidades que un SIG provee consiste en el análisis geográfico de los datos geoespaciales, la mayor parte de estas herramientas traen funciones de visualización en 3D, que aunque son bastante básicas permiten mostrar algunas perspectivas, efectos de nieblas, vuelos animados sobres modelos digitales del terreno, etc[14].

Diseño de la investigación

Teniendo en cuenta el inmenso potencial de estas tecnologías, en la UCI se creó el Polo de Geoinformática para atender los proyectos que pudieran desarrollarse en este sector. En los últimos años dicho Polo sigue una estrategia de migración para colocar toda la infraestructura a plataformas libres; el primer motivo es lograr alcanzar la independencia tecnológica, y no depender de las herramientas de un monopolio como Microsoft; la segunda razón es la seguridad, ya que no hay garantías de saber qué hace exactamente el software privativo; y por último, el costo de las licencias de uso generalmente es excesivo y es lujo que un país pobre y bloqueado como el nuestro no puede darse.

En el mundo de las aplicaciones SIGs libres se ha registrado un crecimiento vertiginoso de software; por ejemplo, hasta el momento están catalogados más de 350 proyectos en http://www.freegis.org[1]. Entre estos proyectos destaca uno por su impacto en la esfera: Geotools, librería que permite desarrollar soluciones enfocadas en los estándares del sector; está escrita en el lenguaje de programación Java y se nutre de una comunidad de usuarios muy dinámica.

Aunque actualmente en el mundo muchas soluciones integran Geotools, en el Polo de Geoinformática no existen proyectos que hagan uso de sus potencialidades en la implementación de funcionalidades de los SIGs y de una herramienta sencilla de modelación de superficies para dichos sistemas - como es común en este tipo de aplicaciones-.

Teniendo en cuenta la situación expuesta anteriormente se define como **problema científico**: la inexistencia de una aplicación que demuestre la factibilidad de crear prototipos SIGs, basados en las funcionalidades básicas que ofrece Geotools, que pueden ser extendidos a partir de la adición de nuevas herramientas.

Para darle respuesta al problema descrito es necesario el desarrollo de una herramienta de modelación de superficies de terrenos para un prototipo de aplicación SIG, a partir de la utilización de las funcionalidades que brinda Geotools, lo cual figura como **objetivo general** de la investigación.

El desarrollo de aplicaciones de SIG con librerías de código libre constituye el **objeto de estudio** del presente trabajo, específicamente la elaboración de herramientas de modelación de superficies de terrenos sobre estos sistemas, siendo éste el **campo de acción**.

Para satisfacer eficazmente el objetivo general del trabajo, se declaran como **objetivos específicos** los que a continuación se relacionan:

- Identificar las principales funcionalidades que ofrece Geotools para el desarrollo de aplicaciones SIG.
- Describir la estructura de Geotools y sus principales componentes para su posterior uso en el desarrollo de aplicaciones SIGs.

[1] A través de este sitio el proyecto FreeGIS busca promover el uso, desarrollo y soporte del software libre en el ámbito de los Sistemas de Información Geográfica.

- Utilizar adecuadamente las funcionalidades y componentes de Geotools en el diseño e implementación de un prototipo de aplicación SIG.
- Aplicar los conocimientos y habilidades adquiridas en el desarrollo de una herramienta de modelación para el prototipo de aplicación desarrollado.

Las **tareas de la investigación** planteadas para satisfacer los objetivos expuestos anteriormente son las siguientes:

- Realizar un estudio de la estructura y las funcionalidades que ofrece Geotools para el desarrollo de aplicaciones SIGs.
- Seleccionar y documentar las principales funcionalidades y componentes que ofrece Geotools para el desarrollo de aplicaciones SIGs.
- Diseñar e implementar un prototipo de aplicación SIG usando de Geotools.
- Desarrollar una herramienta de modelación de terrenos para el prototipo de aplicación implementado.

Para llevar a cabo las tareas anteriormente especificadas se realiza la investigación mediante los siguientes **métodos**:

- Analítico-Sintético: para analizar la estructura e implementación de Geotools, determinar sus partes más esenciales y las relaciones entre ellas, y finalmente descubrir los aspectos más significativos para la investigación.
- Histórico-Lógico: posibilita conocer los antecedentes y tendencias actuales en el desarrollo del software SIG.
- Sistémico: para facilitar la elaboración de los componentes a desarrollar y el establecimiento de las relaciones dinámicas y estructurales entre ellos.
- Experimentación: que permite probar las distintas versiones obtenidas en el trancurso de la implementación del prototipo y la herramienta.

Los **resultados esperados** son:

- Un prototipo de aplicación SIG basado en Geotools.
- Una herramienta de modelación de superficies en 3D, a partir de mallas rectangulares, para la aplicación SIG desarrollada.

El desarrollo exitoso de las tareas expuestas previamente justificará la **idea a defender** en la investigación: es posible desarrollar una herramienta de modelación de terrenos extendiendo un prototipo de un Sistema de Información Geográfica (SIG) implementado sobre Geotools.

El documento está estructurada en 5 capítulos. Comienza con el actual capítulo como introducción; el Capítulo 2, contiene la fundamentación teórica de la investigación, en la cual son expuestos los principales conceptos que esclarecen el objeto de estudio; en el Capítulo 3 se fundamenta la selección de las herramientas y procesos utilizados en el desarrollo; en el Capítulo 4 se presentan las características de la solución propuesta; y en el Capítulo 5 quedan reflejados los resultados de las pruebas realizadas al sistema.

CAPÍTULO 2

Preliminares

El capítulo PRELIMINARES constituye la base teórica de la investigación. Incluye los conceptos fundamentales necesarios para adentrarse en el dominio del problema. Primeramente en la sección 2.1 *Sistema de Información Geográfica* se profundiza un poco más en las características de estos sistemas, aunque sin salirse del contexto del objetivo general del trabajo, pues se abordan los datos que manipula un SIG y las operaciones que pueden realizarse sobre éstos; en la sección 2.2 *Modelación de superficies* se abordan las técnicas fundamentales que existen en la modelación de superficies de terrenos y se mencionan las principales APIs para programar aplicaciones en 3D; y por último se explica qué son las metodologías de desarrollo y se presentan algunos ejemplos de las mismas, en la sección 2.4 *Metodologías de desarrollo del software.*

2.1. Sistema de Información Geográfica

Como plantea la definición de SIG, dada en el capítulo anterior, los datos que éste manipula pueden ser de dos tipos: georeferenciados o no; a los primeros se hace referencia como espaciales y a los no referenciados: de atributo. Estos datos provienen de una fuente, usualmente un mapa; por tanto, un Sistema de Información Geográfica (SIG) es visto, en su esencia, como un grupo de capas de mapas, donde cada capa está relacionada con otra. Generalmente, cada capa contiene información de un tema único; estos temas pueden ser, por ejemplo, topografía, suelos, infraestructuras tales como caminos, redes, etc. En algunos casos el SIG está definido por el tipo de dato que el sistema contiene, por ejemplo, el término Land Information System (LIS) se aplica a los sistemas que tratan información acerca de los suelos.

Los datos espaciales se refieren a dónde ocurre el fenómeno; los datos de atributos dicen qué ocurrió, la naturaleza de los datos espaciales. Por ejemplo, se quiere describir la

localización del pozo del agua municipal como un punto coordenado: “36 grados 11 min 1 segundo de latitud norte, 95 grados 2 min 22 segundos longitud oeste”; además de eso, es posible reportar muchos atributos de ese pozo, como la profundidad, rendimiento, calidad del agua, proximidad a alguna bomba. Todos los SIG tienen la capacidad de almacenar y manipular ambos tipos de datos[9].

2.1.1. Formatos de archivos SIG

Un formato de archivo SIG es un estándar de codificación de información geográfica. Un SIG puede utilizar uno de los dos tipos básicos de modelos para almacenar información georeferenciada en formato digital: vectorial y raster; este último se conoce también como de mallas rectangulares regulares.

La Figura 2.1 muestra cómo el modelo raster está constituido por un conjunto de celdas o pixeles, de forma cuadrada y tamaño idéntico, localizados de forma contigua; la matriz que forman es bidimensional; cada celda es referenciada por índices de línea y columna y contiene un número que representa el valor del atributo mapeado; los valores de los pixeles en la matriz pueden estar dentro de un intervalo numérico dado, como por ejemplo, de 0 a 255 para imágenes de 8 bits, o bien pueden ser valores asignados según una categoría definida para la identificación del elemento o el resultado de modelamientos numéricos que representan el comportamiento de una determinada variable a una condición dada. La resolución de los datos raster depende del tamaño del pixel y puede variar desde valores submétricos hasta kilómetros [5].

El modelo raster se enfoca en las propiedades del espacio más que en la precisión de la localización. Cuanto mayor sea la resolución menor es la precisión o detalle en la representación del espacio geográfico.

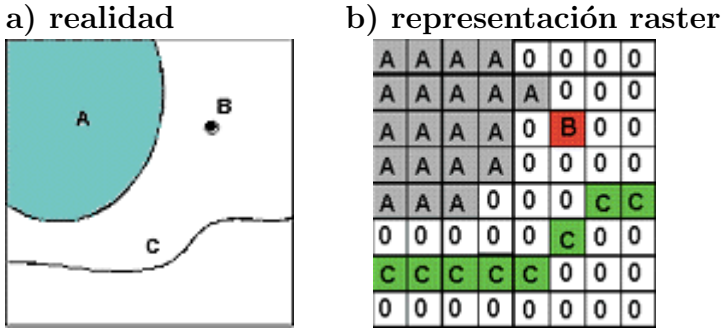


Figura 2.1: Modelo Raster

Los formatos de archivos SIG más populares[27] que implementan el modelo raster son:

- **ARC Digitized Raster Graphics (ADRG)**
- **Compressed ARC Digitised Raster Graphics (CADRG)**

- **Band Interleaved by Line (BIL)**
- **Controlled Image Base (CIB)**
- **Digital Raster Graphic (DRG)**
- **Enhanced Compressed Wavelet (ECW)**
- **ESRI GRID**
- **GeoTIFF**
- **ERDAS IMAGINE (IMG)**
- **Multi-Resolution Seamless Image Database (MrSID)**
- **JPEG2000**

A continuación se abordan brevemente los elementos esenciales de los formatos GeoTIFF y ESRI GRID, ya que a efectos de la investigación resultan suficientes para lograr los objetivos trazados en el trabajo.

Formato de archivo GeoTIFF

El formato GeoTIFF es una variante del formato TIFF, enriquecida con información relevante para los SIGs. GeoTIFF usa un grupo pequeño de etiquetas TIFF para almacenar información georeferenciada y una llamada "MetaEtiqueta" (GeoKey) para codificar docenas de elementos adicionales en sólo 6 etiquetas; estas últimas están diseñadas de una manera similar a las etiquetas TIFF. GeoTIFF utiliza códigos numéricos para describir los tipos de proyecciones, sistemas coordenados, elipsoide, etc. Los códigos de la proyección, el sistema coordenado y la elipsoide se derivan de la lista del European Petroleum Survey Group (EPSG)[22].

Formato de archivo ESRI GRID

ESRI GRID es un formato de archivo raster desarrollado por el Environmental Systems Research Institute (ESRI), que tiene dos variantes:

- **ARC/INFO GRID**: un formato binario propietario, conocido como también como ARC GRID.
- **ARC/INFO ASCII GRID**: formato ASCII no propietario.

  El ASCII GRID consiste en un encabezado que especifica los datos geográficos y la resolución, seguido por los valores de las celdas en la matriz. Estos archivos son de una sola banda [7].

  A continuación se muestra un ejemplo de un archivo de datos ARC/INFO ASCII GRID:

```
ncols         4 # columnas de la matriz
nrows         6 #  filas de la matriz
xllcorner     0.0 #  margen izquierdo de la matriz
yllcorner     0.0 #  margen superior de la matriz
cellsize      50.0 # longitud de cada celda
NODATA_value  -9999 #  representa la ausencia de datos

-9999 -9999 5 2
-9999 20 100 36
3 8 35 10
32 42 50 6
88 75 27 9
13 5 1 -9999
```

2.1.2. Operaciones sobre datos

Compilación

La compilación de datos involucra unir los datos espaciales y de atributos en un formato específico dentro del SIG. Una vez que el usuario establece los requerimientos comunes de los datos, se crea un mapa base que contiene los requisitos estándares para la información y asegura la compatibilidad. Con los datos ensamblados y el mapa base establecido, se lleva a cabo entonces la conversión o digitalización. La digitalización puede realizarse de varias formas, el escaneo es una de ellas. Este proceso de digitalización convierte los datos de mapas en puntos, líneas y celdas almacenables en una computadora.

Almacenamiento

Una vez que los datos han sido digitalizados son guardados. Los modelos de datos utilizados son genéricos, y los más comunes son el formato raster y el vectorial.

Manipulación

Con los datos almacenados es posible consultar y analizar la información a través de funciones que, generalmente, son las siguientes: recuperación de información, medición de áreas y perímetros, superposición y álgebra de mapas, y re-clasificar datos de mapas.

Salida

La tarea final de un SIG es generar una salida, generalmente un mapa. Los mapas generados pueden contener muchos esquemas de símbolos y colores; con una escala y tamaño correspondientes con la especificaciones del usuario. Sin embargo, no se debe ver el sistema sólo con la capacidad de generar mapas de gran calidad, también puede realizar análisis y manipulación de la información.

2.1.3. Categorías de herramientas SIGs

La diversidad actual de implementaciones de SIGs ha suscitado el surgimiento de diversas clasificaciones para los mismos. En [23] se plantea una clasificación que parece la más objetiva, y consiste en:

- **SIG Escritorio**: utilizado para crear, editar, analizar y mostrar datos geoespaciales.
- **Spacial Data Base Management System (SDBMS)**: almacena los datos, pero a veces provee funcionalidades de análisis y manipulación de los mismos. Ej: PostGIS.
- **Servidor WebMap**: para la distribución de los datos a través de internet. Ej: MapServer y GeoServer.
- **Librerías y extensiones**: proporcionan funcionalidades adicionales. Para aplicaciones web existen soluciones como MapFish, OpenLayers o Geomajas; y en entornos no web se pueden usar OpenMap, GDAL, Prij.4 y Geotools.

Geotools

La librería Geotools provee funcionalidades estándares para manipular datos geoespaciales. Está escrita en Java y por tanto se ejecuta dentro de la Java Virtual Machine (JVM). Tiene como base el proyecto GeoAPI, fundado en el 2002 por el mismo fundador de Geotools: James McGill. GeoAPI ha tenido una historia un poco complicada, pues en algún momento fue disuelto por la falta de actividad después de haberse unido a un proyecto similar del Open Geospatial Consortium (OGC); finalmente Geotools tomó el proyecto con el patrocinio del OGC y se creó el OGC GeoAPI Working Group[1], que busca formalizar GeoAPI como un estándar de OGC. El grupo de trabajo de Geotools es el principal contribuidor actualmente de GeoAPI, liberando versiones conjuntamente[2]:

- GeoAPI 2.1 (milestone) para Geotools 2.4
- GeoAPI 2.2 (milestone) para Geotools 2.5

GeoAPI provee un conjunto de interfaces para el desarrollo en Java de aplicaciones geospaciales. En su núcleo están definidas las interfaces necesarias para la manipulación de metadatos e imágenes georeferenciadas; la referenciación, proyección y conversión de información geospacial; la construcción y manejo estructuras de datos; adicionalmente, define interfaces para el almacenamiento y acceso a datos, incluyendo filtros para realizar

[1] El OGC GeoAPI 3.0 Working Group ya ha publicado su visión, que puede ser consultada en http://geoapi.sourceforge.net/charter.html. Este grupo busca producir la versión inicial del estándar 3.0 con sólo algunos paquetes probados y muy estables. Para más información visite http://geoapi.sourceforge.net/.

[2] Estas versiones no son oficiales, pues no parten del grupo de trabajo del OGC. La última versión oficial es la GeoAPI 2.0.

consultas a los mismos y mostrarlos. GeoAPI sigue las especificaciones publicadas por la International Organization for Standardization (ISO) y el OGC, en la serie ISO 19100[3] de la primera y en el proyecto OpenGIS[4] del segundo.

Las principales interfaces definidas son:

Feature y SimpleFeature: Feature es la clase representar cualquier elemento que pueda ser mostrado en un mapa.Es el núcleo del modelo llamado Feature Model que GeoAPI divide en: Data Model, Query Model y Metadata, los cuales a su vez definen las especificaciones:

- ISO 19017: Geographic information – Spatial schema.
- OGC OGC General Feature Model.

Filter: utilizada en la realización de consultas a un DataStore o Catalog.

Geometry: GeoAPI define un conjunto de interfaces correspondientes a la ISO 19107 Geometry Specification. La desventaja de estas interfaces es que no son tan sencillas como la usada por Java Topology Suite (JTS) y las implementaciones no están bien probadas:

- JTS-Wrapper: se usa de igual manera que JTS, incluso con las limitaciones de ésta.
- Geometry: está en desarrollo aún.

ProgressListener: útil en el manejo del progreso de operaciones largas.

Licencia de software

Geotools está disponible bajo licencias libres. A continuación se muestran las licencias que incluye en su uso y comercialización:

- GNU Lesser General Public License (LGPL)
- SOSNOKILLLICENSE
- Apache License v2.0

Para una revisión completa de los términos de cada una, lea el apéndice A **Licencias**.

[3]ISO 19100 es una serie de estándares para la definición, descripción y administración de información geográfica. Para más información consulte el sitio http://www.isotc.iso.org.

[4]OpenGIS agrupa todos los documentos y especificaciones generadas por el OGC.

2.2. Modelación de superficies

En la actualidad, la tendencia en la modelación de superficies es la representación de éstas mediante mallas poligonales, que por lo general son triángulos, pues tres puntos determinan sólo un plano, y así es más eficiente la representación. Por esta razón, se han desarrollado técnicas cada vez más efectivas que hacen de esta abstracción la más popular en el campo de la modelación de superficies. Dichas técnicas quedan fuera del contexto de la investigación, ya que los datos que sirven de entrada a la herramienta de modelación son modelados a través de una matriz, sin necesidad de utilizar las complejas técnicas que a continuación se explican brevemente; sin embargo, se considera necesario mencionarlas pues podría servir a futuras investigaciones derivadas de ésta.

2.2.1. Técnicas de modelación de superficies

Para la representación de modelos de terrenos existen dos formas fundamentales: a través de datos distribuidos irregularmente en el plano o TINs, o formando mallas rectangulares regulares (QuadTrees).

Redes Irregulares Trianguladas

Las TINs creadas a partir del criterio de triangulación de Delaunay, Figura 2.2, son ampliamente utilizadas en la modelación, según [4, 19]. No obstante las TIN tienen como principal desventaja la difícil obtención del modelo con distintos niveles de detalle.

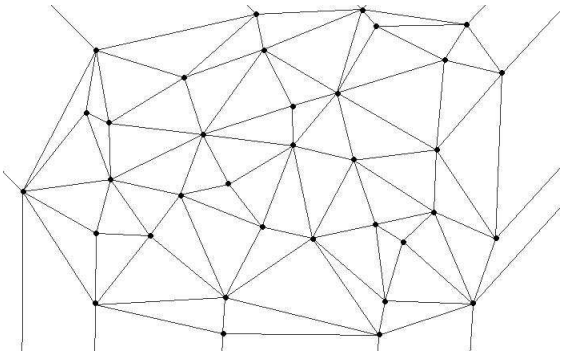


Figura 2.2: Triangulación de Delaunay.

QuadTrees

Los modelos basados en QuadTree son las más eficientes por su alta regularidad. Este tipo de estructura se basa en la división recursiva de la superficie en cuatro regiones, que pueden ser cuadradas o rectangulares, hasta llegar a una dimensión dada de las celdas. En el caso de los QuadTrees restringidos la dimensión de las celdas no debe ser menor que

la unidad. Una vez terminada la fase de división se procede a triangular cada celda de la siguiente forma: cada cuadrante se divide en ocho triángulos, dos triángulos por cada arista, a menos que la arista coincida con la de un vecino cuyo nivel de división menor que el del cuadrante en cuestión.

Entre los principales exponentes en lo que se refiere a triangulaciones basadas en Quadtrees restringidos se encuentran Lindstrom [17, 18, 16] y Pajarola [20, 21]. Ambos autores enfocan su atención en la representación eficiente de modelos construidos sobre conjuntos de datos arbitrariamente extensos que no pueden ser almacenados completamente en memoria, así como en la extracción eficiente de mallas triangulares con múltiples niveles de detalle y de manera progresiva. La principal contribución de Lindstrom radica en la introducción del concepto de dependencias entre vértices, como base para la extracción de las triangulaciones multiresolución sin la presencia de grietas en la superficie.

El hecho de que el proceso de triangulación se lleve a cabo teniendo en cuenta las dimensiones de las regiones adyacentes, trae consigo que no se produzcan anomalías como, por ejemplo, grietas entre los cuadrantes.

Aunque los QuadTrees restringidos han demostrado ser altamente eficientes en la visualización de superficies de terrenos, presentan una dificultad importante. La selección en la estructura de un cuadrante para ser triangulado provoca un efecto cascada en la selección de sus vecinos, dada la restricción impuesta en los niveles de división. Como consecuencia de esto pueden resultar seleccionados cuadrantes adicionales, traduciéndose en un mayor número de triángulos a visualizar, y por consiguiente en una disminución de la velocidad de renderizado de la estructura.

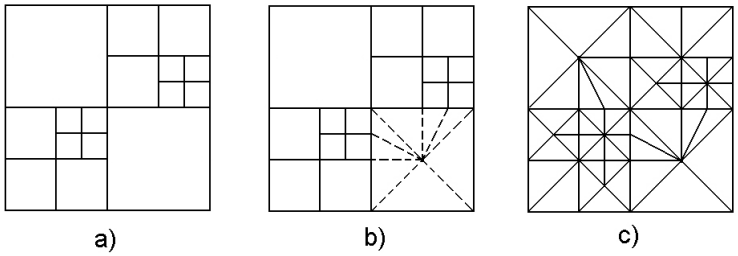


Figura 2.3: Triangulación sobre un QuadTree no restringido.

Una alternativa a los QuadTrees restringidos son las denominadas triangulaciones basadas en Quadtrees no restringidos [10, 2]. El proceso de división de cada cuadrante se realiza independiente de sus vecinos, eliminando de esta forma la restricción de balance sobre la estructura, como muestra la Figura 2.3.

2.3. APIs 3D

Los gráficos 3D se han convertido en algo muy popular, particularmente en juegos de computadora, al punto que se han creado Application Program Interfaces (APIs) especializadas para facilitar los procesos en todas las etapas de la generación de gráficos.

Estas APIs han demostrado ser vitales para los desarrolladores de hardware para gráficos por computadora, ya que proveen un camino al programador para acceder al hardware de manera abstracta.

Existen varias APIs que permiten escribir aplicaciones que produzcan gráficos 2D y 3D, entre las que se destacan OpenGL y Direct3D. OpenGL se utiliza ampliamente en realidad virtual, visualización de información y simulación; también se emplea en el desarrollo de videojuegos, donde compite con Direct3D en plataformas Microsoft Windows.

2.3.1. Open Graphics Library (OpenGL)

Open Graphics Library (OpenGL) es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D. La interfaz consiste en más de 250 funciones diferentes que pueden usarse para dibujar escenas tridimensionales complejas a partir de primitivas geométricas simples, tales como puntos, líneas y triángulos. Fue desarrollada originalmente por Silicon Graphics Inc. (SGI) en 1992.

Entre las características más atrayentes de OpenGL se pueden citar:

- **Confiable y portable**: Todas las aplicaciones desarrolladas en OpenGL, producen resultados de visualización consistentes en cualquier plataforma que soporte su API.
- **Evolución**: Debido a su minucioso diseño, la API de OpenGL permite asimilar rápidamente las innovaciones que se producen en el hardware a través de su mecanismo de extensión. Esto permite a los desarrolladores de aplicaciones y proveedores de hardware incorporar las nuevas características a sus productos.
- **Escalable**: Las aplicaciones que utilizan el API de OpenGL, se pueden ejecutar en una amplia gama de sistemas, que va desde las PC y estaciones de trabajo, hasta supercomputadoras.
- **Fácil de usar**: Posee una gran cantidad de primitivas muy eficientes, que encapsulan información sobre el hardware subyacente, liberando al desarrollador de tener que diseñar teniendo en cuenta la plataforma.
- **Amplia documentación**: El hecho de que numerosos libros que han sido publicados sobre OpenGL sean accesibles libremente, además de una gran cantidad de código de ejemplos, hace que la información sobre la misma sea fácil de obtener.

2.3.2. Direct3D

Direct3D es parte de DirectX, una API propiedad de Microsoft disponible tanto en los sistemas Windows de 32 y 64 bits, como para sus consolas Xbox y Xbox 360 para la programación de gráficos 3D.

El objetivo de esta API es facilitar el manejo y trazado de entidades gráficas elementales, como líneas, polígonos y texturas, en cualquier aplicación que despliegue gráficos en 3D, así como efectuar de forma transparente transformaciones geométricas sobre dichas entidades. Direct3D provee también una interfaz transparente con el hardware de aceleración gráfica.

Se usa principalmente en aplicaciones donde el rendimiento es fundamental, como los videojuegos, aprovechando el hardware de aceleración gráfica disponible en la tarjeta gráfica.

2.4. Metodologías de desarrollo de software

Las metodologías de desarrollo de software son un conjunto de métodos, técnicas y ayudas a la documentación para el desarrollo de productos de software. Define cada una de las actividades que se deben realizar para lograr el producto deseado, especificando los roles de las personas involucradas en los procesos y detallando además la información que se debe obtener de estas actividades y la necesaria para comenzarlas.

El desarrollo de software ocurre en un entorno en el que cada uno de los factores influye directamente en las personas que trabajan en él, y por tanto en el resultado de su trabajo; por ejemplo, las variaciones en el Sistema Operativo, el lenguaje de programación o en los estándares de código afectan cualitativa y cuantitavamente la producción. Es por ello que no sólo de debe mejorar la productividad, también hay que implementar los procesos para aumentar la calidad del resultado en un entorno donde la única constante es el cambio.

2.4.1. Desarrollo ágil de software

Se entiende como *desarrollo ágil* uno de los paradigmas de desarrollo de software que se basa en procesos ágiles, es decir, procesos que se enfocan sólo en las personas y sus resultados. El desarrollo ágil "es un marco de trabajo conceptual que promueve iteraciones en el desarrollo a lo largo de todo el ciclo de vida del software"[24], que evoluciona a partir de los años 1990 como contraparte de otras técnicas de desarrollo más burocráticas y estrictas y que a diferencia de éstas hace más énfasis en la adaptabilidad que en la previsibilidad. Existen actualmente muchas metodologías de desarrollo que se basan en proceso ágiles, siendo XP (Extreme Programming) y Scrum dos de las más utilizadas.

XP

XP o Programación Extrema es el ejemplo más destacado de metodología ágil, y se caracteriza por:

- Desarrollo iterativo e incremental: mejoras sucesivas, aunque pequeñas.
- Pruebas unitarias continuas: generalmente repetidas y automatizadas, incluso se recomienda escribirlas antes de codificar; las pruebas de regresión forman parte de estas pruebas.

- Programación en parejas: se supone que de esta forma el código es discutido mientras se escribe.
- Integración del equipo de desarrollo con el cliente: se recomienda que haya alguna representación del cliente en el equipo de desarrollo.
- Refactorización del código: para mejorar ciertas partes del código, pero sin modificar las funcionalidades.
- Propiedad del código compartida: con el objetivo de que todo el equipo de trabajo se sienta responsable por cualquier parte del proyecto.
- Simplicidad en el código: para que primero funcione y luego se añade la funcionalidad si es necesario; es mejor tener algo simple que funcione que algo complejo que nunca lo haga.

Scrum

Scrum es otra metodología de desarrollo ágil utilizado comúnmente. Se enfoca en la gestión de proyectos aunque puede ser utilizado en equipos de mantenimiento de software. Define un grupo de prácticas y roles que pueden tomarse como punto de partida para el proceso de desarrollo. Los roles principales son el ScrumMaster, que hace las funciones de director del proyecto, el ProductOwner, que representa a los interesados, y el Team que incluye a los desarrolladores. Durante las iteraciones o sprints de desarrollo, de un período de entre 15 y 30 días, se crea un entregable que posee un grupo de características que se definen en una reunión previa. Las mayores ventajas de Scrum es que es muy fácil de aprender y no requiere grandes esfuerzos para empezar a utilizarse.

2.5. Conclusiones del capítulo

La información reflejada a lo largo del capítulo permitió adquirir los conceptos fundamentales necesarios para adentrarse en el dominio del problema. El primer aspecto a tener en cuenta son las ideas relacionadas con los SIG, definición de los mismos, clasificaciones, y operaciones que realizan; además, los datos que manejan pueden seguir dos modelos de almacenamiento, en dependencia de las requerimientos.

Otro aspecto es el relacionado con la modelación de terrenos. Existen diversas técnicas y APIs que mejoran el rendimiento de las aplicaciones que simulan la realidad mediante gráficos en 3D. Dentro del conjunto de técnicas están las que utilizan QuadTree o TINs para realizar una división de la superficie a modelar, cada una con sus ventajas que se aprovechan en dependencia del contexto. Direct3D y OpenGL son las dos APIs fundamentales utilizadas para generar gráficos en computadora; la primera es parte de una conjunto de APIs de Microsoft para la programación en 3D, y por tanto, posee una licencia de software privativa; sin embargo, OpenGL es capaz de ofrecer las mismas prestaciones y además es libre.

El tercer aspecto tiene que ver con la utilización de las metodologías de desarrollo de software. Muchos desarrolladores se resisten a la idea de tener que realizar Ingeniería además de programar sus aplicaciones; sin embargo, la experiencia ha demostrado que estos procesos ingenieriles, lejos de atrasar el progreso, contribuyen con una calidad superior del producto resultante. En entornos de desarrollo libres la tendencia es la selección de metodologías ágiles, las cuales ahorran gran parte de los procesos -innecesarios en ocasiones- que promueven otras metodologías más convencionales; XP y Scrum son ejemplos de metodologías ágiles que se usan en la actualidad.

El potencial que existe en el campo de los Sistemas de Información Geográfica es inmenso, pues existen muchas varientes de desarrollo para lograr un producto que satisfaga las necesidades del cliente al ofrecer una solución con la calidad esperada.

CAPÍTULO 3

Descripción y Análisis de la solución propuesta

En el presente capítulo se realiza una descripción de las tecnologías a utilizar, así como del lenguaje y herramientas considerados esenciales en la solución del problema planteado. Inicialmente, se determina la metolodogía que guía el desarrollo del prototipo SIG y la herramienta de modelación; luego se abordan el lenguaje y el Integrated Development Environment (IDE) que sirven de soporte a la implementación del software; y finalmente se brinda una descripción de las APIs que brindan las funcionalidades acordes con las necesidades de la investigación.

3.1. Metodología de desarrollo

3.1.1. SXP (Scrum+XP)

Teniendo en cuenta las características de las metodologías expuestas anteriormente, y considerando que el equipo de desarrollo del software está integrado por sólo una persona, se decide adoptar el uso de metodologías de desarrollo ágiles, específicamente Scrum para la planificación del proceso y XP para el desarrollo. Sin embargo, como alternativa más viable existe una variante llamada SXP, propuesta en el 2008 por la ingeniera Gladys Marsi Peñalver Romero, y probada en los proyectos que trabajan con Software Libre en la Facultad 10 de la UCI, y que hace uso de las ventajas de gestión de Scrum y las posibilidades en el desarrollo que brinda XP; además posee la ventaja de tener en cuenta el contexto de desarrollo de software libre en la universidad, por lo que ahorra un conjunto de pasos prescindibles de otras metodologías convencionales.

Según el guión de la metodología, SXP se organiza en cuatro fases[1]:

[1]Las dos últimas fases carecen de interés en el documento, por lo que no se reflejarán sus artefactos.

- Planificación: se generan los documentos que se encuentran relacionados con la concepción inicial del sistema, así como la definición del mismo.
  Artefactos: Concepción inicial del sistema, Lista de reserva del proyecto, Historias de usuario, Lista de riesgos y Modelo de diseño.

- Desarrollo: en la primera parte de esta fase se generan todos los documentos relacionados con la planificación de las iteraciones, y además se recogen las principales definiciones que se manejan en la metodología y otras términos de difícil entendimiento para los clientes, así como de las tareas a realizar durante la implementación. Además se genera el código fuente en la etapa de implementación y los documentos relacionados con las pruebas, como última parte de esta etapa.
  Artefactos: Tareas de Ingeniería, Cronograma de producción, Plan de versiones, etc.

- Entrega: se realiza la entrega del software y su documentación, generándose aquellos documentos que son imprescindibles para el entrenamiento y entendimiento del producto.

- Mantenimiento: se realizan las actividades relacionadas con el soporte del software y se generan los documentos relacionados con los cambios que puedan ocurrir en el mismo.

3.2. Herramientas utilizadas

3.2.1. Java

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90 del siglo pasado. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. Las características principales que nos ofrece Java respecto a cualquier otro lenguaje de programación, son:

- Simple
- Orientado a objetos
- Robusto
- Seguro
- Portable
- Dinámico

3.2.2. NetBeans

NetBeans se refiere a una plataforma para el desarrollo de aplicaciones de escritorio usando Java y a un IDE (Figura 3.1) desarrollado usando la Plataforma NetBeans. La plataforma NetBeans 6.5 permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

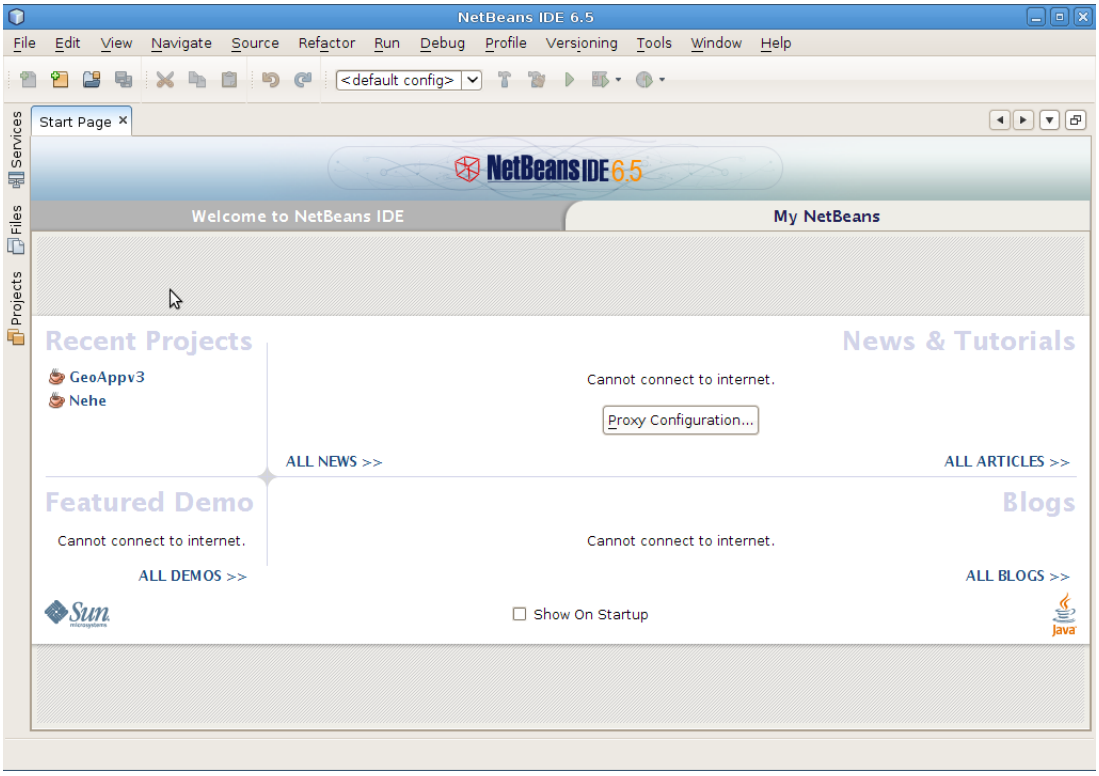


Figura 3.1: IDE NetBeans 6.5

3.2.3. Mercurial

Mercurial es una herramienta de control distribuido de versiones multiplataforma, para desarrolladores de software. Está implementado principalmente haciendo uso del lenguaje de programación Python, pero incluye una implementación binaria de diff escrita en C. Mercurial fue escrito originalmente para funcionar sobre Linux. Ha sido adaptado para Windows, Mac OS X y la mayoría de otros sistemas tipo Unix. Mercurial es, sobre todo,

un programa para la línea de mandatos. Todas las operaciones de Mercurial se invocan como opciones dadas a su programa motor hg, que hace referencia al símbolo químico del mercurio.

Las principales metas de desarrollo de Mercurial incluyen un gran rendimiento y escalabilidad; desarrollo completamente distribuido, sin necesidad de un servidor; gestión robusta de archivos tanto de texto como binarios; y capacidades avanzadas de ramificación e integración, todo ello manteniendo la sencillez conceptual.

La Figura 3.2 muestra el plugin de la plataforma NetBeans 6.5 que integra las funcionalidades de Mercurial.

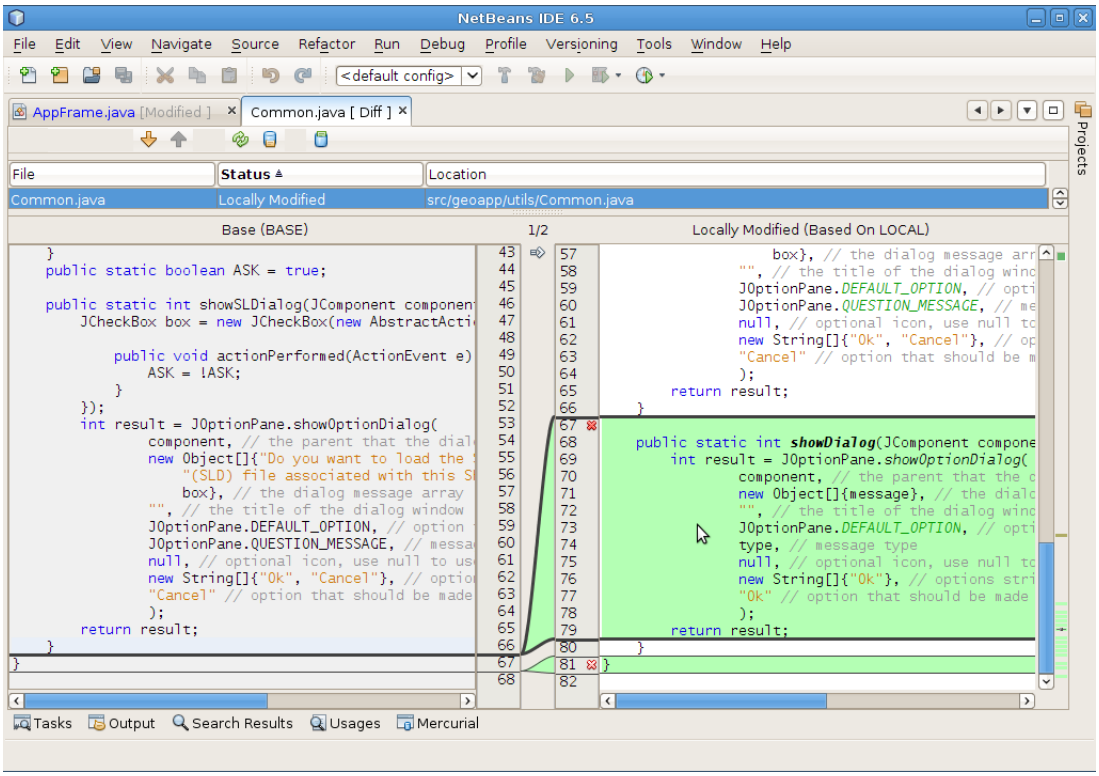


Figura 3.2: Plugin del IDE NetBeans 6.5 que integra Mercurial

3.3. APIs

3.3.1. Geotools

Geotools es la librería seleccionada previa a la investigación sobre la que se llevará a cabo el desarrollo del prototipo SIG. A continuación se realiza una descripción de los elementos más relevantes de su estructura.

Modelo de componentes

La Figura 3.3 muestra la estructura de Geotools desde la perpectiva de los componentes que la forman. La separación en componentes hace que el uso de la misma sea más sencillo, además permite la extensión y/o reemplazo de la implementación de cualquiera de ellos sin afectar los demás.

La estructura está compuesta por un módulo principal y los plugins adjuntos. El módulo principal contiene las librerías núcleos, y los plugins proveen acceso a diferentes versiones de formatos de la base de datos de la EPSG. De igual forma, tiene plugins que brindan acceso a formatos comunes de datos geoespaciales, tales como, imágenes y formatos vectoriales, que estén ya sea en el disco duro local, bases de datos o en Internet.

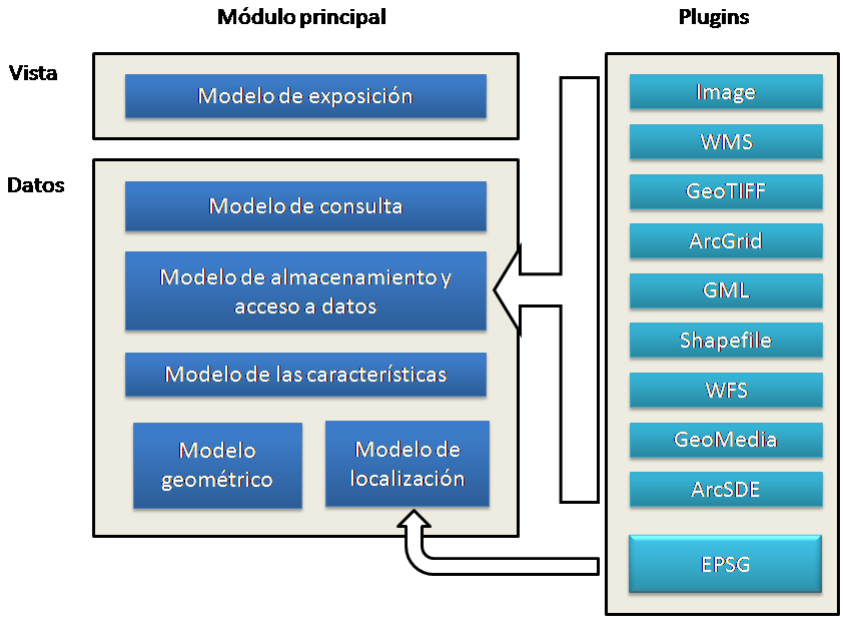


Figura 3.3: Componentes de la arquitectura de Geotools

El módulo principal está dividido a su vez en dos partes: el modelo de datos y la vista de datos. Estos componentes son dos de los tres del patrón de arquitectura Modelo-Vista-Controlador (MVC). Este patrón permite que separar el modelo de datos de la lógica usada para crear y modificar estos datos. El componente controlador se deja explícitamente a ser implementado por los usuarios de la librería.

Modelo de datos

Consiste en el modelo geométrico, el modelo de geoposicionamiento o localización y el modelo de las características o features, los cuales en conjunto proveen el núcleo del

modelo de datos para las entidades geoespaciales. Además, está incluido un modelo de acceso y almacenamiento de datos, con plugin para los formatos de datos más comunes y un modelo de consulta, que brinda facilidades para el obtención y filtrado de datos.

El modelo de geoposicionamiento implementa las vías para llevar a cabo operaciones comunes sobre datos geo-referenciados, como transformaciones y re-proyecciones, contenida en la clase CoordinateReferenceSystem. Este modelo ofrece formas para definir sistemas coordenados de referencia y el plugin EPSG define una fuente estándar para los comúnmente definidos, como muestra la Figura 3.3. La fuente principal de datos es el European Petroleum Survey Group (EPSG), a través de la clase CRS.

El modelo geométrico provee la infraestructura para el uso de figuras geométricas vectoriales. Incluye operadores para realizar operaciones estándares de geometría computacional. En versiones superiores de Geotools 2.2, la librería depende de JTS, y está por tanto, limitada a dos dimensiones.

En el modelo de las características o features se manejan las features (Figura 3.4), las cuales constituyen la unidad principal de los datos en Geotools y están definidas en GeoAPI. Conceptualmente, las features contienen un identificador, un conjunto de atributos en un arreglo de Java y un esquema definiendo los atributos y la disposición de éstos dentro del arreglo. Los atributos incluyen la definición geoespacial de la característica, cualquier otra entidad u atributo. El método estándar de obtener el contenido de una característica es a través de combinaciones de filtros y expresiones.

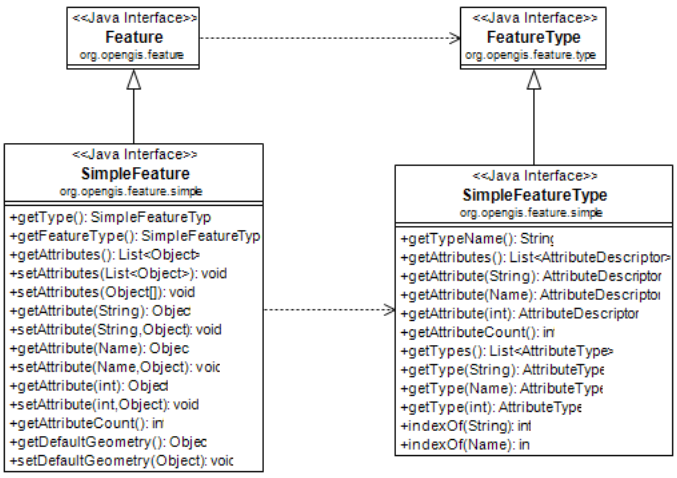


Figura 3.4: Modelo de Feature de Geotools.

El modelo de acceso y almacenamiento incluye métodos para crear, manipular y almacenar datos, implementando las interfaces propuestas por GeoAPI, que siguen la norma ISO 19115 de Metadatos; facilita la interacción de datos en formatos raster o vectoriales, tanto locales, en base de datos o Internet, siendo las interfaces DataAccess y GridCoverage[2] la más importantes para cada tipo de formato, respectivamente. Este mode-

[2]Básicamente, se entiende por GridCoverage a una matriz de valores numéricos con información de

lo brinda varios plugins para acceder a formatos de datos comunes, como por ejemplo: para los archivos GeoTIFF están disponibles las clases GeoTiffReader, GeoTiffWriter, GeoTiffFormat, entre otras; para los archivos ASCII GRID, las clases AsciiGridRaster, AsciiGridsImageReader, AsciiGridsImageWriter, etc.

En el modelo de consulta están los métodos estándares para buscar y obtener el contenido de las estructuras de datos a través de filtros que sólo extraen la información relevante. Geotools hace uso de la interfaz Filter de GeoAPI para implementar las consultas de features a un DataStore, que usualmente se realiza utilizando la interfaz Query.

Vista de datos

EL modelo de exposición es el único en esta parte del patrón. En dicho modelo, el uso de estilos es uno de los atractivos. La renderización en Geotools sigue la especificación Style Layer Descriptor (SLD) del OGC, esquema XML que plantea un lenguaje estándar para describir el conjunto de capas que dan apariencia a un mapa. El SLD ayuda a definir el estilo visual de cada capa de objetos geográficos que componen el mapa, permitiendo, por ejemplo, representar el color de relleno, tipo y ancho de borde, etc.

Algunas de las principales interfaces definidas para la renderización son:

GTRenderer: implementada por las clases ShapefileRenderer y StreamingRenderer, esta última optimizada para no reservar nada y funcionar con grandes cantidades de datos sin desbordar la memoria.

Style, StyledLayer, StyledLayerDescriptor y FeatureStyleImpl: controlan el proceso de renderización, dándole un estilo.

Symbolizer: describe cómo representar una característica o feature en un mapa, basada en su contenido, es decir, la geometría y sus atributos. Existen varios interfaces para los distintos tipos de Symbolizer, por ejemplo, Text Symbolizer, Line Symbolizer, Polygon Symbolizer, Point Symbolizer y Raster Symbolizer.

Clase SLD: se utiliza en la realización de operaciones comunes sobre los objetos que representan estilos (Style).

Extensiones

Las extensiones que a continuación se mencionan no forman parte del diseño de Geotools, pero son muy útiles:

JMapPane: es un componente Swing que puede mostrar un mapa y proveer una interacción limitada con el mismo. Puede funcionar con cualquier tipo de renderización.

Graph: permite construir grafos de una manera flexible a partir de las features.

La Figura 3.5 muestra un diagrama que contiene los paquetes perteneciente a Geotools fundamentales necesarios para desarrollar la aplicación, así como las interfaces o clases que contiene cada uno.

cada valor y su posición geospacial.

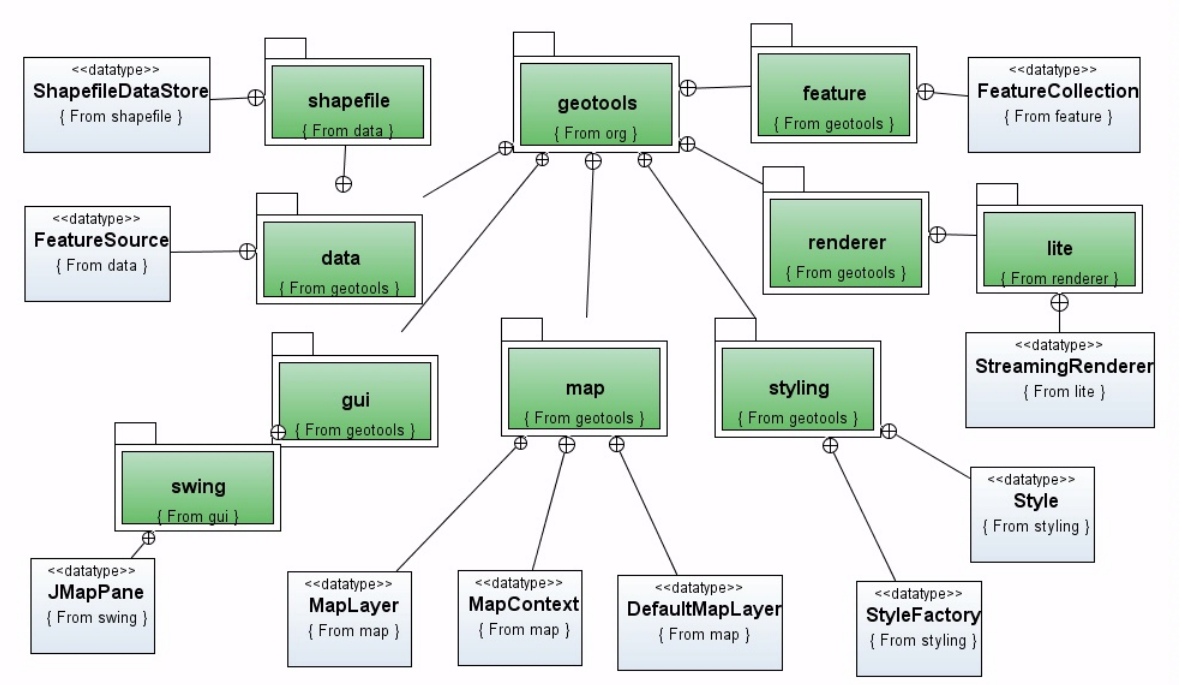


Figura 3.5: Paquetes fundamentales de Geotools.

3.3.2. Java Opengl (JOGL)

En la implementación de la herramienta de modelación de superficies de terrenos en 3D se determina usar Java OpenGL (JOGL), pues permite acceder a la mayoría de las características de OpenGL disponibles para los programadores de C, con la excepción de las llamadas a ventanas realizadas en GLUT (ya que Java contiene sus propios sistemas de ventanas, AWT y Swing), y algunas extensiones de OpenGL; además es la implementación de referencia de Java para OpenGL según la Java Specification Requests (JSR) 231[3].

JOGL se diferencia de otras bibliotecas Java para OpenGL en que simplemente expone las funciones de la OpenGL, basadas en C, por medio de métodos contenidos en unas pocas clases (Figura 3.6), en lugar de intentar realizar un mapeo completo del código OpenGL para transformarlo y adaptarlo al paradigma de orientación a objetos. De hecho, la mayoría del código de JOGL está en realidad autogenerado a partir de las cabeceras de las bibliotecas C de OpenGL, mediante una herramienta llamada Gluegen, que fue programada específicamente para dicho propósito. Esta decisión en el diseño tiene sus ventajas y sus desventajas. La naturaleza procedural y de máquina de estados de OpenGL es inconsistente con la forma habitual de programar en Java, lo cual puede resultar más complicado; sin embargo, la fina capa de abstracción proporcionada por JOGL hace que la ejecución sea muy eficiente[30].

[3]Java Bindigs for OpenGL. Consulte el sitio http://jcp.org/aboutJava/communityprocess/final/jsr231/index.html para más información sobre esta especificación.

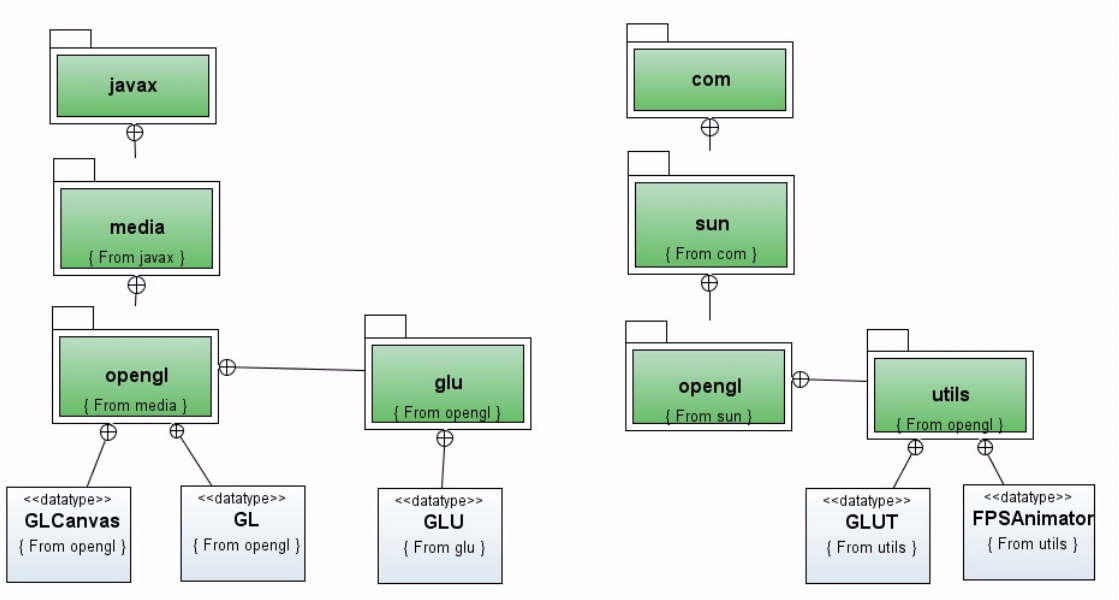


Figura 3.6: Clases fundamentales de JOGL.

3.4. Conclusiones del capítulo

En el capítulo se muestra la minuciosa selección de las herramientas y tecnologías que se utilizan en el desarrollo del software. El criterio de selección se basa en la característica de ser libre o de código abierto, aspecto que forma parte del objeto de estudio de la investigación.

SXP como metodología de desarrollo favorece que los procesos de gestión, planificación, implementación y prueba fluyan de una manera más natural, contribuyendo de esta forma a la calidad del resultado.

La selección de Netbeans como plataforma de desarrollo permite que el prototipo de aplicación SIG y la herramienta de modelación de superficies sean desarrolladas a partir de módulos y que la aplicación pueda ser extendida agregándole nuevos prestaciones por otros desarrolladores; además, Netbeans provee incontables funcionalidades que facilitan la programación de aplicaciones en Java.

Optar por JOGL es, sin duda, una decisión cuestionable, por la fina capa de abstración respecto a OpenGL que posee confiriéndole una naturaleza un tanto procedural y que entra en contradicción con la forma intuitiva de programar Java; sin embargo, se consideran razones de peso que sea eficiente, fácil de usar, portable y que evolucione de una forma transparente con OpenGL.

La estructura basada en componentes y diseñada mediante el patrón MVC de la librería Geotools propicia la independencia de funcionamiento entre las distintas capas lógicas y el mejoramiento del código; pero no sólo su estructura modular constituye una gran ventaja, también lo es su filosofía de implementación de los estándares que definen y describen los

principales conceptos en el ámbito de los SIGs; otro aspecto a destacar es el número significativo de extensiones a los diversos formatos de archivos que procesan los SIGs, que incluyen soporte para archivos raster y vectoriales. Por estas razones, Geotools es una librería que estimula el desarrollo de aplicaciones basadas en estándares de información geográfica, concepto clave en la interoperabilidad de los sistemas en general.

CAPÍTULO 4

Características del sistema

En el presente capítulo se describen las características del prototipo de aplicación SIG, teniendo en cuenta los pasos de la metodología SXP. Se incluye además la propuesta del sistema basado en los requerimientos funcionales y no funcionales que debe satisfacer la aplicación, la definición de roles y las plantillas de las historias de usuarios o casos de uso.

4.1. Concepción inicial del sistema

A continuación en la sección se describe la primera etapa en el desarrollo del prototipo, según la metodología SXP, en la cual se define el objetivo a alcanzar y los requerimientos de la aplicación.

4.1.1. Objetivo a alcanzar

Para ilustrar el proceso de desarrollo de un prototipo de SIG a partir de Geotools se propone la implementación de una aplicación de la cual se derivarán los siguientes beneficios:

- Contar con un prototipo de SIG, que no es más que un facsímil de lo real, pero no tan funcional como para que equivalga a un SIG completo, ya que no lleva a cabo la totalidad de las funciones necesarias del sistema.
- Herramienta de visualización en 3D de archivos raster que sirvan de entrada al prototipo.

4.1.2. Propuesta del sistema

Se desea llevar a cabo el desarrollo de un prototipo de aplicación SIG que cuente con una herramienta de modelación de superficies de terrenos en 3D. Para ello, se implementará una aplicación de escritorio, lo que posibilita que los usuarios usen el prototipo sin depender del uso de un servidor de mapas. Una vez terminado el desarrollo del prototipo y de su herramienta de modelación estos deben permitir la interacción con archivos con los formatos Shapefile y GeoTIFF y la modelación interactiva de archivos ASCII GRID en 3D.

Requerimientos funcionales

Teniendo en cuenta que los requisitos funcionales son capacidades o condiciones que el sistema tiene que satisfacer e indican su comportamiento, se debe analizar cuáles serán las funcionalidades del prototipo que cumplan con los objetivos que se plantearon, enumerando para ello las acciones que la aplicación debe ser capaz de realizar. Dichas acciones se enumeran a continuación:

- Leer y mostrar imágenes desde archivos Shapefile y GeoTIFF: el sistema carga cualquiera de los archivos especificados, y para el caso del formato Shapefile permite cargar el archivo SLD correspondiente.
- Salvar archivos con capas vectoriales: el sistema permite salvar archivos con capas vectoriales creadas por la interacción del usuario y la aplicación.
- Adicionar capas vectoriales en formato Shapefile: el sistema permite adicionar capas vectoriales desde archivos de formato Shapefile.
- Visualizar en 3D archivos de formato ASCII GRID del ESRI: el sistema muestra los datos cargados de los archivos en un sistema coordenado de tres dimensiones.
- Calcular distancia entre dos puntos: el sistema permite calcular la distancia correspondiente al sistema coordeando de referencia de la imagen entre dos puntos.
- Alejar, acercar y reestablecer la imagen: el sistema permite acercar, alejar y reestablecer al tamaño original la imagen mostrada.

Requerimientos no funcionales

Los requerimientos no funcionales detallan las propiedades o cualidades que el producto debe tener, agregando prestaciones al sistema, haciéndolo atractivo, fácil de usar, rápido y confiable.

Apariencia o interfaz externa

- La interfaz externa del sistema será amigable, sencilla y fácil de usar por los usuarios finales, facilitando el control de las operaciones sin necesidad de mucho entrenamiento para utilizar la aplicación.
- La aplicación estará estructurada de forma clara y comprensible, al mismo tiempo permitirá la interpretación correcta e inequívoca de la información.
- El diseño responderá a la ejecución de acciones de una manera rápida, minimizando los pasos a dar en cada proceso.
- Todos los textos y mensajes de la aplicación aparecerán en idioma inglés.

Usabilidad

- Su funcionamiento será intuitivo y requerirá de información mínima para su uso.
- Sistema flexible y de fácil uso para todo tipo de usuario con conocimientos mínimos.

Rendimiento

- Sistema poco cargado garantizando una rápida respuesta por parte del mismo al igual que una rápida velocidad de procesamiento de la información.

Portabilidad

- El sistema podrá será independiente y multiplataforma permitiendo que pueda ser ejecutado sobre cualquier sistema operativo.

Seguridad

- Al contarse con un único usuario cualquier persona podrá acceder al sistema y realizar las operaciones que estime conveniente.

Software

- El sistema se implementará con tecnología Java.

Legales

- Las herramientas escogidas para el desarrollo de la aplicación están respaldadas por licencias libres bajo las condiciones del software libre.

Confiabilidad

- Mecanismo de respuesta rápida ante fallos minimizando las pérdidas de información.

4.2. Desarrollo del sistema

En la segunda fase o etapa de la SXP se lleva a cabo el desarrollo del sistema. A continuación se ilustra los pasos fundamentales que se realizaron en dicha fase para alcanzar los resultados esperados en la investigación.

4.2.1. Roles

El desarrollo de software con SXP exige de la creación de pequeños grupos de trabajo, donde los roles son pocos, pero están bien definidas sus actividades. A continuación se muestran los roles que se establecieron:

Rol	Nombre y Apellidos
Gerente	Felix Alejandro Prieto Carratalá
Cliente	Lic. Yusnier Valle Martínez Felix Alejandro Prieto Carratalá
Programador	Felix Alejandro Prieto Carratalá
Analista	Felix Alejandro Prieto Carratalá
Diseñador de interfaz	Felix Alejandro Prieto Carratalá
Tester	Felix Alejandro Prieto Carratalá
Arquitecto	Felix Alejandro Prieto Carratalá
Consultor	Lic. Yusnier Valle Martínez

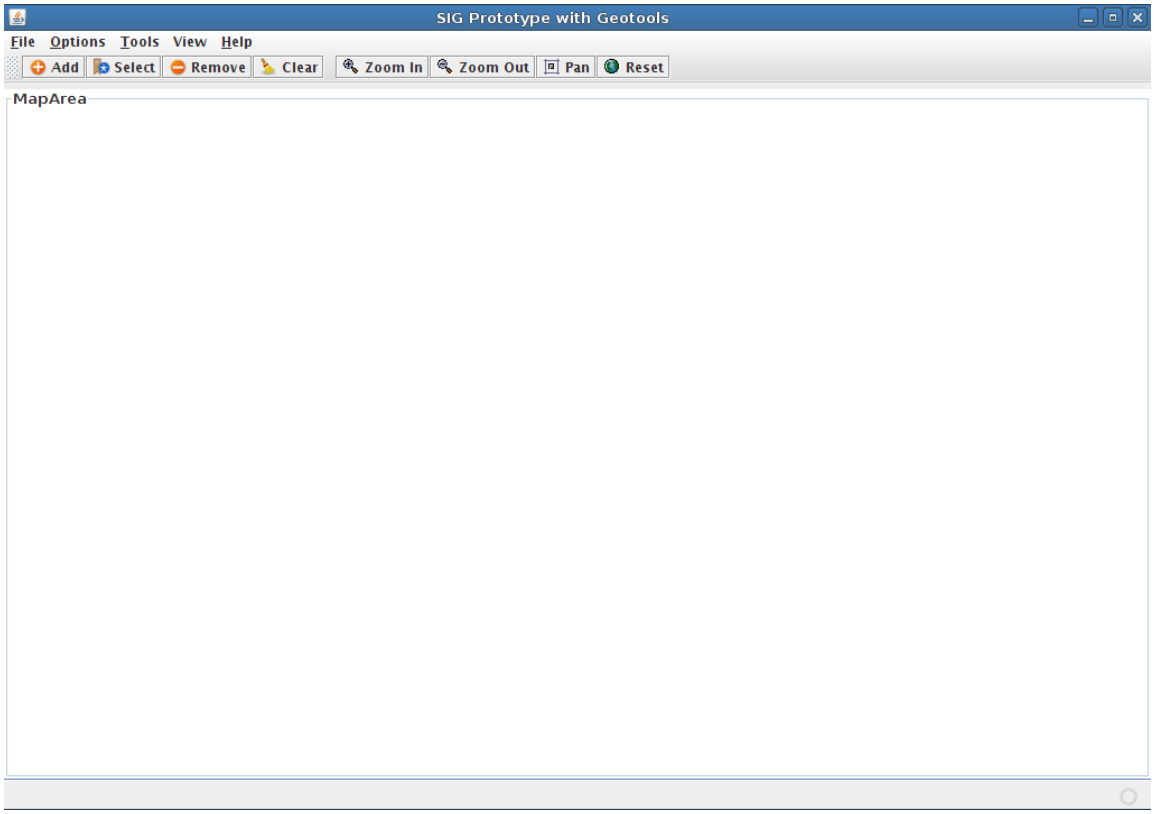
4.2.2. Historias de Usuarios

Las historias de usuario son la técnica utilizada en SXP para especificar los requisitos del software, lo que equivaldría a los casos de uso en el Proceso Unificado[1].

Historia de Usuario	
Número: 1	Nombre Historia de Usuario: Implementar interfaz de usuario
Modificación de Historia de Usuario Número: 1.0	
Usuario: Felix Alejandro Prieto Carratalá	Iteración Asignada: 1
Prioridad en Negocio:	Puntos Estimados: 1
Riesgo en Desarrollo: Alta	Puntos Reales: 1
Descripción: El prototipo de interfaz de usuario es la representación limitada del diseño de la aplicación, y tiene como objetivo probar las partes en situaciones reales y explorar su uso.	

[1] El Proceso Unificado es un marco de desarrollo de software que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental[32].

Prototipo de interfaz:



	Tarea de Ingeniería
Número Tarea: *1*	Número Historia de Usuario: 1
Nombre Tarea: Estudio de los componentes Swing de Java	
Tipo de Tarea : Investigación	Puntos Estimados: 2/7
Fecha Inicio: 11/3/2009	Fecha Fin: 12/3/2009
Programador Responsable: Felix Alejandro Prieto Carratalá	
Descripción: El estudio del uso de los componentes Swing de Java se realiza a través del tutorial oficial de Sun Microsystems disponible en http://java.sun.com/docs/books/tutorial/uiswing/.	

	Tarea de Ingeniería
Número Tarea: *2*	Número Historia de Usuario: 1

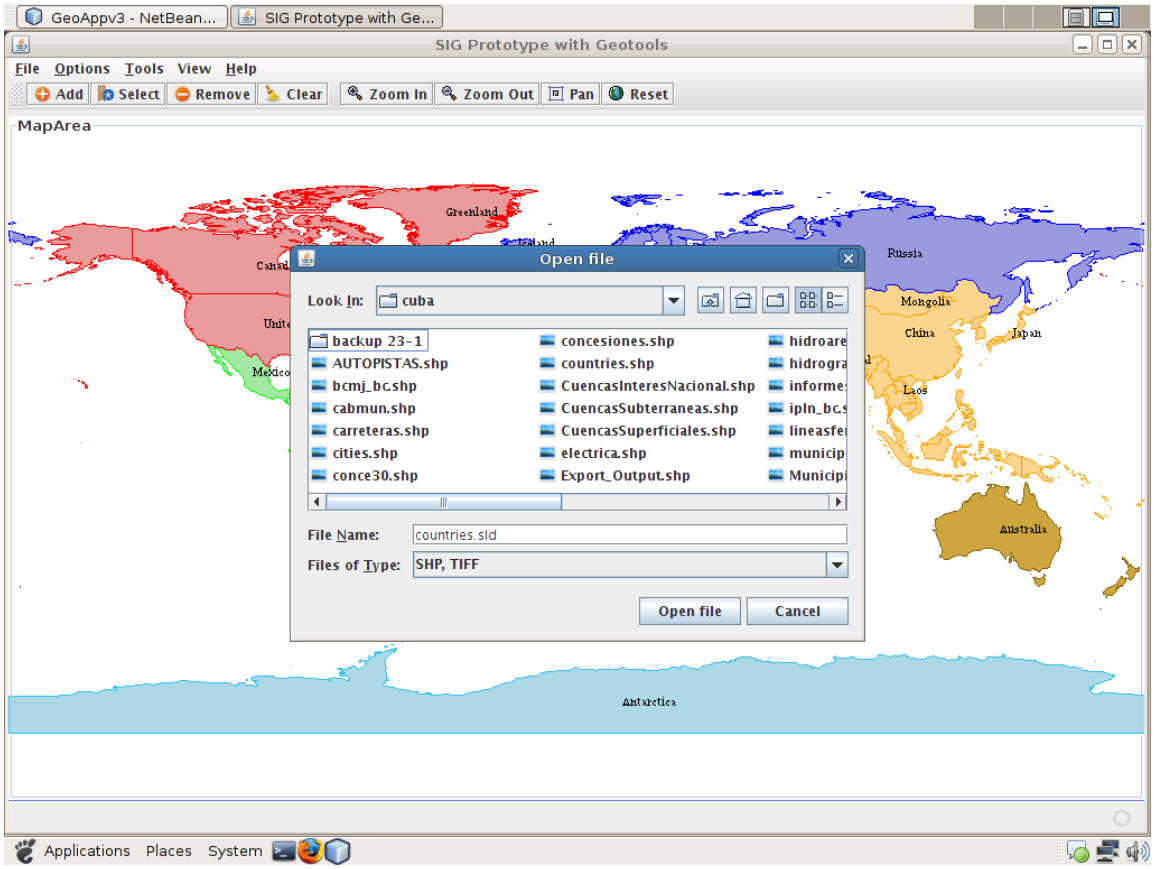
Nombre Tarea: Crear clase GeoApp	
Tipo de Tarea : Desarrollo	Puntos Estimados: 2/7
Fecha Inicio: 13/3/2009	Fecha Fin: 14/3/2009
Programador Responsable: Felix Alejandro Prieto Carratalá	
Descripción: La clase GeoApp extiende org.jdesktop.application.SingleFrameApplication para controlar la interfaz de usuario.	

Tarea de Ingeniería	
Número Tarea: *3*	Número Historia de Usuario: 1
Nombre Tarea: Crear clase GeoView	
Tipo de Tarea : Desarrollo	Puntos Estimados: 2/7
Fecha Inicio: 15/3/2009	Fecha Fin: 16/3/2009
Programador Responsable: Felix Alejandro Prieto Carratalá	
Descripción: La clase extiende GeoView org.jdesktop.application.FrameView para crear una intefaz de usuario simple con una ventana principal.	

Tarea de Ingeniería	
Número Tarea: *4*	Número Historia de Usuario: 1
Nombre Tarea: Crear la clase About	
Tipo de Tarea : Desarrollo	Puntos Estimados: 1/7
Fecha Inicio: 17/3/2009	Fecha Fin: 17/3/2009
Programador Responsable: Felix Alejandro Prieto Carratalá	
Descripción: La clase About extiende javax.swing.JDialog para ofrecer información acerca de la aplicación, como por ejemplo la versión, descripción, etc.	

Historia de Usuario		
Número: 2	Nombre Historia de Usuario: Leer y visualizar imágenes desde archivos locales de los formatos Shapefile y GeoTIFF.	
Modificación de Historia de Usuario Número: 1.0		
Usuario: Felix Alejandro Prieto Carratalá		Iteración Asignada: 1
Prioridad en Negocio:		Puntos Estimados: 2 semanas
Riesgo en Desarrollo: Alta		Puntos Reales: 2 semanas
Descripción: incluye la lectura y visualización de archivos GeoTIFF y Shapefile en el prototipo de interfaz de usuario, a través de las funcionalidades que brinda Geotools para interactuar con estos formatos.		

Prototipo de interfaz:



Tarea de Ingeniería	
Número Tarea: *1*	Número Historia de Usuario: 2
Nombre Tarea: Implementar la clase FileChooser	
Tipo de Tarea : Desarrollo	Puntos Estimados: 1/7
Fecha Inicio: 18/3/2009	Fecha Fin: 18/3/2009
Programador Responsable: Felix Alejandro Prieto Carratalá	
Descripción: La clase FileChooser extiende javax.swing.JFileChooser y es responsable de crear la vista y el filtro de imágenes, a través de sus clases privadas ImageFileView e ImageFileFilter, respectivamente.	

Tarea de Ingeniería

Número Tarea: *2*	Número Historia de Usuario: 2
Nombre Tarea: Implementar la clase ImageFileView	
Tipo de Tarea : Desarrollo	Puntos Estimados: 1
Fecha Inicio: 19/3/2009	Fecha Fin: 19/3/2009
Programador Responsable: Felix Alejandro Prieto Carratalá	
Descripción: La clase ImageFileView extiende javax.swing.filechooser.FileView para proveer a la interfaz de usuario de información acerca de los iconos y el tipo de los archivos de formatos GeoTIFF, Shapefile y ASCII GRID.	

Tarea de Ingeniería	
Número Tarea: *3*	Número Historia de Usuario: 2
Nombre Tarea: Implementar la clase ImageFileFilter	
Tipo de Tarea : Desarrollo	Puntos Estimados: 1/7
Fecha Inicio: 20/3/2009	Fecha Fin: 20/3/2009
Programador Responsable: Felix Alejandro Prieto Carratalá	
Descripción: La clase ImageFileFilter extiende javax.swing.filechooser.FileFilter para proveer un filtro al conjunto de archivos con formatos GeoTIFF, Shapefile y ASCII GRID.	

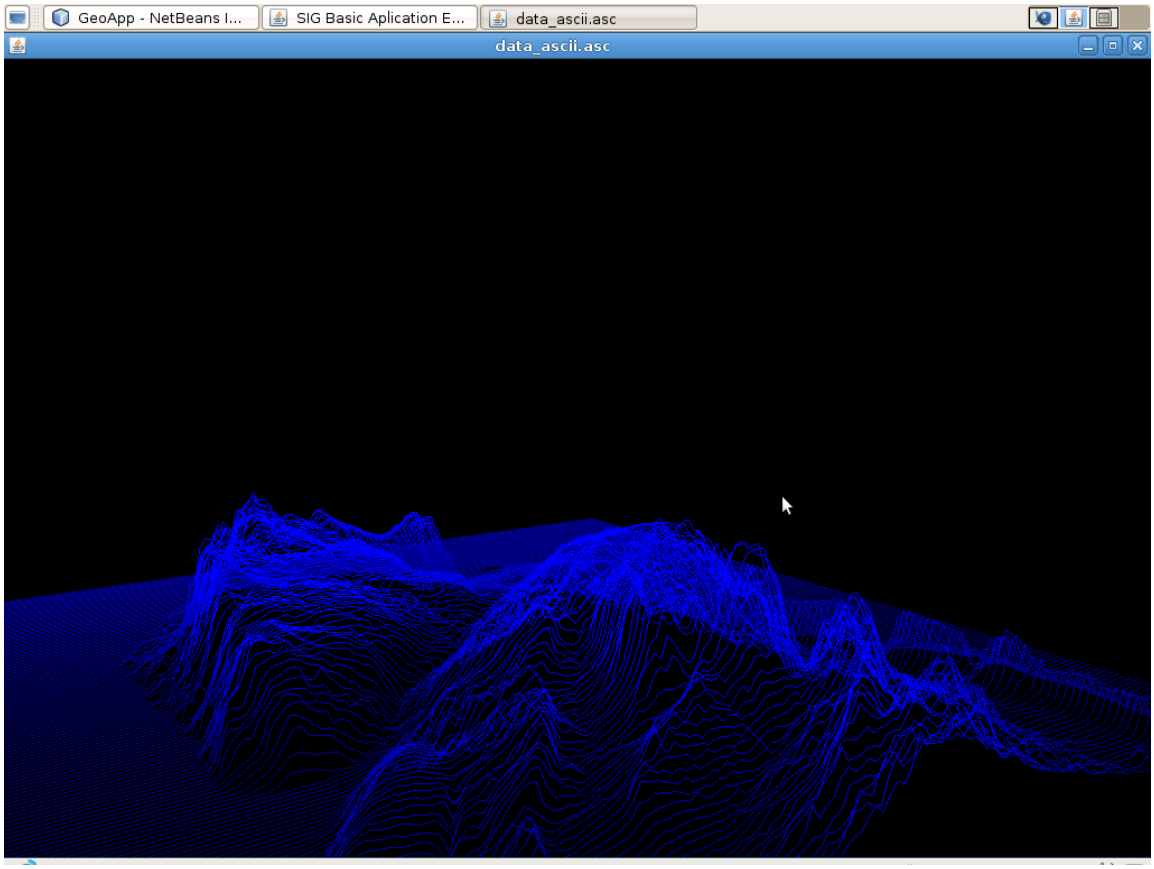
Tarea de Ingeniería	
Número Tarea: *4*	Número Historia de Usuario: 2
Nombre Tarea: Estudio de los pluggins Shapefile y GeoTIFF de la librería Geotools	
Tipo de Tarea : Investigación	Puntos Estimados: 1 semana
Fecha Inicio: 21/3/2009	Fecha Fin: 27/3/2009
Programador Responsable: Felix Alejandro Prieto Carratalá	

Tarea de Ingeniería	
Número Tarea: *5*	Número Historia de Usuario: 2
Nombre Tarea: Implementar clase StyleProvider	
Tipo de Tarea : Desarrollo	Puntos Estimados: 2/7
Fecha Inicio: 28/3/2009	Fecha Fin: 29/3/2009
Programador Responsable: Felix Alejandro Prieto Carratalá	
Descripción: Clase encargada de proveer el estilo de visualización de los datos cargados desde archivos Shapefile y GeoTIFF.	

Tarea de Ingeniería	
Número Tarea: *5*	Número Historia de Usuario: 2
Nombre Tarea: Implementar clase LayerLoadTask	
Tipo de Tarea : Desarrollo	Puntos Estimados: 2/7
Fecha Inicio: 30/3/2009	Fecha Fin: 31/3/2009
Programador Responsable: Felix Alejandro Prieto Carratalá	
Descripción: LayerLoadTask extiende org.jdesktop.application.Task para implementar la lectura de archivos de los formatos Shapefile y GeoTIFF.	

Historia de Usuario	
Número: 3	Nombre Historia de Usuario: Leer y visualizar en 3D archivos de formato ASCII GRID.
Modificación de Historia de Usuario Número: 1.0	
Usuario: Felix Alejandro Prieto Carratalá	Iteración Asignada: 1
Prioridad en Negocio:	Puntos Estimados: 1 semanas
Riesgo en Desarrollo: Alta	Puntos Reales: 2 semanas
Descripción: incluye la lectura y visualización de archivos ASCII GRID en el prototipo de interfaz de usuario.	

Prototipo de interfaz:



Tarea de Ingeniería	
Número Tarea: *1*	Número Historia de Usuario: 3
Nombre Tarea: Estudio del pluggins ArcGrid de la librería Geotools	
Tipo de Tarea : Investigación	Puntos Estimados: 3/7 semanas
Fecha Inicio: 1/4/2009	Fecha Fin: 2/4/2009
Programador Responsable: Felix Alejandro Prieto Carratalá	
Descripción: Estudio de las funcionalidades que brinda el plugin ArcGrid de la librería Geotools para procesar archivos de formato ASCII GRID.	

Tarea de Ingeniería	
Número Tarea: *2*	Número Historia de Usuario: 3

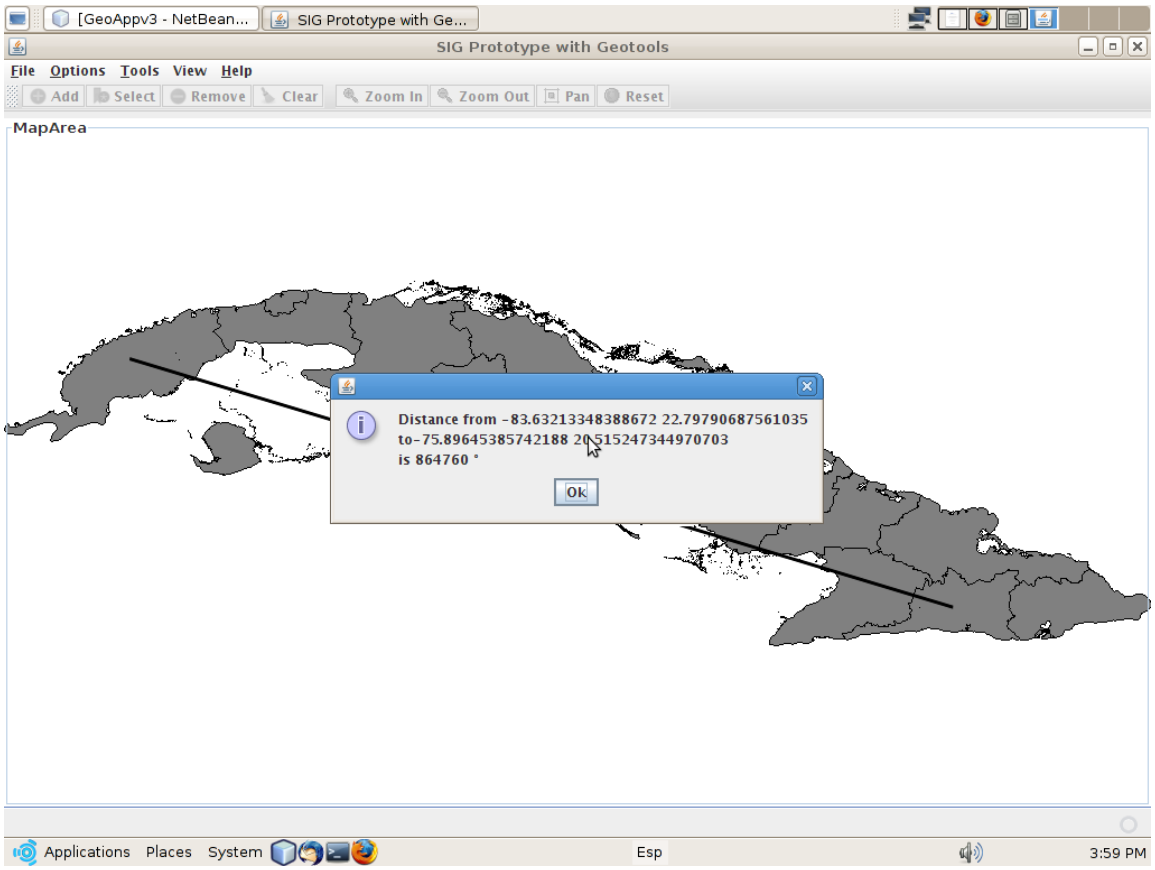
Nombre Tarea: Implementar clase GLDisplay	
Tipo de Tarea : Desarrollo	Puntos Estimados: 1/7 semana
Fecha Inicio: 3/4/2009	Fecha Fin: 3/4/2009
Programador Responsable: Felix Alejandro Prieto Carratalá	
Descripción: La clase GLDisplay crea los componentes necesarios para visualizar en una ventana la matriz de datos con JOGL.	

Tarea de Ingeniería	
Número Tarea: *3*	Número Historia de Usuario: 3
Nombre Tarea: Implementar clase RenderMode	
Tipo de Tarea : Desarrollo	Puntos Estimados: 1/7
Fecha Inicio: 4/4/2009	Fecha Fin: 4/4/2009
Programador Responsable: Felix Alejandro Prieto Carratalá	
Descripción: La clase RenderMode establece el modo de renderización de JOGL a modelo de alambre o a modelo en colores.	

Tarea de Ingeniería	
Número Tarea: *4*	Número Historia de Usuario: 3
Nombre Tarea: Implementar clase RasterRenderer	
Tipo de Tarea : Desarrollo	Puntos Estimados: 2/7
Fecha Inicio: 5/4/2009	Fecha Fin: 5/4/2009
Programador Responsable: Felix Alejandro Prieto Carratalá	
Descripción: La clase RasterRenderer lee la matriz de datos desde un archivo ASCII GRID, a través del plugin de Geotools para este formato.	

Tarea de Ingeniería	
Número Tarea: *5*	Número Historia de Usuario: 3
Nombre Tarea: Implementar clase ModelLoadTask	
Tipo de Tarea : Desarrollo	Puntos Estimados: 2
Fecha Inicio: 6/4/2009	Fecha Fin: 6/4/2009
Programador Responsable: Felix Alejandro Prieto Carratalá	
Descripción: ModelLoadTask extiende org.jdesktop.application.Task y fusiona las funcionalidades de las clases RasterRenderer, GLDisplay e InputHandler para visualizar en una ventana la matriz de datos en 3D.	

Historia de Usuario	
Número: 4	Nombre Historia de Usuario: Calcular distancia entre dos puntos
Modificación de Historia de Usuario Número: 1.0	
Usuario: Felix Alejandro Prieto Carratalá	Iteración Asignada: 1
Prioridad en Negocio:	Puntos Estimados: 1 semana
Riesgo en Desarrollo: Alta	Puntos Reales: 5/7 semana
Descripción: Permite calcular la distancia entre dos puntos de una imagen mostrada en el prototipo de interfaz.	
Prototipo de interfaz:	



Tarea de Ingeniería	
Número Tarea: *1*	Número Historia de Usuario: 4
Nombre Tarea: Implementar clase DistanceCalculator	

Tipo de Tarea : Desarrollo	Puntos Estimados: 1/7 semana
Fecha Inicio: 9/4/2009	Fecha Fin: 9/4/2009
Programador Responsable: Felix Alejandro Prieto Carratalá	
Descripción: DistanceCalculator es una clase abstracta que contiene el comportamiento común de las clases EuclideanDistanceCalculator y GeodeticDistanceCalculator.	

Tarea de Ingeniería	
Número Tarea: *2*	Número Historia de Usuario: 4
Nombre Tarea: Implementar clase EuclideanDistanceCalculator	
Tipo de Tarea : Desarrollo	Puntos Estimados: 1/7 semana
Fecha Inicio: 10/4/2009	Fecha Fin: 10/4/2009
Programador Responsable: Felix Alejandro Prieto Carratalá	
Descripción: La clase EuclideanDistanceCalculator calcula la distancia ordinaria entre dos puntos.	

Tarea de Ingeniería	
Número Tarea: *3*	Número Historia de Usuario: 4
Nombre Tarea: Implementar clase GeodeticDistanceCalculator	
Tipo de Tarea : Desarrollo	Puntos Estimados: 1/7 semana
Fecha Inicio: 11/4/2009	Fecha Fin: 11/4/2009
Programador Responsable: Felix Alejandro Prieto Carratalá	
Descripción: La clase GeodeticDistanceCalculator permite calcular la distancia geodética entre dos puntos que formen parte de la elipsoide de la tierra.	

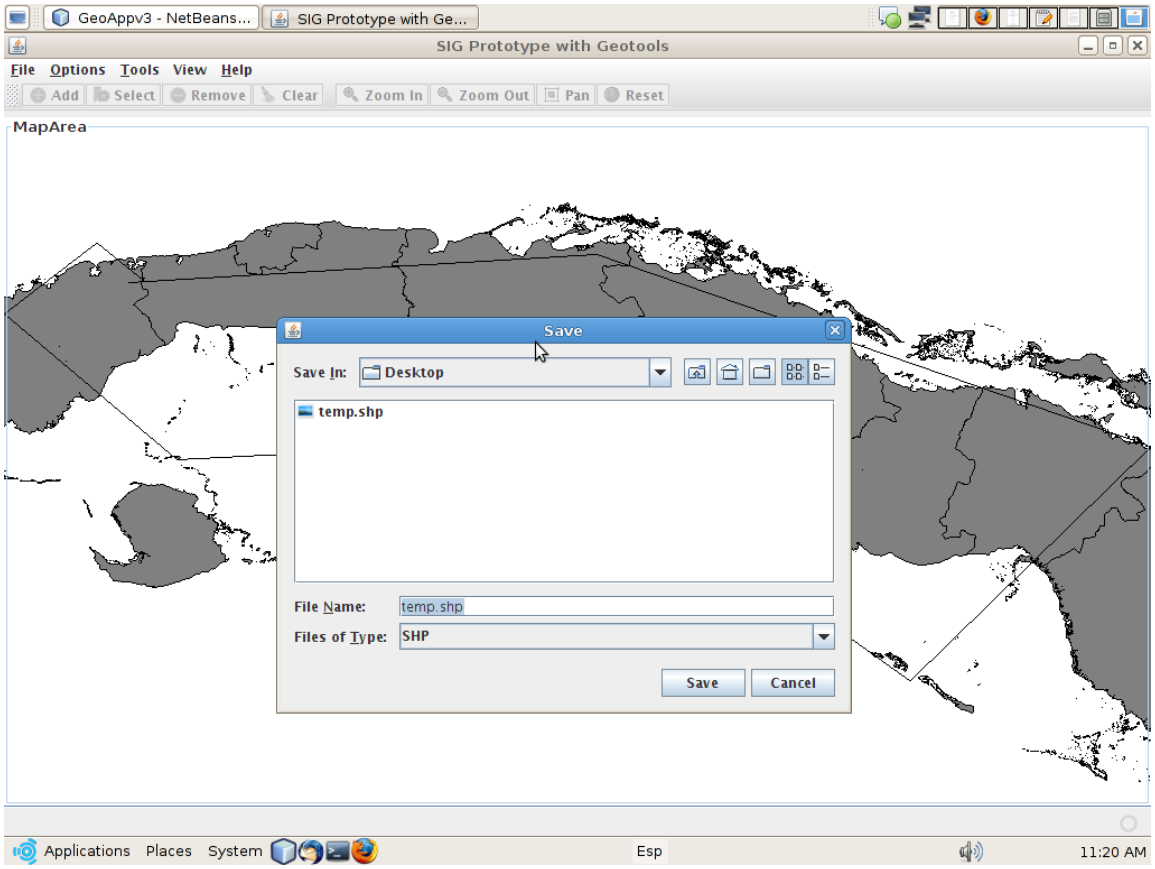
Tarea de Ingeniería	
Número Tarea: *4*	Número Historia de Usuario: 4
Nombre Tarea: Implementar clase CalculatorFactory	
Tipo de Tarea : Desarrollo	Puntos Estimados: 1/7 semana
Fecha Inicio: 12/4/2009	Fecha Fin: 12/4/2009
Programador Responsable: Felix Alejandro Prieto Carratalá	
Descripción: Implementa el patrón de diseño Factory para proveer la calculadora apropiada según la unidad de medida del sistema coordenado.	

Tarea de Ingeniería	
Número Tarea: *5*	Número Historia de Usuario: 4

Nombre Tarea: Implementar clase WorldDistanceCalculator	
Tipo de Tarea : Desarrollo	Puntos Estimados: 1/7 semana
Fecha Inicio: 13/4/2009	Fecha Fin: 13/4/2009
Programador Responsable: Felix Alejandro Prieto Carratalá	
Descripción: La clase WordDistanceCalculator, utilizando las clases CalculatorFactor y DistanceCalculator -con sus subclases-, permite calcular la distancia entre dos puntos dados en un sistema coordenado de referencia.	

Historia de Usuario	
Número: 5	Nombre Historia de Usuario: Crear y salvar archivos con capas vectoriales
Modificación de Historia de Usuario Número: 1.0	
Usuario: Felix Alejandro Prieto Carrataláa	Iteración Asignada: 1
Prioridad en Negocio:	Puntos Estimados: 1/2 semana
Riesgo en Desarrollo: Alta	Puntos Reales: 1/2 semana
Descripción: gestiona la creación de capas vectoriales.	

Prototipo de interfaz:



Tarea de Ingeniería	
Número Tarea: *1*	Número Historia de Usuario: 5
Nombre Tarea: Implementar clase ShapeWriter	
Tipo de Tarea : Desarrollo	Puntos Estimados: 3/7 semana
Fecha Inicio: 14/4/2009	Fecha Fin: 17/4/2009
Programador Responsable: Felix Alejandro Prieto Carratalá	
Descripción: La clase ShapeWriter crea un archivo de formato Shapefile a partir de un conjunto de puntos que formen parte de una capa vectorial.	

4.3. Conclusiones del capítulo

En el capítulo se describen las características principales del prototipo de aplicación SIG y la herramienta de modelación de terrenos en 3D. Teniendo como base las funcionalidades de la librería Geotools se muestra el proceso de implementación de un conjunto de requerimientos funcionales de la aplicación. El proceso de desarrollo se ejemplifica con las plantillas de los roles que se definieron, las historias de usuarios y sus descripciones.

En el desarrollo del prototipo se emplearon los plugins GeoTIFF y Shapefile de Geotools, con la idea de visualizar mapas interactivos a través del componente visual JMapPane, perteneciente también a dicha librería. Para la herramienta de visualización de terrenos se usó el plugin ArcGrid para archivos de raster con dicho formato.

CAPÍTULO 5

Validación de la solución propuesta

A continuación se plasman los casos de pruebas o test de aceptación a las que fue sometida la aplicación en cada una de las iteraciones; el cumplimiento de estos casos de pruebas fue el hito para avanzar hacia la próxima iteración.

5.1. Casos de prueba

La metodología SXP define entre iteración e iteración un conjunto de casos de pruebas o tests de aceptación para poder avanzar a una iteración superior. Durante el desarrollo del prototipo de SIG se diseñaron un conjunto de casos de prueba a las que fue sometido el sistema para comprobar el funcionamiento de acuerdo a las Historias de Usuario. Se definieron casos de prueba para todas las historias de usuario, a continuación se relacionan las pruebas más significativas realizadas.

5.1.1. Caso de Prueba: Historia de Usuario 1

Esta sección cubre el conjunto de pruebas funcionales relacionadas con la historia de usuario: Implementar interfaz de usuario. Se comprueba que la aplicación se ejecute en una ventana principal y muestre un diálogo de información de la misma.

Caso de Prueba de Aceptación	
Código Caso de Prueba: 1	Nombre Historia de Usuario: Implementar interfaz de usuario
Nombre de la persona que realiza la prueba: Felix Alejandro Prieto Carratalá	
Descripción de la Prueba: al lanzar la aplicación debe mostrarse la ventana con un menú y una barra de herramientas.	

Condiciones de Ejecución: ninguna
Entrada / Pasos de ejecución: ejecutar el comando "java -jar file.jar", siendo *file* el nombre del archivo con formato jar.
Resultado Esperado: se muestra la ventana principal.
Evaluación de la Prueba: Satisfactoria

Caso de Prueba de Aceptación	
Código Caso de Prueba: 2	Nombre Historia de Usuario: Implementar interfaz de usuario
Nombre de la persona que realiza la prueba: Felix Alejandro Prieto Carratalá	
Descripción de la Prueba: una vez que la aplicación esté visible se comprueba que el menú "Help-About" funciona brindando información acerca de la aplicación.	
Condiciones de Ejecución: la aplicación en ejecución.	
Entrada / Pasos de ejecución: seleccionar en el menú "Help", la opción "About".	
Resultado Esperado: se muestra un ventana de diálogo con información de la aplicación	
Evaluación de la Prueba: Satisfactoria	

5.1.2. Caso de Prueba: Historia de Usuario 2

Esta sección cubre el conjunto de pruebas funcionales relacionadas con la historia de usuario: Leer imágenes desde archivos locales de los formatos Shapefile y GeoTIFF. Se comprueba que la aplicación lea y muestre los datos de imágenes de dichos formatos.

Caso de Prueba de Aceptación	
Código Caso de Prueba: 1	Nombre Historia de Usuario: Leer y visualizar imágenes desde archivos locales de los formatos Shapefile y GeoTIFF.
Nombre de la persona que realiza la prueba: Felix Alejandro Prieto Carratalá	
Descripción de la Prueba: Se inicia la aplicación y se selecciona en el menú "File" la opción "Open File", y se muestra un diálogo para abrir archivos de los formatos GeoTIFF y Shapefile. Una vez seleccionado el archivo se lee y se visualiza la imagen en el área de trabajo de la aplicación.	
Condiciones de Ejecución: la aplicación debe estar ejecutándose.	
Entrada / Pasos de ejecución: Seleccionar en el menú "File", la opción "Open File" y escoger un archivo de imagen.	
Resultado Esperado: Se muestra los datos de la imagen en el área de trabajo.	
Evaluación de la Prueba: Satisfactoria	

5.1.3. Caso de Prueba: Historia de Usuario 3

Esta sección cubre el conjunto de pruebas funcionales relacionadas con la historia de usuario: Salvar imágenes a archivos de capas vectoriales. Se comprueba la aplipación permita crear capas vectoriales a partir de la interacción con el usuario y salvarlas como archivos de formato Shapefile.

Caso de Prueba de Aceptación	
Código Caso de Prueba: 1	Nombre Historia de Usuario: Crear y salvar archivos con capas vectoriales
Nombre de la persona que realiza la prueba: Felix Alejandro Prieto Carratalá	
Descripción de la Prueba: se procede a hacer clicks sucesivos con el botón izquierdo en distintos puntos de la imagen mostrada en el área de trabajo de la aplicación para crear una capa vectorial de datos, y se termina con un click derecho.	
Condiciones de Ejecución: la aplicación debe estar ejecutándose y mostrando una imagen previamente cargada.	
Entrada / Pasos de ejecución: seleccionar en el menú "Tools", la opción "Write shape", realizar clicks sucesivos con el botón izquierdo del ratón en distintos puntos de la imagen mostrada en el área de trabajo y terminar con el derecho.	
Resultado Esperado: en el área de trabajo se visualizan los vectores de la capa creada a partir de los clicks del usuario.	
Evaluación de la Prueba: Satisfactoria	

Caso de Prueba de Aceptación	
Código Caso de Prueba: 2	Nombre Historia de Usuario: Crear y salvar archivos con capas vectoriales
Nombre de la persona que realiza la prueba: Felix Alejandro Prieto Carratalá	
Descripción de la Prueba: una vez creada la capa a partir de la interacción del usuario a través del ratón se procede a mostrar un diálogo para salvar el archivo Shapefile que va a contener la capa vectorial creada.	
Condiciones de Ejecución: capas de vectores creada previamente a partir de la interacción con el usuario a través del ratón.	
Entrada / Pasos de ejecución: escribir un nombre de archivo y seleccionar "Save".	
Resultado Esperado: se crean los archivos con la extensiones file.shp, file.shx, file.prj, file.quix, file.dbf y file.fix, siendo *file* el nombre de los archivos.	
Evaluación de la Prueba: Satisfactoria	

5.1.4. Caso de Prueba: Historia de Usuario 4

Esta sección cubre el conjunto de pruebas funcionales relacionadas con la historia de usuario: Calcular distancia entre dos puntos. Se comprueba la aplicación permita calcular la distancia entre dos puntos cualesquiera de la imagen mostrada en el área de trabajo.

Caso de Prueba de Aceptación	
Código Caso de Prueba: 2	Nombre Historia de Usuario: Calcular distancia entre dos puntos
Nombre de la persona que realiza la prueba: Felix Alejandro Prieto Carratalá	
Descripción de la Prueba: se procede a hacer un click con el botón izquierdo en un punto de la imagen mostrada en el área de trabajo de la aplicación y se termina con un click derecho, para determinar la distancia entre ambos puntos.	
Condiciones de Ejecución: la aplicación debe estar ejecutándose y mostrando una imagen previamente cargada.	
Entrada / Pasos de ejecución: seleccionar en el menú "Tools", la opción "Calculator", realizar un click con el botón izquierdo del ratón en un punto de la imagen mostrada en el área de trabajo y terminar con el derecho.	
Resultado Esperado: se muestra un diálogo con la distancia entre los dos puntos.	
Evaluación de la Prueba: Satisfactoria	

5.1.5. Caso de Prueba: Historia de Usuario 5

Esta sección cubre el conjunto de pruebas funcionales relacionadas con la historia de usuario: Visualizar en 3D archivos de formato ASCII GRID. Se comprueba la aplicación lea la matriz de datos contenida en un archivo ASCII GRID y las modele en 3D, visualizando el resultado en una ventana interactiva secundaria.

Caso de Prueba de Aceptación	
Código Caso de Prueba: 1	Nombre Historia de Usuario: Leer y visualizar en 3D archivos de formato ASCII GRID
Nombre de la persona que realiza la prueba: Felix Alejandro Prieto Carratalá	
Descripción de la Prueba: se inicia la aplicación y se selecciona en el menú "Tools" la opción "Raster3D", y se muestra un diálogo para abrir archivos del formato ASCII GRID. Una vez seleccionado el archivo se lee y se visualizan en 3D los datos en una ventana secundaria.	
Condiciones de Ejecución: la aplicación debe estar ejecutándose.	
Entrada / Pasos de ejecución: seleccionar en el menú "Tools", la opción "Raster3D".	
Resultado Esperado: se muestra un ventana con los datos leídos modelados en 3D, que permite interactuar con ella acercando y alejando el modelo.	
Evaluación de la Prueba: Satisfactoria	

5.2. Conclusiones del capítulo

En el capítulo se presentaron algunos casos de pruebas en las que se centraron el control de calidad del prototipo y que sirven de base para demostrar los resultados obtenidos en el período que se enmarca. Las pruebas realizadas al prototipo y que involucran algunas de las funcionalidades básicas de un SIG resultaron satisfactorias; igualmente ocurre con la herramienta de modelación de superficies de terrenos en 3D.

Conclusiones generales

En el trancurso de la investigación se realizó un estudio de las principales funcionalidades de la librería Geotools para el desarrollo de Sistemas de Información Geográfica. Luego de haber utilizado parte del conjunto de interfaces y clases que provee en el desarrollo de un Sistema de Información Geográfica (SIG) prototipo se concluye que su utilización es una alternativa para desarrollar SIGs, ya que:

- Está basada en estándares y posee una arquitectura bien definida, garantizando interoperabilidad y robustez en los productos de software.
- Tiene una organización modular, por lo que es sencillo añadirle o quitarle funcionalidades a las soluciones que se basen en su implementación.
- Provee un conjunto de interfaces y clases que facilitan el proceso de diseño e implementación de aplicaciones orientadadas a los SIGs.
- Su implementación es libre, lo cual es perfecto en el marco del proceso de migración a software libre que se efectúa actualmente en Cuba, y propicia además una mejora continua del código por la comunidad.

Teniendo en cuenta la adhesión de una herramienta de modelación de superficies de terrenos en 3D al prototipo desarrollado se plantea que:

- Geotools permite extender su alcance a otras esferas fuera del ámbito de los SIGs de una forma fácil y consistente con su arquitectura.
- Es posible desarrollar una herramienta que se integre con un Sistema de Información Geográfica (SIG) implementado sobre Geotools.

Por tanto, se concluye que se cumplieron los objetivos trazados satisfactoriamente al inicio de la investigación, ya que, luego de diseñar e implementar un prototipo de aplicación

SIG basado en Geotools, se demostró la posibilidad de extender las soluciones creadas con dicha librería al añadirle una herramienta sencilla de modelación de superficies de terrenos en 3D.

Recomendaciones

Aunque se cumplieron los objetivos de las investigación y se alcanzaron los resultados esperados, a continuación se incluyen varias recomendaciones a tener en cuenta:

- Utilizar el prototipo SIG como ejemplo de estudio en la utilización de Geotools, en el Polo de GeoInformática de la UCI.
- Enriquecer el prototipo, añadiéndole nuevas extensiones y funcionalidades.
- Implementar la modelación de superficies en 3D usando las técnicas mencionadas en el capítulo Preliminares.
- Realizar un estudio de las nuevas funcionalidades de Geotools, en sus versiones estables más recientes para actualizar el prototipo.

Glosario de términos

M | P | Q | R | T

M

modelación de terrenos representación digital de la topografía de la superficie terrestre a partir de conjuntos de datos. 3

P

pixel es la menor unidad homogénea en color que forma parte de una imagen digital, ya sea esta una fotografía, un fotograma de vídeo o un gráfico. 6

Q

QuadTree estructura de datos en árbol dónde cada nodo tiene cuatro hijos, usada en los Sistemas de Información Geográfica. 11, 12

R

renderización proceso de generar una imagen desde un modelo. 23

resolución tamaño mínimo de un elemento en una imagen o mapa. 6

T

TIN Red Irregular Triangulada (del inglés Triangulated Irregular Network) se refiere a la estructura de datos usada en los Sistemas de Información Geográfica para representar una superficie, que se basa en nodos y líneas distribuidos irregularmente. 11

Siglas

A | B | C | D | E | I | J | L | M | N | O | S

A

ADRG ARC Digitized Raster Graphics. 6

API Application Program Interface. 12, 13

B

BIL Band Interleaved by Line. 7

C

CADRG Compressed ARC Digitised Raster Graphics. 6

CIB Controlled Image Base. 7

D

DRG Digital Raster Graphic. 7

E

ECW Enhanced Compressed Wavelet. 7

EPSG European Petroleum Survey Group. 7, 22

ESRI Environmental Systems Research Institute. 7

I

IDE Integrated Development Environment. 17, 19

IMG ERDAS IMAGINE. 7

ISO International Organization for Standardization. 10, 22

J

JOGL Java OpenGL. 24

JSR Java Specification Requests. 24

JTS Java Topology Suite. 10, 22

JVM Java Virtual Machine. 9

L

LGPL GNU Lesser General Public License. 10

LIS Land Information System. 5

M

MrSID Multi-Resolution Seamless Image Database. 7

MVC Modelo-Vista-Controlador. 21, 25

N

NCGIA National Center of Geographic Information and Analysis. 1

O

OGC Open Geospatial Consortium. 9, 10, 23

S

SDBMS Spacial Data Base Management System. 9

SIG Sistema de Información Geográfica. 1–3, 5–7, 9, 15, 17, 20, 25, 26, 47–50

SLD Style Layer Descriptor. 23

Bibliografía

[1] GeoAPI. Disponible en http://geoapi.sourceforge.net/, [ONLINE]. [Consultado x Marzo de 2009].

[2] A. Aguilera, F.R. Feito y J.C. Torres. Visualización de terrenos mediante niveles de detalle. *In XII Congreso Nacional de Tecnología de la Información Geográfica* (Septiembre 2006), 315 322.

[3] Ana Beatriz Ochoa G. Métodos de investigación. Disponible en http://www.monografias.com/trabajos11/metods/metods.shtml, [ONLINE]. [Consultado x Marzo de 2009].

[4] Daniel Cohen-Or and Yishay Levanoni. Temporal continuity of levels of detail in delaunay triangulated terrain. *Roni Yagel and Gregory M. Nielson, editors, IEEE Visualization* (1996), 3742.

[5] Dr F. Escobar, Assoc Prof G. Hunter, Assoc Prof I. Bishop, Dr A. Zerger. Introducción a los SIG. Disponible en http://www.sli.unimelb.edu.au/gisweb/, [ONLINE]. [Consultado x Marzo de 2009].

[6] ESRI. GIS Dictionary. Disponible en http://support.esri.com/index.cfm?fa=knowledgebase.gisDictionary.search&searchTerm=desktop%20GIS, [ONLINE]. [Consultado x Marzo de 2009].

[7] ESRI. ASCII Grid File Format Specification, 2005.

[8] jive. GeoTools 2.5 Users Guide. Disponible en http://docs.codehaus.org/display/GEOTDOC/Home, [ONLINE]. [Consultado x Marzo de 2009].

[9] Lloyd P. Queen, Charles R. Blinn. The Basics of Geographic Information Systems. Disponible en http://www.extension.umn.edu/distribution/naturalresources/DD5926.html, [ONLINE]. [Consultado x Marzo de 2009].

[10] Mariano Perez, Ricardo Olanda y Marcos Fernandez. Visualization of large terrain using non-restricted quadtree triangulations. *In Heidelberg Springer Berlin, editor, Computational Science and Its Applications ICCSA* (2004), 671 681.

[11] NCGIA. NCGIA CORE CURRICULUM 1990 Version. Disponible en http://www.geog.ubc.ca/courses/klink/gis.notes/ncgia/toc.html, [ONLINE]. [Consultado x Marzo de 2009].

[12] Open Geospatial Consortium. About OGC. Disponible en http://www.opengeospatial.org/ogc, [ONLINE]. [Consultado x Marzo de 2009].

[13] Open Geospatial Consortium. OpenGIS Standards and Specifications. Disponible en http://www.opengeospatial.org/standards, [ONLINE]. [Consultado x Marzo de 2009].

[14] Ortiz, G. Visualización 3d avanzada: Propuestas para la interoperatividad entre aplicaciones. Disponible en http://www.gabrielortiz.com/index.asp?Info=034, [ONLINE]. [Consultado x Marzo de 2009].

[15] Ortiz, G. Visualización 3D Avanzada: Propuestas para la interoperatividad entre aplicaciones. Disponible en http://geotools.codehaus.org/, [ONLINE]. [Consultado x Marzo de 2009].

[16] P. Lindstrom and V. Pascucci. Terrain simplification simplified: A general framework for view-dependent out-of-core visualization. *IEEE Transaction on Visualization and Computer Graphics* (Julio 2002), 239 254.

[17] P. Lindstrom, D. Koller, Larry F. Hodges, W. Ribarsky, Nick L. Faust, and G. Turner. Level-of-detail management for real-time rendering of phototextured terrain. *Gvu technical report* (1995).

[18] P. Lindstrom, D. Koller, Larry F. Hodges, W. Ribarsky, Nick L. Faust, and G. Turner. Real-time continuous level of detail rendering of height fields. *Proceedings of SIGGRAPH 1996* (1996), 109 118.

[19] Paolo Cignoni, Enrico Puppo y Roberto Scopigno. Representation and visualization of terrain surfaces at variable resolution. *The Visual Computer* (1997), 199217.

[20] Renato Pajarola. Overview of quadtree based terrain triangulation and visualization. *Technical report uci-ics tr 02-01* (January 2002), 239 254.

[21] Renato Pajarola, Marc Antonijuan, and Roberto Lario. Quadtin: quadtree based triangulated irregular networks. *In VIS 2002: Proceedings of the conference on Visualization 2002* (2002), 395 402.

[22] Ritter, N. GeoTIFF Format Specification, GeoTIFF Revision 1.0.

[23] STEFAN STEINIGER, ERWAN BOCHER. Terrain simplification simplified: A general framework for view-dependent out-of-core visualization. *Int. J. of Geographical Information Science* (Sept 5 2008).

[24] WIKIMEDIA FOUNDATION. Desarrollo ágil de software. Disponible en http://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software, [ONLINE]. [Consultado x Marzo de 2009].

[25] WIKIMEDIA FOUNDATION. Esri grid. Disponible en http://en.wikipedia.org/wiki/ESRI_grid, [ONLINE]. [Consultado x Marzo de 2009].

[26] WIKIMEDIA FOUNDATION. Geographic information system. Disponible en http://en.wikipedia.org/wiki/Geographic_information_system#Free_and_Open-source_GIS_software, [ONLINE]. [Consultado x Marzo de 2009].

[27] WIKIMEDIA FOUNDATION. GIS file formats. Disponible en http://en.wikipedia.org/wiki/GIS_file_formats, [ONLINE]. [Consultado x Marzo de 2009].

[28] WIKIMEDIA FOUNDATION. GIS software categories. Disponible en http://en.wikipedia.org/wiki/GIS_software#GIS_software_categories, [ONLINE]. [Consultado x Marzo de 2009].

[29] WIKIMEDIA FOUNDATION. Java topology suite. Disponible en http://es.wikipedia.org/wiki/Java_Topology_Suite, [ONLINE]. [Consultado x Marzo de 2009].

[30] WIKIMEDIA FOUNDATION. Jogl. Disponible en http://es.wikipedia.org/wiki/JOGL, [ONLINE]. [Consultado x Marzo de 2009].

[31] WIKIMEDIA FOUNDATION. Open geospacial consortium. Disponible en http://es.wikipedia.org/wiki/OGC, [ONLINE]. [Consultado x Marzo de 2009].

[32] WIKIMEDIA FOUNDATION. Proceso Unificado. Disponible en http://es.wikipedia.org/wiki/Proceso_Unificado, [ONLINE]. [Consultado x Marzo de 2009].

[33] WIKIMEDIA FOUNDATION. Sistema de Información Geográfica. Disponible en http://es.wikipedia.org/wiki/Sistema_de_Informaci%C3%B3n_Geogr%C3%A1fica, [ONLINE]. [Consultado x Marzo de 2009].

[34] WIKIMEDIA FOUNDATION. Styled Layer Descriptor. Disponible en http://es.wikipedia.org/wiki/Styled_Layer_Descriptor, [ONLINE]. [Consultado x Marzo de 2009].

Apéndices

Las páginas siguientes contienen información accesoria o dependiente, pero que no se aborda directamente en la investigación.

APÉNDICE A

Licencias

A.1. SOSNOKILLLICENSE

The file DateUtils requires that this license be included with your GeoTools documentation.

Copyright (c) 2002-2004, Dennis M. Sosnoski. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of JiBX nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS .ᴬS IS.ᴬND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CON-

TRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.