Universidad de las Ciencias Informáticas

Facultad 6

Título:

"Herramienta para la Visualización de Información Geográfica basada en MapWindow"

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

Autora:
Annaliet Castellón Naranjo

Tutor:
Ms.C. Yesnier Bravo García

Junio 2012

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Annaliet Castellón Naranjo

_____

Firma del Autor

Yesnier Bravo García

_____

Firma del Tutor

Datos de Contacto

Nombre del tutor: Yesnier Bravo García.

Título del tutor: Master en ciencias.

Cargo del tutor en la universidad: Profesor.

Correo electrónico del tutor: yesnier@uci.cu.

Apartamento en la universidad: 23203.

Teléfono: 8358756.

Agradecimientos

A mami, abuela y papi.

Al resto de mi familia que los considero maravillosos.

A los amigos, compañeros y profesores de estos cinco años.

A Juan Manuel Sierra Martínez por su apoyo incondicional.

A mi tutor, Yesnier Bravo García, que tanto me ayudó.

A todos los que estuvieron ahí cuando los necesité.

A los que no necesité pero sé que estuvieron siempre.

Nunca he de olvidarlos.

Anna.

Dedicatoria

A las madres más especiales del mundo,

Rafaela Medina Peña y **Miriam Naranjo Medina**.

A esas madres por enseñarme a soñar y a luchar por esos sueños.

Las amo entrañablemente, siempre de ustedes, su niña.

Resumen

El desarrollo de software en el proyecto SIG-Desktop ha estado enfocado exclusivamente al trabajo con la herramienta Quantum GIS y el framework Qt, pero estas tecnologías han venido aparejadas con problemas de productividad y falta de motivación para los programadores del proyecto. Durante el desarrollo del presente trabajo de diploma se busca una alternativa a esta situación. Su objetivo es implementar una herramienta que permita sentar las bases para mejorar la productividad y las oportunidades de negocio en el proyecto SIG-Desktop. Se estudian algunos Sistemas de Información Geográfica (SIG) de escritorio y de software libre encontrando la tecnología correcta para implementar la herramienta que es objetivo de este trabajo. El framework MapWindow es la librería que podría dar solución al problema existente en SIG-Desktop. Se determinan los conceptos que guardan relación con el dominio del problema introduciendo el concepto de controles de usuarios como una alternativa para la solución. Se realiza el levantamiento de requisitos y el análisis de los mismos. Luego se exponen los artefactos que permiten modelar la herramienta, definiendo así la arquitectura y haciendo un detallado diseño del software. Se definen las pruebas que podrán validar el sistema y se exponen los resultados después de aplicarle dichas pruebas a la aplicación.

Palabras Claves: SIG de escritorio y software libre, framework MapWindow y controles de usuarios.

Índice de contenidos

Índice de ilustraciones

Índice de tablas

Introducción

El desarrollo de la computación ha permitido enriquecer las funcionalidades y el campo de aplicación para los Sistemas de Información Geográfica (SIG), por lo que cada día se hace más cotidiano en todo el mundo el uso de ellos. Los SIG permiten hacer un análisis exhaustivo de un área geográfica. El análisis de la información geográfica permite aliviar necesidades estratégicas de muchos negocios, proporcionando respuestas a preguntas como: ¿Qué hay? ¿Dónde se encuentra? ¿Por qué ocurre allí? De esta forma, se convierten en herramientas con un amplio campo de aplicación en cualquier actividad que conlleve un componente geoespacial.

En Cuba, una institución destacada en el desarrollo de software es la Universidad de las Ciencias Informáticas. La misma cuenta con el Centro de Desarrollo Geo-informática y Señales Digitales (GEySED), el cual dirige proyectos para la construcción de SIG a la medida. Uno de estos proyectos es SIG-Desktop, designado a desarrollar SIG de escritorios que sean competitivos en el mercado actual. Estos deben ser fácilmente personalizables de acuerdo con las exigencias de los clientes, poder integrarse con otras aplicaciones y permitir adicionar funcionalidades nuevas en dependencia de las oportunidades de negocio. Actualmente, se está liberando su primer producto GeoQ basado en la plataforma Quantum GIS (Qgis). Esta herramienta ha sido desarrollada en C++, usando el framework Qt. Su versión libre es la que se usa en SIG-Desktop y sus capacidades son limitadas cuando se cargan medianos o grandes volúmenes de datos. Se tiene la experiencia reciente con la Organización Nacional de Recursos Mineros (ONRM), los cuales quedaron insatisfechos con una personalización de GeoQ, por los problemas de rendimiento que presentaba al cargar los mapas vectoriales con los que esta entidad trabaja.

El desarrollo sobre la plataforma Quantum GIS tiene sus dificultades. Lo más importante es la falta de documentación referente a su arquitectura. Los programadores deben interpretar directamente el código, el cual no muestra uniformidad en cuanto a estándares de codificación. Además, no es posible depurar las soluciones y la búsqueda de un error suele ser una tarea engorrosa. En este escenario, encontrar personal idóneo para desarrollar nuevas funcionalidades de forma rápida y eficiente es una tarea difícil. La mayoría de los estudiantes, que son los programadores del proyecto, no arriban al mismo con las habilidades necesarias y les resulta difícil desarrollarlas bajo estas condiciones. Como resultado muestran rechazo con respecto al rol de programador. Este aspecto es muy importante para la Universidad, pues la función formativa de la actividad productiva se ve afectada.

Por otro lado, muchos proyectos en la UCI requieren el uso de SIG en sus soluciones, y la política de

la universidad está enfocada a la colaboración entre distintos proyectos y a favor del desarrollo de soluciones integrales. Aunque se está abogando por la independencia tecnológica y el software libre, realmente hay muchas tecnologías diferentes involucradas en las soluciones actuales en la universidad, y el éxito de muchos de los proyectos indica la permanencia de este escenario por más tiempo.

Con propósito de esta colaboración entre proyectos de la universidad, recientemente se diseñó una componente de mapificación basada en GeoQ para el Sistema de Atención a Emergencias de Venezuela (Proyecto 171). Esta estuvo marcada por la integración de diferentes tecnologías. El Sistema de Gestión de Emergencias de Seguridad Ciudadana (SIGESC) está implementado sobre la plataforma .NET, y cuenta con un número considerable de módulos, que se ejecutan e interoperan de forma segura dentro de un domino de aplicación, un recurso de .NET. Además, se firman los ensamblados de .NET para mayor seguridad. La componente que se desarrolló en el proyecto SIG-Desktop está implementada en C++, y no se puede usar la firma, ni ejecutar en un dominio de aplicaciones, permitiendo su suplantación con facilidad. Lo anterior evidencia la necesidad de considerar distintas tecnologías para el desarrollo de SIG, a fin de dar soluciones acordes a los sistemas con los que se quiera lograr una integración. Además, de esta forma se incrementan las oportunidades de nuevos clientes de aplicaciones SIG fuera de la universidad.

Otro elemento tomado en cuenta en esta investigación es que en el proyecto generalmente no se implementan funcionalidades SIG de bajo nivel o que requieran un manejo óptimo de memoria o tomar control del rendimiento, para lo cual el lenguaje C++ ofrece grandes ventajas. El objetivo es el desarrollo rápido de aplicaciones (RAD, por sus siglas en inglés) a la medida, haciendo uso de las características provistas por la plataforma Quantum GIS. En este entorno es más apropiado el desarrollo en lenguajes como java, c# o Visual Basic, que se destacan por la gestión automática de memoria, sobre plataformas J2E o .NET, siendo esta última de referencia mundial actualmente para el desarrollo de aplicaciones SIG.

El proyecto SIG-Desktop debería contar con alguna propuesta de solución o alguna alternativa, que siente las bases para desarrollar un sistema que resuelva las necesidades de la situación mencionada. De esta manera queda identificado el siguiente **problema a resolver**: ¿Cómo mejorar la productividad de los desarrolladores y las oportunidades de negocio del proyecto SIG-Desktop?

Partiendo del problema de la investigación se propone como **objeto de estudio**: las plataformas SIG de escritorio. Y el **campo de acción**: las plataformas SIG de escritorio de interés para SIG-Desktop.

Para dar solución al problema se define como **objetivo general**: Implementar una aplicación SIG usando una plataforma base (alternativa a Quantum GIS) sobre .NET que permita mejorar la productividad y las oportunidades de negocio en el proyecto SIG-Desktop.

Se defiende la siguiente **idea**: Si se utilizara en el proyecto SIG-Desktop una plataforma base (alternativa a Quantum GIS) sobre .NET entonces sería posible mejorar la productividad y las oportunidades de negocio en dicho proyecto.

Las **tareas** que se definen para la investigación son:

- Caracterizar el estado del arte de los procesos de desarrollo de visores de información geográfica libre y de escritorio.
- Identificar y argumentar las tendencias y tecnologías actuales.
- Identificar las funcionalidades de la herramienta.
- Analizar y diseñar el sistema.
- Implementar las funcionalidades del sistema.
- Validar el resultado obtenido.

Para solucionar el problema planteado se utilizan como **métodos científicos**:

Teóricos

- El histórico – lógico para estudiar las características esenciales de otros SIG de escritorio y mejorar el desarrollo de la herramienta que se pretende lograr.
- El analítico – sintético para analizar otros SIG de escritorio, tomar lo mejor de cada uno y unir esas características formando una mejor solución.
- La modelación que se aplicará creando abstracciones del sistema que se desea obtener para explicar ingenierilmente el resultado que se quiere lograr.

Empíricos

- La observación para identificar el curso natural del proceso de elaboración de SIG de escritorio en el proyecto SIG-Desktop y así conocer mejor cómo resolver el problema de la investigación.

Se esperan como **resultados**:

- La selección de una plataforma alternativa a Quantum GIS basada en .NET que satisfaga los

requerimientos del proyecto SIG-Desktop.

- Una herramienta SIG de escritorio con funcionalidades básicas de visualización, con componentes altamente reutilizables, que siente las bases para el desarrollo rápido de aplicaciones a la medida en el proyecto SIG-Desktop.
- Documentación ingenieril abundante para facilitar el desarrollo de otras aplicaciones basadas en la misma tecnología.

Este trabajo de diploma está estructurado en 4 capítulos, los cuales se describen brevemente a continuación:

Capítulo 1: En este capítulo se valoran algunos aspectos que son importantes para seleccionar una alternativa a la plataforma Quantum GIS, se analizan los SIG libres y de escritorios más utilizados en el mundo y se selecciona la tecnología que puede solucionar la situación antes descrita.

Capítulo 2: En este capítulo se presenta el Modelo de Dominio, se identifican los requisitos funcionales y no funcionales, se agrupan dichos requisitos en Casos de Uso del Sistema (CUS) y se describen algunos de estos Casos de Uso.

Capítulo 3: En este capítulo se expone la arquitectura propuesta, se explica el diseño de clases de cada paquete arquitectónico, se representan los diagramas de secuencias de algunos CUS y se muestra el Diagrama de despliegue de la herramienta.

Capítulo 4: En este capítulo se definen las pruebas que se le realizaron a la aplicación, los casos de prueba según los Casos de Uso del Sistema y se describen los resultados obtenidos.

Capítulo 1: Alternativas tecnológicas para SIG-Desktop

1.1 Introducción al capítulo

Un SIG, según David Rhind en 1989, se define como: *"un sistema de hardware, software y procedimientos, diseñados para soportar la captura, el manejo, la manipulación, el análisis, el modelado y el despliegue de datos espacialmente referenciados, para la solución de los problemas complejos del manejo y planeamiento territorial"*. Los datos espacialmente referenciados son aquellos que pasan por un proceso de georreferenciación, o sea que localizan objetos físicos en determinadas coordenadas geográficas. ***[1]***

Durante el desarrollo de este trabajo se estará utilizando el término de SIG como un software que permite a los usuarios integrar, analizar y visualizar en un mapa la información geográfica referenciada asociada a un territorio. En este capítulo se analizan las aplicaciones SIG más utilizadas mundialmente y que se consideran más importantes de acuerdo con el objetivo de este trabajo. Se realizan comparaciones entre las características de SIG según las necesidades del proyecto SIG-Desktop para finalmente llegar a la conclusión de la tecnología más conveniente.

1.2 Aspectos a considerar en la búsqueda de la tecnología

A la hora de escoger la plataforma correcta hay que considerar la **licencia** del software porque se necesita que se puedan comercializar las personalizaciones implementadas. Algunas licencias de software libre son: la Licencia Pública General (GPL) donde el usuario puede usar un programa, modificarlo y distribuir las versiones modificadas pero siempre bajo esta misma licencia, o sea, siempre se tiene que entregar el código fuente; la Licencia Pública General Menor (LGPL) permite que sus sistemas puedan ser utilizados o enlazados con software propietario; la Distribución de Software de Berkeley (BSD) no impone ninguna restricción a los desarrolladores en lo referente a la utilización posterior del código y licencias de estos programas; La Licencia Pública de Mozilla (MPL) es una licencia de código abierto y software libre, pero deja abierto el camino a una posible reutilización comercial y no libre del software, o sea, solo se entrega el código fuente si el propietario lo desea. ***[2]***

Dentro de las políticas de la universidad está el no **entregar el código fuente** de los productos que se comercializan. En el proyecto se distribuyen las personalizaciones de Qgis y este es un software desarrollado bajo las normativas GPL. En el párrafo anterior se plantea la obligación de entregar el

código fuente de las aplicaciones distribuidas bajo la licencia GPL, lo cual va en contra de las normas de la universidad. Esta es una característica a tener en cuenta a la hora de seleccionar una plataforma alternativa para el proyecto.

En los últimos años se ha visto un gran desarrollo de los SIG libres, ejemplos de ellos ka-Map, Mapbender, MapFish, OpenLayers, Jump, Grass, Kosmo, GvSIG, Qgis y Mapwindows. Entre estos SIG libres se han seleccionado específicamente los de escritorio, ellos son: Jump, Grass, Kosmo, GvSIG, Qgis y Mapwindows. Viendo la interfaz gráfica de estos SIG se perciben comportamientos comunes. (Ver ilustraciones de la 1 a la 6)

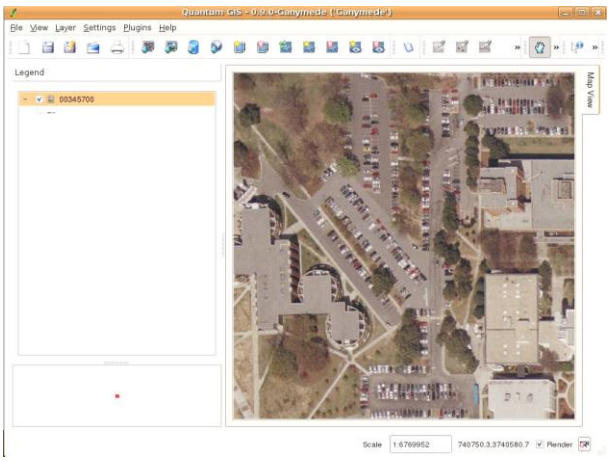

Ilustración 1: Interfaz de usuario de Qgis.

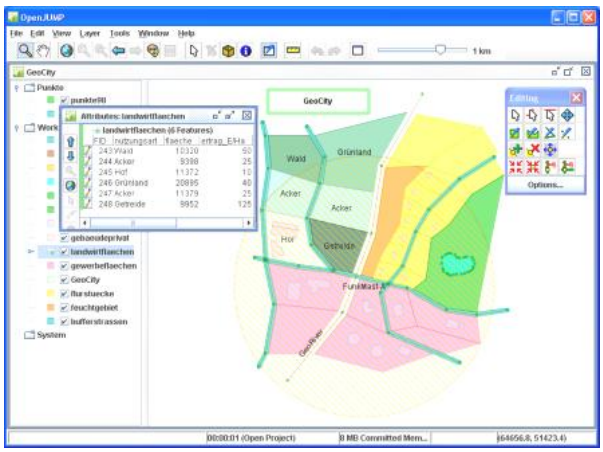

Ilustración 2: Interfaz de usuario de Jump

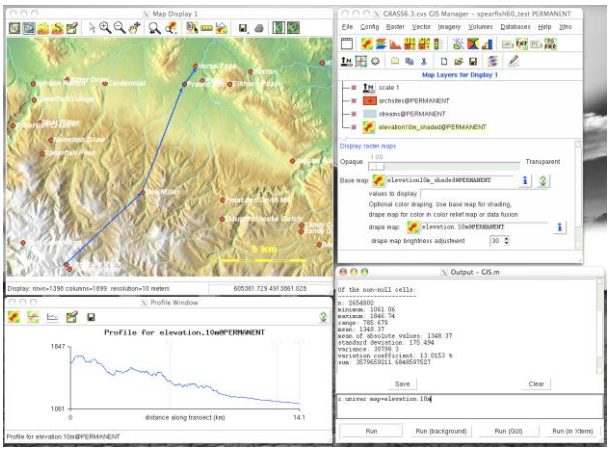

Ilustración 3: Interfaz de usuario de Grass

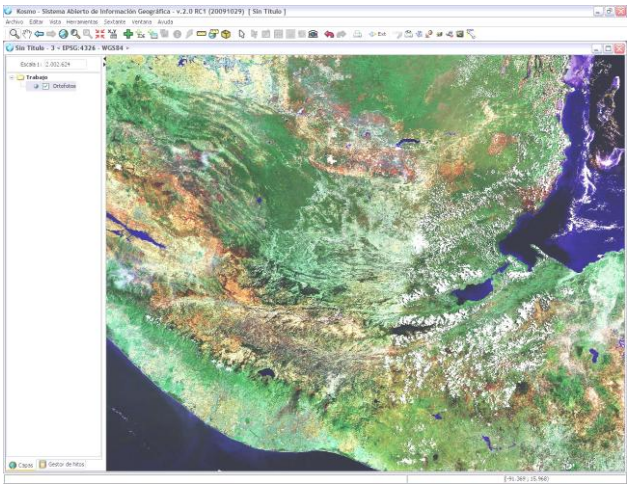

Ilustración 4: Interfaz de usuario de Kosmo

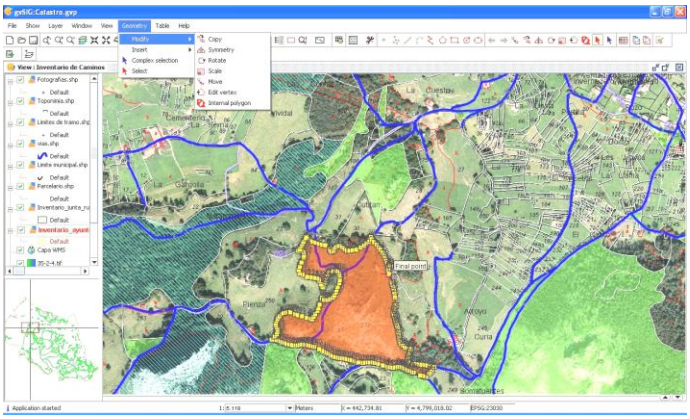

Ilustración 5: Interfaz de usuario de Gvsig

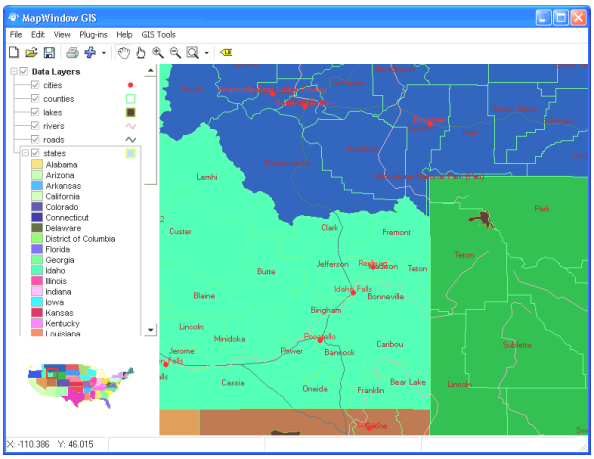

Ilustración 6: Interfaz de usuario de MapWindow GIS

Se puede observar que todas las interfaces tienen un menú de herramientas, una lista de capas y un mapa. Es más fácil personalizar aplicaciones si las funcionalidades comunes están agrupadas en componentes de software como objetos COM (Component Object Model). COM es una manera de implementar objetos neutrales que pueden ser usados en entornos distintos del que fueron creados. El objeto es responsable de su propia creación y destrucción. El método más usado en COM es la creación de sub-objetos a los que se delegan las llamadas a métodos, o sea la agregación. De esta forma se pueden llamar los métodos ya implementados desde cualquier aplicación. Se puede decir que el uso de tecnologías que empleen **objetos COM** mejoraría la productividad de los programadores en el proyecto. ***[3]***

1.2 Selección de una plataforma apropiada para el proyecto SIG-Desktop

Quantum GIS o QGis está desarrollado en C++, usando la biblioteca Qt para su interfaz de usuario. Es de código libre y multiplataforma. Permite manejar formatos ráster y vectoriales a través de las bibliotecas GDAL y OGR.

Jump fue programado en JAVA y puede ejecutarse en entornos distintos como Windows, Linux o Mac. Su arquitectura modular permite crear plugins. Con él es posible conectarse a servidores de cartografía WMS (Web Map Service) y las especificaciones están recogidas en OGC (Open Geoespatial Consortium). ***[4]***

Grass (Geographic Resources Analysis Support System) es utilizado para la gestión y análisis de datos geoespaciales, procesamiento de imágenes y gráficos, producción de mapas, modelado espacial y visualización. Está disponible principalmente para plataformas GNU/Linux, aunque existe un proyecto

denominado winGRASS GIS que ha portado el programa a versiones basadas en la tecnología .NET. ***[5]***

Kosmo ha sido implementado en el lenguaje de programación Java. Fue desarrollado a partir de JUMP y de una serie de bibliotecas de código libre de reconocido prestigio, entre ellas Geotools y Java Topology Suite (JTS). Es un SIG multiplataforma. ***[6]***

GvSIG Desktop se puede acceder a información vectorial, ráster y a servidores de mapas con especificaciones del OGC. Desarrollado en Java y funciona con los sistemas operativos Windows, Linux y Mac. Actualmente es impulsado por un conjunto de empresas, administraciones y universidades. ***[7]***

MapWindow 4 es una herramienta basada en Windows y Open Source, con el que se puede visualizar, gestionar, editar y analizar datos. Incluye plugins para tareas de geo-procesamiento, delineación de las cuencas, acceso a fuentes de datos en línea y una base de datos geográfica experimental. Es fácil de incorporar dentro de los productos basados en MS-Office como MS-Excel y MS-Access, así como programas basados en VB6, C++, C#, VB.NET, y Delphi, ya que utiliza tecnología de controles ActiveX. ***[8]***

A continuación se muestra la tabla 1 en la que se reflejan los aspectos a considerar en la búsqueda de la tecnología adecuada para el proyecto SIG-Desktop y se comparan los SIG mencionados.

Característica	QGis	Jump	Grass	Kosmo	GvSIG	MapWindow
Software libre	Sí	Sí	Sí	Sí	Sí	Sí
Comercializable según su licencia	Sí (GNU_GPL 2.0)	Sí (GNU_GPL 2.0)	Sí (GNU_GPL 2.0)	Sí (GNU_GPL 2.0)	Sí (GNU_GPL 2.0)	Sí (Mozilla Public License)
Entregar código fuente	Sí	Sí	Sí	Sí	Sí	No
Plataforma .NET	No	No	Sí	No	No	Sí
Uso de objetos COM	No	No	No	No	No	Sí

Tabla 1: Comparación de los SIG referenciados.

Los cinco primeros sistemas mencionados no cumplen con la totalidad de las variables que se habían definido. Sus licencias no cumplen las directivas de la universidad en cuanto a la entrega del código fuente y no utilizan objetos COM que facilite a los programadores la personalización de SIG. De ellos solo Grass y MapWindows GIS se basan en la plataforma .NET. Sin embargo MapWindows GIS es un sistema de software libre y su licencia, Mozilla Public License, le permite redistribuir sus versiones de forma comercializable y sin entregar su código fuente. Este SIG está implementado sobre el framework MapWindows, cuenta con una arquitectura basada en componentes que provee un objeto COM que facilita el trabajo del programador.

1.3 Caracterización del framework MapWindow

MapWinGIS es el nombre del control ActiveX del framework MapWindow y su componente básico. Fue implementado en C++. Este se puede agregar a un formulario en Visual Basic, Delphi, o en otros idiomas que soporte ActiveX. Se puede utilizar en diferentes entornos de desarrollo de Microsoft y con

los lenguajes de programación Visual Basic 6.0, Visual Basic. NET 2003, Visual Basic 2005, C #. NET 2003, C #. NET 2005 y Visual C++. También se ha utilizado en Microsoft Access, Excel y PowerPoint utilizando Visual Basic Applications (VBA), incluso puede utilizase MapWinGIS en los formularios desarrollados dentro de ESRI y ArcGIS con VBA. Su interfaz se ha tratado de optimizar para obtener un modelo totalmente funcional. Está compuesto por una Interfaz de Programación de Aplicaciones (API) que permite el acceso a objetos, funciones, propiedades y métodos para la visualización y manipulación de información geográfica en un mapa. ***[9]***

Algunas funciones de este componente ActiveX son:

- Abrir, crear, editar y guardar la imagen geo-referenciada directamente.
- Ver, marcar, colorear y dar relieve a los datos en el mapa.
- Realizar consultas sobre los datos espaciales.
- Buscar características con atributos específicos.
- Editar los datos espaciales y ver inmediatamente los cambios en el mapa.
- Interactuar con los datos a través del mapa.
- Construir Redes Trianguladas Irregulares (TIN).

1.4 Conclusiones parciales

Se consideró la opción de utilizar el framework MapWindows y no el SIG MapWindow 4 porque se desea implementar una arquitectura propia para el proyecto, que agrupe las funcionalidades comunes que se muestran en las fotos de los SIG analizados anteriormente. Se puede confeccionar la herramienta, elaborando controles de usuarios que se encuentren en el cuadro de herramientas de un IDE de desarrollo. Esos controles tendrán implementadas un conjunto de acciones o funcionalidades de visualización de información geográfica que se utilizan en todos los SIG.

Capítulo 2: Características de la solución.

2.1 Introducción al capítulo

En éste capítulo se describe el sistema propuesto. Para poder comprender el marco conceptual y por la inexistencia de un negocio, se determina desarrollar un Modelo de Dominio. Se enumeran los requerimientos funcionales y no funcionales. Se presenta el Diagrama de CUS y se describen dichos casos de uso.

2.2 Modelo de dominio

La ilustración 7 expone los conceptos que se manejan en las aplicaciones SIG de escritorio. El Modelo de Dominio ayuda a comprender los conceptos que utilizan los usuarios y con los que trabaja la aplicación. ***[10]*** Para ello se han estudiado las interfaces de usuarios de un conjunto de aplicaciones, entre las cuales destacan las citadas en la sección 1.2 del capítulo anterior.

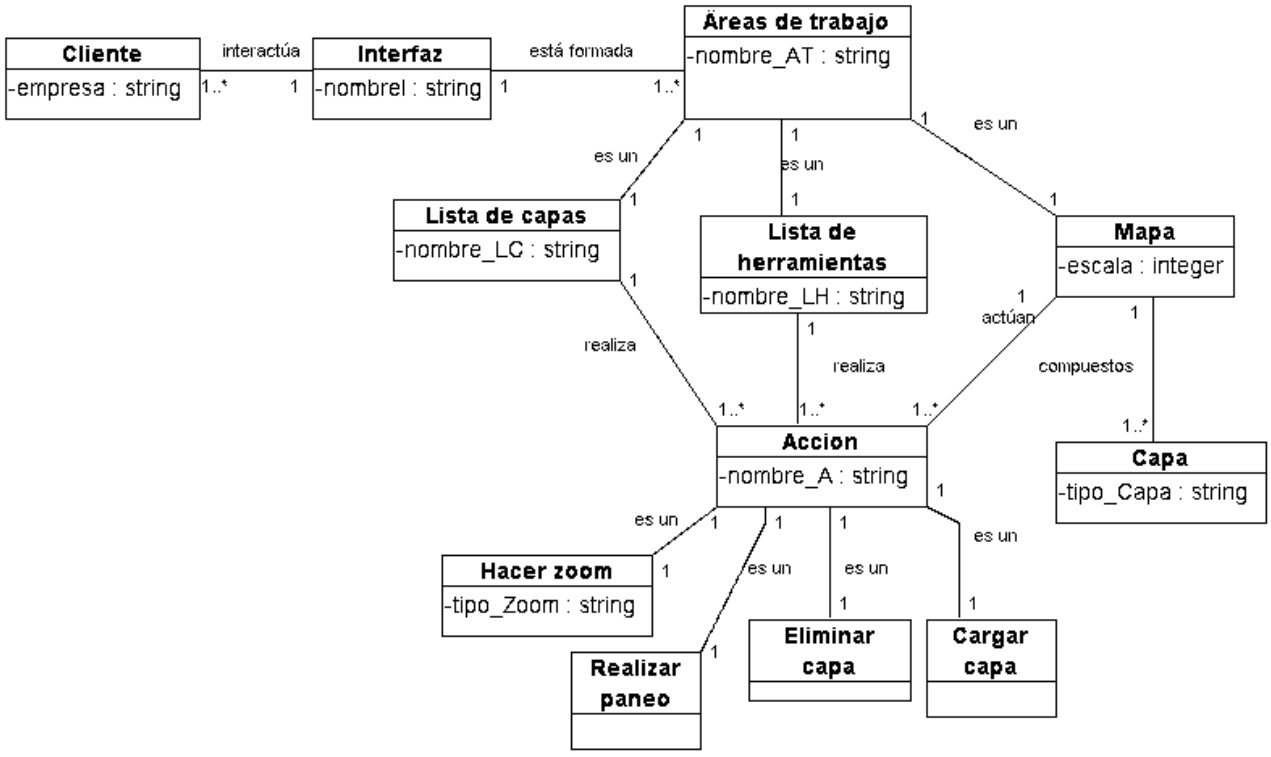

Ilustración 7: Modelo de Dominio de la aplicación.

Un **cliente** de un proyecto que implementa SIG de escritorio interactúa con la **interfaz** de usuario de la

aplicación SIG. La interfaz está formada por **áreas de trabajo que pueden ser:** un **mapa**, una **lista de capas** y/o una **lista de herramientas**. Dichas áreas pudieran ser controles de usuarios. La lista de capas y la de herramientas realizan **acciones** que actúan sobre el **mapa**. **Las acciones** pueden ser, hacer **zoom**, realizar **paneo**, **cargar** o **eliminar capa**, entre otras. A su vez, un **mapa** está compuesto por una o varias **capas**.

2.3 Requisitos de la herramienta

Los requisitos se clasifican en funcionales y no funcionales. Los funcionales son las capacidades o condiciones que el sistema debe cumplir. ***[11]*** Ellos se identificaron realizando un análisis de los requerimientos de visualización de información geográfica más usados en los SIG. Los no funcionales son las propiedades o cualidades que el sistema debe tener ***[12]*** Estos últimos se tienen en cuenta con el objetivo de que la herramienta sea usable, rápida, confiable, agradable y otros.

2.3.1 Requisitos Funcionales

Los requisitos funcionales deseados son aquellos que se consideren básicos a la hora de visualizar información en los SIG. En la tabla 2 se listan los requerimientos seleccionados para la herramienta.

Clave	**Descripción**
RF1	Cargar capas a la aplicación.
RF2	Acercar el mapa.
RF3	Alejar el mapa.
RF4	Centrar el mapa.
RF5	Mover el mapa.
RF6	Quitar una capa de la aplicación.
RF7	Cambiar el nombre de una capa.
RF8	Ocultar la información de una capa y volver a mostrarla si se desea.

RF9	Seleccionar un objeto del mapa y ver la información del mismo.
RF10	Ver las coordenadas de una posición del mapa.

Tabla 2: Requisitos de la herramienta.

2.3.2 Requisitos No Funcionales

- **Usabilidad**: El sistema podrá ser usado por personas con conocimientos básicos en el manejo de SIG. Su uso será intuitivo y agradable.
- **Eficiencia**: Se escribirá un código óptimo para mejorar el tiempo de respuesta y la velocidad de procesamiento, aunque estos factores también están dados por la cantidad de información que tenga que procesar la aplicación.
- **Rendimiento**: El tiempo de respuesta para cargar una capa en la pantalla y realizar acciones sobre el mapa será menor que 3 segundos en dependencia de la cantidad de información que tenga que actualizar.
- **Apariencia**: El diseño de la interfaz debe ser sencillo, donde no es necesario mucho entrenamiento para utilizarlo. Adema, debe estar bien organizado de modo que el usuario se sienta familiarizado con la herramienta.
- **Hardware**: Se requiere al menos 1 GB de RAM y 40 GB de disco duro.
- **Software**: Se debe tener instalado el sistema operativo Windows.

2.4 Diagrama de Casos de Uso del Sistema (CUS)

En la ilustración 8 se muestra que el actor del sistema es el usuario. Este es el encargado de ejecutar los Casos de Uso y será quien utilice las funcionalidades implementadas. La descripción del usuario y la totalidad de los CUS se encuentran en el documento “Modelo del Sistema de la herramienta basad en MapWindows” que fue generado para el proyecto SIG-Desktop.

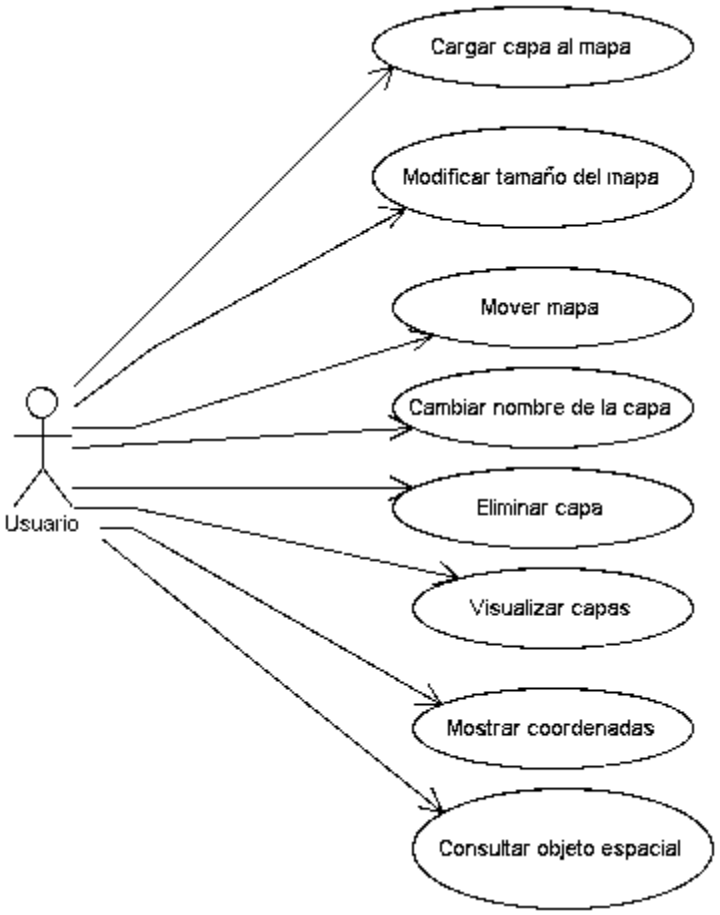

Ilustración 8: Diagrama de CUS de la aplicación.

2.4.1 Descripción del CUS: Cargar capa al mapa

Caso de Uso 1:	Cargar capa al mapa
Actores:	Usuario
Resumen:	El CUS inicia cuando el usuario selecciona la opción de “Cargar capa”, permite escoger una capa y finaliza cuando se visualiza la capa en el mapa.
Precondiciones:	Debe estar ejecutándose correctamente el programa.
Referencias	RF1

Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El usuario selecciona la opción “Cargar capa”. (ver ilustración 9)	2- El sistema abre una ventana de búsqueda de ficheros.
3- El usuario busca y selecciona el fichero deseado, luego presiona el botón “Aceptar”. (ver ilustración 10)	4- El sistema carga la capa y la muestra en el mapa y en una lista de capas. (ver ilustración 11)

Prototipo de Interfaz

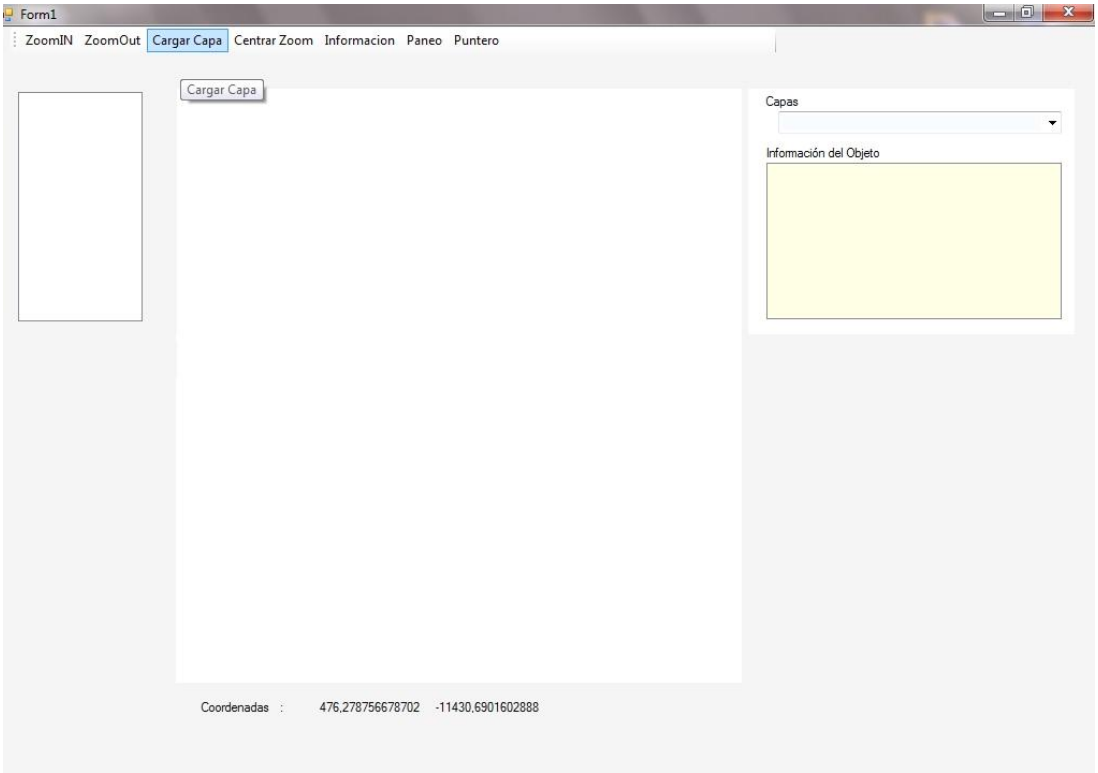

Ilustración 9: Vista inicial de la aplicación.

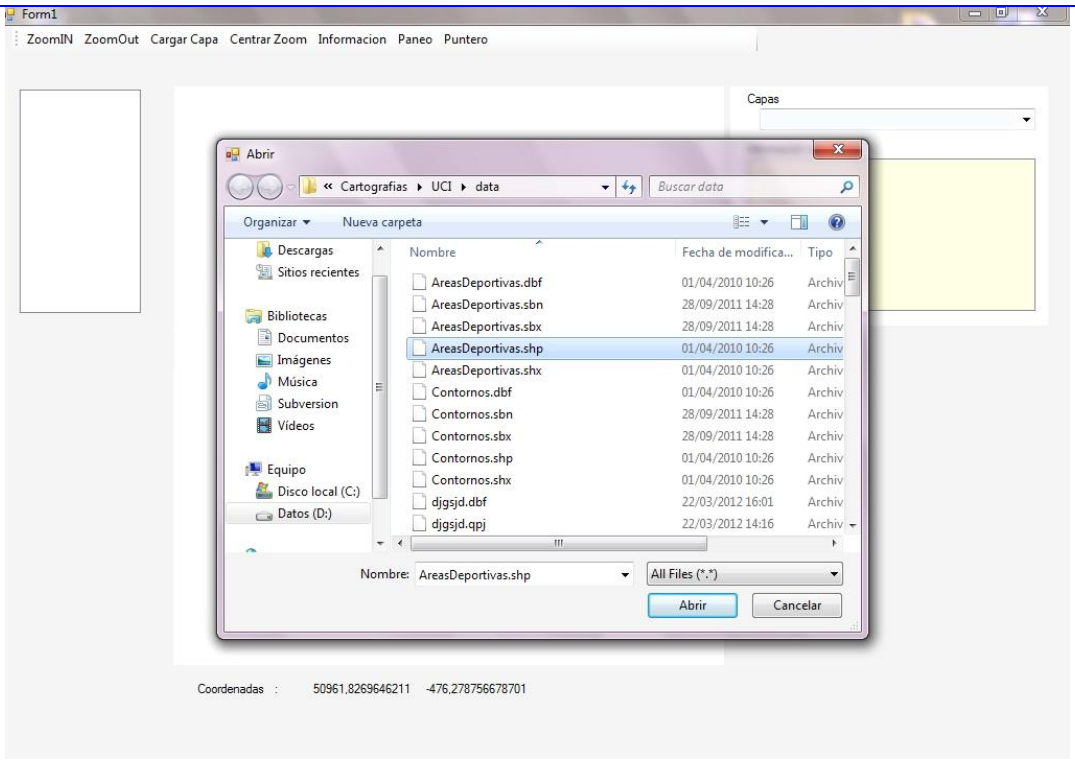

Ilustración 10: Cargando una capa en la aplicación.

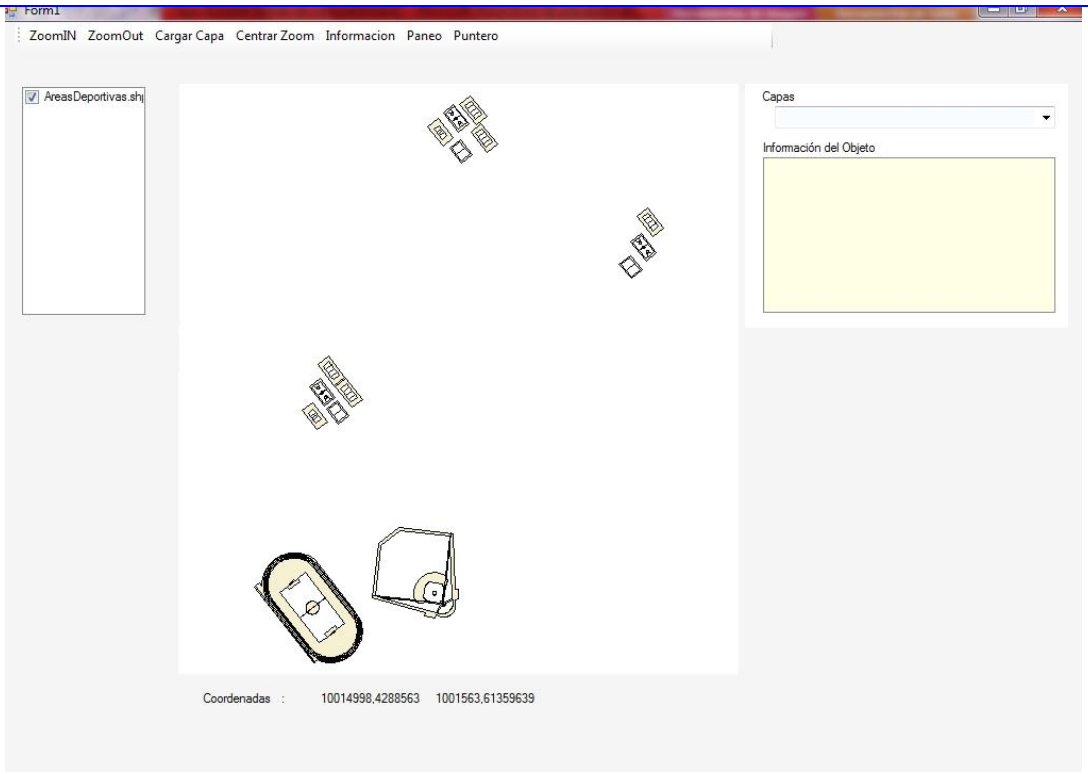

Ilustración 11: Vista de una capa cargada en la aplicación.

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3- El usuario selecciona el botón “Cancelar”.	4- El sistema cierra la venta de búsqueda de ficheros.
3- El usuario cierra la aplicación.	4- El sistema cierra la venta de búsqueda de ficheros.
Poscondiciones	Se ha cargado una capa al sistema.

Tabla 3: Descripción del CUS “Cargar capa”.

2.4.2 Descripción del CUS: Visualizar capas

Caso de Uso 5:	Visualizar capas

Actores:	Usuario
Resumen:	El CUS inicia cuando el usuario activa o desactiva una capa en la lista de capas y finaliza cuando se ha cambiado la visualización de dicha capa en el mapa.
Precondiciones:	Debe haberse cargado al menos una capa.
Referencias	RF8
Prioridad	Critico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El usuario activa o desactiva una capa de la lista de capas cargadas.	2- El sistema Si la capa estaba desactivada entonces la activa en la lista de capas y la visualiza en el mapa. (ver ilustración 12) Si la capa estaba activada entonces la desactiva en la lista de capas y la no la visualiza en el mapa. (ver ilustración 13)
Prototipo de Interfaz	

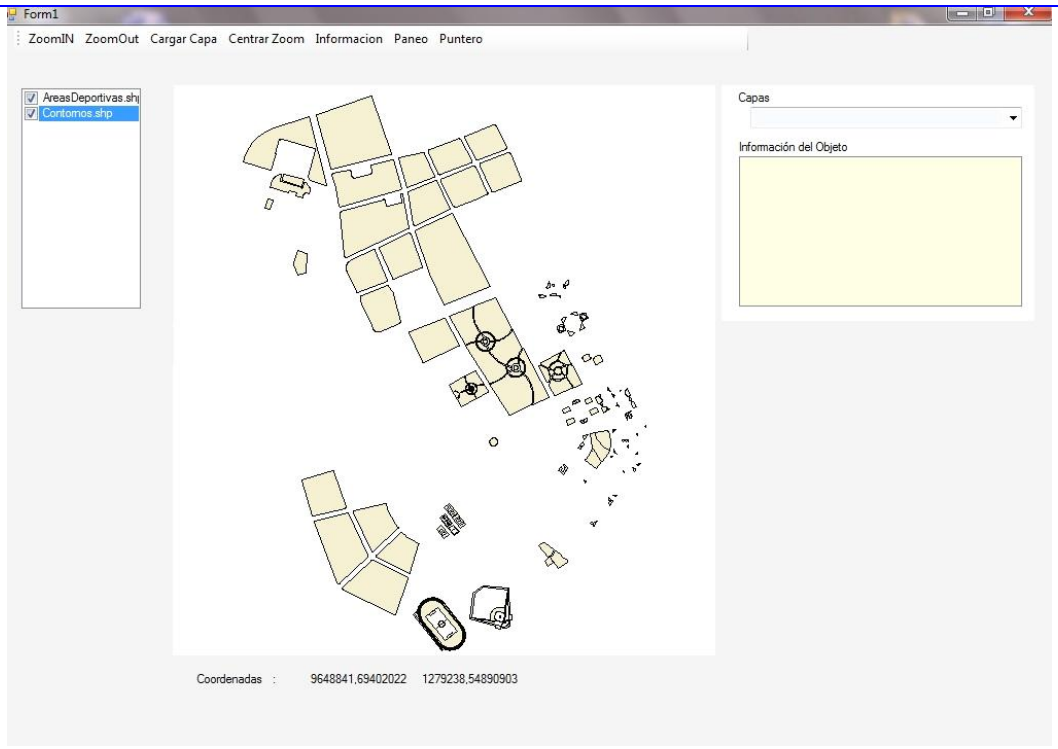

Ilustración 12: Activando una capa del mapa.

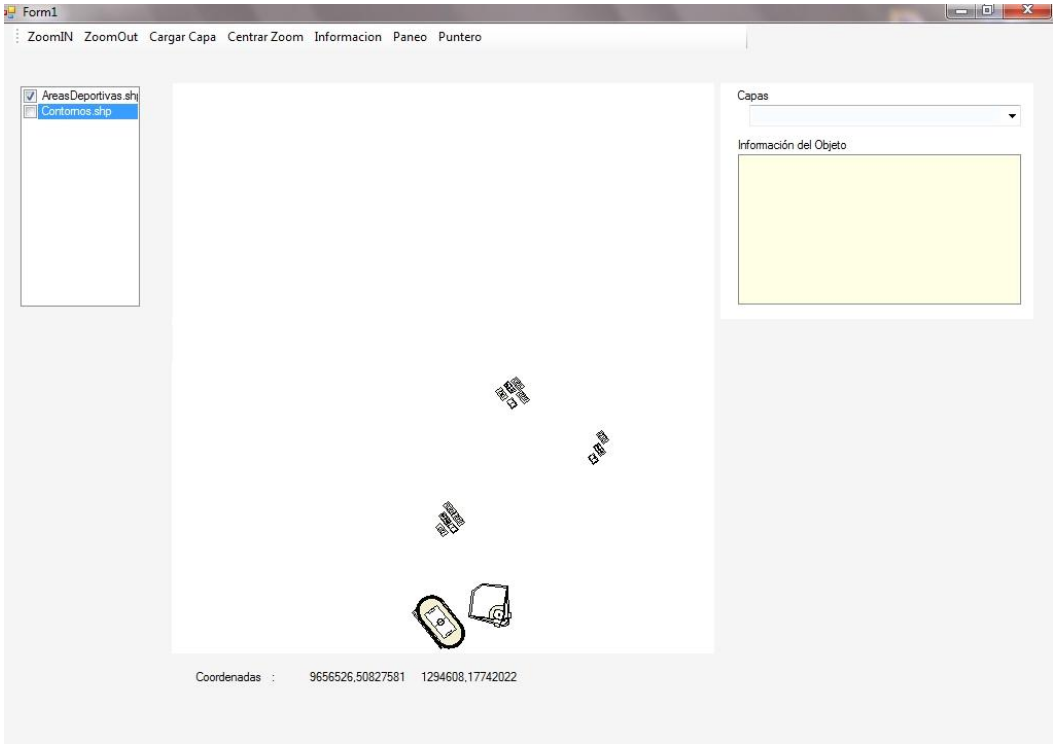

Ilustración 13: Desactivando una capa del mapa.	
Poscondiciones	Se ha ocultado o visualizado una capa.

Tabla 4: Descripción de CUS "Visualizar capa".

2.4.3 Descripción del CUS: Modificar tamaño del mapa

Caso de Uso 6:	Modificar tamaño del mapa
Actores:	Usuario
Resumen:	El CUS inicia cuando el usuario presiona alguna de las opciones: "Acercar zoom", "Alejar zoom" o "Centrar zoom" y finaliza cuando se ha modificado el tamaño de las capas.
Precondiciones:	Debe haberse cargado la capa.
Referencias	RF2, RF3 y RF4
Prioridad	Critico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El actor selecciona una de las opciones "Acercar zoom", "Alejar zoom" o "Centrar zoom".	2- Si selecciona: • Acercar zoom ir a "Sección Acercar Zoom". • Alejar zoom ir a "Sección Alejar Zoom". • Centrar zoom ir a "Sección Centrar Zoom".

Sección “Acercar Zoom”	
Acción del Actor	**Respuesta del Sistema**
1- El usuario selecciona la opción “Acercar zoom” y hace clic en una posición del mapa.	2- El sistema realiza la acción de acercar la posición seleccionada en el mapa. (ver ilustración 14)
Prototipo de Interfaz	

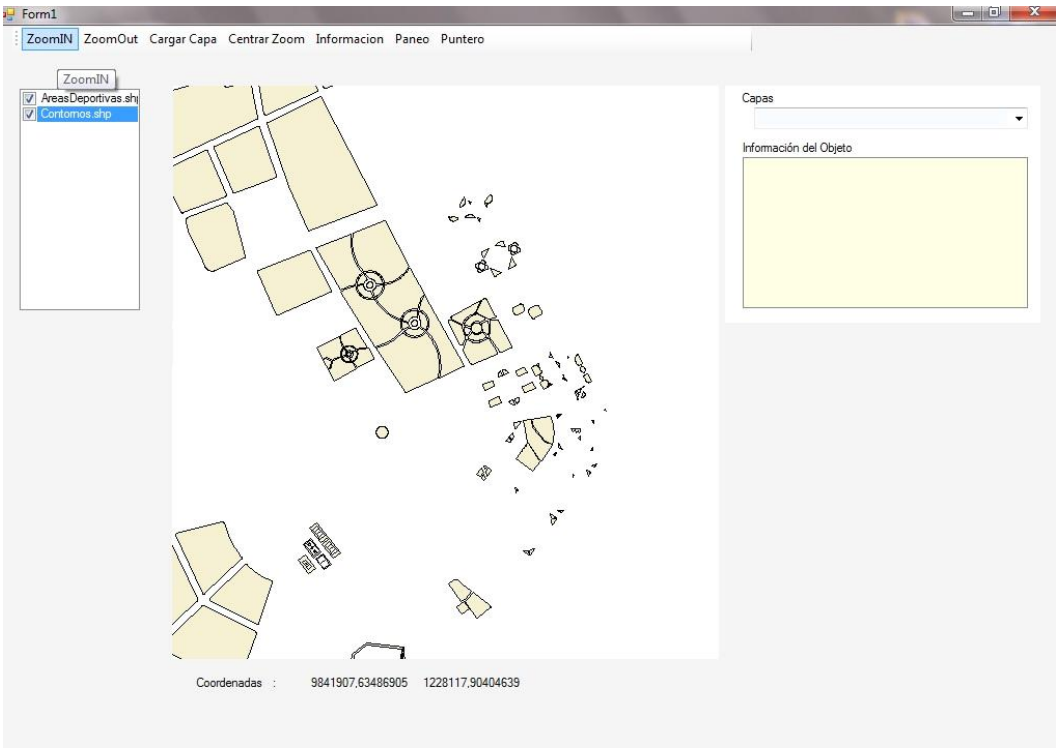

Ilustración 14: Acercando el zoom en la aplicación.

Flujos Alternos	
1- El usuario selecciona la opción Acercar zoom”, hace clic en una posición del mapa y arrastra el ratón formando un rectángulo.	2- El sistema realiza la acción de acercar el rectángulo seleccionado en el mapa.
Sección “Alejar Zoom”	

Acción del Actor	Respuesta del Sistema
1- El usuario selecciona la opción “Alejar zoom” y hace clic en una posición del mapa.	2- El sistema realiza la acción de alejar la posición seleccionada en el mapa. (ver ilustración 15)

Prototipo de Interfaz

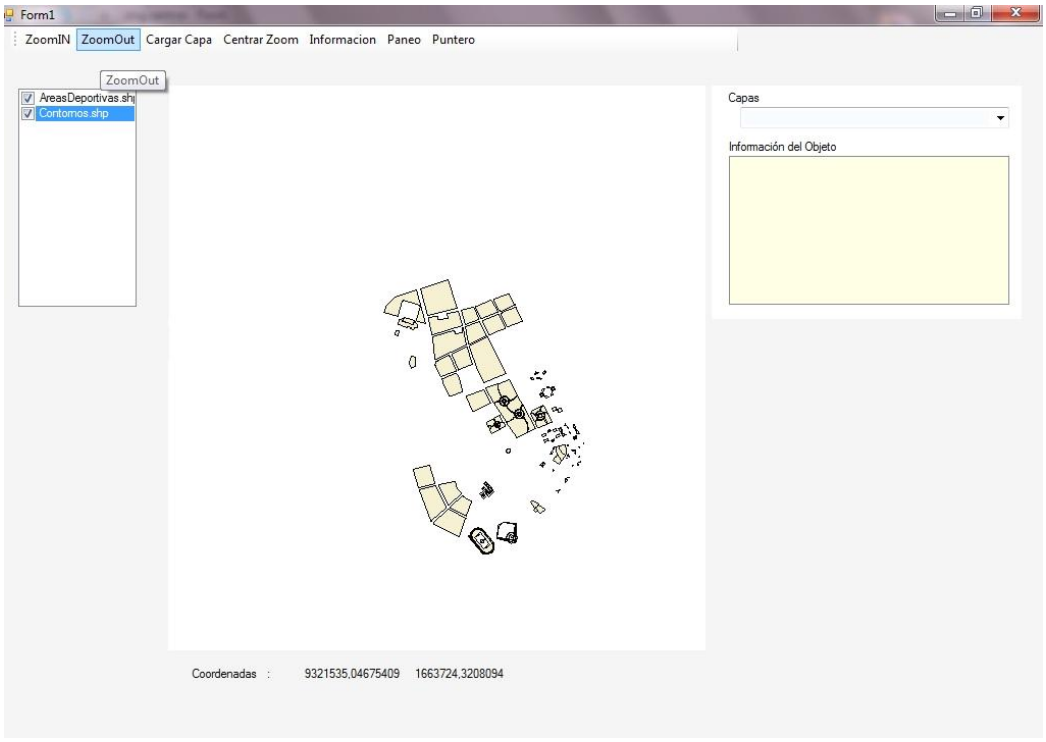

Ilustración 15: Alejando el zoom en la aplicación.

Flujos Alternos	
1- El usuario selecciona la opción “Alejar zoom”, hace clic en una posición del mapa y arrastra el ratón formando un rectángulo.	2- El sistema realiza la acción de alejar el rectángulo seleccionado en el mapa.

Sección “Centrar Zoom”	
Acción del Actor	**Respuesta del Sistema**

1- El usuario selecciona la opción “Centrar zoom”.	2- El sistema coloca el mapa en la posición y el tamaño inicial. (ver ilustración 16)

Prototipo de Interfaz

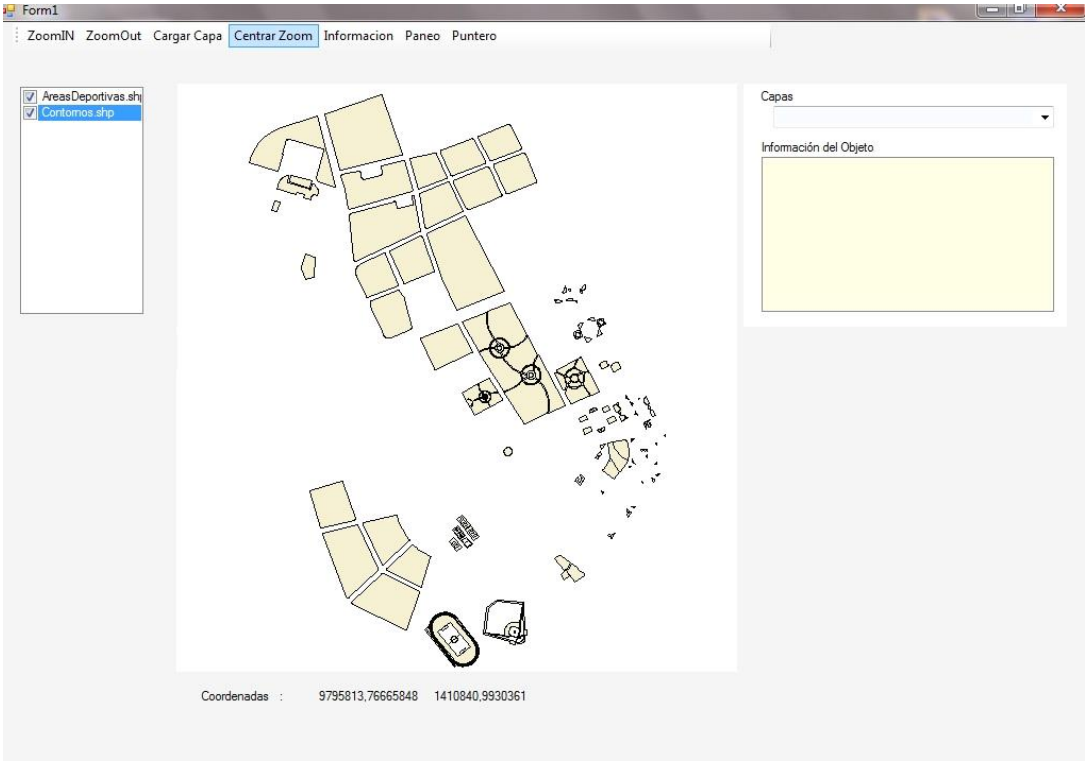

Ilustración 16: Centrando el zoom en la aplicación.

Poscondiciones	Se ha modificado el tamaño del mapa.

Tabla 5: Descripción del CUS “Modificar tamaño del mapa”.

2.5 Conclusiones parciales

En este capítulo se plantearon los conceptos que guardan relación con el dominio del problema, dentro de ellos se introdujo el concepto de controles de usuarios. Se realizó además un levantamiento de requisitos proponiendo algunas funcionalidades esenciales para los SIG, que podrán incrementarse en próximas versiones. Finalmente se expusieron los artefactos que permitieron describir las funcionalidades que se desean implementar, creando las condiciones para comenzar el diseño del sistema.

Capítulo 3: Diseño e implementación de la solución

3.1 Introducción al capítulo

En éste capítulo se exponen los artefactos que permiten modelar el sistema. Se define el estilo arquitectónico a utilizar y se realiza un modelado de la arquitectura teniendo en cuenta algunos patrones de diseño. Luego se realiza la implementación del sistema usando como lenguaje de programación Visual Basic y a Visual Studio como entorno de desarrollo integrado (IDE).

Fue seleccionado Visual Basic con el fin de explotar todas las potencialidades del framework MapWindows, especificamente el uso de VBA. Con esta oportunidad se puede utilizar MapWinGIS en los formularios desarrollados dentro de ArcGIS que es el SIG de referencia a nivel mundial y a través de macros puede usarse también en Microsoft Access, Excel y PowerPoint. También se valora que Visual Basic es el lenguaje de programación que se seleccionó para implementar el SIG MapWindow. Algunas de las ventajas que se destacan para Visual Basic son:

- Integra el diseño e implementación de formularios de Windows.
- Tiene acceso casi total a la API de Windows, incluidas librerías actuales.
- Es fácilmente extensible mediante librerías DLL y componentes ActiveX de otros lenguajes.
- Existe una versión, Visual Basic para Aplicaciones (VBA), integrada en las aplicaciones de Microsoft Office, que permite programar macros para extender y automatizar funcionalidades en documentos, hojas de cálculo y bases de datos. ***[13]***

Visual Studio soporta varios lenguajes de programación, dentro de ellos Visual Basic. Es un punto de referencia para la productividad de los programadores por sus amplias ventajas, por ejemplo cuenta con un cuadro de herramientas, un depurador y una ventana de tareas comunes, reduciendo las dificultades de aprendizaje y garantizado elegir el lenguaje más apropiado. Además, su función para completar instrucciones y la comprobación automática de errores de sintaxis, informan a los programadores cuando el código es incorrecto. También está el explorador de soluciones que ayudan a reutilizar fácilmente el código a través de diferentes proyectos. ***[14]***

3.2 Arquitectura propuesta

La arquitectura de un software indica la estructura, el funcionamiento e interacción entre las partes del

software. En el libro "An introduction to Software Architecture", se define que la Arquitectura es un nivel de diseño que se centra en aspectos "más allá de los algoritmos y estructuras de datos de la computación; el diseño y especificación de la estructura global del sistema es un nuevo tipo de problema". En la ilustración 17 se muestra el diagrama de arquitectura del sistema.

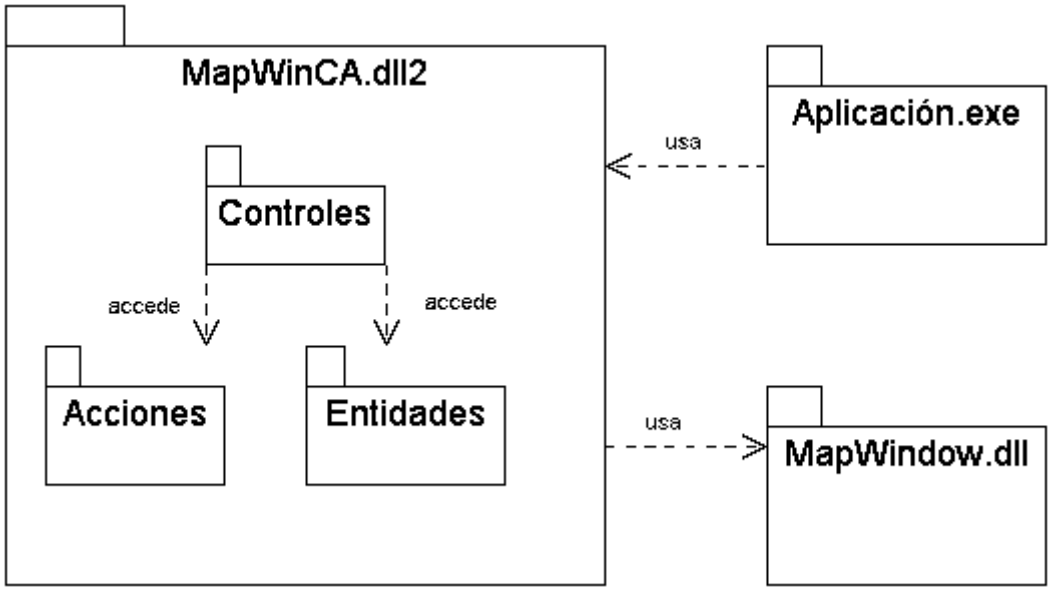

Ilustración 17: Diagrama de Paquetes de la Arquitectura de la aplicación

Aplicación se refiere a la herramienta de visualización de información geográfica que va a tener acceso **MapWinCA.dll** que contiene las interfaces de los controles de usuarios en el paquete de **Controles**, estos últimos tendrán acceso al de **Acciones** y **Entidades.** En el paquete de Acciones es donde se implementan las funcionalidades del sistema y se gestiona el mapa que estará en la interfaz de la aplicación. En el de Entidades solo se encuentra una clase Capa que almacena la posición y el nombre de una capa.

Se utiliza una **arquitectura Orientada a Objetos** porque los objetos y sus interacciones son el centro del diseño, basándose en la herencia, el polimorfismo y el encapsulamiento. Otro estilo arquitectónico empleado es el **basado en componentes** donde las interfaces y sus interacciones son el centro de incumbencias. Los componentes proveen funcionalidades y propiedades que pueden ser descubiertas y utilizadas en tiempo de ejecución. Con este estilo una interfaz puede ser implementada por múltiples componentes como es el caso de la herramienta que se desea.

La idea general de la arquitectura propuesta es que las componentes principales del SIG estén agrupadas en la librería de controles de usuarios MapWinCA.dll. Así, cualquier aplicación SIG que se quiera crear basada en MapWindows, puede conseguirse fácilmente a partir de una orquestación de

estos controles. Por lo tanto, en el resto del capítulo vamos a presentar el diseño de clases propuesto para los paquetes Controles y Acciones concebidos en la librería de controles (ver Ilustraciones 18 y 19).

3.3 Paquete de Interfaces de Usuario

En el diseño del paquete de Controles se definen los controles de usuario que han sido concebidas para satisfacer los RF. Ellos están relacionadas mediante una jerarquía de clases cuya clase base es ControlAxMap y esta hereda a su vez de la clase UserControl del framework .NET. El uso de la herencia permite la especialización-generalización de determinados conceptos, ahorrando código y tiempo, pues la clase padre ha sido verificada con anterioridad y la clase derivada hereda el comportamiento y los atributos de la clase base. En este caso ControlAxMap tendrá un atributo mapa que será heredado y usado por todos sus hijos.

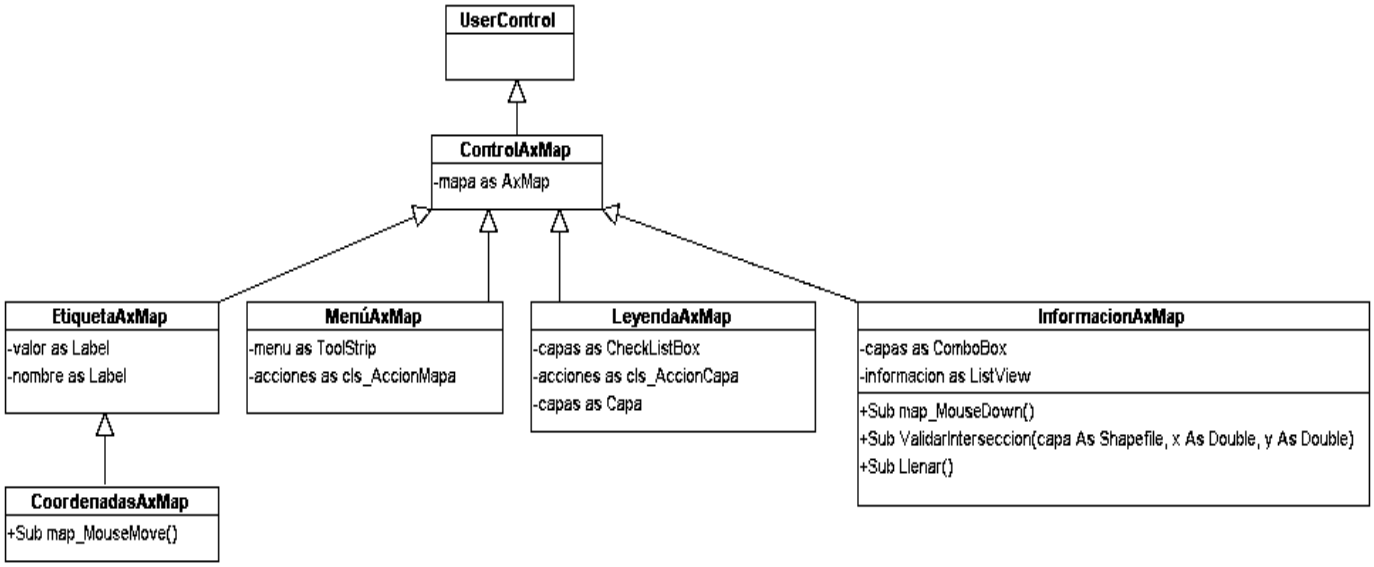

Ilustración 18: Diagrama de clases del paquete de Controles de la aplicación

Para satisfacer el RF10 se define un control Etiqueta, del cual hereda el control CoordenadasAxMap. Toda etiqueta se visualiza en el sistema de igual forma “Propiedad:Valor” y este aspecto se incluye en la clase Etiqueta mediante el diseño de la interfaz. Sin embargo, tanto el nombre de la propiedad como el valor dependen del tipo de control específico. Por tanto, se define la clase etiqueta como abstracta y se implementa la propiedad Nombre y los métodos que calculan el valor en el control hijo. El control InformacionAxMap satisface el RF9, este lista todas las capas de modo que se pueda seleccionar una y un objeto de ella para ver su información.

En el control MenuAxMap se satisfacen los RF del 1 el 5, todos ellos responden a acciones específicas sobre el mapa y se visualizan en el sistema con una barra de menú. LeyendaAxMap satisface los RF del 6 al 8 que son acciones sobre cada capa del mapa, aquí es donde el sistema muestra la lista de las capas que se han cargado en la aplicación. MenuAxMap y LeyendaAxMap tendrán una lista de acciones, haciendo uso del **patrón Delegación** que es la abstracción que da soporte a la composición o agregación y es de los patrones fundamentales, no entra dentro de las categorías GRASP (General Responsibility Assignment Software Patterns), ni GoF (Gang of Four). ***[15]***

3.4 Paquete de Acciones de los controles propuestos

En este paquete se encuentran las clases que implementan las acciones de los controles anteriores. Todas las acciones tendrán un nombre. Las que se ejecutan sobre un mapa heredan directamente de cls_AccionMapa y también hereda de ella la clase cls_AccionCapa. Las especificaciones de esta última, además del mapa usan la capa sobre la que se ejecutará la acción.

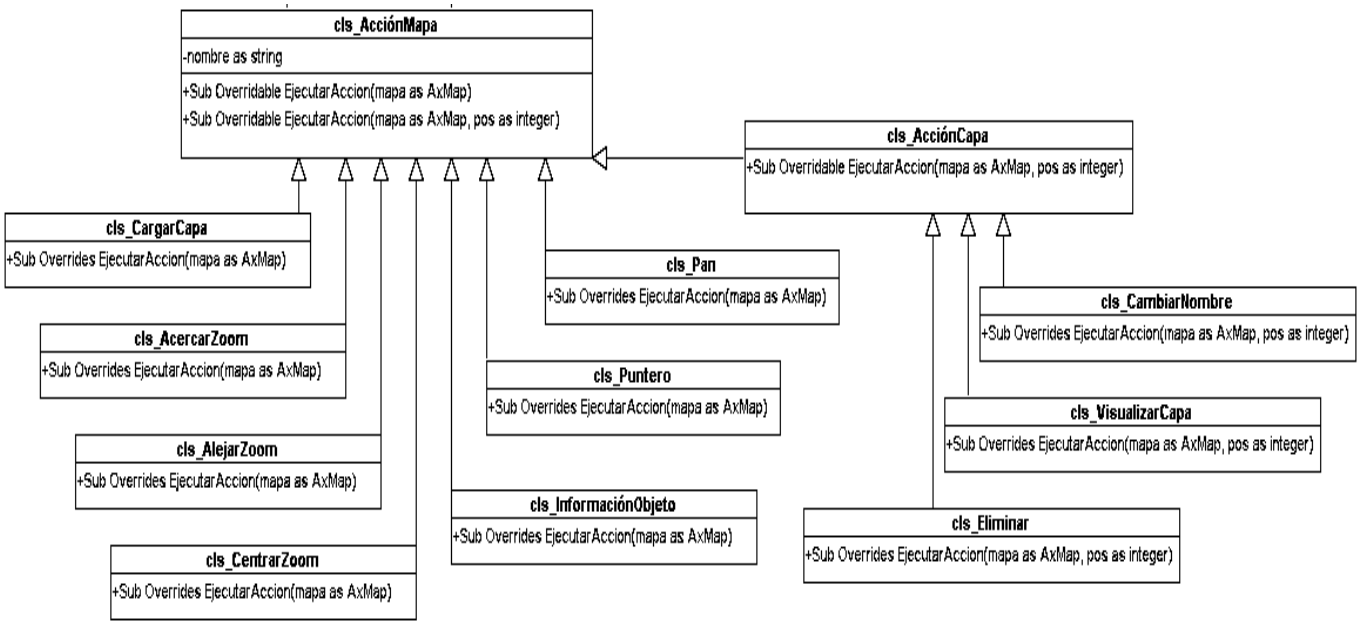

Ilustración 19: Diagrama de clases del paquete de Acciones de la aplicación.

Como se puede observar en la Ilustración 10, en el diseño de clases se utiliza el **patrón Estrategia** que permite mantener un conjunto de algoritmos de los cuales se puede elegir aquel que conviene ejecutar e intercambiarlo dinámicamente. Este es un patrón de tipo GoF., esto lo hace creando una superclase que almacena el comportamiento común y que servirá de paso hacia las clases concretas. ***[15]*** En este caso se hace referencia a la clase cls_AccionMapa y la acción común es el método

EjecutarAccion() que se le pasa como parámetro en algunos casos el mapa sobre el que se ejecutará la acción y en otros se le pasa además la posición de la capa.

3.5 Interacción entre objetos

Los diagramas de secuencia muestran la interacción de un conjunto de objetos en una aplicación a través del tiempo y contiene detalles de la implementación. ***[16]*** A continuación se presentan diagramas de secuencia reflejando clases, métodos y mensajes intercambiados entre los objetos de la herramienta. En los anexos se pueden encontrar los diagramas de clases del diseño de cada CUS.

3.5.1 Cargar capa en el menú

La ilustración 20 muestra el comportamiento cuando se selecciona la acción de cargar capa. Primeramente se hace una llamada al método EjecutarAccion de la clase cls_CargarCapa, en ese método se crea una ventana de cargar fichero, se abre el fichero seleccionado, se le adiciona al mapa y se inserta en la lista de capas. A continuación el diagrama de secuencia del CUS “Cargar capa”:

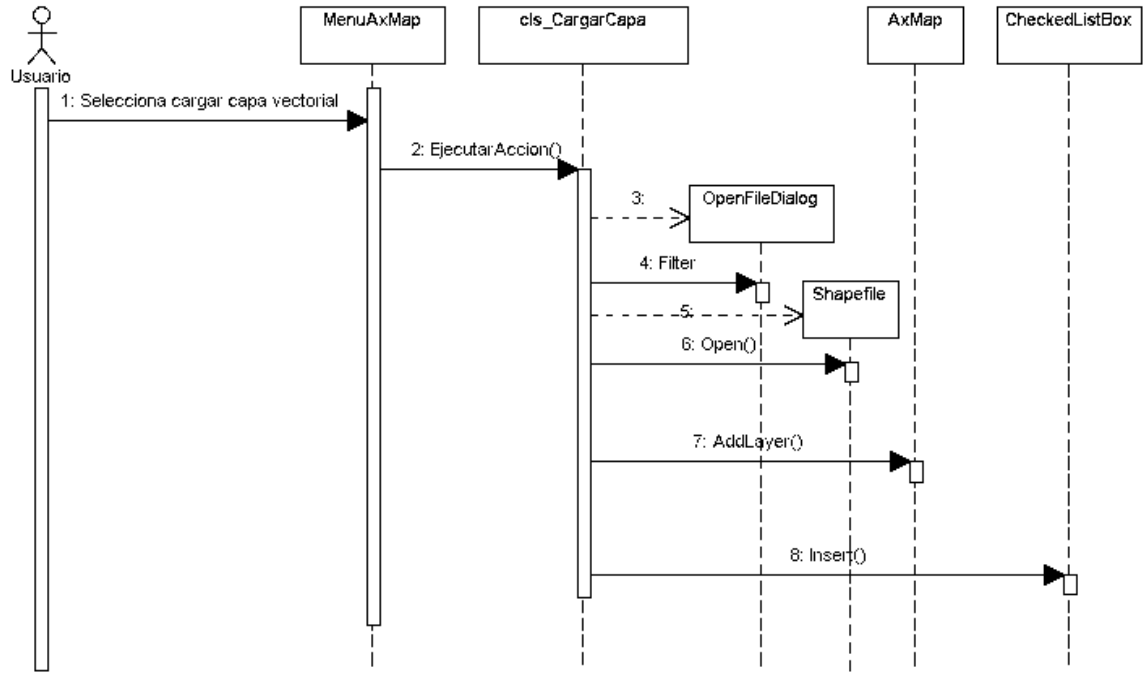

Ilustración 20: Diagrama de Secuencia del CUS "Cargar capa vectorial"

3.5.2 Visualizar capa en la leyenda

La ilustración 21 muestra el comportamiento cuando se lanza el evento que chequea un elemento de la

lista de capas del control Leyenda. Se hace una llamada al método EjecutarAccion de la clase cls_Visible y en ese método se actualiza la visibilidad de la capa en el mapa. A continuación el diagrama de secuencia del CUS “Visualizar capa”:

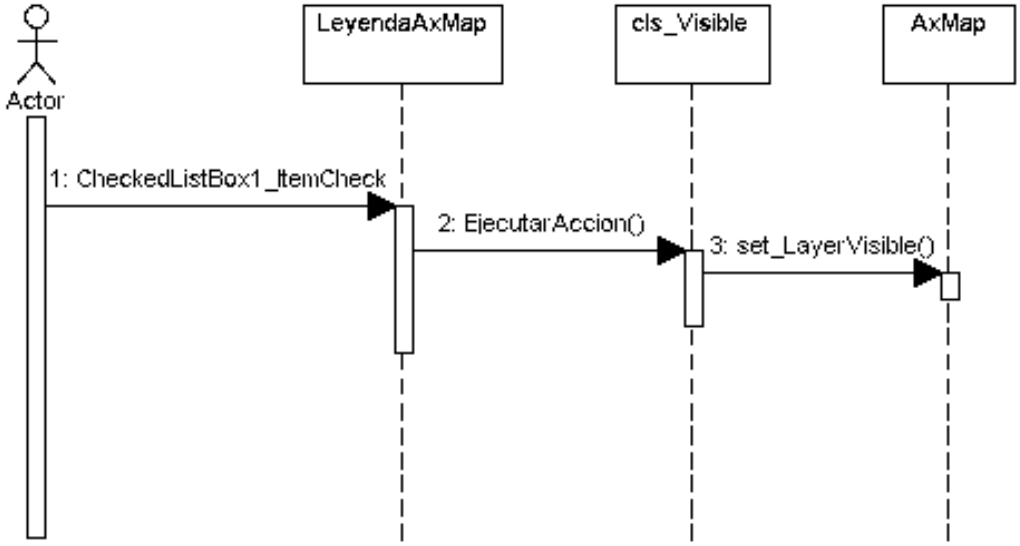

Ilustración 21: Diagrama de Secuencia del CUS "Visualizar capa en la leyenda"

3.5.3 Modificar zoom en el menú

La ilustración 22 muestra el comportamiento cuando se selecciona la acción de acercar una posición del mapa. Se hace una llamada al método EjecutarAccion de la clase cls_AcercarZoom, en ese método se cambia el modo del cursor del mapa, esto se realiza tomando el valor cmZoomIn del enum tkCursorMode. A continuación el diagrama de secuencia de una de las secciones del CUS “Modificar tamaño de las capas”:

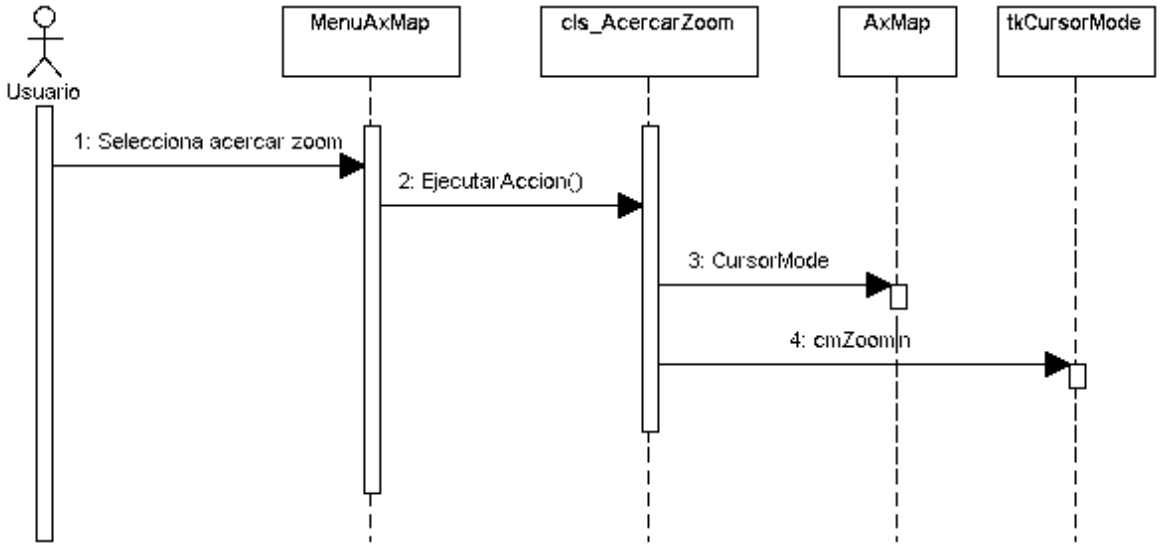

Ilustración 22: Diagrama de Secuencia de la funcionalidad "Acercar zoom"

3.6 Modelo de despliegue

El modelo físico o de despliegue es un mapa de la instalación física del sistema. Un diagrama de despliegue muestra dónde se ubicarán los componentes, en qué servidores, máquinas o hardware. ***[17]*** Al terminar la implementación de la aplicación debe ser instalada en una máquina que tenga como sistema operativo Windows. Ver ilustración 23.

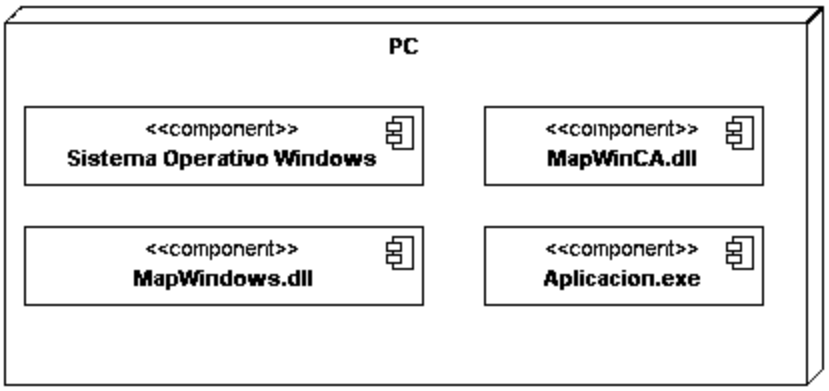

Ilustración 23: Modelo de Despliegue de la aplicación.

3.7 Conclusiones parciales

En este capítulo se han expuesto los artefactos que permiten modelar la herramienta que se desea obtener. Se definió la arquitectura, se realizó el diseño y la implementación del software. En el próximo capítulo se abordan los aspectos referentes a las pruebas y validaciones del sistema.

Capítulo 4: Validación de la solución.

4.1 Introducción al capítulo

La validación de un sistema se utiliza para evaluar al software durante o al final de su desarrollo, así se determina si satisface los requisitos iniciales y responde a las interrogantes: ¿Es esto lo que el cliente quiere? o ¿Estamos construyendo el producto correcto? En este capítulo se van a confeccionar los casos de prueba y se realizan las pruebas necesarias al software para comprobar que fue implementado lo que se analizó con anterioridad.

4.2 Métodos de pruebas para la herramienta

Casos de prueba para los requisitos funcionales

Los diseños de casos de prueba es una técnica que se utiliza para evaluar cada funcionalidad del sistema partiendo de la descripción de los CUS. Esta técnica se corresponde con los tipos de pruebas de software denominados **pruebas de caja negra** que realizan el estudio de un sistema desde el punto de vista de las entradas que recibe y las respuestas que produce, sin tener en cuenta su funcionamiento interno. Las pruebas de caja negra van enfocadas en saber qué es lo que hace el software, sin dar importancia a cómo lo hace. Una de sus ventajas es que se realizan desde el punto de vista del usuario, teniendo en cuenta las funciones y su usabilidad. Dentro de este grupo de pruebas, específicamente se realizaron las pruebas de partición equivalente. ***[18]***

Pruebas de rendimiento para los requisitos no funcionales

- **Pruebas de carga:** Se realiza con el objetivo de observar el comportamiento de la aplicación al cargar una cantidad esperada de capas y ver el tiempo que demora en cargarlas. Esta prueba permite conocer el rendimiento del producto.
- **Prueba de estrés:** Se realiza con el objetivo de determinar la solidez de la aplicación en los momentos de carga extrema, o sea cuando la carga real supere la carga esperada. Se ejecuta doblando el número de capas que se cargan en el sistema hasta que se rompe.
- **Prueba de estabilidad:** Se realiza con el objetivo de determinar si hay fuga de memoria en la aplicación, o sea se observa si el sistema puede aguantar una carga esperada continuada.

4.3 Casos de prueba para validar los requisitos funcionales

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC 1: Cargar capa	EC 1.1: Aceptar cargar la capa.	En esta sección el sistema permite cargar una capa en el mapa.	Seleccionar la opción “Cargar capa”. Se abre una ventana de búsqueda de ficheros. Se busca y se selecciona un fichero. Se presiona la opción Aceptar. Se muestra la capa en el mapa.
	EC 1.2: Cancelar la opción de cargar la capa.	En esta sección el sistema no permite cagar capas en el mapa.	Seleccionar la opción “Cargar capa”. Se abre una ventana de búsqueda de ficheros. Se presiona la opción Cancelar. Se cierra la ventana de búsqueda de ficheros.
	EC 1.3: Cerrar	En esta sección el sistema cerrar la ventana de búsqueda de ficheros.	Seleccionar la opción “Cargar capa”. Se abre una ventana de búsqueda de ficheros. Se presiona la opción Cerrar de la ventana. Se cierra la ventana de búsqueda de ficheros.

Tabla 6: Caso de prueba del CUS “Cargar capa”.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC 1: Acercar	EC 1.1: Acercar	En esta sección el	Seleccionar la opción de “Acercar zoom”.

zoom	zoom de una posición.	sistema permite acercar una posición del mapa.	Seleccionar una posición del mapa. Se acercar la vista del mapa centrando la posición donde se haga clic.
	EC 1.2: Acercar zoom de un rectángulo.	En esta sección el sistema permite acercar el área de un rectángulo en el mapa.	Seleccionar la opción de “Acercar zoom”. Seleccionar una posición del mapa y arrastrar formando un rectángulo. Se acercar la vista del mapa centrando el rectángulo.
SC 2: Alejar zoom.	EC 2.1: Alejar zoom de una posición.	En esta sección el sistema permite alejar una posición del mapa.	Seleccionar la opción de “Alejar zoom”. Seleccionar una posición del mapa. Se aleja la vista del mapa centrando la posición donde se haga clic.
	EC 2.2: Alejar zoom de un rectángulo.	En esta sección el sistema permite alejar el área de un rectángulo en el mapa.	Seleccionar la opción de “Alejar zoom”. Seleccionar una posición del mapa y arrastrar formando un rectángulo. Se aleja la vista del mapa centrando el rectángulo.
SC 3: Centrar zoom.	EC 3.1: Centrar zoom.	En esta sección el sistema permite centrar el mapa.	Seleccionar la opción de “Centrar zoom”. Se centra la vista del mapa como estaba cuando fue cargada la última capa.

Tabla 7 Caso de prueba del CUS “Modificar tamaño del mapa”.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC 1: Mover mapa.	EC 1.1: Mover mapa.	En esta sección el sistema permite	Seleccionar la opción “Mover mapa”. Seleccionar una posición del mapa y

		mover una posición del mapa.	arrastrar. Se mueve el mapa en la dirección del ratón.

Tabla 8: Caso de prueba del CUS “Mover mapa”.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC 1: Eliminar capa.	EC 1.1: Eliminar capa.	En esta sección el sistema permite eliminar una capa del mapa y de la lista de capas..	Dar clic secundario sobre la capa que se desea eliminar. Seleccionar la opción “Eliminar capa” del menú contextual. Se elimina la capa seleccionada de la lista de capas y del mapa.

Tabla 9: Caso de prueba del CUS “Eliminar capa”.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC 1: Cambiar nombre de la capa.	EC 1.1: Aceptar cambiar el nombre de la capa.	En esta sección el sistema permite cambiar el nombre de la capa.	Dar clic secundario sobre la capa que se le desea cambiar el nombre. Seleccionar la opción “Cambiar nombre” del menú contextual. Sale una ventana para escribir el nuevo nombre. Se escribe el nombre. Se presiona la opción Aceptar. Se cambia el nombre de la capa seleccionada en la lista de capas.
	EC 1.2:	En esta sección el	Dar clic secundario sobre la capa que se

	Cancelar la opción de cambiar el nombre de la capa.	sistema no permite cambiar el nombre de la capa.	le desea cambiar el nombre. Seleccionar la opción “Cambiar nombre” del menú contextual. Sale una ventana para escribir el nuevo nombre. Se presiona la opción Cancelar. Se cierra la ventana.
	EC 1.2: Cerrar.	En esta sección el sistema permite cerrar la ventana de cambiar el nombre de una capa.	Dar clic secundario sobre la capa que se le desea cambiar el nombre. Seleccionar la opción “Cambiar nombre” del menú contextual. Sale una ventana para escribir el nuevo nombre. Se presiona la opción Cerrar de la ventana. Se cierra la ventana.

Tabla 10: Caso de prueba del CUS “Cambiar nombre de una capa”.

La tabla 10 muestra que el CUS “Cambiar nombre de una capa” tiene una variable de entrada que es el nuevo nombre, en la tabla 11 se muestran los diferentes valores de entrada de dicha variable y las posibles respuestas del sistema.

ID del escenario	Escenario	Variable 1	Respuesta del sistema	Resultado de la prueba
EC 1.1	Aceptar cambiar el nombre de la capa.	Capa	Se espera que el sistema cambie el nombre de la capa seleccionada en	Se obtuvo como resultado de la prueba que el sistema cambió el nombre de la capa
		1234		
		Capa1*		

			la lista de capas.	seleccionada en la lista de capas.

Tabla 11: Variables de entrada del CUS “Cambiar nombre de una capa”.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC 1: Visualizar capas.	EC 1.1: Visualizar capas.	En esta sección el sistema permite ocultar o visualizar capas en el mapa.	Inicialmente todas las capas están marcadas y todas se visualizan en el mapa. Se desmarca la capa que se desea que no esté visualizada y deja de verse dicha capa en el mapa.

Tabla 12: Caso de prueba del CUS “Visualizar capa”.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC 1: Obtener información de un objeto.	EC 1.1: Obtener información de un objeto.	En esta sección el sistema permite mostrar la información de un objeto seleccionado.	Seleccionar la capa de la que se desea obtener información. Seleccionar la opción Información del menú. Seleccionar un objeto de esa capa en el mapa. Se muestra la información del objeto.
	EC 1.2: No seleccionar objeto de la capa	En esta sección el sistema muestra un mensaje diciendo que no se ha seleccionado ningún	Seleccionar la capa de la que se desea obtener información. Seleccionar la opción Información del menú. Seleccionar un objeto que no es de la

		objeto.	capa. Se muestra un mensaje de alerta.

Tabla 13: Caso de prueba del CUS "Obtener información de un objeto".

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC 1: Mostrar coordenadas.	EC 1.1: Mostrar coordenadas.	En esta sección el sistema muestra las coordenadas del ratón sobre el mapa.	Siempre que el ratón pase sobre el mapa se muestran las coordenadas sobre las que se encuentra el ratón en el mapa.

Tabla 14: Caso de prueba del CUS "Mostrar coordenadas".

4.4 Resultados de las validaciones de los requisitos funcionales

En la tabla 15 se tiene por las columnas los casos de prueba (CPx.x) descritos anteriormente y en las filas la cantidad de veces (Y) que se probaron los requisitos. En la intersección se encuentra un valor distinto de Ok en aquellos casos donde hubo problemas, encontrándose una notación que se corresponde con Y.x.x y en la tabla 16 se explica lo ocurrido.

Cantidad de pruebas / Casos de prueba										
Y/CPx.x	**CP1.0**	**CP2.1**	**CP2.2**	**CP2.3**	**CP3.0**	**CP4.0**	**CP5.0**	**CP6.0**	**CP7.0**	**CP8.0**
1.	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok
2.	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok
3.	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok
4.	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	4.7.0	Ok
5.	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok
6.	6.1.0	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok

Tabla 15: Resultados de las pruebas de Caja Negra.

Explicación del problema	
Identificador del error	**Explicación**
6.1.0	Luego de haber sido eliminada una capa diferente a la última cargada y se vuelve a cargar, no la visualiza en el mapa y sí en el CheckBox.
4.7.0	No valida cuando no se selecciona alguna capa en el ComboBox y se selecciona un objeto en el mapa.

Tabla 16: Errores detectados en las pruebas de Caja Negra.

Una vez terminadas las pruebas y detectados los errores se realiza una segunda iteración de la implementación para rectificar los problemas encontrados. Esta revisión del código se enfocó en:

- Validar que cuando no haya ninguna capa seleccionada en el ComboBox y se selecciona un objeto en el mapa aparezca un mensaje de advertencia.
- Verificar que siempre que se carga una capa se muestre en el mapa.

4.5 Resultados de las validaciones de los requisitos no funcionales

Para realizar las **pruebas de carga** se mide el tiempo que demora la aplicación en cargar 6 capas, cada una por separado y luego se determina el promedio que fue menor que 3 segundos, lo cual muestra un alto rendimiento de la aplicación.

Numero de capas cargadas	**Tiempo de carga** (segundos)	**Promedio** (segundos)
1.	2.9	**1.98**
2.	2.1	
3.	2.6	
4.	1.3	

5.	1.9	
6.	2.1	

Tabla 17: Resultado de las pruebas de carga.

Para realizar las **pruebas de estrés** se cargan 12 capas que es el doble de las cargadas en las pruebas anteriores y luego se probaron todas las funcionalidades del sistema. Durante esta validación se encuentra como resultado que los controles realizan todas sus funciones con velocidades menores que 3 segundos y de la forma esperada según los casos de prueba descritos. Ver ilustración 24.

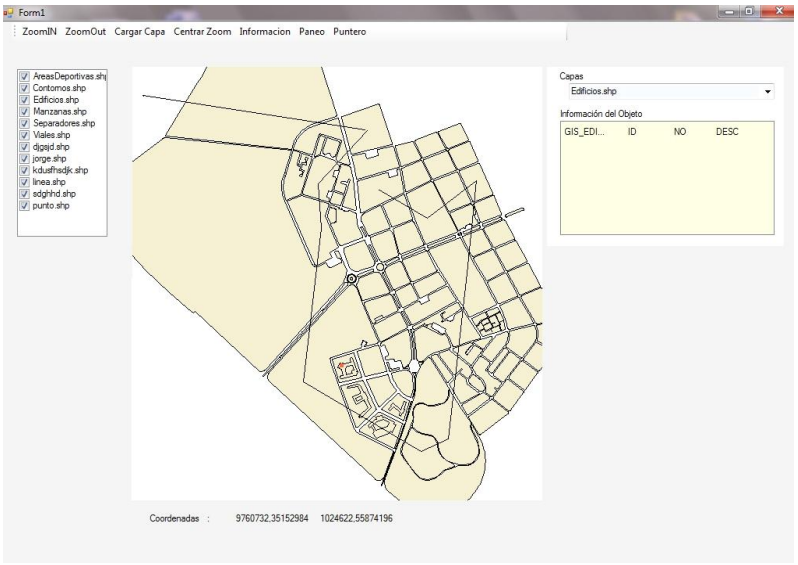

Ilustración 24: Vista de la herramienta bajo una prueba de estrés.

Para realizar las **pruebas de estabilidad** se cargan 6 capas a la aplicación, se realizan todas las acciones implementadas, luego se minimiza la aplicación y se esperan 15 minutos para volver a probar las mismas acciones. Esta prueba valida que no existe fuga de información en la aplicación.

4.6 Conclusiones parciales

En este capítulo se definieron las pruebas que podrían validar el sistema. Estas pruebas arrojaron resultados positivos aunque con algunas recomendaciones para mejorar el funcionamiento de la herramienta y el entendimiento de los usuarios. Dichos señalamientos fueron corregidos y probados nuevamente, verificando finalmente que la arquitectura y los requisitos funcionales de la librería de controles de usuarios funcionan correctamente.

Conclusiones generales

Las ventajas que produjeron los resultados del presente trabajo de diploma fueron:

- Se detectó problemas de productividad y posibles pérdidas de oportunidad de negocio para el proyecto SIG-Desktop lo que produjo una investigación sobre los SIG libres y de escritorio más utilizados en el mundo, de ese estudio se selecciona MapWindow GIS cómo la plataforma que podría dar solución a la situación mencionada.
- Se determinó al framework MapWindow como nueva alternativa tecnológica para el proyecto SIG-Desktop, de esta forma quedan solucionados los problemas de seguridad que puedan surgir en próximas integraciones con sistemas que se basan en la plataforma .NET.
- Se diseñó e implementó una librería de controles de usuario, que puede facilitar el desarrollo de SIG para los programadores del proyecto SIG-Desktop que trabajen sobre el framework MapWindow.
- Se elaboró la documentación ingenieril necesaria de la herramienta de visualización de información geográfica diseñada e implementada para el proyecto SIG-Desktop, de modo que resulte sencillo para los programadores estudiar su arquitectura y adicionarle funcionalidades.

Recomendaciones

Se recomienda:

- Hacer nuevas iteraciones sobre la librería de controles de usuario con el objetivo de añadirle nuevas funcionalidades.
- Implementar alguna personalización de la herramienta para un negocio con el objetivo de validar si aumenta la productividad de los programadores del proyecto.

Bibliografía

1. P. Ames, Daniel. “Getting Started With the MapWinGIS”. 2006.
2. Batista Silva, Dr. José Luis. Revista internacional de ciencias de la tierra, “APLICACIÓN DE SISTEMAS DE INFORMACIÓN GEOGRÁFICA EN CUBA”. 2005.
3. Pérez San-José, Pablo; Gutiérrez Borge, Cristina; Álvarez Alonso, Eduardo; De la Fuente Rodríguez, Susana; García Pérez, Laura. INTECO. “Guía sobre seguridad y privacidad de las herramientas de geolocalización”. España. Marzo 2011.
4. Peña Llopis, Juan**.** “Sistemas de Información Geográfica aplicados a la gestión del territorio”. 2006.
5. Lazo Nodarse, Yoandri. “GeoQ_Proyecto Técnico”. La Habana, Cuba. 2010.
6. Martínez Llario, José Carlos; Coll Aliaga, Eloína; Irigoyen Gaztelumendi, Jesús. Servicio de Publicaciones de la Universidad Politécnica de Valencia. “Sistemas de Información Geográfica y Urbanismo (SIG III)”. 2000.
7. Instituto del patrimonio inmobiliario de la administración pública del estado de quintana roo. “Manuel de políticas para el uso de las tecnologías de la información y telecomunicaciones”. 2005.
8. Pressman, Roger. “Ingeniería de Software un enfoque práctico”. Capítulo 04: “Desarrollo Ágil”.
9. Pressman, Roger. “Ingeniería de Software un enfoque práctico”. Capítulo 14: “Técnicas de Prueba”.
10. Isaura Tasé Figueredo, Liuver Romel Sañudo Ortiz. TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS “Desarrollo de una solución arquitectónica base e ingenieril para la concesión de un SIG con la herramienta Open Jump”. 2011.
11. P. Sun ; Z. Yu ; S. Liu ; X. Wei ; J. Wang ; N. Zegre. “Climate change, growing season water deficit and vegetation activity along the north-south transect of Eastern China from 1982 through 2006”. 2012.
12. A. Gómez Labrador: “Software Libre en el Escritorio de Inpro”. 2005.
13. Philippe Kruchten. “The Rational Unified Process: An introduction”. 2000.
14. Confederación de Empresarios de Andalucía. “Sistemas de información geográfica, tipos y

aplicaciones empresariales". 2010. Disponible en: http://sig.cea.es/. Consultado en fecha: Nobiembre del 2011.

15. Compañía Headquarters. Octubre 2008. Disponible en: http://www.visual-paradigm.com/. Consultado en fecha: Noviembre del 2011.
16. Ruiz, Ernest; Comas, David. "Fundamentos de los sistemas de información geográfica". España. 1993.
17. Revista Cuba: Medio ambiente y desarrollo. 2002. Disponible en: http://www.medioambiente.cu/. Consultado en fecha: Noviembre del 2011.
18. ESRI. 2010. Disponible en: http://www.esri.es/ . Consultado en fecha: Noviembre del 2011.
19. Kosmo. Disponible en: http://www.opengis.es/. Consultado en fecha: Noviembre del 2011.
20. Kosmo. 2006. Disponible en: http://www.kosmo.com/. Consultado en fecha: Noviembre del 2011.
21. Asociación gvSIG. Julio 2011. Disponible en: http://www.gvsig.org/. Consultado en fecha: Noviembre 2011.
22. Fowler, Martin. "Metodología de desarrollo de software Programación Extrema". 2003. Disponible en: http://www.programacionextrema.org/. Consultado en fecha: Noviembre del 2011.
23. SuperMap. Disponible en: http://www.supermap.com/. Consultado en fecha: Noviembre del 2011.
24. P. Ames, Daniel. 1998. Disponible en: http://www.mapwindow.org/. Consultado en fecha: Noviembre del 2011.
25. Federal Geographic Data Comittee (FGDC). Disponible en: http://www.fgdc.gov. Consultado en fecha: Noviembre del 2011.
26. Internacional Organization for Standarization (ISO). Disponible en: http://www.iso.org. Consultado en fecha: Noviembre del 2011.
27. Lennert, Moritz. "Grass Tutorial", 2003.
28. GRASS Development Team. 1999. Disponible en: http://grass.itc.it/. Consultado en fecha: Noviembre del 2011.

29. Organización JUMP Pilot Project. 2011. Disponible en: http://www.jumpproject.org/. Consultado en fecha: Noviembre del 2011.

30. Corporación Microsoft. 2010. Disponible en: http://www.microsoft.com/spain/visualstudio/. Consultado en fecha: Noviembre del 2011.

31. Voon Kiong, Dr. Liew. 2010. Disponible en: http://www.vbtutor.net/. Consultado en fecha: Noviembre del 2011.

32. The New York Times Company. 2011. Disponible en: http://visualbasic.about.com/ . Consultado en fecha: Noviembre del 2011.

Referencias Bibliográficas

1. Pablo Pérez San-José; Cristina Gutiérrez Borge; Eduardo Álvarez Alonso; Susana De la Fuente Rodríguez; Laura García Pérez. INTECO. “Guía sobre seguridad y privacidad de las herramientas de geolocalización”. España. Marzo 2011.
2. Ramón M. Gómez Labrador. “Tipos de licencia de Software”. Septiembre 2005.
3. Dan Berger. “COM: A Brief Introduction”. 2004.
4. Asuquo Essien Eyo. “Model for Dredging a Horizontal Trapezoidal Open Channel with Hydraulic Jump”. 2012.
5. Armin H. Seydack; Cornelia C. Grant; Izak P. Smit; Wessel J. Vermeulen; Johan Baard y Nick Zambatis. “Development of a climate–vegetation response model linking plant metabolic performance to climate and the effects on forage availability for large herbivores”.
6. Kosmo. 2006. Disponible en: http://www.kosmo.com/. Consultado en fecha: Noviembre del 2011.
7. Fernández Muñoz, Alejandro Luis Publicación. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas: “Análisis de un Sistema de Información Geográfica sobre GVSIG en el Departamento Geoinformática”. 2010.
8. Daniel P. Ames. 1998. Disponible en: http://www.mapwindow.org/. Consultado en fecha: Noviembre del 2011.
9. Daniel P. Ames. “Getting Started With the MapWinGIS”. 2006.
10. Craig Larman. 3rd Ed: “Applying UML and Patterns”. 2004.
11. Karl E Wiegers. 2nd Ed: “Software Requirements 2: Practical techniques for gathering and managing requirements throughout the product development cycle”. 2003.
12. RUMBAUGH, I.J.G.B.J. “El Proceso Unificado de Desarrollo de Software”. Capítulos 7, 8. 2000.
13. Andrew Troelsen. “Pro VB 2008 and the .NET 3.5 Platform: The expert's voice in .NET”. 2008.
14. Corporación Microsoft. 2010. Disponible en: http://www.microsoft.com/visualstudio/es-es. Consultado en fecha: Noviembre del 2011.
15. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Addison Wesley. GoF- Gang of Four. “Design Patterns. Elements of Reusable Object-Oriented Software”.

16. OBM. “FEA Consolidated Reference Model Document”. Mayo 2005.

17. Martin Fowler. “UML Destilled”.

18. Pressman, Roger. “Ingeniería de Software un enfoque práctico”. Capítulo 14: “Técnicas de Prueba”.

Anexos

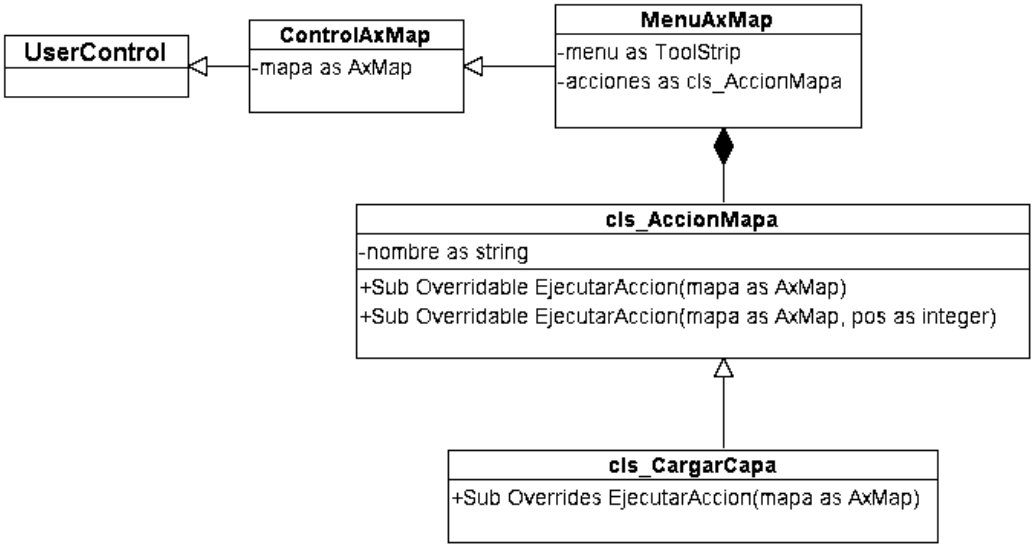

Ilustración 25: Diagrama de clases del CUS "Cargar capa"

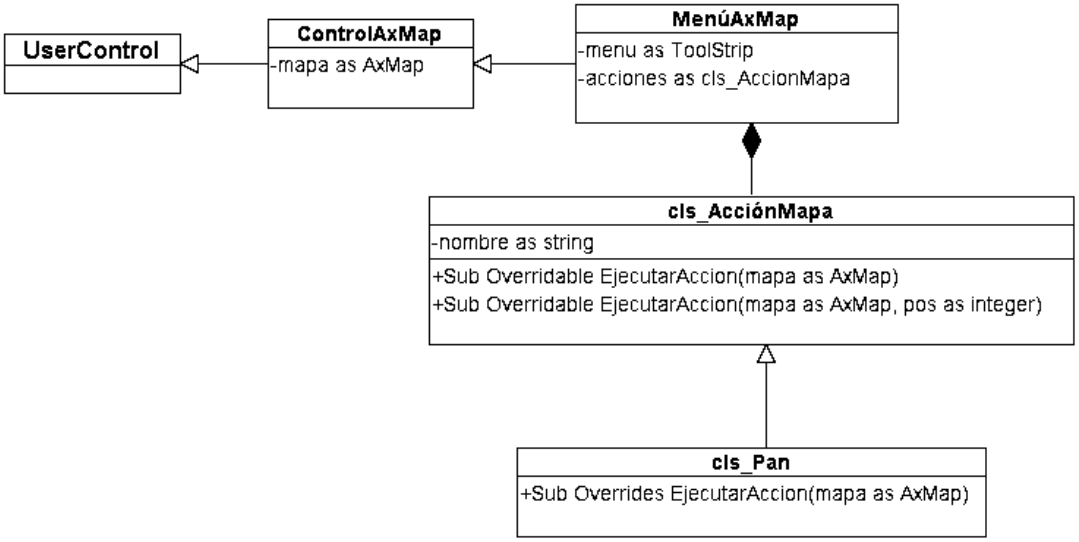

Ilustración 26: Diagrama de clases del CUS "Mover mapa"

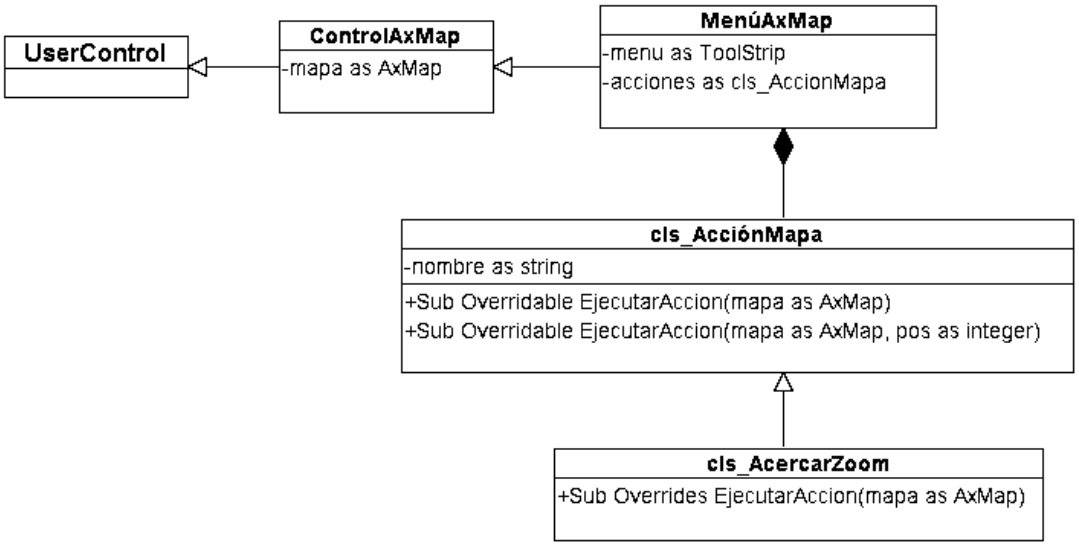

Ilustración 27: Diagrama de clases del CUS "Acercar zoom"

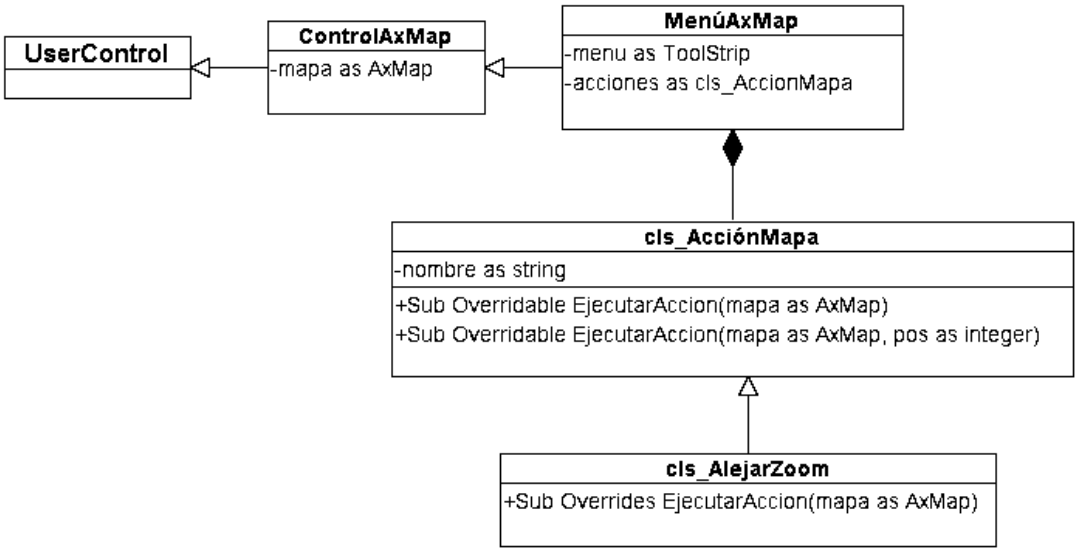

Ilustración 28: Diagrama de clases del CUS "Alejar zoom"

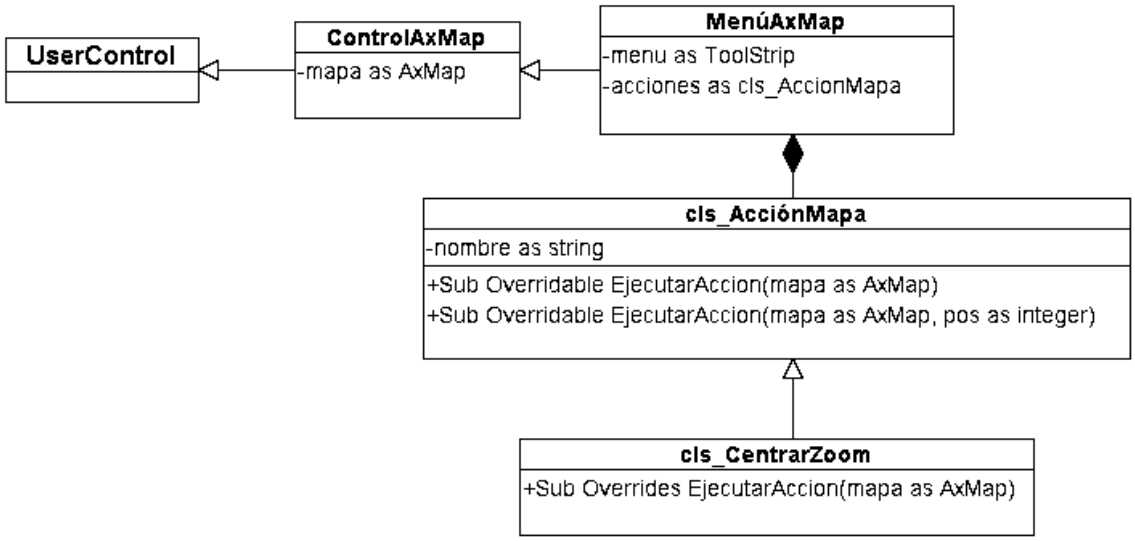

Ilustración 29: Diagrama de clases del CUS "Centrar zoom"

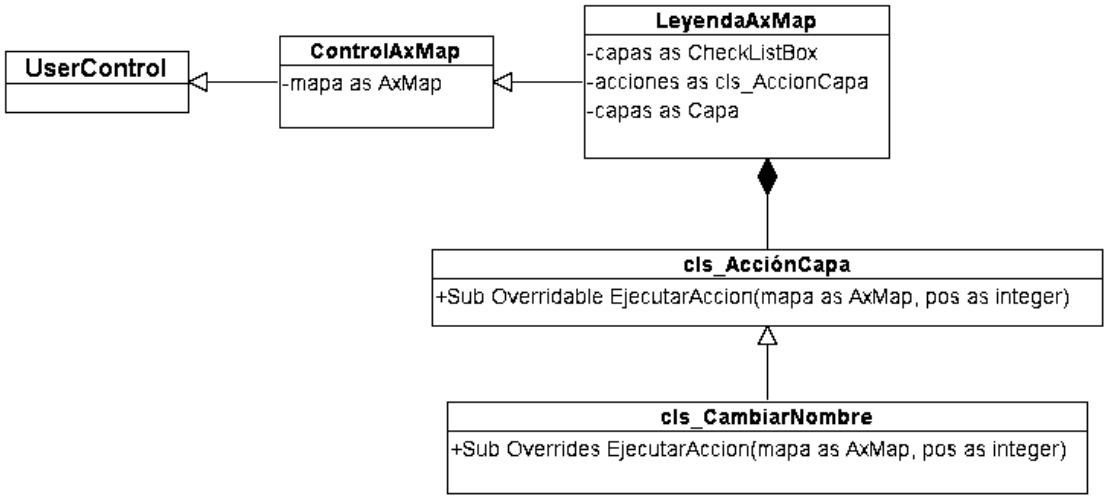

Ilustración 30: Diagrama de clases del CUS "Cambiar nombre"

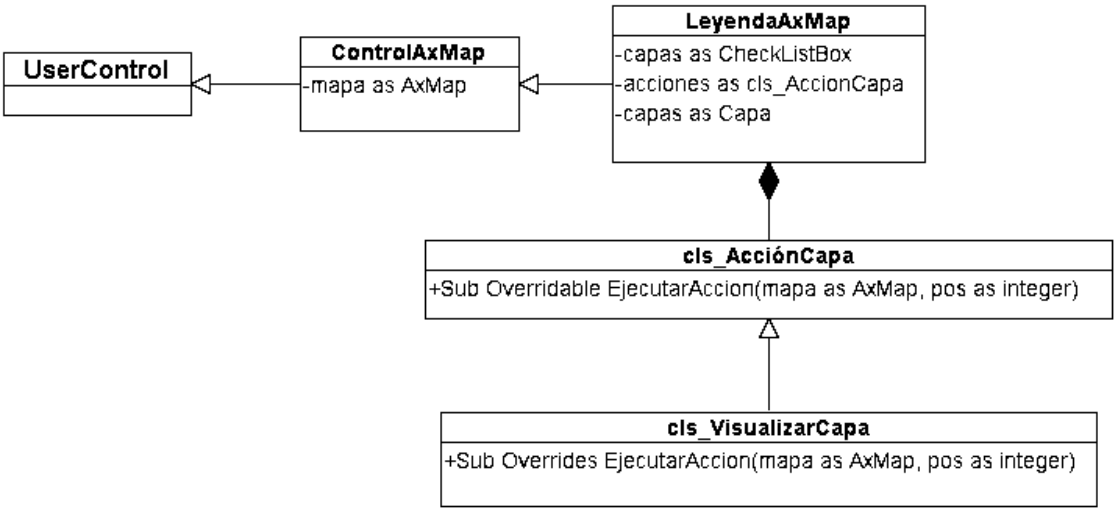

Ilustración 31: Diagrama de clases del CUS "Visualizar capa"

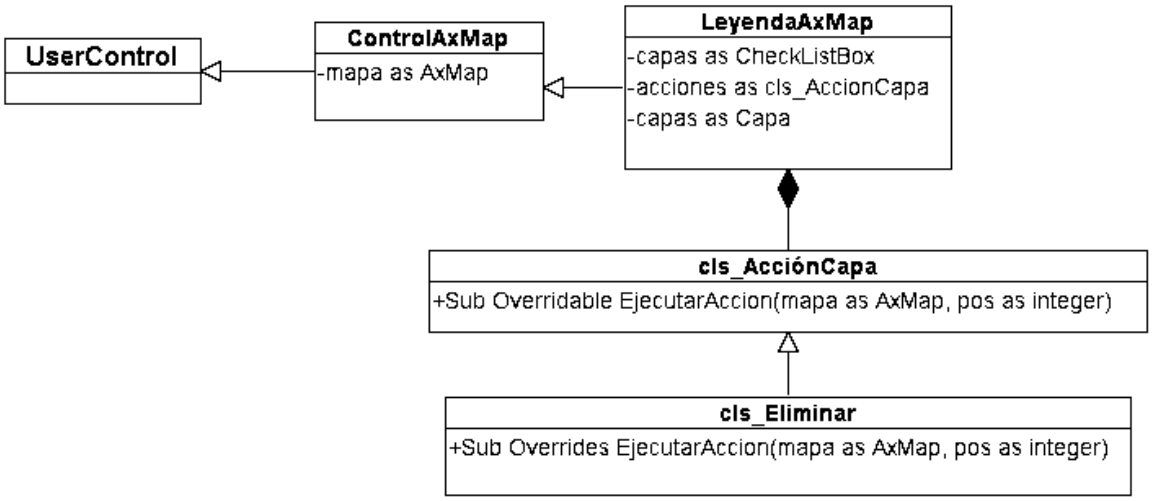

Ilustración 32: Diagrama de clases del CUS "Eliminar capa"

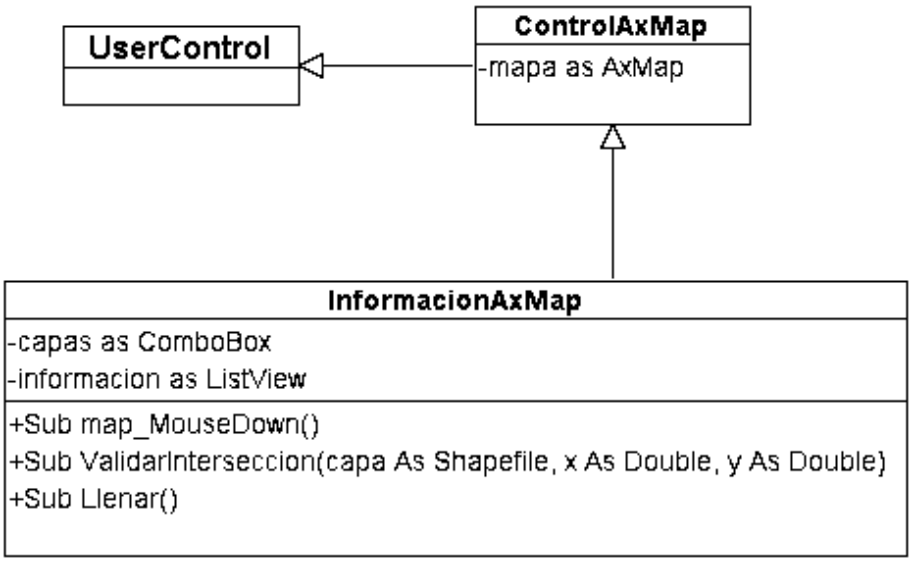

Ilustración 33: Diagrama de clases del CUS "Obtener información"

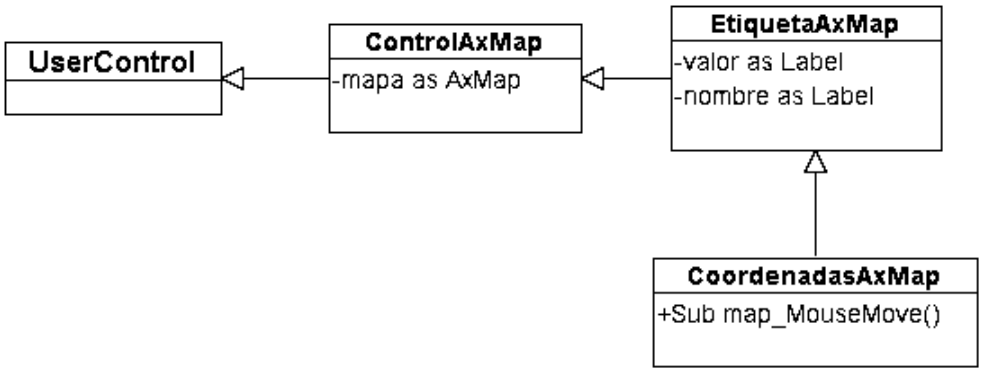

Ilustración 34: Diagrama de clases del CUS "Mostrar coordenadas"

Glosario de términos

Datos espaciales:

Son entidades espacio-temporales que cuantifican distribución, estado y vínculos de fenómenos u objetos. Tienen posición absoluta sobre un sistema de coordenadas (x,y,z) y posición relativa frente a elementos del paisaje, por ejemplo topología: incluido, adyacentes, (etc.). Estas entidades tienen atributos o características que los describen.

Estándares OGC (Open Geospatial Consortium):

Es un organismo formado por empresas, universidades y administraciones públicas que permiten la interoperabilidad en el ámbito de los Sistemas de Información Geográfica desarrollado una serie de protocolos y estándares. Por ejemplo: WMS, WMC, WFS, GML.

SIG Integrable:

Es aquel SIG que facilita la integración con servicios y aplicaciones de terceros, incorporando sus ventajas en diversos entornos, garantizando interoperabilidad y mejorando la eficiencia y las áreas de influencia del SIG.

SIG de código abierto:

Son los SIG distribuidos y desarrollados libremente. Tiene un punto de vista orientado a los beneficios prácticos de compartir el código teniendo en cuenta las cuestiones éticas y morales del llamado software libre. Los requisitos para que un SIG sea de código abierto son:

- Libre redistribución: el software debe poder ser regalado o vendido libremente.
- Código fuente: el código fuente debe estar incluido u obtenerse libremente.
- Trabajos derivados: la redistribución de modificaciones debe estar permitida.
- Integridad del código fuente del autor: las licencias pueden requerir que las modificaciones sean redistribuidas sólo como parches.
- Sin discriminación de personas o grupos: nadie puede dejarse fuera.
- Sin discriminación de áreas de iniciativa: los usuarios comerciales no pueden ser excluidos.
- Distribución de la licencia: deben aplicarse los mismos derechos a todo el que reciba el programa.

- La licencia no debe ser específica de un producto: el programa no puede licenciarse solo como parte de una distribución mayor.
- La licencia no debe restringir otro software: la licencia no puede obligar a que algún otro software que sea distribuido con el software abierto deba también ser de código abierto.
- La licencia debe ser tecnológicamente neutral: no debe requerirse la aceptación de la licencia por medio de un acceso por clic de ratón o de otra forma específica del medio de soporte del software.

API (Application Programming Interface):

Son, en la programación orientada a objetos, las funciones y procedimientos que ofrece alguna biblioteca para ser utilizada como una capa de abstracción en otro software. Con ellos se tiene la capacidad de comunicación entre componentes del sistema. Ofrecen acceso entre los niveles o capas inferiores y los superiores del software. Uno de sus propósitos principales consiste en proporcionar funciones de uso general, así el programador se beneficia haciendo uso de su funcionalidad sin programar desde el principio. Los sistemas que proporcionan un API generalmente son llamados la implementación de esa API.

Librerías DLL (Dynamic Link Library):

Son archivos ejecutables que se cargan bajo demanda de un software por parte del sistema operativo. Este término es de los sistemas operativos Windows, la extensión de estos ficheros es ".dll".

Componentes ActiveX:

Es un entorno para definir componentes de software reusables de forma independiente del lenguaje de programación. En cada aplicación pueden ser usados uno o más de esos componentes. Se usa generalmente en el sistema operativo Windows, aunque la tecnología no está atada a él. Aplicaciones Windows como Internet Explorer, Microsoft Office, Microsoft Visual Studio y Windoows Media Player usan controles ActiveX.