UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 6

Módulo de Seguridad para el Sistema de Información Geográfica Quantum GIS

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

AUTORA: Yisell Viamontes Lores

TUTOR: Ing. Vladimir Martell Fernández

La Habana, 27 de junio de 2011
"Año 53 de la Revolución"

"El futuro tiene muchos nombres. Para los débiles es lo inalcanzable. Para los temerosos, lo desconocido. Para los valientes es la oportunidad".

Victor Hugo

DEDICATORIA

Dedico el presente trabajo de diploma a mi familia por su apoyo, cariño, comprensión y ejemplo a lo largo de mi vida.

A la Revolución y en especial a Fidel por permitirme hacer realidad mis sueños.

AGRADECIMIENTOS

A mis padres Carlos y Margarita por su ternura, guía, ejemplo y por constituir ambos mi fortaleza y mi espíritu de consagración.

A mis abuelos Delia y Viamontes, a mi tía Cachi y tío Martín por su preocupación y entereza, por su ímpetu y desvelo ante cada paso mío en la vida.

A mi familia en general por ser mi orgullo y mi mayor felicidad.

A las Kankas (Lisandra, Ivelin, Krysna, Saily, Adaily, Cecilia, Glenda) por demostrarme lo hermoso que es la amistad.

A Yami por su hermandad entrañable.

A Omarito por convertirse en mi Ángel de la guarda en momentos difíciles.

A mi tutor Vladimir por su apoyo incondicional y guía certera.

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

A todas mis amistades de la Universidad que durante 5 años se convirtieron en mi familia y principal apoyo, y lo seguirán siendo; resulta realmente imposible listarlos a todos y agradecerles con la magnitud que se merecen.

A la Universidad y a la Revolución por darme la oportunidad de formarme como una profesional, de poder hacer realidad mis sueños y haberme permitido el privilegio de conocer tantas personas maravillosas.

DECLARACIÓN DE AUDITORÍA

Declaro que soy la única autora de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los 27 días del mes de junio del año 2011.

Yisell Viamontes Lores	**Ing. Vladimir Martell Fernández**
Autor	**Tutor**

DATOS DE CONTACTO

TUTOR: Ing. Vladimir Martell Fernández.

Correo electrónico: vmartell@uci.cu

Titulación: Ingeniero en Ciencias Informáticas.

Datos de interés: Instructor. Imparte clases en el departamento Técnicas de Programación de la Facultad 6. Actualmente se desempeña como 2do Jefe del Departamento de Geoinformática del Centro de Desarrollo de Software “Geoinformática y Señales Digitales”. Cuenta con cuatro años de experiencia en la producción.

RESUMEN

La seguridad informática se ha convertido en una temática de vital importancia en las organizaciones modernas. Para una organización, prescindir de prácticas adecuadas de seguridad informática puede tener un impacto enorme; la mayoría de las organizaciones se verían gravemente afectadas si su información fuera modificada o accedida sin autorización de manera accidental o intencional por empleados o por terceros. La no aplicación de políticas de seguridad traería como consecuencia un sistema informático susceptible y vulnerable a ataques que comprometerían la integridad de los datos almacenados, impediría el acceso a recursos a usuarios legítimos, o peor aún, podría hacerse un mal uso de los recursos informáticos.

Tan necesaria disciplina no se encuentra aplicada a Quantum GIS (QGIS) al constituir una herramienta de código abierto en fase de ampliación. Por tanto el objetivo del presente trabajo de diploma persigue desarrollar un módulo que garantice los parámetros de seguridad necesarios en la mencionada herramienta QGIS.

El documento que se presenta coleccionará los resultados de la investigación desarrollada. Se identificará y analizará exhaustivamente la situación problemática y el dominio donde ocurren los principales procesos del negocio. Se realiza un análisis de las diferentes herramientas y tecnologías actuales candidatas para el modelado y desarrollo del sistema. Posteriormente se enumeran las distintas funcionalidades que debe poseer el sistema y se realiza su diseño para determinar cómo quedará estructurada para su mejor desarrollo, se realiza el diseño del sistema propuesto basado en las funcionalidades planteadas anteriormente, se modela su implementación y finalmente se procede con la implementación del sistema propuesto.

Palabras clave: permisos, Quantum GIS, seguridad, Sistema de Información Geográfica (SIG).

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE CONTENIDO

INTRODUCCIÓN

En el transcurso de la historia de la ciencia, han existido determinados momentos clave en los que el ingenio de ciertas personas, aunados a toda una situación social e ideológica propicias, les han permitido percibir algo que nadie antes había sido capaz de ver. En muchas ocasiones esta revolución en los conceptos viene acompañada de innovaciones tecnológicas que constituyen grandes saltos en el desarrollo de nuevos aparatos destinados a revolucionar el progreso de la sociedad.

La propia necesidad humana de agilizar actividades, mejorar su forma de vida y transformar el entorno circundante ha incitado la creación de artefactos con características capaces de humanizar los procesos; he aquí que el surgimiento de la computación ha marcado un hito significativo en el desarrollo de la especie humana.

El mundo moderno le debe parte de su evolución a las computadoras, ya que a través de ellas se ha brindado un mecanismo, el cual permite que los niveles de desarrollo tanto económicos, electrónicos como culturales se acerquen de forma más rápida. De tal modo que el rápido desarrollo de la ciencia y la tecnología ha llevado a la sociedad a entrar en un nuevo milenio inmerso en lo que se ha dado en llamar "*Era de la Informatización*". Este desarrollo tecnológico acelerado ha propiciado el surgimiento de las Tecnologías de la Información y las Comunicaciones, las cuales están inundando el mundo referencial del ser humano a la vez que le ayudan a conquistar conocimientos y acciones que ayer mismo parecían inaccesibles.

Se puede denominar como Tecnologías de la Información y las Comunicaciones (en lo adelante TIC) al *conjunto de procesos y productos derivados de las nuevas herramientas (hardware y software), soportes de la información y canales de comunicación relacionados con el almacenamiento, procesamiento y transmisión digitalizados de la información* (GONZÁLEZ 1996).

"Las TIC son tecnologías de gestión e innovación que se basan en sistemas o productos que son capaces de captar información multidimensional, de almacenarla, de elaborarla, de tomar decisiones, de transmitirlas, difundirlas y de hacerlas inteligibles, accesibles y aplicables en correspondencia con el fenómeno a transformar. Su singularidad es la constante innovación que posibilitan y la cada vez mayor capacidad de tratamiento de la información. Abarcan una gran variedad de herramientas de datos, y de símbolos que representan información para sus usuarios, por lo que sus sistemas y productos guardan

relación, y afectan el pensamiento, la comunicación y la práctica cotidiana convirtiéndose en un eminente proceso cultural" (ARENCIBIA 2006).

Cuba, a pesar de constituir un país subdesarrollado con disímiles limitantes que han intentado frenar su desarrollo, implementa de manera eficiente las TIC para la formación y bienestar de la sociedad, proponiéndose la premisa de lograr avances significativos en los diversos sectores sociales, fomentando la preparación del pueblo en aras de lograr una cultura general integral.

El propio desarrollo de la informática en todos los ámbitos requiere llevar aparejado un desarrollo necesario del software. Por ello hoy se habla del fortalecimiento de la Industria Cubana del Software con dos objetivos fundamentales:

1. Hacer cumplir la estrategia de informatización de la sociedad.
2. Convertirla en una significativa fuente de ingresos para el país como resultado del correcto aprovechamiento de las ventajas que ofrece el alto capital humano disponible en la actualidad.

Una muestra de lo anterior lo constituye el surgimiento de la Universidad de las Ciencias Informáticas (en lo adelante UCI), centro de estudios que presenta un diseño estratégico de vínculo docencia-producción para la formación de un profesional calificado en las ciencias informáticas, incorporando más del 60% del estudiantado a proyectos productivos e investigativos de software, en interés o por encargo de la sociedad cubana y de otros países.

La UCI se encuentra estructurada por facultades, las mismas cuentan con una infraestructura productiva que las divide en centros de desarrollo de software, los cuales se especializan en diversas temáticas de acuerdo a un determinado perfil.

La facultad 6 cuenta con el Centro de Desarrollo "Geoinformática y Señales Digitales" (en lo adelante GEySED), el cual presenta como misión principal: *lograr el desarrollo de productos, servicios y soluciones informáticas en el campo del procesamiento de Señales Digitales y la Geoinformática*; para ello contiene diversos proyectos especializados en temáticas específicas de estas dos áreas.

Dentro de las temáticas que se materializan en el área de la Geoinformática se encuentra el desarrollo de Sistemas de Información Geográfica (en lo adelante SIG). Un SIG es 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 (geo-referenciados), para la solución de problemas complejos del manejo y planeamiento territorial (SILVA 2005).

Los SIG han evolucionado de manera rápida y en forma paralela al crecimiento vertiginoso de las TIC, dando una perspectiva totalmente nueva y dinámica de la información y ayudando a la toma de decisiones. Un SIG puede mapear cualquier información almacenada en planillas o bases de datos, que tenga un componente geográfico que permita ver patrones, relaciones y tendencias, que no pueden verse en un formato de tabla o lista.

Dadas las disímiles ventajas que proporciona la aplicación de un SIG, permitiendo el almacenamiento y representación de la información por niveles temáticos y agilizando procesos, se desarrolló una Plataforma Soberana (denominada geneSIG), que constituye un producto web que cumplimenta las exigencias de un SIG.

A pesar de las funcionalidades que ofrece esta aplicación, posee diversas limitantes dadas las características propias que presenta cualquier sistema tipo web, tales como:

1. Dificultad en la edición de la cartografía.
2. Incompatibilidad de navegadores (geneSIG depende del navegador Mozilla Firefox para su funcionamiento. No es posible ejecutarlo en otros navegadores).
3. Dependencia de conectividad de red para el hospedaje de servidores.

Luego del estudio del estado del arte, de consultas y análisis de información en cuanto a beneficios y oportunidades, se concluyó que era necesario la implementación de un SIG en ambiente de escritorio, que presentara funcionalidades similares a las implementadas en geneSIG pero que diera al traste con las limitantes antes expuestas, además, que permitiera ampliar las posibilidades de negocios en el mercado de adquisición de SIG. A partir de ese momento surge la línea de desarrollo SIG-DESKTOP como parte de la infraestructura productiva del departamento de Geoinformática.

Luego de haberse realizado un estudio detallado de las diversas tecnologías existentes en este campo se seleccionó como herramienta base para el desarrollo, por sus múltiples funcionalidades, Quantum GIS (en lo adelante QGIS) entre otras, por constituir un SIG de código abierto y libre de costo, muy potente y completo, lo que permitirá su personalización y/o ampliación de funcionalidades de acuerdo a las exigencias de los diversos clientes.

Un tema sensible a analizar en el desarrollo de SIG, por las múltiples consecuencias que puede acarrear en el tratamiento de la información que este contendrá, lo constituye la seguridad.

La seguridad informática se ha convertido en una temática de vital importancia en las organizaciones modernas. Para una organización, prescindir de prácticas adecuadas de seguridad informática puede tener un impacto enorme: por ejemplo, dejarían de funcionar por completo si no estuvieran disponibles los servicios que proveen las TIC o la información que estas soportan; la mayoría de las organizaciones se verían gravemente afectadas si su información fuera modificada o accedida sin autorización de manera accidental o intencional por empleados o por terceros.

Actualmente, debido a la gran dependencia tecnológica, la seguridad informática está directamente ligada a la supervivencia organizacional, y por eso es competencia de la alta gerencia.

Tan necesaria disciplina no se encuentra aplicada a QGIS al constituir una herramienta de código abierto en fase de ampliación. La aplicación no cumplimenta las exigencias de seguridad ya que:

- no tiene en cuenta el control de usuarios o control de acceso a la información, por lo tanto, su seguridad queda restringida a lo que pueda brindar el sistema operativo, de modo que cualquier personal con acceso a la computadora puede alterar los datos que allí se generan y/o procesan;
- como resultado de lo anterior, no delimita niveles de accesibilidad que garanticen que los diferentes usuarios interactúen con la información autorizada;
- no contiene la posibilidad de controlar administrativamente todos los perfiles de los usuarios, lo cual, de poseerse, permitiría una personalización de la aplicación de acuerdo a las necesidades y exigencias propias de cada usuario y una mejor organización de la información que se maneja y/o procesa;
- no presenta una forma de auditar las violaciones en el acceso a la información y lo que es peor, las infracciones en la manipulación de esta;
- al no contar con un módulo centralizado de seguridad no es posible evitar que en el futuro cada sistema tenga que implementar sus propios elementos de seguridad, revirtiéndose esto en un mayor costo y tiempo de desarrollo, además de no permitir concentrar todos los esfuerzos de seguridad en un solo punto.

La no aplicación de políticas de seguridad traería como consecuencia un sistema informático susceptible y vulnerable a ataques que comprometerían la integridad de los datos almacenados, impediría el acceso a recursos a usuarios legítimos o peor aún podría hacerse un mal uso de los recursos informáticos.

A partir de todo lo anteriormente explicado, se plantea como **problema de la investigación:** *¿Cómo garantizar el control por niveles de acceso a los datos geográficos que se manipulen en el Sistema de Información Geográfica QGIS?* Definiéndose como **objeto de estudio** *el Sistema de Información Geográfica QGIS* y como **campo de acción** *la implementación de la seguridad en el Sistema de Información Geográfica QGIS.*

Luego de lo anterior, se propone como **objetivo general de la investigación**: *Desarrollar el módulo de seguridad para el Sistema de Información Geográfica QGIS.*

Se defiende la siguiente idea: *Con el desarrollo del módulo de seguridad para el Sistema de Información Geográfica QGIS se garantizará el control por niveles de acceso a los datos geográficos que se manipulen en él.*

Para lograr los objetivos propuestos se acometen las siguientes **tareas**:

1. Caracterizar el estado actual de las tecnologías existentes para el control por niveles de acceso a datos con el objetivo de evaluar las posibles soluciones existentes que den respuesta al problema y definir las principales tendencias internacionales en temas de seguridad.
2. Caracterizar la Arquitectura base de QGIS para conocer el funcionamiento interno y la base tecnológica de la herramienta.
3. Elaborar la documentación técnica correspondiente al módulo de Seguridad de QGIS con el objetivo de dejar constancia del proceso ingenieril para futuras modificaciones y/o ampliaciones.
4. Implementar el módulo de seguridad según la documentación técnica generada.

Para el desarrollo de la investigación se utilizaron diferentes métodos científicos:

Métodos Teóricos

Analítico – Sintético: Principalmente para una mejor organización de la investigación, durante el estudio bibliográfico de las temáticas asociadas al tema en cuestión, para una clasificación acertada de la información consultada.

Histórico-Lógico: Para poder definir una sucesión lógica, necesaria para conocer la evolución y desarrollo de la temática de la aplicación de la seguridad en los sistemas informáticos, las principales etapas de su desenvolvimiento y las conexiones históricas fundamentales.

Modelación: Para expresar la realidad mediante modelos que facilitan el proceso investigativo. Esto facilita la creación de los modelos ingenieriles.

Métodos Empíricos

Entrevista: Para definir los posibles requisitos a implementar, así como la obtención de propuestas de soluciones.

Una vez finalizada la investigación se esperan obtener como resultados los enumerados a continuación:

1. Caracterización del estado actual de las tecnologías asociadas a la seguridad en aplicaciones de escritorio.
2. La documentación técnica del proceso ingenieril.
3. Un módulo para la seguridad del Sistema de Información Geográfica QGIS.

El documento se encuentra estructurado de la siguiente manera:

En el **capítulo 1** se especifican y explican los principales conceptos asociados al campo de acción definido, se realiza una descripción del objeto de estudio identificado, se detalla la situación problemática actual, además, se expone el estudio y las principales conclusiones que se obtuvieron luego de que se valoraran las soluciones existentes relacionadas con la temática.

El **capítulo 2** detalla las principales tecnologías, lenguajes de programación y herramientas que se utilizarán para la construcción de la solución.

En el **capítulo 3** se describe propiamente la solución propuesta en términos de modelo de dominio, requisitos funcionales y no funcionales y los casos de uso del sistema, todo esto unido a los correspondientes diagramas que lo modelan.

El **capítulo 4** presenta la construcción propuesta en función de diagramas de clases y estándares del diseño, generalidades y modelo de implementación, así como pruebas realizadas.

CAPÍTULO 1:
FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se abordan conceptos asociados al dominio del problema, así como los aspectos y conceptos generales relacionados con el tema de la seguridad informática aplicada a la herramienta QGIS, además una descripción del estado del arte del tema a tratar. Se pretende en este capítulo dejar sentadas las bases teóricas para un correcto análisis.

1.2 Conceptos asociados al dominio del problema

Seguridad

El término seguridad tiene múltiples usos. A grandes rasgos, puede afirmarse que este concepto que proviene del latín *securĭtas* se refiere a la cualidad de seguro, es decir aquello que está exento de peligro, daño o riesgo. Algo seguro es algo cierto, firme e indubitable. La seguridad, por lo tanto, es una certeza (*Informe sobre el Desarrollo Mundial de las Telecomunicaciones/TIC, 2006.Evaluación de las TIC para el desarrollo económico y social* 2008).

La seguridad implica la cualidad o estado de estar seguro, es decir, la evitación de exposiciones a situaciones de peligro y la actuación para quedar a cubierto frente a contingencias adversas (GAGNE 1999).

Según la definición presentada por la ISO 9126, se entiende por seguridad a la "Subcaracterística de funcionalidad[1], que indica el grado en que un acceso no autorizado (accidental o deliberado) se prevenga y se permita un acceso autorizado" (INFORMÁTICA 2006).

Seguridad Informática

Es el área de la informática que se enfoca en la protección de la infraestructura computacional y todo lo relacionado con esta (incluyendo la información contenida). Para ello existen una serie de estándares,

[1](functionality, ISO-9126): Grado en que las necesidades asumidas o descritas se satisfacen. Se divide en las subcaracterística idoneidad, precisión, interoperabilidad, seguridad.

protocolos, métodos, reglas, herramientas y leyes concebidas para minimizar los posibles riesgos a la infraestructura o a la información. La seguridad informática comprende software, bases de datos, metadatos, archivos y todo lo que la organización valore (activo) y signifique un riesgo si esta llega a manos de otras personas. Este tipo de información se conoce como información privilegiada o confidencial (MACEDO 2010).

La seguridad de la información persigue proteger la información de posibles accesos y modificaciones no autorizadas.

Constituye el conjunto de procedimientos, estrategias y herramientas que permiten garantizar la integridad, la disponibilidad y la confidencialidad de la información de una entidad, los mismos constituyen los principales objetivos de la seguridad de la información y se pueden argumentar de la siguiente manera:

- Confidencialidad: Describe el estado en el cual la información está protegida de revelaciones no autorizadas, por ejemplo una falta de confidencialidad ocurre cuando el contenido de una comunicación o de un fichero es revelado a personas no autorizadas.
- Integridad: Significa que la información no ha sido alterada o destruida, por una acción accidental o por un intento malicioso.
- Disponibilidad: Referencia al hecho de que una persona autorizada pueda acceder a la información en un apropiado período de tiempo. Las razones de la pérdida de disponibilidad pueden ser ataques o inestabilidades del sistema.
- Responsabilidad: Se define para asegurar que las acciones realizadas en el sistema por una entidad se puedan asociar únicamente a esa entidad, que será responsable de sus acciones. Es decir que una entidad no pueda negar su implicación en una acción que realizó en el sistema (SÁNCHEZ 2007).

La seguridad informática está concebida para proteger los activos informáticos, entre los que se encuentran:

- La información contenida:

  Se ha convertido en uno de los elementos más importantes dentro de una organización. La seguridad informática debe ser administrada según los criterios establecidos por los administradores y supervisores, evitando que usuarios externos y no autorizados puedan acceder a ella sin autorización. De lo contrario correrá el riesgo de que la información sea utilizada maliciosamente para obtener ventajas de ella o que sea manipulada, ocasionando lecturas erradas o incompletas de la misma. Otra

función de la seguridad informática en esta área es la de asegurar el acceso a la información en el momento oportuno, incluyendo respaldos de la misma en caso de que esta sufra daños o pérdida producto de accidentes, atentados o desastres.

- La infraestructura computacional:

  La función de la seguridad informática en esta área es velar que los equipos funcionen adecuadamente y prever medidas objetivas en caso de fallas o cualquier otro factor que atente contra la infraestructura informática.

- Los usuarios:

  Se definen como las personas que utilizan la estructura tecnológica, la zona de comunicaciones, y que gestionan la información. La seguridad informática debe establecer normas que minimicen los riesgos a la información o infraestructura informática. Estas normas incluyen horarios de funcionamiento, restricciones a ciertos lugares, autorizaciones, denegaciones, perfiles de usuario, planes de emergencia, protocolos y todo lo necesario que permita un buen nivel de seguridad informática minimizando el impacto en el desempeño de los funcionarios y de la organización en general y como principal contribuyente al uso de programas realizados por programadores.

  Términos relacionados con la seguridad informática:

  - Activo: recurso del sistema de información o relacionado con este, necesario para que la organización funcione correctamente y alcance los objetivos propuestos.
  - Amenaza: es un evento que puede desencadenar un incidente en la organización, produciendo daños materiales o pérdidas inmateriales en sus activos.
  - Impacto: medir la consecuencia al materializarse una amenaza.
  - Riesgo: posibilidad de que se produzca un impacto determinado en un activo, en un dominio o en toda la organización.
  - Vulnerabilidad: posibilidad de ocurrencia de la materialización de una amenaza sobre un activo.
  - Ataque: evento, exitoso o no, que atenta contra el buen funcionamiento de un sistema determinado.
  - Desastre o Contingencia: interrupción de la capacidad de acceso a la información y procesamiento de la misma a través de computadoras necesarias para la operación normal de un negocio (COMPUTADORAS 2008).

Control por niveles de acceso

Para el cumplimiento eficiente de los objetivos de la seguridad informática se hace necesario la existencia y puesta en marcha de un conjunto de mecanismos de seguridad entre los cuales figura de forma relevante el control de acceso.

El control de acceso surge por la necesidad de garantizar la protección de los recursos del sistema de accesos no autorizados; proceso por el cual los accesos a los recursos del sistema, así como, a la información en el flujo de trabajo son regulados según unas políticas de seguridad y permitiendo el acceso solamente a entidades autorizadas.

Se definen de tal manera una serie de modelos de control de acceso con particularidades, los cuales se compararon y estudiaron para definirse cuál es el óptimo a utilizar dada la necesidad de cumplimentar las normativas de seguridad y la relevancia de la información que se maneje.

- Modelo de Control de Acceso Discrecional ó Discretionary Access Control.

Este modelo (en lo adelante DAC), también llamado modelo de seguridad limitada, constituye un modelo no orientado al control del flujo de información. Todos los sujetos y objetos en el sistema se controlan y se especifican reglas de autorización de acceso para cada sujeto y objeto. Los sujetos pueden ser usuarios, grupos o procesos.

Los modelos DAC están basados en la idea de que el propietario de un objeto, su autor, tiene el control sobre los permisos del objeto. Esto determina que el autor se encuentre autorizado a permitir u otorgar permisos para este objeto a otros usuarios.

DAC admite la copia de datos desde un objeto a otro por usuarios autorizados de manera que un usuario puede permitir el acceso para copiar datos a otro usuario no autorizado. Este riesgo puede ser extendido a todo el sistema violando un conjunto de objetos de seguridad. La principal ventaja de DAC es que el usuario se beneficia de la flexibilidad del modelo. DAC es apropiado en ambientes donde la compartición de información es más importante que su protección (SÁNCHEZ 2007).

- Modelo de Control de Acceso Obligatorio ó Mandatory Access Control.

En el modelo (en lo adelante MAC) todos los sujetos y objetos se clasifican basándose en niveles predefinidos de seguridad que son usados en el proceso de obtención de los permisos de acceso. Para

describir estos niveles de seguridad todos los sujetos y objetos se marcan con etiquetas de seguridad que siguen el modelo de clasificación de la información militar (desde “desclasificado” hasta “alto secreto”), formando lo que se conoce como política de seguridad multinivel. Por este motivo se define MAC como un modelo “multinivel”. Este modelo puede ser implementado usando mecanismos de seguridad multinivel que usan reglas “no read-up[2]” y “no write-down[3]” también conocidas como restricciones Bell-Lapadula.

Estas reglas son diseñadas para asegurar que la información no fluya desde un nivel alto de sensibilidad a un nivel más bajo de sensibilidad.

La regla "no read-up”, establece que los usuarios tienen la capacidad de acceder a cualquier fragmento de información que se encuentre en o por debajo de su nivel de seguridad. La regla “no write-down”, declara que un sujeto con un nivel de seguridad dado no debe escribir en ningún objeto etiquetado con un nivel más bajo de seguridad.

- Modelo de Control de Acceso Basado en Roles ó Rol Based Access Control.

El principal objetivo de este modelo (en lo adelante RBAC) es prevenir que los usuarios tengan libre acceso a la información de la organización.

El modelo introduce el concepto de rol y asocia a los usuarios con los roles por los que va pasando durante la vida del sistema. Los permisos de acceso están asociados a los roles. El rol es un concepto típico usado en empresas para ordenar y estructurar sus actividades organizativas. RBAC permite modelar la seguridad desde una perspectiva empresarial puesto que se puede conectar los requerimientos de seguridad con los roles y las responsabilidades existentes en la organización.

RBAC está basado en la definición de un conjunto de elementos y de relaciones entre ellos. A nivel general describe un grupo de usuarios que pueden estar actuando bajo un conjunto de roles y realizando operaciones en las que utilizan un conjunto de objetos como recursos. En una organización, un rol puede ser definido como una función que describe la autoridad y responsabilidad dada a un usuario en un instante determinado.

Entre estos cuatro elementos se establecen relaciones del tipo:

[2] Acceso de lectura a la información

[3] Escritura de información

- Relaciones entre usuario y roles, modelando los diferentes roles que puede adoptar un usuario.
- Conjunto de operaciones que se pueden realizar sobre cada uno de los objetos. A los elementos de esta relación se les denomina permisos.
- Relaciones entre los permisos y los roles. Se modela cuando un usuario, por estar en un rol determinado, tiene permiso para realizar alguna operación específica sobre un objeto en particular.

El modelo RBAC incluye un conjunto de sesiones donde cada sesión es la relación entre un usuario y un subconjunto de roles que son activados en el momento de establecer dicha sesión. Cada sesión está asociada con un único usuario, mientras que un usuario puede tener una o más sesiones asociadas. Los permisos disponibles para un usuario son el conjunto de permisos asignados a los roles que están activados en todas las sesiones del usuario, sin tener en cuenta las sesiones establecidas por otros usuarios en el sistema.

RBAC añade la posibilidad de modelar una jerarquía de roles de forma que se puedan realizar generalizaciones y especializaciones en los controles de acceso y se facilite la modelización de la seguridad en sistemas complejos (SÁNCHEZ 2007).

Otro aspecto importante en el modelo RBAC es la posibilidad de especificar restricciones sobre la relación usuario/rol y sobre la activación de un conjunto de roles de usuario. Estas restricciones son un fuerte mecanismo para establecer políticas organizacionales de alto nivel. Las restricciones pueden ser de dos tipos: estáticas o dinámicas. Las restricciones estáticas permiten solucionar conflictos de intereses y reglas de cardinalidad de roles desde una perspectiva de política de seguridad. La asociación de un usuario con un rol puede estar sujeta de las siguientes restricciones:

- Un usuario es autorizado para un rol solo si el rol no es mutuamente excluyente con cualquier rol autorizado del usuario (Static Separation of Duty, SSD).
- El número de usuarios autorizados para un rol no puede exceder la cardinalidad del rol (Rol Cardinality)

- Modelo de Control de Acceso Basado en Tareas ó Task Based Access Control.

El siguiente modelo (en lo adelante TBAC) permite controlar el acceso en entornos representados por flujos de trabajos (workflow). El modelo TBAC extiende los tradicionales modelos de control basados en sujetos/objetos incluyendo aspectos que aportan información contextual basada en las actividades o tareas.

El control de acceso en TBAC es garantizado por medio de “Etapas de autorización”. Las “Etapas de autorización” son un concepto abstracto introducido por TBAC para modelar y manejar un sistema de permisos relacionados con el progreso de las tareas o actividades dentro del contexto de un flujo de trabajo (SÁNCHEZ 2007).

Los conceptos, características y componentes que hacen a TBAC un modelo de seguridad activo son:

- Incluye la noción de control de acceso basado en el uso. Controlando que un permiso activo otorgado por una autorización no implique una licencia ilimitada de accesos para ese permiso, establece atributos de validez, uso y expiración para evitar que la activación de un permiso por una etapa de autorización no implique una licencia ilimitada de accesos para ese permiso. Además estos atributos permitirán monitorizar el uso de los permisos en tiempo de ejecución.
- Permite mantener por separado los estados de protección de cada “etapa de autorización”.
- Da soporte para el modelado de la autorización en las tareas y en flujos de trabajo, así como, la monitorización y gestión del procesado de la autorización como del progreso de las tareas.
- Activación y desactivación de los permisos de los estados de protección de forma dinámica en tiempo de ejecución.
- También brinda soporte para características de control de acceso basado en tipos.

Conclusiones parciales

A diferencia de DAC, los modelos MAC proporcionan mecanismos más sólidos para la protección de datos, y tratan con requerimientos de seguridad más específicos, así como, los requerimientos derivados de las políticas de control de los flujos de información. Además, en los modelos MAC es el sistema quien protege los recursos u objetos, el administrador es el que impone las reglas de forma segura, a diferencia del DAC en el cual el dueño es quien protege los recursos.

Se puede concluir que el modelo factible a utilizar es el RBAC, ya que el mismo constituye un componente independiente de control de acceso que puede convivir sin problemas con DAC y/o MAC, por lo que cumplimenta las exigencias de seguridad que se han definido. Por consiguiente dada la alta integración entre los roles y las responsabilidades de los usuarios, se pueden seguir los principios del mínimo privilegio y de la separación de responsabilidades; principios de vital importancia para alcanzar el objetivo de integridad, al requerir que a un usuario no se le otorguen mayores privilegios que los necesarios para efectuar su trabajo. De igual forma el uso de RBAC para administrar los privilegios de los usuarios en un sistema o aplicación es ampliamente aceptado como práctica recomendada; por lo que algunos sistemas

reconocidos y de un alto grado de aceptación como Microsoft Active Directory, SELinux, Solaris, y PostgreSQL lo implementan de alguna forma.

Tipos de autenticación

La autenticación es el proceso que debe seguir un usuario para tener acceso a los recursos de un sistema o de una red de computadores. Este proceso implica identificación (decirle al sistema quién es) y autenticación (demostrar que el usuario es quien dice ser). La autenticación por sí sola no verifica derechos de acceso del usuario; estos se confirman en el proceso de autorización.

Se puede efectuar autenticación usando uno o varios de los siguientes métodos:

- Autenticación por conocimientos: basada en información que sólo conoce el usuario.
- Autenticación por pertenencia: basada en algo que posee el usuario.
- Autenticación por características: basada en alguna característica física del usuario.

De lo anterior se deduce que la autenticación involucra aspectos físicos y lógicos relacionados con el acceso, la utilización y la modificación de los recursos de la red o sistema.

Autenticación física: Se basa en algún objeto físico que posee el usuario, o en alguna característica física del usuario; en tal caso utiliza algún tipo de mecanismo biométrico. La información capturada en el proceso de autenticación pasa al proceso de autorización realizado por personas, dispositivos electrónicos de seguridad o sistemas de seguridad informática.

Autenticación lógica: Puede utilizarse para identificar personas o sistemas y se basa en información que sólo conoce el usuario. La autenticación y autorización se realiza con la utilización de un software especializado.

Si se combinan dos o más métodos de autenticación, esta se denomina autenticación múltiple (multi-factor authentication) y es una autenticación más segura. Por ejemplo, autenticación doble si el usuario debe presentar dos tipos de identificación, una física (una tarjeta) y la otra, algo que el usuario ha memorizado como una clave de seguridad o un número de identificación personal (PIN—Personal Identification Number). Este es el caso de una tarjeta bancaria que se utiliza con un cajero automático (ATM—Automatic Teller Machine). Más aún, algunos sistemas utilizan autenticación triple (con tres factores): un objeto físico, una contraseña y algún dato biométrico como la huella digital (CELY 2006).

Estándar de seguridad

- SAML

SAML es un marco de trabajo que permite expresar asertos acerca de la identidad, los atributos y las autorizaciones de un sujeto con el objetivo de facilitar las relaciones entre distintas empresas, así como las relaciones de estas con sus usuarios. Este marco de trabajo permite a las compañías crear identidades federadas, lo cual les facilita las tareas de gestión de perfiles, autenticación y autorización de usuarios.

Las especificaciones de SAML se basan en el estándar XML, y definen esquemas y formatos para los distintos elementos y mensajes. Se puede destacar que en la actualidad, es el único estándar abierto y es la base de otras especificaciones de federación de identidad como las de Liberty Alliance[4] (ALLIANCE 2010), y propuso una extensión de SAML v1.0 en su especificación ID-FF 1.1 en enero del 2003 (GUERRERO 2009).

El estándar SAML define cuatro elementos fundamentales:

- Asertos (Assertions): Definen las afirmaciones de seguridad de una entidad dentro de un sistema. Estas afirmaciones pueden ser de tres tipos: asertos de autenticación, de atributos y de decisiones de autorización. Los asertos de autenticación indican que el usuario ha sido autenticado por alguna aplicación, mientras que los asertos de atributo son más específicos e incluyen información sobre los atributos del usuario y por último los asertos de decisión de autorización, más específicos todavía, que incluyen directamente los privilegios que posee un usuario. Estos asertos y otros componentes del estándar se basan en XML, lo que permite que puedan ser utilizados en otros contextos.
- Protocolos (Protocols): SAML define un conjunto de protocolos de solicitud/respuesta que describen un mecanismo de intercambio automático de asertos. A continuación se describen algunos protocolos seleccionados por su nivel de relevancia y por considerarse dentro de los principales:

  -Protocolo de petición de autenticación (Authentication Request Protocol): Se utiliza cuando un proveedor de servicios debe autenticar a un usuario. La respuesta a este mensaje de petición de autenticación debe contener por lo menos un aserto de autenticación y puede contener, además, uno o varios asertos de atributo y de decisión de autorización.

[4] Surgió en Septiembre de 2001, tras la unión de un consorcio de empresas, proveedores e instituciones con un interés común: proporcionar estándares para gestión de identidades federadas.

-Protocolo de registro único de salida (Protocol Single Logout): Especifica la manera de realizar un logout[5] simultáneo de todas las sesiones asociadas a un determinado usuario. El logout puede ser iniciado directamente por el usuario o iniciado por un IdP[6] o SP[7], por ejemplo, debido a la expiración de la sesión por tiempo o por decisión de un administrador.

-Protocolo de gestión de identificadores (Name Identifier Management Protocol): Define mecanismos para modificar el valor o formato del identificador utilizado para referirse al usuario. Además, especifica la forma de terminar una asociación de la identidad del usuario entre un IdP y un SP.

-Protocolo de mapeo de identificadores (Name Identifier Mapping Protocol): Define un mecanismo que permite mapear un identificador de usuario usado entre un IdP y un SP, así como de obtener el utilizado entre el IdP y otro SP.

-Protocolo de Resolución de Artefacto (Protocolo Artifact Resolution): Este protocolo permite que se puedan transportar los asertos por referencia (a esta referencia se le denomina Artifact) en vez de valor. El protocolo especifica cómo obtener el aserto dada la referencia.

- Vinculaciones (Bindings): Los asertos y los protocolos están definidos como un esquema XML. Para que se puedan utilizar en la práctica, es necesario realizar un mapeo a los distintos protocolos de transporte utilizados (HTTP-GET, HTTP-POST, SOAP, entre otras).
- Perfiles (Profiles): Los protocolos y los bindings por sí solos no permiten la interoperabilidad de SAML. Los perfiles definen la secuencia específica de mensajes y los bindings requeridos en cada caso, para completar cada uno de los casos de uso definidos en el estándar. Por cada combinación de caso de uso y bindings se pueden obtener variaciones de un mismo perfil.

Sistema de Información Geográfica

La revolución digital de finales del siglo pasado permitió que la información geográfica se volviera más accesible para la mayoría de las personas. La utilización de SIG permite a personas y organizaciones, de

[5] cierre de sesión

[6] Identity Provider (Proveedor de Identidad): Entidad encargada de emitir, mantener y gestionar la información de identidad relativa a los usuarios en una federación. Es el que posee la infraestructura de autenticación e implementa la funcionalidad para llevar a cabo el almacenamiento de credenciales, el borrado y la recuperación de las mismas por parte del usuario. Además, los IdPs emiten asertos SAML para que puedan ser utilizados por los SPs a la hora de tomar decisiones sobre un usuario final.

[7] Service Provider (Proveedor de Servicios): Entidad encargada de proporcionar un determinado servicio a aquellos usuarios o entidades SAML que confían en el IdP para llevar a cabo las tareas de identificación. Las decisiones tomadas por un SP se basan en la información que le ha proporcionado otra entidad del sistema acerca de un determinado usuario.

una parte, analizar hechos y oportunidades; y de la otra, resolver problemas y conflictos utilizando información proveniente de un rango amplio de disciplinas. Esta tecnología, se ha convertido para muchos en una herramienta fundamental de análisis y de toma de decisiones.

Las ideas relacionadas con el surgimiento de los SIG ya tienen una historia que continúa su evolución necesaria para el progreso social. En 1986, el afamado y experto profesor titular de la Universidad de Oxford, reconocido por sus disímiles aportes a esta temática, Peter Burrough, brindaba sus primeros acercamientos a un concepto más rebuscado de SIG, lo definió como un "poderoso conjunto de herramientas para coleccionar, almacenar, recuperar, transformar y exhibir datos espaciales referenciados al mundo real" (BURROUGH 1998).

Luego de un estudio de las múltiples definiciones existentes del tema se puede afirmar que es 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 (geo-referenciados), para la solución de problemas complejos del manejo y planeamiento territorial" (SILVA 2005).

Una definición más sencilla es: Un sistema de computador capaz de mantener y usar datos con localizaciones exactas en una superficie terrestre.

Un SIG constituye una herramienta de análisis de información. La información debe tener una referencia espacial y debe conservar una inteligencia propia sobre la topología y representación.

Un SIG particulariza un conjunto de procedimientos sobre una base de datos no gráfica o descriptiva de objetos del mundo real que tienen una representación gráfica y que son susceptibles de algún tipo de medición respecto a su tamaño y dimensión relativa a la superficie de la tierra.

Se pueden identificar varios componentes de un SIG, cada uno con diferentes funciones, incluyendo la capacidad de:

- Digitalización de mapas, lo que implica que el sistema permite la captura de datos espaciales para elaborar mapas.
- Almacenar, manejar e integrar datos referenciados geográficamente, los que pueden estar ubicados en distintas fuentes, es decir, que tiene algunas funciones de un sistema manejador de bases de datos.

- Recuperar o localizar datos geo-referenciados, lo cual quiere decir que a partir de puntos indicados en un plano espacial se puede saber el número de atributos de esa unidad del sistema.
- Producir diversos tipos de análisis de datos, aspecto que incluye la definición de condiciones de adyacencia, de contenido y de proximidad.
- Producir resultados o salidas en diversos formatos, ya sea en mapas, gráficos o cuadros.
- Producir mapas temáticos de alta calidad, ya que pueden conjugarse simultáneamente formatos de salida y se cuenta con herramientas de edición versátiles.

Datos geográficos

"Los datos geográficos se definen como cualquier información sobre objetos o fenómenos que tengan una ubicación relativa con respecto a la superficie de la Tierra" (WESTFALEN 1995).

"El dato geográfico es toda información que nos permite conocer lo que ocurre (qué) en una determinada posición del espacio (dónde), de una determinada manera (cómo) y en un tiempo (cuándo). Estas preguntas son básicas en el día a día de la sociedad actual, por lo que la producción de datos, y de productos geográficos de valor añadido, es cada vez más común. Por este motivo se necesitan mayores esfuerzos para obtener los beneficios sociales que se derivan de la normalización. La calidad del dato geográfico, entendida en su acepción más amplia, limita la forma en que puede y debe ser usada, tratada y analizada una información geoespacial" (LÓPEZ 2000).

Conceptualmente, los datos geográficos se pueden dividir en dos elementos: observación o entidad y atributo o variable. Los SIG son capaces de gestionar ambos elementos. Los datos espaciales constan de dos componentes: espacial y temática.

Componente espacial: Las observaciones tienen dos aspectos en referencia a su localización: la localización absoluta, basada en un sistema de coordenadas y las relaciones topológicas con respecto a otras entidades.

Componente temática: Las variables o atributos de las entidades se pueden estudiar considerando el aspecto temático (estadística), su localización (análisis espacial) o ambos (SIG).

Los SIG son capaces de manejar ambos conceptos mientras que los programas de diseño asistido por ordenador (CAD) solo utilizan la localización absoluta.

1.3 Objeto de estudio

1.3.1 Descripción General

QGIS surge oficialmente en mayo del 2002 a partir de la creación del proyecto que lleva el mismo nombre, cuando se iniciaron los primeros proyectos de codificación. La idea se concibe cuando Gary Sherman se proyecta en la búsqueda de un visualizador SIG para Linux que fuese rápido y se apoyara en una amplia gama de almacenes de datos. Esto, junto con un interés en la codificación de una aplicación SIG llevó a la creación del proyecto.

Inicialmente QGIS se estableció como un proyecto en SourceForge en junio del 2002. El primer código se verificó en el Sistema Control de Versiones (CVS) en SourceForge el 6 de julio 2002, y el primero, en su mayoría no de liberación en su funcionamiento, llegó en julio del 2002. La primera versión compatible estaba disponible sólo para capas PostGIS.

QGIS se encuentra licenciado bajo la autoridad Pública GNU y Fundación OSGeo[8]. QGIS constituye una cruz-plataforma (Linux, Windows, Mac) aplicación de código abierto que ofrece muchas de las características comunes de los SIG y los servicios públicos necesarios para ver y editar los datos y generar un mapa: escala de zoom, el mapa, el panel de la leyenda, las apariencias, capa (colores, símbolos, entre otros), mapas temáticos, etiquetado, medición (incluyendo área), fácilmente trabaja con datos en varios formatos (SHP, KML, TAB, TIFF), permite realizar consultas espaciales, crear nuevas capas, editar y eliminar la capa, generar mapas, imprimir y exportar mapas.

QGIS posee toda una gama de funcionalidades indispensables para cualquier SIG, las cuales se listan a continuación:

1. Vector de superposición y de datos de trama en diferentes formatos y proyecciones, sin conversión a un formato interno o común. Soportando los siguientes formatos:
   - Espacio habilitado para las tablas PostgreSQL con PostGIS y SpatiaLite, la mayoría de los formatos vectoriales con el apoyo de la biblioteca OGR[9], incluyendo ESRI shapefiles, MapInfo, SDTS y GML.

[8] Open Source Geospatial Foundation*:* creada para promover y construir el software geoespacial abierto de la más alta calidad. El objetivo de la fundación es promover el uso y desarrollo colaborativo de proyectos liderados por la comunidad.

[9] es una librería escrita en C++, de código abierto, que permite manipular una gran variedad de formatos en archivos vectoriales.

- Formatos de mapa de bits con el apoyo de la biblioteca de GDAL[10], tales como modelos de elevación digital, la fotografía aérea o imágenes Landsat, lugares de GRASS[11] y mapsets, en la línea de datos espaciales se desempeñó como OGC-WMS compatible.

2. Crear mapas y explorar interactivamente los datos espaciales con una interfaz gráfica de usuario. Las muchas herramientas útiles disponibles en la interfaz gráfica de usuario incluyen:
    - Compositor de impresión.
    - Panel de vista.
    - Marcadores espaciales.
    - Identificar y seleccionar las características.
    - Editar / Ver / atributos de búsqueda.
    - Características de etiquetado.
    - Superposición de diagrama de vectores.
    - Vector de cambio y la simbología de trama.
    - Añadir una capa de retícula.
    - Decorar el mapa con una flecha al norte, la barra de escala y la etiqueta de derecho de autor.
    - Guardar y restaurar los proyectos.
3. Crear, editar y exportar datos espaciales usando:
    - Herramientas para la digitalización de GRASS y formatos shapefile,
    - Herramientas de GPS para la importación y exportación en formato GPX, convertir otros formatos de GPS para GPX, o bajar / subir directamente a una unidad de GPS.
4. Realizar el análisis espacial utilizando el plugin para fToolsShapefiles o el plugin de GRASS integrado, incluyendo:
    - Álgebra de mapas.
    - Análisis del terreno.
    - Modelos hidrológicos.
    - Análisis de redes.

Quantum GIS es adaptable a las necesidades requeridas ya que presenta una arquitectura de plugin extensible, además posee un importante editor gráfico. Constituye una herramienta implementada sobre la

[10] Geospatial Data Abstraction Library: biblioteca de software para la lectura y escritura de formatos de datos geoespaciales, publicada bajo la licencia X/MIT style Open Source por la fundación geoespacial de código abierto.

[11] Geographic Resources Analysis Support System: Es un software SIG bajo licencia GPL (software libre). Puede soportar información tanto ráster como vectorial y posee herramientas de procesado digital de imágenes.

base del Software Libre. Para el logro de una necesaria integración permite la conexión con otros SIG reconocidos a nivel mundial como MapInfo. Se encuentra disponible en dos idiomas: inglés y español.

1.4 Software Libre como opción para el desarrollo de aplicaciones seguras

Cuando se ofrecen servicios o información se abre la puerta a posibles intrusos que puedan introducirse en estos sistemas. Protegerse de esta posibilidad implica tener un especial cuidado con todo el software empleado, desde el sistema operativo hasta la última de las aplicaciones instaladas, y cuidar en gran medida su configuración.

Pero tampoco se debería olvidar la posibilidad de que existan intrusos que accedan físicamente al sistema.

Es por ello que, se hace necesario adoptar las medidas de seguridad adecuadas sobre el propio hardware para evitar robos, o pérdidas de información por estos accesos inadecuados.

En definitiva un buen sistema de seguridad debe proteger los sistemas vulnerables ante el posible acceso físico de intrusos no autorizados. Evidentemente, el nivel de seguridad establecido tendrá que ser consecuente con un análisis previo de los riesgos, considerando el impacto de dicho acceso no deseado contra las posibilidades de que este se produzca.

Se puede realizar un análisis de los fallos de seguridad que se identificaron como posibles a materializarse en el software y que pueden influir de manera negativa en el funcionamiento normal de la aplicación. De una forma simplista, se pueden dividir en tres bloques:

- Fallos debidos a errores desconocidos en el software, o conocidos sólo por terceras entidades hostiles.
- Fallos debidos a errores conocidos pero no arreglados en la copia en uso del software.
- Fallos debidos a una mala configuración del software, que introduce vulnerabilidades en el sistema.

El primero de ellos se puede asociar a la calidad del código, el segundo a la capacidad y celeridad de arreglo de los errores descubiertos en el código por parte del proveedor del mismo y a la capacidad del administrador de recibir e instalar nuevas copias de este software actualizado. El tercer tipo de vulnerabilidades puede estar determinado, por una falta de documentación del software o una falta de

formación adecuada de los administradores para hacer una adaptación correcta del mismo a sus necesidades.

Los fallos pueden dar lugar a un mal funcionamiento del programa, siendo en el ámbito de la seguridad preocupantes por cuanto:

- Pueden implementarse algoritmos de forma incorrecta lo que puede llevar a una pérdida de seguridad (por ejemplo, un algoritmo de generación de claves que no se base en números totalmente aleatorios).
- Pueden diseñarse servicios que, en contra de sus especificaciones, ofrezcan funcionalidades no deseadas o que puedan vulnerar la seguridad del servidor que los ofrezca.
- Pueden no haberse tomado las medidas de precaución adecuadas para asegurar el correcto tratamiento de los parámetros de entrada, lo que puede hacer que un atacante externo abuse de ellos para obligar al programa a realizar operaciones indeseadas.

La aplicación de software libre en temática de seguridad trae aparejado una serie de ventajas que son de necesario estudio para optimizar su uso, por lo que se pueden pautar a continuación:

- Al disponer del código fuente de los programas en su totalidad, este puede ser analizado por terceras personas ajenas a sus autores en busca de fallos de diseño o de implementación.
- La posibilidad de realizar modificaciones libremente al código fuente y distribuirlas permite que cualquiera pueda ofrecer mejoras sobre este. Estas mejoras podrán ser nuevas funcionalidades que se incorporen al mismo o parches que corrijan problemas detectados anteriormente.
- Las características del software libre hacen que no sea lógico cargar costes sobre el software en sí (dado que se ha de distribuir sin cargo), lo que permite que este tipo de software pueda ser utilizado por organizaciones y personas con menos recursos económicos. Esto se presenta como una ventaja cuando se compara con los precios de lo que cuesta el software de seguridad propietario hoy día (licencias de cortafuegos, sistemas de detección de intrusos, entre otros). El software libre pone en manos de cualquiera el tipo de tecnología que, hoy por hoy, sólo podían tener grandes corporaciones.
- De igual forma, la posibilidad de modificar libremente el software le permite a las organizaciones que lo adapten a sus propias necesidades, pudiendo eliminar funcionalidades que no le sean de interés.

Frente al análisis de fallos que puede sobrevenir en la realización del software, el software libre protege a sus usuarios con una serie de mecanismos determinados. Entre estos:

- La posibilidad de una auditoría de código en las herramientas software reduce los riesgos de seguridad debido a la aparición de fallos desconocidos, a la introducción de funcionalidades no deseadas en el código o la incorrecta implementación de algoritmos públicos. Aunque no se pueda asegurar que el código esté carente de errores, sí es posible garantizar que tantas posibilidades tiene de encontrar un fallo de programación en este (que lleve implícito un riesgo de seguridad) un atacante externo como la organización lo utilice.
- La posibilidad de corregir los programas y distribuir dichas correcciones permite que los programas evolucionen de una forma más abierta. En el mundo de la seguridad, un fallo en el sistema significa exponer a este a una "ventana de vulnerabilidad" que tiene lugar desde la detección del fallo (por parte de sus usuarios legítimos o de terceras partes, hostiles incluso) a la aplicación de la medida correctiva, que pueda ser la instalación del parche adecuado que arregle el problema, pasando por la generación de dicho parche.
- El hecho de que exista una cierta independencia entre el software y su fabricante, o distribuidor original, permite que los usuarios de este software, en caso de pérdida de soporte, puedan realizar el mantenimiento de este ellos mismos o subcontratarlo a una tercera empresa. Este hecho es de gran importancia en el mundo de la seguridad dado que la seguridad de una entidad no debe depender de la solvencia de terceras compañías a las que adquiere productos de seguridad.

Esto afirma la necesidad de implementar un mecanismo de seguridad propio de QGIS que no cree dependencias de terceros y que garantice la total protección de los datos que se manejarán.

A pesar de las múltiples ventajas que ofrece el uso de software libre, no está exento de desventajas. Así se podrían enumerar las siguientes:

- La posibilidad de una generación más fácil de troyanos, dado que el código fuente también puede ser modificado con intenciones maliciosas. Si el troyano logra confundirse con la versión original puede haber problemas graves.

  La fuente del programa, en realidad, será el método de distribución de software, que, de no ser seguro, permitirá que un tercer agente lo manipule. La distribución de software se asegura añadiendo posibilidad de firmado de hashes de la información distribuida.

- El método de generación de software libre suele seguir, en la mayoría de los casos, el modelo bazar, es decir, muchas personas trabajan sobre partes concretas e integrando sus cambios o personas desde el exterior contribuyen con mejoras al proyecto global. Esto puede dar lugar a que se realice una mala gestión del código fuente del software por no seguir métodos formales de seguimiento, la consecuencia final es que falten piezas clave (que nadie ha contribuido) como es el caso de la documentación.
- Al no tener un respaldo directo, la evolución futura de los componentes software no está asegurada o se hace demasiado despacio.

Por tanto, si bien el software libre en la actualidad tiene una cobertura desigual de las distintas necesidades de seguridad de una empresa o corporación, este es, definitivamente, una apuesta de futuro provechosa en aquellas áreas aún no desarrolladas y una oportunidad real e inmediata en las demás áreas para utilizar soluciones equivalentes a las propietarias con:

- un menor coste;
- mayores garantías de seguridad, debido a la posibilidad de auditar el código en uso;
- una mayor flexibilidad en la adaptación e integración, gracias a la posibilidad de modificar dicho código;
- la posibilidad del mantenimiento asegurado de una solución de seguridad con independencia del origen del producto en sí.

1.5 Situación problemática

QGIS constituye una herramienta de gran auge en el mundo del desarrollo de los Sistemas de Información Geográfica de escritorio por los diversos aportes que realiza en la evolución del software libre, pero no posee la implementación de un módulo de seguridad que garantice el uso eficiente del software para el logro de confiabilidad, integridad y disponibilidad de los datos que se manejen.

Actualmente la herramienta no tiene asociada un módulo que garantice el control de acceso a la información, por lo cual la temática de la seguridad queda restringida a lo que pueda brindar el sistema operativo, por lo que la información que se maneja y/o procesa puede ser consultada o alterada por cualquier persona con acceso al computador; como resultado de lo anterior, no delimita niveles de accesibilidad lo cual propicia la creación de una zona de total desconfianza sin poder definirse si el usuario que modifica o consulta la información susceptible es el correcto.

Otra limitante importante lo constituye el no contener la posibilidad de configurar los perfiles de los usuarios, lo cual, de poseerse, permitiría una personalización de la aplicación de acuerdo a las necesidades y exigencias propias de cada usuario y una mejor organización de la información que se maneja y/o procesa; no posee un mecanismo para el control y registro de las violaciones en el acceso a la información y lo que es peor, las infracciones en la manipulación de esta.

Finalmente, no se cuenta con un módulo centralizado de seguridad por lo que las proyecciones se definen enfocadas a la imposibilidad de evitar que en el futuro cada sistema tenga que implementar sus propios elementos de seguridad, lo que traería consigo pérdidas valoradas en costo y tiempo de desarrollo, además de no permitir concentrar todos los esfuerzos de seguridad en un solo punto.

1.5.1 Análisis de otras soluciones existentes

Se realizó un estudio de variantes existentes y se reconoció la existencia en la Universidad de las Ciencias Informáticas de un Sistema de Gestión Integral de Seguridad (Acaxia), el cual constituye un sistema diseñado exactamente para gestionarle la seguridad a cualquier aplicación web que se suscribiese a él; surgido al calor de lo que se conocería como el primer ERP cubano[12], para brindarle seguridad a todos sus módulos o subsistemas.

Constituye un sistema novedoso desarrollado sobre tecnologías libres como el lenguaje PHP, gestor de base de datos PostgreSQL, servidor web Apache, aún cuando en el mundo los principales sistemas de seguridad están implementados sobre otras tecnologías. Su administración centralizada de la seguridad es algo nuevo para las aplicaciones de este tipo. Permite realizar un número de funcionalidades que hace de este sistema muy aplicable para cualquier entorno web que necesite seguridad.

A pesar de las múltiples funcionalidades que proporciona este sistema no se adecua a las necesidades reales de la herramienta Quantum GIS, ya que no existe compatibilidad que pudiese hacer posible la integración de ambos sistemas, por lo que luego de haberse realizado un exhaustivo estudio, se concluyó que no existe aún una variante factible para Quantum GIS que garantice los parámetros de seguridad propicios para la conservación y manipulación correcta de la información que se maneja. Al constituir este una herramienta desarrollada bajo la premisa del Software Libre no existe aún un mecanismo de seguridad que optimice el control de acceso.

[12] Sistema de Gestión de Recursos Empresariales Cedrux.

1.6 Conclusiones

1. El tipo de autenticación a implementar es la autenticación por conocimientos: basada en información que sólo conoce el usuario, en específico la definición de un usuario y contraseña para brindar los servicios pertinentes a cada usuario de acuerdo al rol al cual pertenezca (sistemas biométricos costosos, carencia de equipamiento especializado para aplicarlo).
2. Modelo de control de acceso a utilizar: RBAC, por la necesidad de asignar roles a los usuarios en dependencia de las funcionalidades y tareas a las cuales tenga acceso.
3. Teniendo en cuenta las ventajas que proporciona el software libre, pero considerando las debilidades que este pueda poseer se requiere de la implementación de un mecanismo de seguridad robusto que garantice la integridad, confidencialidad y disponibilidad de la información que se maneja y/o procesa.

CAPÍTULO 2:

HERRAMIENTAS Y TECNOLOGÍAS ACTUALES A UTILIZAR

2.1 Introducción

En este capítulo se realiza un análisis del estado del arte de los métodos, técnicas y herramientas empleadas para el modelado del subsistema así como de las tecnologías que se emplean en el desarrollo y documentación de la aplicación.

2.2 Lenguajes de programación

Un lenguaje de programación es una técnica de comunicación estandarizada para expresar las instrucciones a una computadora. Se trata de un conjunto de reglas sintácticas y semánticas utilizadas para definir los programas de ordenador (INFORMÁTICA 2006).

Un lenguaje le permite a un programador especificar con precisión los datos que una computadora tomará sobre una decisión, cómo estos datos serán almacenados o transmitidos, y precisamente las acciones a tomar en diversas circunstancias.

2.2.1 C++

C++[13] constituye un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. Fue creado para extender al lenguaje de programación C para que permitiese la manipulación de objetos.

Posteriormente se añadieron facilidades como la programación genérica, que se sumó a los otros dos paradigmas que ya se encontraban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje multiparadigma.

En la actualidad, el C++ es un lenguaje versátil, potente y general. Su éxito entre los programadores profesionales le ha llevado a ocupar el primer puesto como herramienta de desarrollo de aplicaciones. El

[13] significa "incremento de C" y se refiere a que C++ es una extensión de C.

C++ mantiene las ventajas del C en cuanto a riqueza de operadores y expresiones, flexibilidad, concisión y eficiencia. Además, ha eliminado algunas de las dificultades y limitaciones del C original (DE JALÓN 1998).

2.2.2 XML

XML[14] es un lenguaje extensible de etiquetas desarrollado por el W3C[15]. XML no es realmente un lenguaje en particular, sino una forma de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, Scalable Vector Graphics(SVG) y MathML (ROQUE 2010).

XML además de ser pensado para aplicaciones en INTERNET, se propone como un estándar para el intercambio de información entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil. XML ofrece además validación de los datos y realiza el recorrido de las estructuras, de manera que el implementador no tiene que preocuparse por estas cuestiones.

2.3 Los frameworks como ayuda en el desarrollo del software

El desarrollo de frameworks en el mundo informático ha marcado un hito significativo, definiendo ventajas significativas que aminorizan el proceso de implementación y estandarizan un poco las definiciones y los procesos.

Una definición de framework es la de un marco de aplicación o conjunto de bibliotecas orientadas a la reutilización a muy gran escala de componentes de software para el desarrollo rápido de aplicaciones (AUTORES 2008).

El término framework presenta una acepción más amplia, en donde además de incluir una biblioteca de componentes reutilizables, es toda una tecnología o modelo de programación que contiene máquinas virtuales, compiladores, bibliotecas de administración de recursos en tiempo de ejecución y

[14] Extensible Markup Language

[15] World Wide Web Consortium

especificaciones de lenguajes. Otra ventaja de los frameworks, y en especial de esta acepción que se define, es la portabilidad de aplicaciones de una arquitectura a otra.

Luego del estudio de las diferentes conceptualizaciones de este término, y aplicaciones prácticas que desarrolla se pueden precisar las principales ventajas de la utilización de un framework, las cuales se enuncian a continuación:

- El desarrollo rápido de aplicaciones. Los componentes incluidos en un framework constituyen una capa que libera al programador de la escritura de código de bajo nivel.
- La reutilización de componentes de software al por mayor. Los frameworks son los paradigmas de la reutilización.
- El uso y la programación de componentes que siguen una política de diseño uniforme. Un framework orientado a objetos logra que los componentes sean clases que pertenezcan a una gran jerarquía de clases, lo que genera como resultado bibliotecas más fáciles de aprender a usar.

2.3.1 Qt

Qt constituye un framework para el desarrollo de aplicaciones que requieran de una interfaz gráfica de usuario, es desarrollado y comercializado por la compañía noruega Trolltech y está disponible para Windows, varios dialectos Unix y para el Macintosh, por otra parte, sigue el paradigma de una sola fuente: Un programa escrito con Qt se puede compilar y ejecutar en cualquiera de las plataformas compatibles, sin cambios en el código fuente.

Se presenta como una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario y también para el desarrollo de programas sin interfaz gráfica como herramientas de la consola y servidores. Qt cuenta actualmente con un sistema de triple licencia: GPL v2/v3 (General Public License) para el desarrollo de software de código abierto y software libre, la licencia de pago QPL para el desarrollo de aplicaciones comerciales, y a partir de la versión 4.5 una licencia gratuita pensada para aplicaciones comerciales, GNU/LGPL (Lesser General Public License, según sus siglas en inglés).

El API[16] de la biblioteca cuenta con métodos para acceder a bases de datos mediante SQL, así como uso de XML, gestión de hilos, soporte de red, una API multiplataforma unificada para la manipulación de archivos y una multitud de otros para el manejo de ficheros, además de estructuras de datos tradicionales.

Otras ventajas significativas que fundamentan su selección como base para el desarrollo son:

- Disponibilidad del código fuente.
- Excelente documentación.
- Fácilmente internacionalizable.
- Arquitectura lista para plugin.
- Viene acompañado de una serie de herramientas que facilitan su uso tales como: Qt Extend (para el desarrollo de herramientas), Qt Designer (para la creación visual de formularios), Qt Linguist (para la traducción rápida de programas), Qt Assistant (para lograr un acceso rápido a la documentación), qmake (simplifica el proceso de construcción de proyectos en las diferentes plataformas soportadas).
- Las librerías no sólo están disponibles en C++, sino también ofrece soluciones para utilizarse con otros lenguajes tales como Java (QJambi), Phyton (PyQt), JavaScript (módulo Qt Script), PHP, entre otros.

2.3.2 Entorno de Desarrollo Integrado (IDE): Qt Creator

Qt Creator constituye una multiplataforma que se ajusta a las necesidades de los desarrolladores. Se centra en proporcionar características que ayudan a los nuevos usuarios de Qt a aprender y comenzar a desarrollar rápidamente, también aumenta la productividad de los desarrolladores con experiencia en Qt (CORPORATION 2011).

Dentro de las múltiples funcionalidades que brinda se encuentran:

- Editor de código con soporte para C++ y QML, proporcionando ayuda sensible al contexto.
- Permite la integración con la mayoría de los sistemas de control de versiones más difundidos internacionalmente como Git, Subversion, Perforce, CVS y Mercurial.
- Herramientas para la rápida navegación del código.

[16]Application Program Interface: Conjunto de convenciones de programación que definen cómo se invoca un servicio desde un programa.

- Resaltado de sintaxis y auto-completado de código.
- Control estático de código y estilo a medida que se escribe.
- Soporte para refactorización de código.
- Plegado de código (codefolding).

2.4 Herramienta CASE

En “Terminology for Software Engineering and Computer-Aided Software Engineering” por B. Terry & D. Logee, CASE se define como:

“Herramientas individuales para ayudar al desarrollador de software o administrador de proyecto durante una o más fases del desarrollo de software (o mantenimiento)” (FOUNDATION 2008).

En el libro “The CASE Experience”, el Dr. Carma L. McClure ofrece la siguiente definición:

“Una combinación de herramientas de software y metodologías de desarrollo” (MCCLURE 2008).

Entre los beneficios ofrecidos por la tecnología CASE se encuentran los siguientes:

Facilidad para la revisión de aplicaciones: Contar con un depósito central agiliza el proceso de revisión ya que este proporciona bases para las definiciones y estándares para los datos. Las capacidades de generación interna, si se encuentran presentes, contribuyen a modificar el sistema por medio de las especificaciones más que por los ajustes al código fuente.

Soporte para el desarrollo de prototipos de sistemas: Los ajustes necesarios al diseño se hacen con rapidez para alterar la presentación y las características de la interfaz. Sin embargo, no se prepara el código fuente, de naturaleza orientada hacia procedimientos, como una parte del prototipo. Como disyuntiva, el desarrollo de prototipos puede producir un sistema que funcione. Las características de entrada y salida son desarrolladas junto con el código orientado hacia los procedimientos y archivos de datos.

Generación de código: La ventaja más visible de esta característica es la disminución del tiempo necesario para preparar un programa. Sin embargo, la generación del código también asegura una estructura estándar y consistente para el programa (lo que tiene gran influencia en el mantenimiento) y disminuye la ocurrencia de varios tipos de errores, mejorando de esta manera la calidad. Ninguna de las herramientas que existen en el presente es capaz de generar un código completo en los dominios.

Mejora en la habilidad para satisfacer los requerimientos del usuario: Tener los requerimientos correctos mejora la calidad de las prácticas de desarrollo. Las herramientas CASE disminuyen el tiempo de desarrollo, una característica que es importante para los usuarios. Las herramientas afectan la naturaleza y cantidad de interacción entre los encargados del desarrollo y el usuario. Las descripciones gráficas y los diagramas, así como los prototipos de reportes y la composición de las pantallas, contribuyen a un intercambio de ideas más efectivo.

Soporte interactivo para el proceso de desarrollo: Las herramientas CASE soportan pasos interactivos al eliminar el tedio manual de dibujar diagramas, elaborar catálogos y clasificar. Como resultado de esto, se anticipa que los analistas repasarán y revisarán los detalles del sistema con mayor frecuencia y en forma más consistente.

2.4.1 Visual Paradigm

Visual Paradigm para UML versión 8.0 (Unified Modeling Language) es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste, así como una serie de funcionalidades y ventajas que se listan a continuación:

- Disponible en Software Libre.
- Compatible con Win98/2000/XP/Vista/7
- Permite la creación de Diagramas de Procesos de Negocio.
- Modelado colaborativo con CVS y Subversion.
- Interoperabilidad con modelos UML 2 a través de XML.
- Ingeniería inversa (Código a modelo, código a diagrama) para Java, C++, Esquemas XML, XML, .NET exe/dll, CORBA IDL.
- Generación de código (Modelo a código, diagrama a código).
- Editor de Detalles de Casos de Uso (Entorno para la especificación de los detalles de los Casos de Uso, incluyendo la especificación del modelo general y de las descripciones de los Casos de Uso)
- Diagramas EJB[17] (Visualización de sistemas EJB).

[17] Enterprise JavaBeans: son componentes del contexto de servidor que cubren la necesidad de intermediar entre la capa web y diversos sistemas empresariales.

- Generación de código y despliegue de EJB´s (Generación de beans para el desarrollo y despliegue de aplicaciones).
- Diagramas de flujo de datos
- Generación de bases de datos (Transformación de diagramas de Entidad-Relación en tablas de base de datos).
- Ingeniería inversa de bases de datos (Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación).
- Generador de informes para generación de documentación.
- Distribución automática de diagramas (Reorganización de las figuras y conectores de los diagramas UML).
- Importación y exportación de ficheros XML.
- Grupo Glosario de Términos por la etiqueta.
- Permite escribir reglas de negocio.
- Permite dibujar el diagrama de datos.
- Diseño de la empresa con el diagrama de ArchiMate.
- Brinda la opción de proteger el diagrama por contraseña.
- Permite examinar los elementos del modelo de red.
- Brinda la opción de ajustar el tamaño de la aplicación de la fuente (COMPANY 2011).

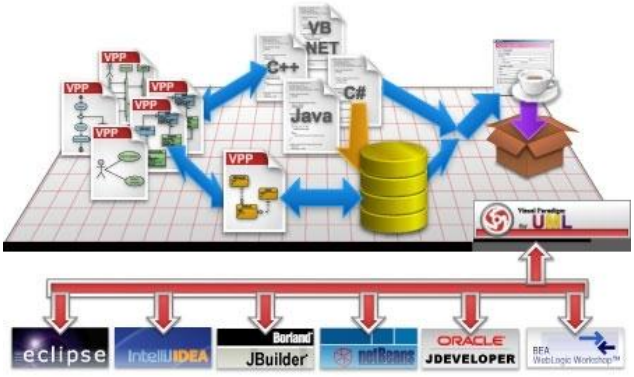


Figura 1 Funcionalidades y ventajas de Visual Paradigm para UML versión 8.0

2.5 El Proceso Unificado de Desarrollo: metodología a seguir

Según Jacobson, Booch y Rumbaugh, padres por excelencia de la Ingeniería de Software, se puede definir como un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas

software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos (JACOBSON 2000).

Es una infraestructura flexible de desarrollo de software que proporciona prácticas recomendadas probadas y una arquitectura configurable, es un proceso práctico. El también conocido como RUP (Rational Unified Process), es un proceso de desarrollo de software para la Ingeniería del Software Orientada a Objetos. RUP proporciona un enfoque disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo de software. Define, para cada etapa y cada disciplina, el flujo de trabajo, los trabajadores que intervienen, las actividades que realizan y los artefactos que se necesitan o producen. Su meta es asegurar la producción de software con la más alta calidad, que reúna las necesidades de los usuarios dentro del cronograma planeado y la inversión prevista. RUP deja bien definido y estructurado el proceso de diseño de un software. Define claramente quién es responsable, cómo se hacen las cosas y cuándo hacerlas.

2.6 Lenguaje Unificado de Modelado: UML

UML es un lenguaje para especificar (cuáles son las características de un sistema antes de su construcción), construir (a partir de los modelos especificados se pueden construir los sistemas diseñados), visualizar (permite expresar de una forma gráfica un sistema de forma que otro pueda entenderlo) y documentar (los propios elementos gráficos sirven como documentación del sistema desarrollado que puede servir para su futura revisión) los artefactos de un sistema de software Orientado a Objetos (OO). En UML los procesos de desarrollo son diferentes según los distintos dominios de trabajo; no puede ser el mismo el proceso para crear una aplicación en tiempo real, que el proceso de desarrollo de una aplicación orientada a gestión, por poner un ejemplo. Las diferencias son muy marcadas y afectan a todas las fases del proceso. El método del UML recomienda utilizar los procesos que otras metodologías tienen definidos (LAFUENTE 2001).

El lenguaje UML tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático: desde el análisis con los Casos de Uso, el diseño con los diagramas de clases, objetos, entre otros hasta la implementación y configuración con los diagramas de despliegue.

UML es ante todo un lenguaje. Un lenguaje proporciona un vocabulario y unas reglas para permitir una comunicación. En este caso, este lenguaje se centra en la representación gráfica de un sistema. UML resuelve de forma bastante satisfactoria un viejo problema del desarrollo de software como es su

modelado gráfico. Además, se ha llegado a una solución unificada basada en lo mejor que había hasta el momento, lo cual lo hace todavía más excepcional (ORALLO 2002).

2.7 Sistemas Gestores de Base de Datos

Se puede definir a un Sistema Gestor de Bases de Datos (en lo adelante SGBD) o Data Base Management System (DBMA) como una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos.

De esta manera se puede definir y detallar que las características principales que presentan los SGBD son las siguientes:

- Abstracción de la información: Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Así, se definen varios niveles de abstracción.
- Independencia: La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.
- Redundancia mínima: Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.
- Consistencia: En aquellos casos en los que no se ha logrado redundancia nula anteriormente expuesta, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.
- Seguridad: Los SGBD deben garantizar que la información contenida se encuentra almacenada de forma segura frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero despistado.
- Integridad: Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos

por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.

- Respaldo y recuperación: Los SGBD deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.
- Control de la concurrencia: En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información, bien para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias (INTEGRATOR 2009).

2.7.1 PostgreSQL

PostgreSQL es un Sistema Gestor de Bases de Datos Objeto-Relacional (ORDBMS) basado en el proyecto POSTGRES, de la Universidad de Berkeley. Es una derivación libre (OpenSource) de este proyecto, y utiliza el lenguaje SQL92/SQL99.

Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido, más tarde en otros sistemas de gestión comerciales. PostgreSQL se presenta como un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como pueden ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de esto, PostgreSQL no es un sistema de gestión de bases de datos puramente orientado a objetos.

A continuación se enumeran las principales características de este gestor de bases de datos:

1. Implementación del estándar SQL92/SQL99.
2. Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP), cadenas de bits, entre otros. También permite la creación de tipos propios.
3. Incorpora una estructura de datos array.
4. Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, entre otras.
5. Permite la declaración de funciones propias, así como la definición de disparadores.
6. Soporta el uso de índices, reglas y vistas.

7. Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
8. Permite la gestión de diferentes usuarios, como también los permisos asignados a ellos.

2.8 Conclusiones

El conocimiento de las ventajas que traería el uso de las herramientas y tecnologías antes expuestas para el desarrollo de la aplicación constituye una guía necesaria para un mejor trabajo. Se concluye, luego del estudio de las diferentes tecnologías de mayor auge a nivel mundial, que la utilización de tecnologías libres permitirá la obtención de un mayor rendimiento y un menor costo de producción del producto final. Luego de todo lo detallado anteriormente se puede afirmar que las bases para comenzar la construcción de la futura aplicación, en términos ingenieriles, están debidamente instauradas.

CAPÍTULO 3:

PRESENTACIÓN DE LA SOLUCIÓN PROPUESTA

3.1 Introducción

En el capítulo que se propone a continuación se realiza el proceso de modelación del sistema. Para ello se especifican los requerimientos funcionales y no funcionales que se consideran necesarios, se identificarán los actores, se describirán los principales Casos de Uso y se construirán varios diagramas para su mejor entendimiento.

3.2 Modelo de dominio

Un modelo del dominio puede ser pensado como un modelo conceptual de un sistema que describe las entidades implicadas en ese sistema y sus relaciones. El modelo del dominio se crea para documentar los conceptos dominantes y el vocabulario del sistema.

El modelo identifica las relaciones entre todas las entidades importantes dentro del sistema e identifica generalmente sus métodos y cualidades importantes. Esto significa que el modelo proporciona una vista estructural del sistema. Una ventaja importante de un modelo del dominio es describir y obligar alcance del sistema.

El modelo de dominio puede ser tomado como el punto de partida para el diseño del sistema. Esto es así ya que cuando se realiza la programación orientada a objetos, se supone que el funcionamiento interno del software va a imitar en alguna medida a la realidad, por lo que el mapa de conceptos del modelo de domino constituye una primera versión del sistema.

Por otra parte, cuando se sigue una aproximación *Centrada en Casos de Uso* como RUP/UP, el modelo de dominio es utilizado como entrada en la tarea análisis de los casos de uso en la construcción de los llamados escenarios de análisis (GARCERANT 2008).

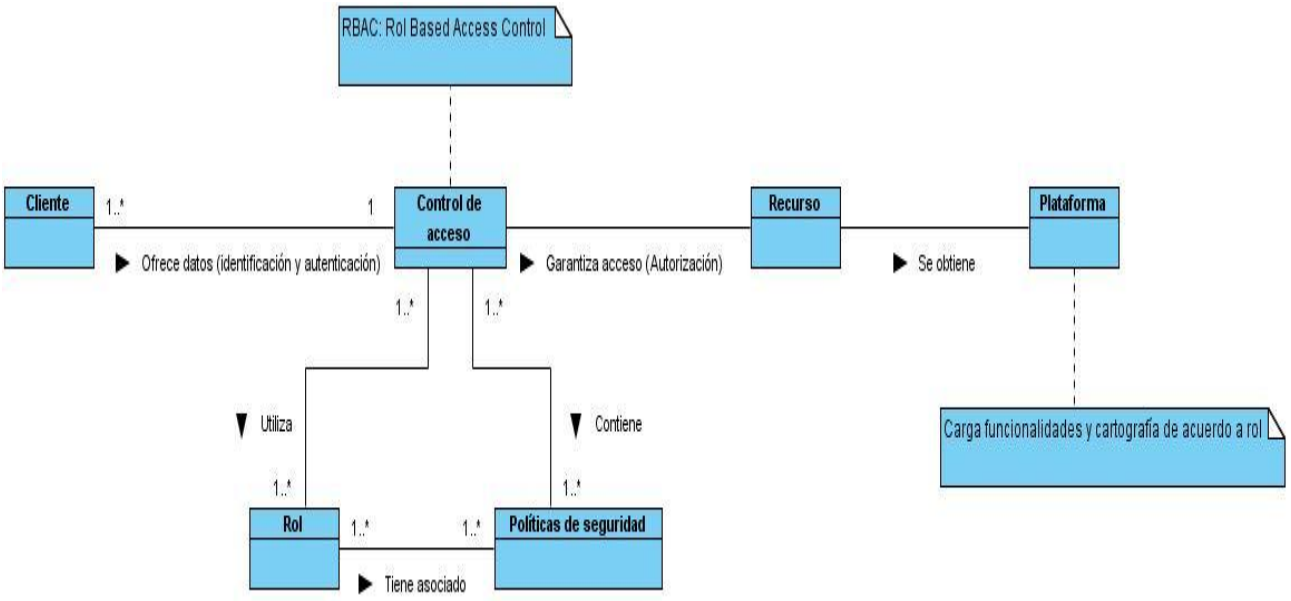


Figura 2 Diagrama del Modelo de Dominio

3.2.1 Descripción de las clases del dominio

Cliente: Persona que necesita un software para trabajar o consultar información incluida en el mapa. Tiene asignado un rol de acuerdo a sus funciones y a la información a la cual tiene acceso.

Control de acceso: Proceso por el cual se le concederán los permisos a los usuarios o grupos de usuarios de acceder a determinadas funcionalidades y cartografía.

Rol: Determina los permisos de acciones que puede desempeñar un usuario frente a una aplicación específica.

Políticas de Seguridad: Contiene los permisos definidos para cada grupo de usuarios.

Recurso: Los datos contenidos en el sistema de información, así como:

a) Un servicio del sistema.

b) Un elemento del equipo de sistema (como hardware, software, o documentación).

c) Una instalación que alberga las operaciones del sistema y el equipo.

Plataforma: Constituye un software donde se ejecutarán y desarrollarán todos los procesos para personalizar un cliente determinado.

3.3 Requerimientos Funcionales

El subsistema debe ser capaz de:

1. Eliminar el acceso de un usuario a un mapa determinado.
2. Actualizar el acceso a las capas de acuerdo a un rol determinado.
3. Adicionar usuario al subsistema.
4. Eliminar usuario del subsistema.
5. Modificar datos de un usuario presente en el subsistema.
6. Listar usuarios registrados en el subsistema.
7. Buscar un usuario registrado en el subsistema.
8. Autenticar usuario.
9. Adicionar un rol al subsistema.
10. Eliminar rol del subsistema.
11. Modificar la información registrada de un rol del subsistema.
12. Listar los roles registrados en el subsistema.
13. Asociar un determinado rol a un usuario.
14. Disociar un rol de un usuario.
15. Visualizar reporte de datos de los usuarios.
16. Dar permisos de acceso a los objetos de la Base de Datos.
17. Denegar permisos de acceso a los objetos de la Base de Datos
18. Modificar los permisos de acceso a los objetos de la Base de Datos.
19. Eliminar permisos de acceso a los objetos de la Base de Datos.
20. Listar los permisos asignados a un rol determinado.

21. Gestionar datos de la conexión con la Base de Datos.

22. Asignar permiso de acceso a funcionalidades a roles registrados.

23. Eliminar permiso de acceso a funcionalidades registradas.

3.4 Requerimientos No Funcionales

Requerimientos de usabilidad

1. El subsistema podrá ser usado por personas con conocimientos básicos en el manejo de computadoras.
2. La interfaz gráfica de usuario debe permitir completar las interacciones necesarias en la menor cantidad de secuencias posibles, para el logro de una mejor interacción con el usuario.

Apariencia o Interfaz externa

1. El subsistema debe tener una apariencia profesional, un diseño gráfico sencillo y agradable a la vista.
2. La interfaz debe ser de fácil comprensión en su funcionamiento permitiendo la utilización del subsistema sin mucho adiestramiento.
3. El subsistema debe contar con el nombre de la aplicación en la parte superior mediante una barra de título así como el nombre de cada una de las ventanas en dependencia de la funcionalidad que esté presente.

Requerimientos de portabilidad y operatividad

1. El subsistema debe ser multiplataforma, disponible fundamentalmente para las distribuciones más populares de GNU/Linux y Windows NT 2000/XP/Vista.

Requerimientos de rendimiento

1. El tiempo de respuesta, la velocidad para procesar la información, la actualización y la recuperación oscilará entre 1 y 5 segundos excepto la carga inicial del sistema que pudiera llegar hasta los 10 segundos.

Requerimientos de software del sistema

Para las PCs clientes:

- Sistema operativo: GNU/Linux y Windows NT 2000/XP/Vista.

Para los servidores:

- Sistemas operativos GNU/Linux o Windows Server 2000 o superior.
- PostgreSQL versión 8.4 como Sistema Gestor de Base de Datos con su extensión espacial PostGIS.

Requerimientos de confiabilidad y seguridad

1. Al subsistema se accederá a través de la autenticación convencional: usuario y contraseña.
2. Cada usuario debe tener exclusivamente los permisos necesarios para realizar las operaciones que le sean permitidas.
3. Debe mantenerse la consistencia de los datos en correspondencia con la realidad.
4. La herramienta a implementar debe tener soporte para recuperación ante fallos y errores.
5. La información manejada por el subsistema estará protegida de accesos no autorizados y divulgación.
6. Es importante que el subsistema presente un mecanismo de respuesta rápida ante fallos y que en caso de ocurrencia se minimicen las pérdidas de información por lo que deberá existir un plan de salvas y mantenimiento, garantizando con esto una rápida protección y recuperación ante un problema dado.

Requerimientos de hardware del sistema

Para las PCs clientes:

- Se requiere tengan tarjeta de red con velocidad de 100 Mbps
- 512 Mb de memoria RAM como mínimo.
- 100 Mb de almacenamiento interno como mínimo.
- Procesador 512 MHz como mínimo.

Para los servidores:

- Se requiere tarjeta de red con velocidad de 100 Mbps
- El Servidor de BD tenga como mínimo 2 Gb de RAM y 10 Gb de disco duro.
- Procesador 3 GHz como mínimo.

3.5 Descripción del sistema propuesto

El modelo de Casos de Uso ayuda al cliente, a los usuarios y a los desarrolladores a llegar a un acuerdo sobre cómo utilizar el sistema. La mayoría de los sistemas tienen muchos tipos de usuarios. Cada tipo de usuario se representa mediante un actor. Los actores utilizan el sistema al interactuar con los Casos de Uso (JACOBSON 2000).

3.5.1 Descripción de los actores

Actores del Sistema	Descripción
Usuario	Constituye cualquier cliente que desee interactuar con la aplicación.
Administrador General	Es el encargado de gestionar los permisos de todos los roles y usuarios que se definan, controlar el acceso a la aplicación, así como gestionar los permisos de los diferentes usuarios sobre las capas que se carguen de los mapas, controlar y definir el acceso a la cartografía base.

Tabla 1 Actores del sistema

3.5.2 Diagrama de Casos de Uso del Sistema

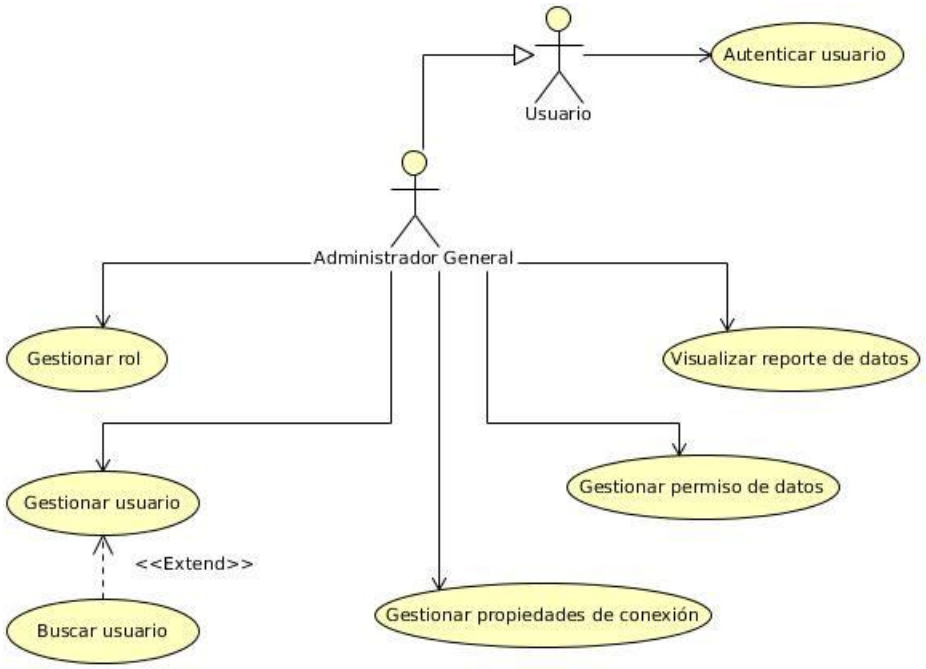


Figura 3 Diagrama de Casos de Uso del Sistema

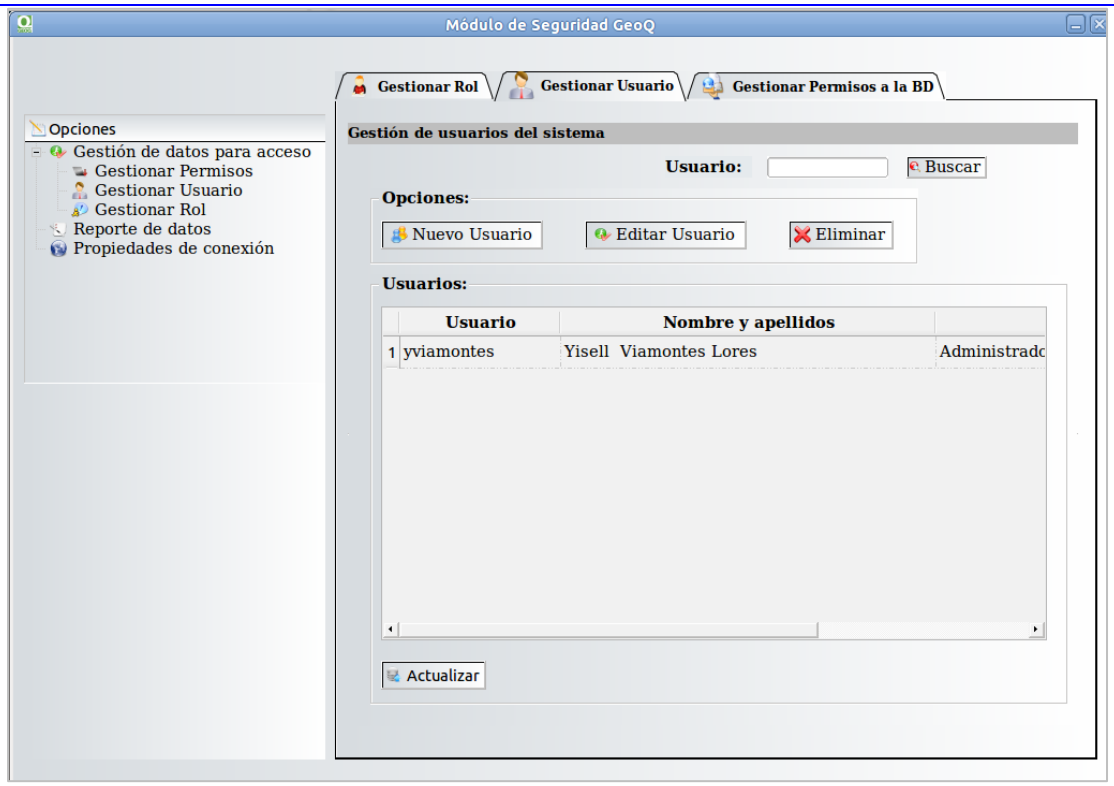
3.5.3 Descripción textual de los Casos de Uso del Sistema

A continuación se muestra la descripción textual del Caso de Uso *Gestionar usuario*, el resto de las descripciones pueden ser consultadas en el Anexo 1.

Caso de Uso: Gestionar usuario

Caso de Uso:	Gestionar usuario
Actores:	Administrador General
Resumen:	El Caso de Uso se inicia cuando el Usuario selecciona la opción Gestionar usuario, y termina cuando el Usuario finaliza las operaciones guardando los cambios.
Precondiciones:	El Usuario debe estar autenticado y debe ser Administrador del Sistema.
Referencias	RF 3, RF 4, RF 5, RF 6, RF 13, RF 14
Prioridad	Crítico

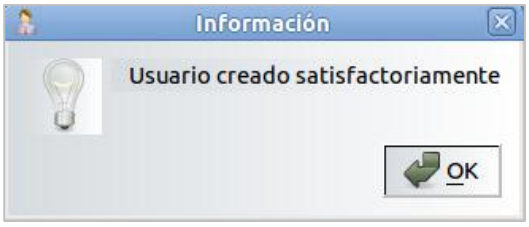
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador elige la opción “Gestionar usuario”.	2. El Sistema muestra una lista con los usuarios registrados y además permite realizar las operaciones de creación de usuarios, edición y eliminación.(Ver Prototipo de interfaz: Gestionar usuario) - Si el usuario elige “Nuevo Usuario” ver Sección “Nuevo usuario”. - Si el usuario elige “Editar Usuario” ver Sección “Editar usuario”. - Si el usuario elige “Eliminar Usuario” ver Sección “Eliminar usuario”.



Prototipo de interfaz: Gestionar usuario

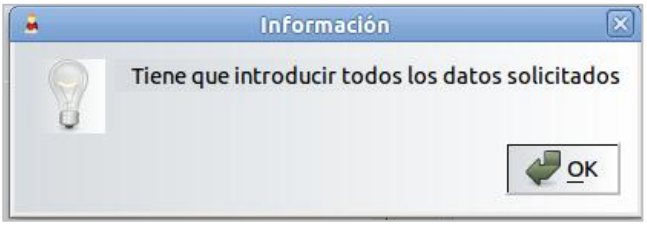
Sección "Nuevo usuario"	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador elige la opción "Nuevo Usuario".	2. El Sistema muestra la opción "Nuevo usuario" con un listado "Roles" con todos los roles existentes, un campo "Usuario", "Nombre", "Apellidos" y otro "Contraseña". (Ver Prototipo de interfaz: Nuevo usuario)
3. El Administrador ingresa los datos correspondientes y selecciona la opción "Aceptar".	4. El Sistema adiciona el nuevo usuario a la Base de Datos, actualiza el listado de usuarios y muestra un mensaje de confirmación de que la acción se consumó satisfactoriamente. (Ver Prototipo de interfaz: Mensaje de confirmación)

Prototipo de Interfaz: Nuevo usuario

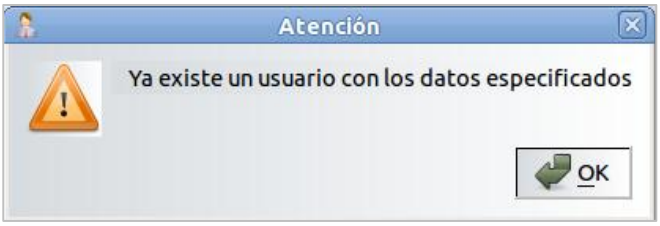


Prototipo de interfaz: Mensaje de confirmación

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3. El Administrador no ingresa los datos correspondientes y selecciona la opción “Aceptar”.	4. El Sistema muestra un mensaje de error informando que es necesario ingresar todos los datos solicitados.(Ver Prototipo de interfaz: Mensaje de error de ausencia de datos)
3. El Administrador ingresa el nombre de un usuario existente y selecciona la opción “Aceptar”.	4. El Sistema muestra un mensaje indicando que ya el usuario existe en el Sistema.(Ver Prototipo de interfaz: Mensaje de existencia de usuario)



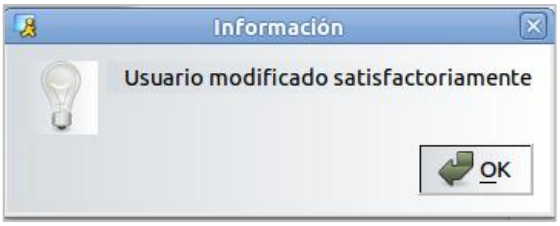
Prototipo de interfaz: Mensaje de error de ausencia de datos



Prototipo de interfaz: Mensaje de existencia de usuario

Sección "Editar usuario"	
Acción del Actor	**Respuesta del Sistema**
	1. El Sistema muestra un listado con los usuarios registrados.
2. El Administrador escoge el usuario del cual desea editar sus datos y selecciona la opción "Editar Usuario".	3. El Sistema muestra los datos registrados del usuario seleccionado. (Ver Prototipo de interfaz: Editar usuario)
4. El Administrador ingresa los datos que desea modificar del usuario y selecciona la opción "Aceptar".	5. El Sistema guarda los nuevos datos del usuario en la Base de Datos, actualiza el listado de usuarios y muestra un mensaje de confirmación de que la acción se consumó satisfactoriamente. (Ver Prototipo de interfaz: Mensaje de confirmación)

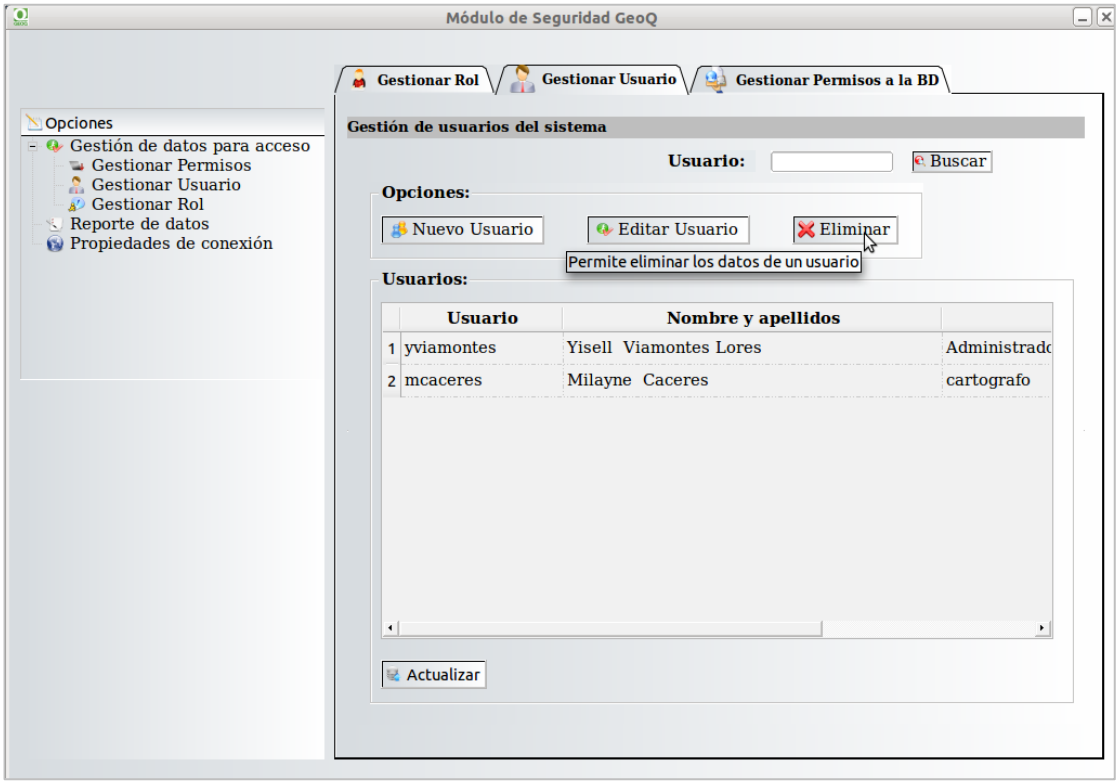
Prototipo de interfaz: Editar usuario



Prototipo de interfaz: Mensaje de confirmación

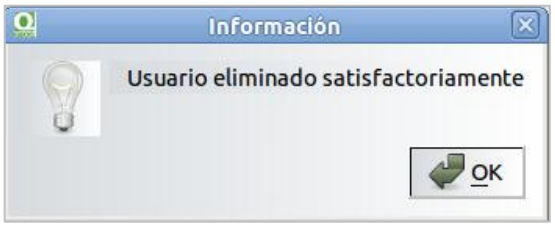
Sección "Eliminar usuario"	
Acción del Actor	**Respuesta del Sistema**
	1. El Sistema muestra un listado de todos los usuarios registrados en el Sistema.(Ver Prototipo de interfaz: Eliminar usuario)
2. El Administrador elige el usuario a eliminar y selecciona la opción "Eliminar Usuario".	3. El Sistema muestra un mensaje preguntando si se desea eliminar permanentemente los datos del usuario seleccionado.(Ver Prototipo de interfaz: Mensaje de selección de opción)
4. El Administrador selecciona la opción "Sí".	5. El Sistema elimina el usuario de la Base de Datos y muestra un mensaje de confirmación de que la acción se consumó satisfactoriamente. (Ver Prototipo de interfaz:

	Mensaje de confirmación)



Prototipo de interfaz: Eliminar usuario

Prototipo de interfaz: Mensaje de selección de opción



Prototipo de interfaz: Mensaje de confirmación

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
2. El Administrador no elige el usuario	3. El Sistema muestra un mensaje indicando

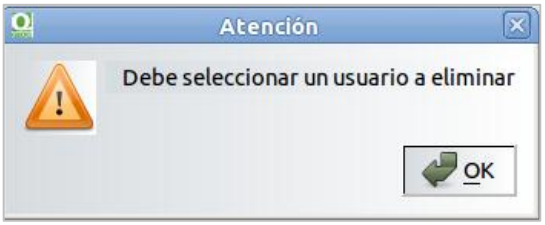
a eliminar y selecciona la opción “Eliminar”.	que se debe seleccionar un Usuario. (Ver Prototipo de interfaz: Mensaje de error no selección de usuario)
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
4. El Administrador selecciona la opción “No”.	5. El Sistema cierra la ventana y no realiza ningún cambio.
 **Prototipo de interfaz: Mensaje de error no selección de usuario**	
Pos-condiciones	Se gestionan las configuraciones de los datos del usuario.

Tabla 2 Descripción Textual CUS: Gestionar usuario

3.6 Conclusiones

1. Todos los Casos de Uso del Sistema, excepto el de “Visualizar reporte de datos” y “Buscar usuario”, se consideran críticos, constituyendo estos la propuesta para una arquitectura candidata.
2. El subsistema propuesto es multiplataforma, intuitivo y sencillo, para lograr una manipulación favorable al usuario.
3. El subsistema no requiere de elevadas prestaciones de hardware para su ejecución.

CAPÍTULO 4:

CONSTRUCCIÓN DE LA SOLUCIÓN PROPUESTA

4.1 Introducción

En el presente capítulo se describe la solución propuesta en términos de diagramas de clases del diseño, cada uno asociado a cada Caso de Uso del Sistema definido anteriormente por su nivel de complejidad. Se describen los patrones de diseño que se utilizaron en el desarrollo de la aplicación, en consonancia con la arquitectura definida. Se presenta la Base de Datos tanto a través del Diagrama Entidad-Relación como con el Diagrama de Clases Persistentes. Se presenta el modelo de implementación resultante del diseño.

4.2 Arquitectura y diseño

La arquitectura no es el software operativo, en cambio, es una representación que permite que un ingeniero de software: 1) analice la efectividad del diseño para cumplir con los requisitos establecidos, 2) considere opciones arquitectónicas en una etapa en que aún resulta relativamente fácil hacer cambios al diseño y 3) reduzca los riesgos asociados con la construcción del software (PRESSMAN 2005).

La Arquitectura de Software (en lo adelante AS) es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones (REYNOSO Marzo 2004).

Según un documento emitido por la IEEE Std. 1471-2000 se define que la AS es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución.

El profesor Roger S. Pressman, autoridad internacionalmente reconocida en la mejora de procesos de software y en tecnologías de Ingeniería de Software, define que: “El diseño arquitectónico representa la estructura de datos y los componentes del programa necesarios para construir un sistema computacional. Asume el estilo arquitectónico que tomará el sistema, la estructura y las propiedades de los componentes

que constituyen el sistema y las interrelaciones entre todos los componentes arquitectónicos de un sistema".

Finalmente, se asume como arquitectura de un programa o sistema de cómputo la estructura o las estructuras del sistema, que incluyen los componentes del software, las propiedades visibles externamente de esos componentes y las relaciones entre ellos.

4.3 Patrones arquitectónicos

Un estilo arquitectónico, a veces denominado un patrón arquitectónico, es un conjunto de principios, un patrón de grano grueso que proporciona un marco abstracto para una familia de sistemas. Un estilo arquitectónico mejora la separación y promueve la reutilización del diseño, aportando soluciones a los problemas que se repiten con frecuencia. Se puede conceptualizar de la siguiente manera: "familia de sistemas en términos de un patrón de organización estructural. Más específicamente, un estilo arquitectónico determina el vocabulario de los componentes y conectores que se pueden utilizar en casos de ese estilo, junto con un conjunto de restricciones sobre cómo pueden ser combinados. Estos pueden incluir restricciones topológicas en las descripciones arquitectónicas (por ejemplo, sin ciclos). Otras restricciones, por ejemplo, que tienen que ver con la semántica de la ejecución, también podría ser parte de la definición de estilo" (GARLAN 1994).

Una vez realizado el estudio de las características principales de los patrones arquitectónicos reconocidos se determinó que era factible la aplicación de la arquitectura en tres capas, y la implementación de patrones orientados a objetos y basados en componentes. A continuación se argumentan teóricamente los criterios asumidos para las selecciones anteriores y los principios básicos presentes en los mismos.

Arquitectura en tres capas

La arquitectura de una aplicación es la vista conceptual de la estructura de esta. Toda aplicación contiene código de presentación, código de procesamiento de datos y código de almacenamiento de datos. La arquitectura de las aplicaciones difiere según como está distribuido este código (CORNEJO 2006).

La estrategia tradicional de utilizar aplicaciones compactas causa gran cantidad de problemas de integración en sistemas software complejos como pueden ser los sistemas de gestión de una empresa o los sistemas de información integrados consistentes en más de una aplicación. Estas aplicaciones suelen encontrarse con importantes problemas de escalabilidad, disponibilidad, seguridad e integración.

Para solventar estos problemas se ha generalizado la división de las aplicaciones en capas: una capa que sirve para guardar los datos (base de datos), una capa para centralizar la lógica de negocio (modelo) y por último una interfaz gráfica que facilite al usuario el uso del sistema.

Esto permite distribuir el trabajo de creación de la aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto de los niveles, de forma que basta con conocer la API que existe entre niveles.

- **Capa de Presentación:** Esta capa se comunica únicamente con la capa de negocio llevando y trayendo los datos o registros necesarios, es la interfaz gráfica del programa y debe ser lo más amena posible para una mejor comunicación con el usuario.
- **Capa de negocio:** Constituye la capa donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todos los procesos que deben realizarse.
- **Capa de acceso a datos:** Constituye la capa donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Una vez concluido el estudio acerca de la aplicación de dicha arquitectura y los resultados de dicho ejercicio se definen una serie de ventajas que se listan a continuación:

- Esta arquitectura permite que el desarrollo se pueda llevar a cabo en varios niveles.
- Desarrollos paralelos (en cada capa).
- Permite la obtención de aplicaciones más robustas debido al encapsulamiento.
- En caso de que sobrevenga algún cambio sólo se ataca al nivel requerido sin tener que revisar entre el código mezclado.
- Mantenimiento y soporte más sencillo (es más sencillo cambiar un componente que modificar una aplicación monolítica).
- Mayor flexibilidad (se pueden añadir nuevos módulos para dotar al sistema de nueva funcionalidad).

- La principal ventaja de una aplicación distribuida bien diseñada es su buen escalado, es decir que pueda manejar muchas peticiones con el mismo rendimiento simplemente añadiendo más hardware.

Arquitectura basada en componentes

La arquitectura basada en componentes consiste en una rama de la Ingeniería de Software en la cual se trata con énfasis la descomposición en componentes funcionales. Esta descomposición permite convertir componentes del software pre-existentes en piezas más grandes de software. Este proceso de construcción de una pieza de software con componentes ya existentes, da origen al principio de reutilización del software, mediante el cual se promueve que los componentes sean implementados de una forma que permita su utilización funcional sobre diferentes sistemas en el futuro.

De forma evidente se puede determinar que, el principal elemento de software dentro de una arquitectura basada en componentes es precisamente el componente de software, es por ello que se pueden definir diversos principios que definen a un componente de software como elemento de la arquitectura:

1. Reutilizables: Los componentes son usualmente diseñados para ser reutilizados en diferentes escenarios en diferentes aplicaciones. Sin embargo, algunos componentes pueden ser diseñados para una tarea específica.
2. Reemplazables: Los componentes pueden ser fácilmente sustituidos por otros componentes similares.
3. No contexto específico: Los componentes son diseñados para operar en diferentes ambientes y contextos. Información específica como el estado de los datos deben ser pasadas al componente en vez de incluirlos o permitir al componente acceder a ellos.
4. Extensible: Un componente puede ser ampliado a partir de los actuales componentes para proporcionar un comportamiento nuevo.
5. Encapsulado: Los componentes exponen interfaces que permiten a la persona que llama utilizar su funcionalidad, y no revelar los detalles de los procesos internos o variables internas o estatales.
6. Independiente: Los componentes están diseñados para tener un mínimo de dependencia de otros componentes. Por lo tanto los componentes pueden ser desplegados en cualquier ambiente adecuado sin afectar otros componentes o sistemas.

Arquitectura Orientada a Objetos

Un diseño orientado a objetos define al sistema como una serie de objetos cooperantes, en lugar de un conjunto de rutinas o instrucciones de procedimiento. Los objetos son discretos, independientes, y ligeramente acoplados; se comunican a través de interfaces, llamando a los métodos o accediendo a las propiedades de otros objetos, y mediante el envío y recepción de mensajes. Los principios claves de la arquitectura orientada a objetos son los siguientes:

1. Abstracción: Esto le permite reducir una compleja operación en una generalización que conserva las características de base de la operación. Por ejemplo, una interfaz abstracta puede ser una definición bien conocida que apoya las operaciones de acceso a datos utilizando métodos simples, tales como obtener y actualizar. Otra forma de abstracción de los metadatos: podría ser utilizado para proporcionar una asignación entre los dos formatos que contienen datos estructurados.
2. Composición: Los objetos pueden ser montados a partir de otros objetos, y pueden optar por ocultar estos objetos internos de otras clases o exponerlos como interfaces simples.
3. Herencia: Los objetos pueden heredar de otros objetos, y usar funcionalidades del objeto base o reemplazarlas implementando un nuevo comportamiento. Por otra parte, la herencia permite que el mantenimiento y la actualización sean más sencillos, ya que los cambios en el objeto base se propagan automáticamente a los objetos heredados.
4. Encapsulación: Los objetos exponen funcionalidades sólo a través de métodos, propiedades y eventos, y ocultan los detalles internos tales como el estado y las variables de otros objetos. Esto hace que sea más fácil de actualizar o reemplazar los objetos, siempre que sus interfaces sean compatibles, sin afectar a otros objetos y código.
5. Polimorfismo: Esto le permite reemplazar el comportamiento de un tipo de base que apoya las operaciones en la aplicación mediante la implementación de nuevos tipos que son intercambiables con el objeto existente.
6. Disociación: Los objetos pueden ser disociados de los consumidores mediante la definición de una interfaz abstracta que implementa el objeto y el consumidor pueda entender. Esto le permite proporcionar implementaciones alternativas sin afectar a los consumidores de la interfaz.

Los usos comunes del estilo orientado a objetos incluyen la definición de un modelo de objetos compatible con operaciones complejas, y la definición de los objetos que representan objetos del mundo real dentro de un dominio de negocio (por ejemplo, un cliente o una orden).

4.3.1 Patrones de diseño

Se pueden definir como: soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan. Los patrones de diseño no son fáciles de entender, pero una vez entendido su funcionamiento, los diseños serán mucho más flexibles, modulares y reutilizables (GRACIA 2005).

Según la perspectiva presentada por Analistas de Sistemas que trabajan con tecnologías Microsoft en la página oficial de esta compañía se hace referencia a los patrones de diseño como *el esqueleto de las soluciones a problemas comunes en el desarrollo de software (…), brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares* (TEDESCHI 2011).

Se definió que era necesaria la aplicación de patrones GoF[18] y GRASP[19], los cuales se relacionan a continuación:

- Fachada (Facade): Proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar.
- Observador (Observer): Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos.
- Alta Cohesión: Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificables.
- Singleton: Está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella. La instrumentación del patrón puede ser delicada en programas con múltiples hilos de ejecución. Si dos hilos de ejecución intentan crear la instancia al mismo tiempo y esta no existe todavía, sólo uno de ellos debe lograr crear el objeto. La solución clásica para este problema es utilizar exclusión mutua en el método de creación de la clase que implementa el patrón.
- Table Data Gateway: Se encarga de generar todo el SQL para acceder a una sola tabla: select, insert, update y delete. También puede tener otros métodos para interactuar con la base de datos. En principio, no representa mayor lógica de negocio. Permite generar un objeto que actúa como

[18] Gang of Four

[19] General Responsibility Assignment Software Patterns

Gateway (puerta de enlace) a la base de datos, una instancia que maneja todas las filas en una tabla.

- Registro (Registry): El Registro proporciona un mecanismo para almacenar datos de manera global de una manera bien gestionada. Permite generar un objeto conocido que otros objetos pueden utilizar para encontrar objetos y servicios comunes. Registro es en esencia un objeto global o al menos parece uno, (Singleton) el cual es accedido directamente por otros objetos.
- Controlador (Controller): Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones: el "sistema" global (controlador de fachada); la empresa u organización global (controlador de fachada), algo en el mundo real que es activo (por ejemplo; el papel de una persona) y que pueda participar en la tarea (controlador de tareas); un manejador artificial de todos los eventos del sistema de un caso de uso(controlador de casos de uso) (LARMAN 1999).

4.4 Modelo de Diseño

El modelo de diseño constituye un modelo de objetos que describe la realización física de los Casos de Uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas en el entorno de la implementación, tienen impacto en el sistema a considerar. Además, el modelo de diseño se muestra como una abstracción de la implementación del sistema y es, de ese modo, utilizado como una entrada fundamental de las actividades de implementación.

4.4.1 Diagrama de Clases del Diseño

El diagrama de clases de diseño permite describir gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Dado que los diagramas de clases son los más utilizados en el modelo de sistemas orientados a objetos y que unos de los patrones arquitectónicos propuesto es Orientado a Objetos se proponen los siguientes diagramas de clases que muestran un conjunto de clases identificadas para el desarrollo del subsistema propuesto, interfaces y colaboraciones, así como sus relaciones. Los diagramas de clases permiten modelar la vista de diseño estática del sistema y además se obtuvieron como resultado del refinamiento del modelo conceptual. Los diagramas de clases del diseño correspondientes a los demás Casos de Uso del Sistema pueden ser consultados en el Anexo 2.

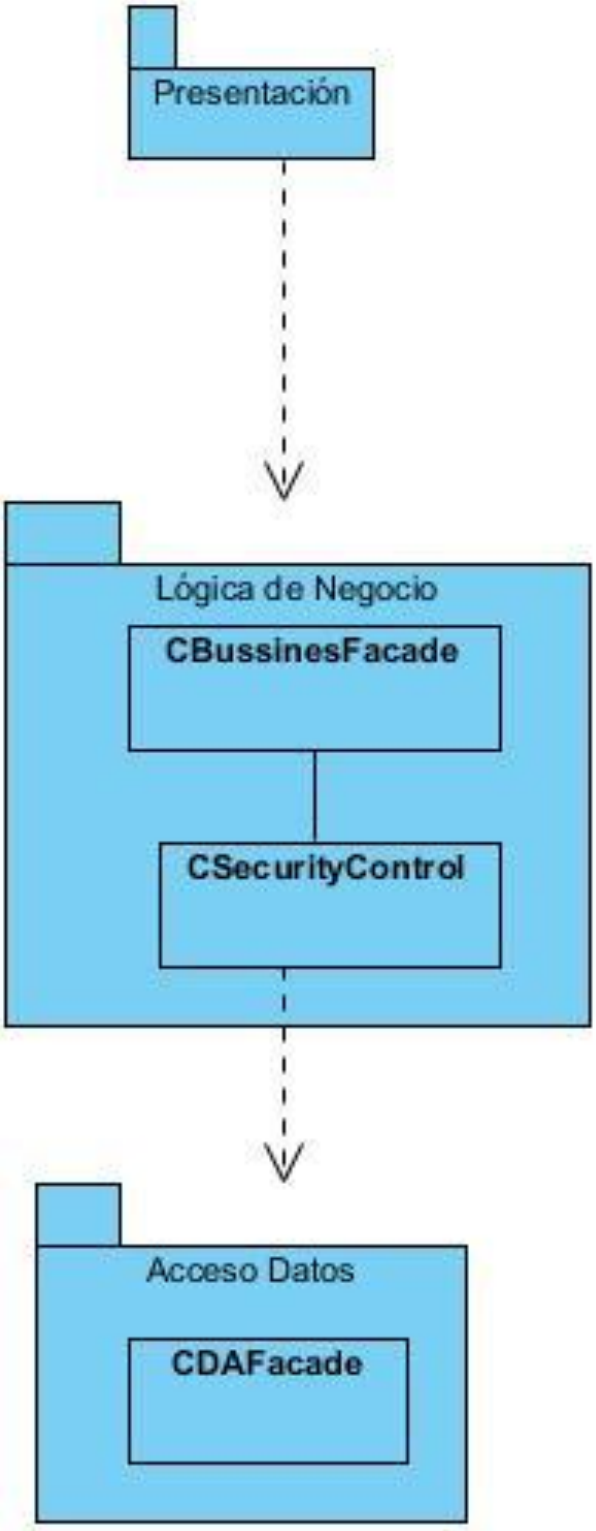


Figura 4 Diagrama de paquetes del diseño. (Vista general)

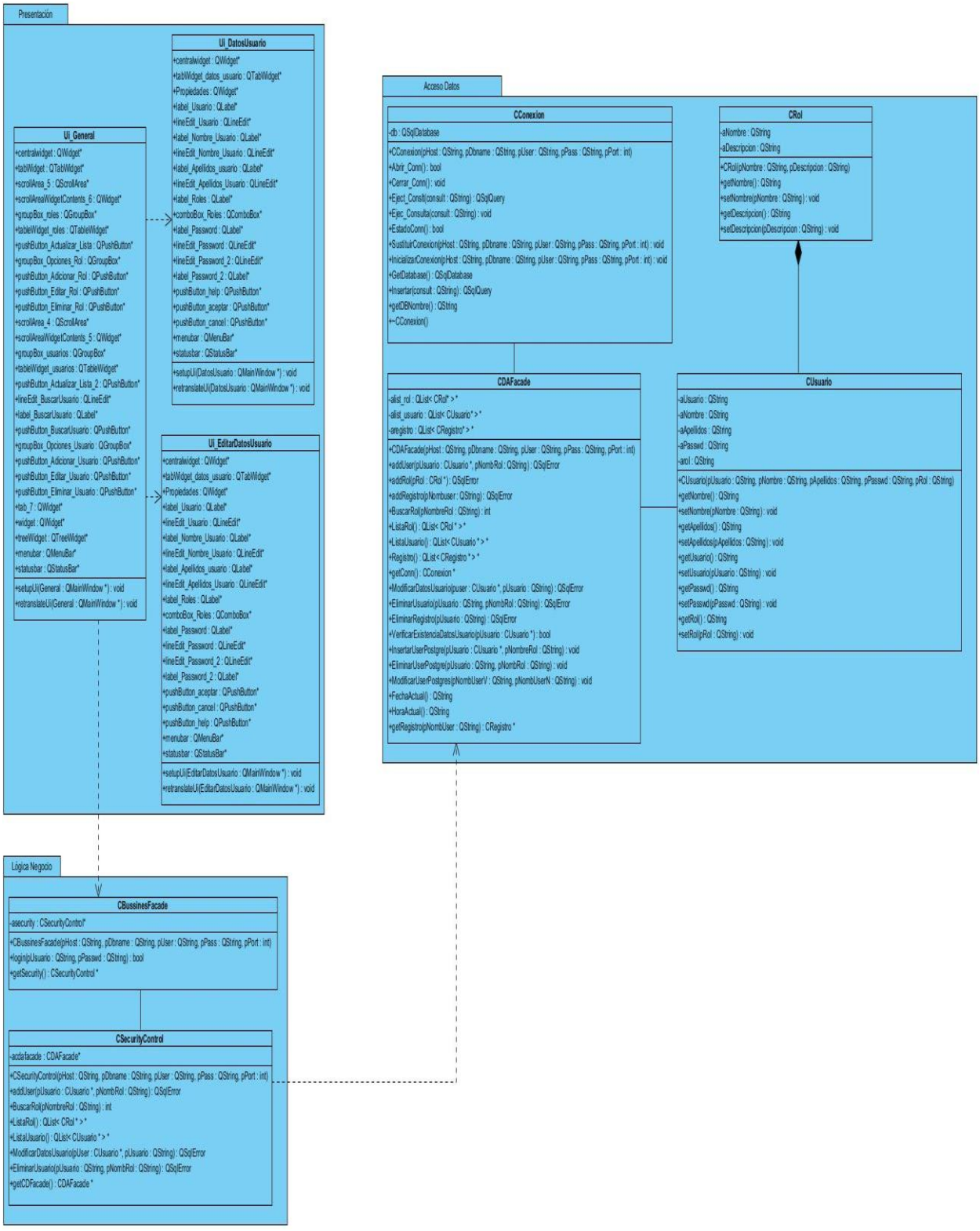


Figura 5 Diagrama de clases del diseño: CU Gestionar usuario

4.5 Principios del Diseño

4.5.1 Estándares de la Interfaz de la Aplicación

Las interfaces que presentará el subsistema (interfaz de autenticación e interfaz de gestión de la seguridad), de acuerdo con las normas establecidas dentro de la arquitectura definida y las consideraciones del diseño asumidas por el proyecto productivo, deben ir en concordancia con los colores presentes en la interfaz principal de la plataforma GeoQ, incluyendo sus logos, imágenes representativas y algunos otros elementos.

El contenido a mostrar se distribuirá en tres partes fundamentales:

El encabezado: Estará situado ocupando toda la parte superior de las interfaces y en él se encontrarán el logo de GeoQ y la identificación de la interfaz.

El menú lateral: La interfaz de gestión de seguridad presentará un menú lateral en consonancia con las funcionalidades presentes, dicho menú estará situado ocupando la parte izquierda, desde donde termina el encabezado hasta el final de la página.

El área de trabajo: Como cualquier diseño estándar, estará situada al centro, desde donde termina el encabezado hasta el final de la página. Ocupa la mayor parte del espacio ya que incluirá todas las actividades a considerar dentro del sistema que aquí se diseña.

Para lograr un adecuado diseño de la aplicación propuesta se requiere seguir una serie de estándares que van dirigidos a garantizar la consistencia de dicha aplicación y una mayor aceptación por parte de los interesados. Estos estándares se listan a continuación:

- Brindar una interfaz sencilla, de manera tal que cualquier persona con un mínimo dominio de la computación pueda aprender a trabajar con la aplicación.
- Garantizar la legibilidad de manera que exista contraste de los colores de los textos con el fondo y el tamaño de la fuente para que sea lo suficientemente adecuado a la vista del usuario.
- Mostrar al usuario solamente aquellas opciones a las que, dado los permisos asignados a su rol, tiene derecho a acceder.
- Mostrar al usuario, siempre que vaya a realizar una acción relevante sobre el sistema, un mensaje de confirmación que le permita asegurarse que es correcta la opción seleccionada.

4.6 Diseño de la Base de Datos

La Base de Datos necesita de una definición de su estructura, de manera que permita almacenar datos, reconocer el contenido, y recuperar la información. Para lograr un diseño de la Base de Datos lo más acorde a las necesidades reales propuestas se hace necesario seguir un conjunto de pasos que comienzan con definir las clases persistentes, luego refinarlas y clasificarlas junto con sus atributos, para más tarde realizar el diagrama de clases persistentes.

Con la aplicación de la persistencia se puede lograr que los objetos mantengan su valor en el tiempo y el espacio. Durante el diseño se identifican todas las clases persistentes, específicamente en el diagrama de clases persistentes; este diagrama muestra la estructura lógica de la base de datos mediante clases, traduciendo sus atributos a columnas de las tablas.

A continuación se muestra el diagrama de clases persistentes asociado a la aplicación a desarrollar y posteriormente se representa el diagrama entidad relación de la Base de Datos que se genera del diagrama anterior.

4.6.1 Diagrama de Clases Persistentes

La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. Las clases persistentes por lo general tienen como origen las clases clasificadas como entidad porque ellas modelan la información y el comportamiento asociado de algún fenómeno o concepto, como una persona, un objeto del mundo real o un suceso (GONZÁLEZ, A.H, 2004).

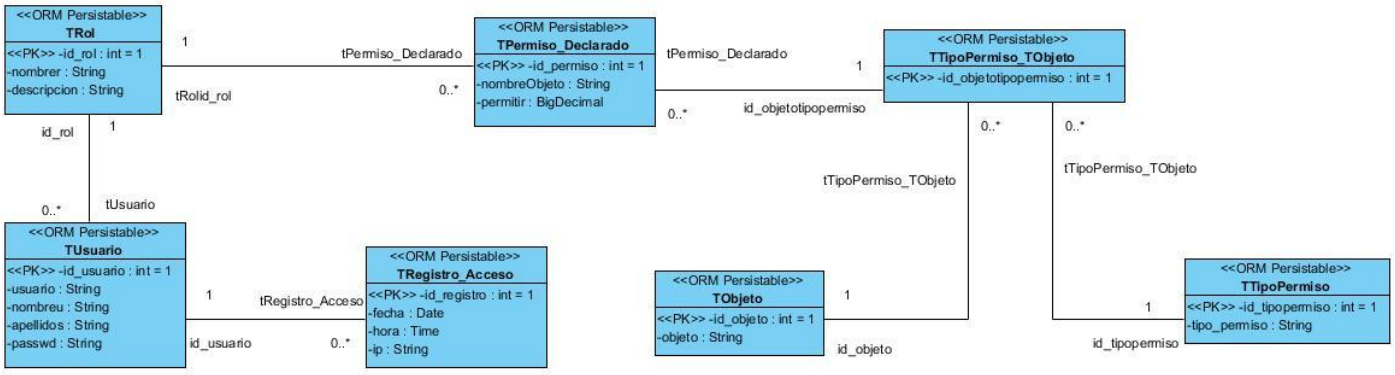


Figura 6 Diagrama de clases persistentes: Subsistema de seguridad

4.6.2 Diagrama Entidad-Relación

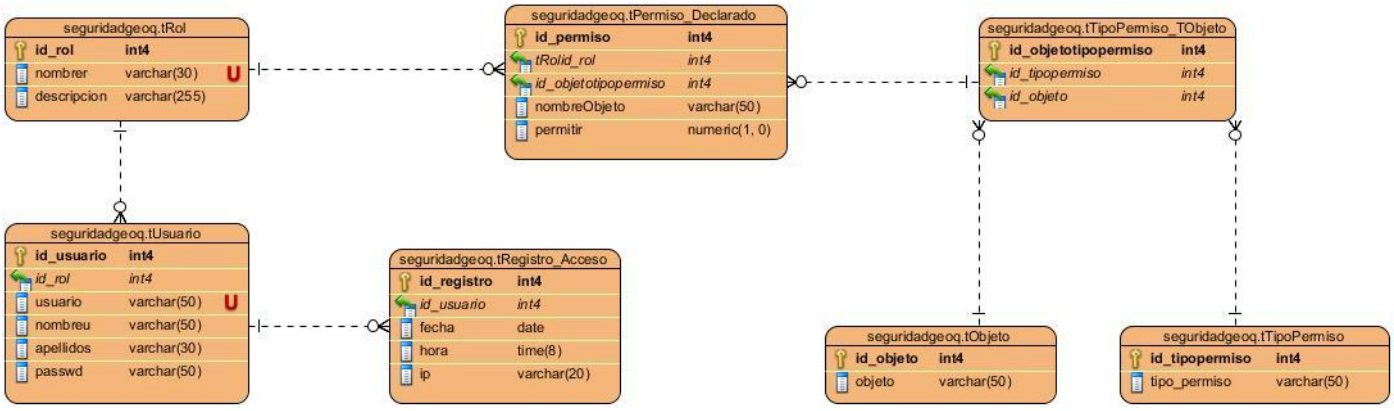


Figura 7 Diagrama de Entidad-Relación: Subsistema de seguridad

4.7 Generalidades de la Implementación

4.7.1 Modelo de Despliegue

El diagrama de distribución o diagrama de despliegue muestra la disposición física de los distintos nodos que componen el sistema y el reparto de los componentes sobre dichos nodos. Un nodo es un elemento físico que existe en tiempo de ejecución y representa un recurso computacional, que generalmente tiene algo de memoria y, a menudo, capacidad de procesamiento.

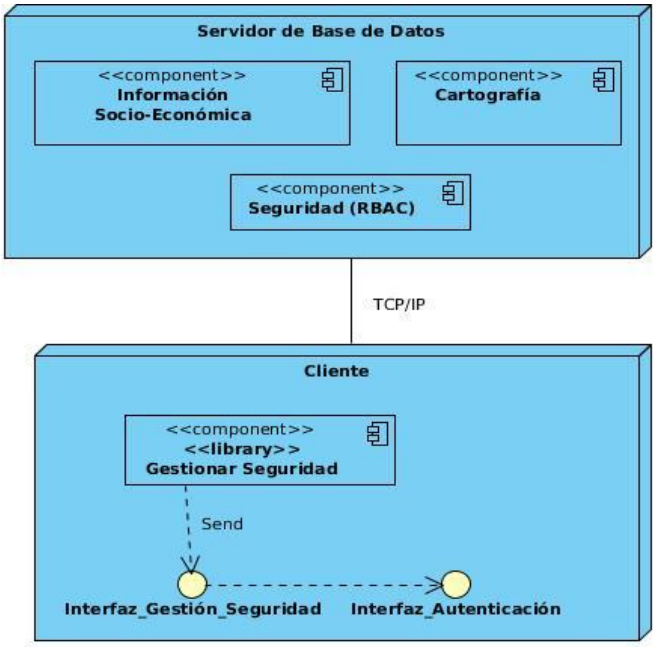


Figura 8 Modelo de despliegue: Subsistema de seguridad

4.7.2 Modelo de Implementación

El modelo de implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes se pueden encontrar datos, archivos, ejecutables, código fuente y los directorios.

Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos.

Un modelo de implementación brinda una visión general de lo que tiene que ser implementado, y un apartado para cada iteración con los componentes y subsistemas a implementar durante esa iteración, así como de los resultados software que se han de obtener y el testeo que se ha de realizar sobre ellos.

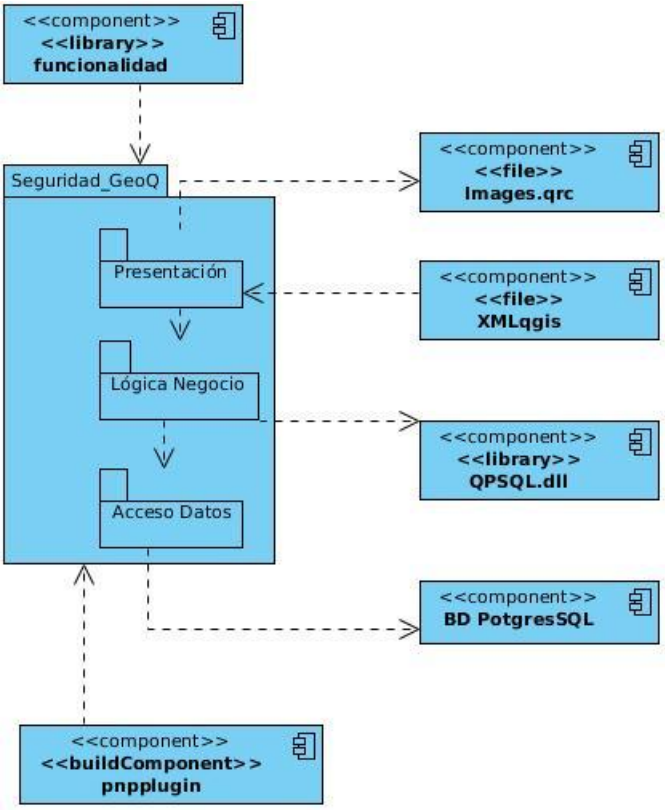


Figura 9 Diagrama de componentes por paquetes

A continuación se muestra el contenido de cada uno de los paquetes.

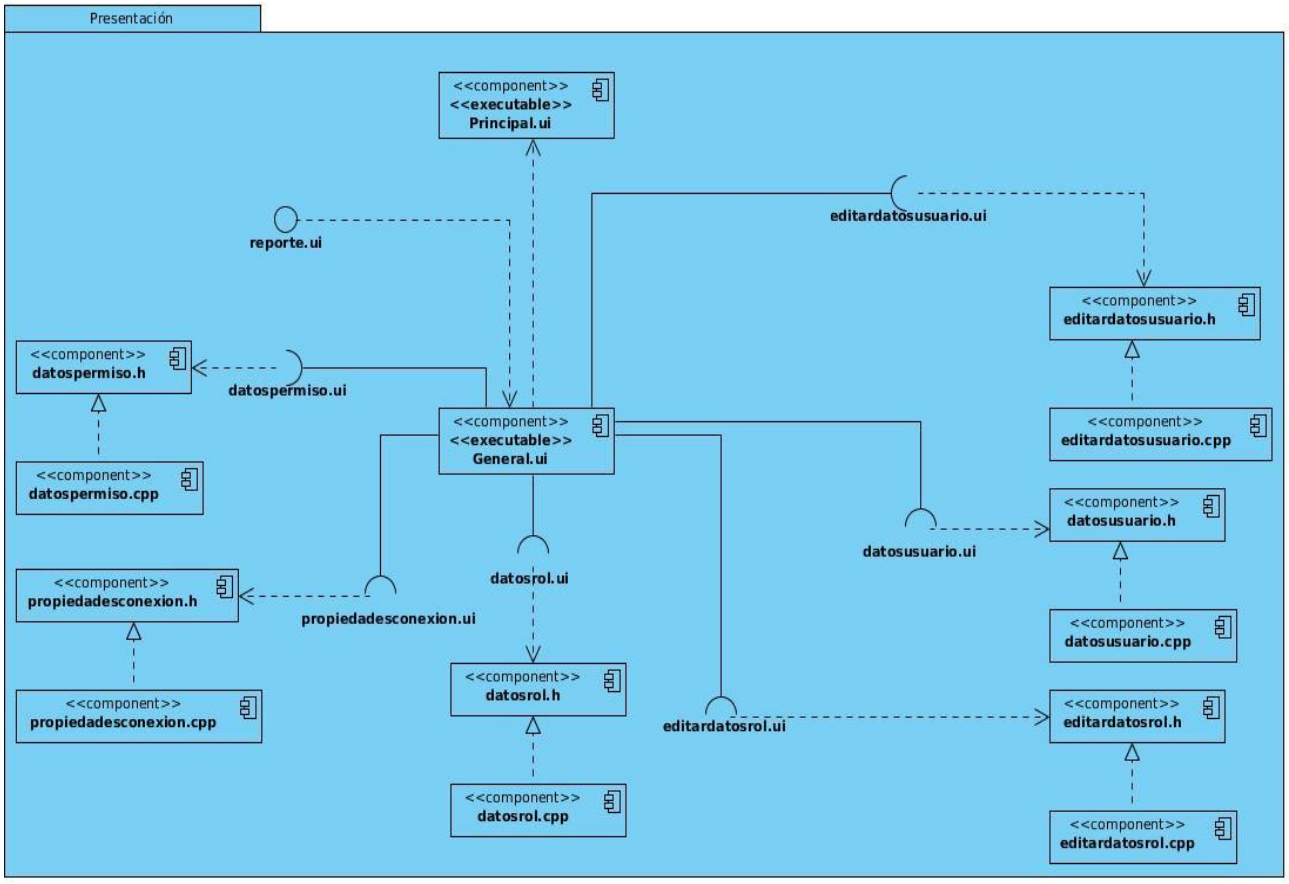


Figura 10 Paquete de Presentación

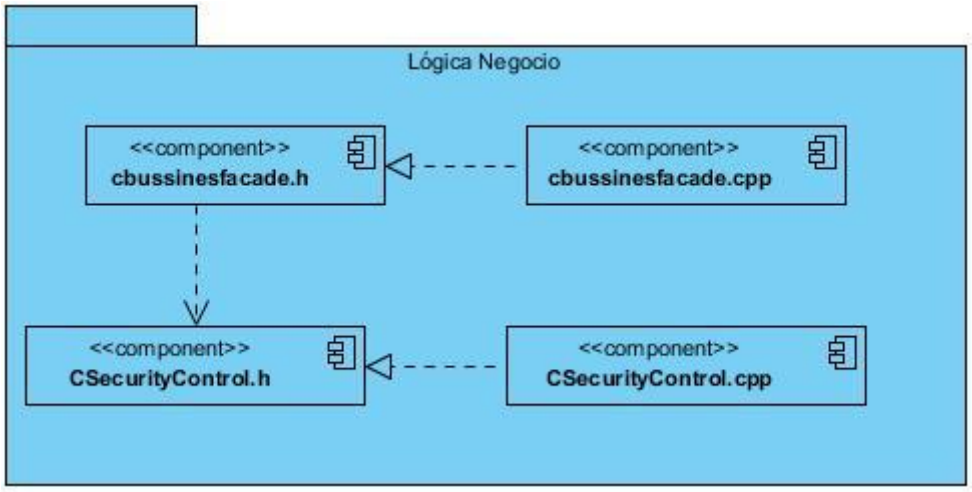


Figura 11 Paquete de Lógica de Negocio

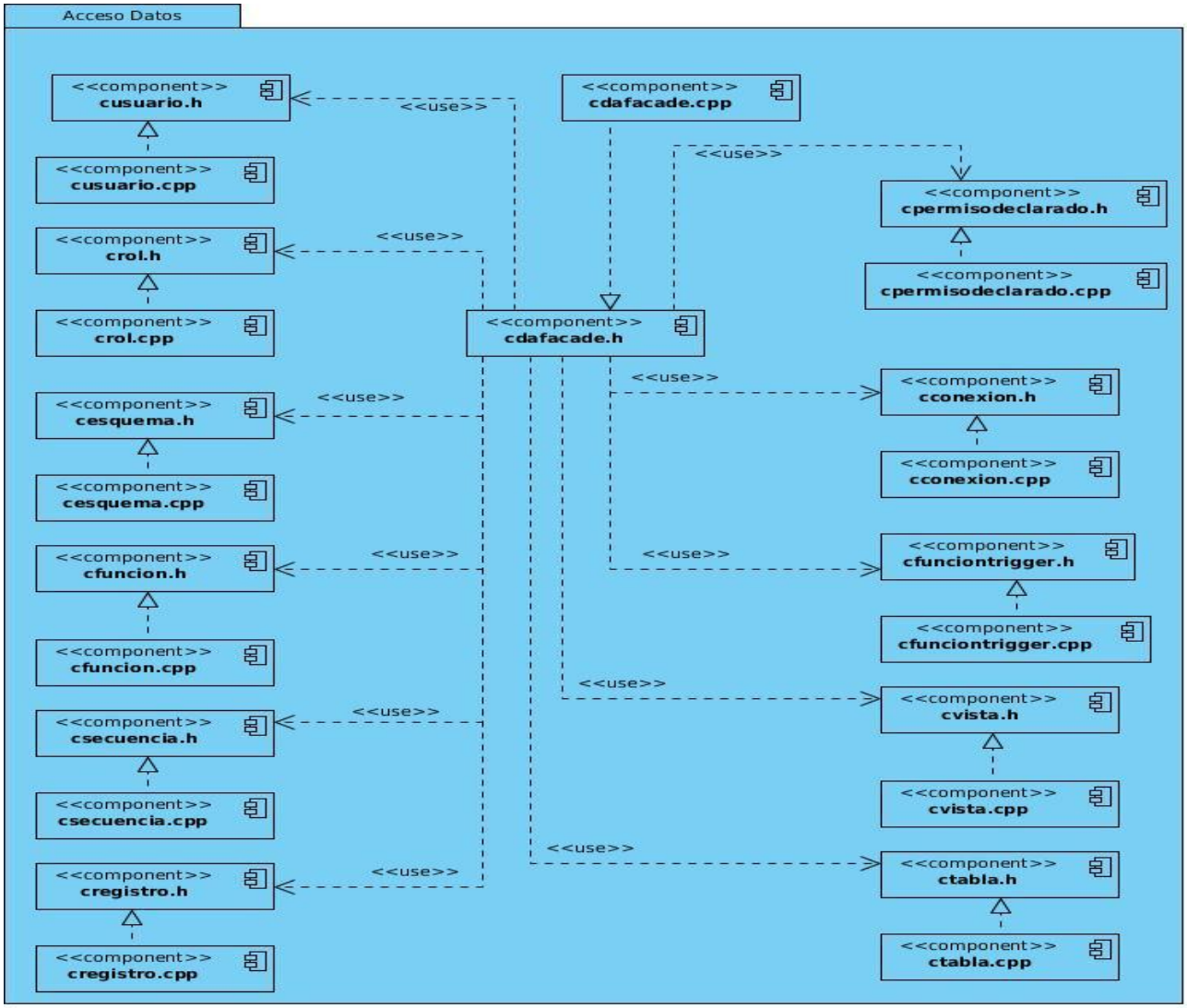


Figura 12 Paquete de Acceso a Datos

4.8 Pruebas

La prueba es el proceso de ejecución de un programa con la intención de descubrir un error. Las pruebas del software son un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones, del diseño y la codificación. La creciente percepción del software como un elemento del sistema y la importancia de los costes asociados a un fallo del propio sistema, están motivando la creación de pruebas minuciosas y bien detalladas (PRESSMAN 2005).

Uno de los métodos más utilizados en la realización de pruebas de software es el llamado pruebas unitarias (unit testing). La base de este método es el hacer pruebas en pequeños fragmentos del

programa, estos fragmentos deben ser unidades estructurales del programa encargados de una tarea específica; en programación procedural u orientada a objetos se puede afirmar que estas unidades son los métodos o las funciones que se definen.

La prueba de unidad centra el proceso de verificación en la menor unidad del diseño del software: el componente software o módulo.

Las pruebas unitarias no revelan errores en la integración de las partes unitarias ni tampoco otros problemas como el bajo rendimiento de las aplicaciones o problemas derivados del sistema sobre el que está ejecutándose el programa; el objetivo principal de las pruebas unitarias es el aislamiento de partes del código y la demostración de que estas partes no contienen errores.

El framework de desarrollo que se utilizó para el desarrollo de la solución presenta una librería denominada QTestLib la cual constituye una herramienta para pruebas unitarias de aplicaciones basadas en Qt y bibliotecas. QTestLib proporciona todas las funcionalidades que se encuentran comúnmente en los marcos de las pruebas unitarias, así como extensiones de las pruebas con usuarios interfaces gráficas.

Teniendo en cuenta lo explicado anteriormente se muestra las pruebas unitarias realizadas al subsistema. Se tomó como muestra los siguientes métodos:

- BuscarIduser(QString pUsuario): el mismo obtiene como parámetro el nombre de un usuario devolviendo un número entero con el identificador de dicho usuario.
- BuscarRol(QString pNombreRol): el mismo obtiene como parámetro el nombre de un rol devolviendo un número entero con el identificador de dicho rol.
- CantidadPermisos(QString pUsuario): el mismo obtiene como parámetro el nombre de un usuario devolviendo un número entero con la cantidad de permisos asignados a dicho usuario.
- countUserByUserNameAndPassword(QString pUsuario, QString pPassword): el mismo obtiene como parámetros el nombre de un usuario y su contraseña devolviendo verdadero o falso de acuerdo a la validez de los datos introducidos (verdadero, si los datos existen y coinciden con los registrados y falso si ocurre todo lo contrario).
- VerficarExistenciaDatosUsuario(CUsuario *pUser): el mismo obtiene como parámetro un objeto de tipo CUsuario con todos sus datos y devuelve verdadero en caso de encontrar un usuario registrado exactamente con dichos datos o falso en caso contrario.

Para la realización de dichas pruebas se utilizó el método QVERIFY definido en Qt Creator, así como datos predefinidos para corroborar la veracidad de las funcionalidades implementadas obteniéndose resultados satisfactorios.

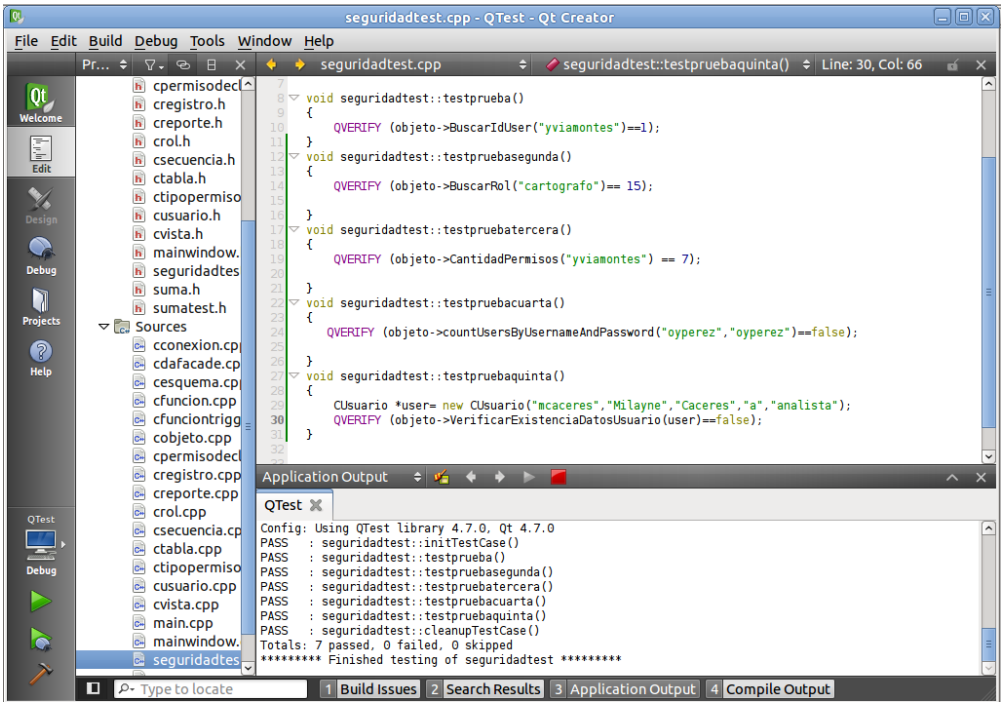


Figura 13 Ejemplo de realización de pruebas unitarias utilizando Qt

4.9 Conclusiones

1. El módulo desarrollado está diseñado siguiendo buenas prácticas de diseño a partir de la implementación de varios patrones y estilos arquitectónicos.
2. La implementación de la arquitectura en capas permitirá aumentar la escalabilidad, disponibilidad, seguridad e integración de la solución propuesta.
3. La aplicación informática tendrá un diseño sencillo, con una distribución estándar del contenido, aumentando así los niveles de usabilidad y adaptación del usuario final.
4. Las pruebas realizadas confirmaron la factibilidad de la solución propuesta.

CONCLUSIONES

Una vez concluido el proceso investigativo, se resaltan una serie de conclusiones las cuales son expuestas a continuación:

1. El resultado desarrollado cumple con las normas establecidas en el Centro GEySED e impulsadas por la Universidad debido a que ha sido llevado a término utilizando plataformas, herramientas y tecnologías libres, lo cual propicia un paso de avance hacia la tan necesaria soberanía tecnológica.
2. Las herramientas y tecnologías utilizadas en el desarrollo de la aplicación se adecuaron a las exigencias del proceso investigativo ahorrando considerablemente tiempo, esfuerzo y costos.
3. Todos los requerimientos funcionales previstos fueron implementados satisfactoriamente teniendo en cuenta las restricciones no funcionales especificadas a lo largo de la investigación.
4. Los requerimientos funcionales implementados se ajustan a las necesidades de la Línea SIG-DESKTOP del Departamento Geoinformática.
5. La utilización de la arquitectura en capas propició un aumento en la escalabilidad, disponibilidad, seguridad e integración del resultado desarrollado.
6. La Línea SIG-DESKTOP cuenta con un módulo para la administración de QGIS que permite controlar centralizadamente el acceso a la aplicación y establecer niveles de acceso de manera que cada usuario del sistema pueda realizar solamente las acciones autorizadas.
7. El resultado desarrollado es fácil de utilizar, intuitivo, con un diseño sencillo, puede ser ejecutado en computadoras con bajas prestaciones y desde cualquier sistema operativo.

RECOMENDACIONES

La autora del trabajo de diploma recomienda:

1. Implementar la autenticación utilizando protocolo LDAP para propiciar la identificación en el sistema a partir del dominio establecido en el ambiente del usuario.
2. Implementar el registro de acceso de los usuarios con el objetivo de conocer sus acciones y movimientos dentro del SIG.
3. Implementar una versión en idioma inglés para aumentar la usabilidad y el mercado potencial de la solución.
4. Socializar el resultado de esta investigación en algún espacio de la universidad para que sirva de consulta a investigaciones afines o similares.

REFERENCIAS BIBLIOGRÁFICAS

ALLIANCE, L., 2010. [2011]. Disponible en: http://www.liberty-alliance.com/

ARENCIBIA, M. G. *Mundo de unos y ceros en la gerencia empresarial*. EUMED.NET, 2006.

AUTORES, G. D. *CODEBOX.Web coders in the box*, 2008. [2011]. Disponible en: http://www.codebox.es/glosario

CELY, C. P. S. *Autenticación de usuarios*. *Comisión Iberoamericana de Telecomunicaciones.Organización de los Estados Americanos.*, 2006.

COMPANY, A. *Visual Paradigm for UML (Enterprise Edition) 8.0*, Apple Inc., 2011. [2011]. Disponible en: http://www.apple.com/downloads/macosx/development_tools/visualparadigmforumlenterpriseedition.html

COMPUTADORAS, S. *Concepto de seguridad Informática* 2008. [2011]. Disponible en: http://www.punchador.com/index.php?option=com_content&view=article&id=54&Itemid=71

CORNEJO, J. E. G. *Arquitectura en Capas DNA.Un camino hacia los procesos distribuidos*. *DocIRS*, 2006.

CORPORATION, N. *Qt Creator IDE and tools* 2011. [2011]. Disponible en: http://qt.nokia.com/products/developer-tools/

DAVID GARLAN , M. S. *An Introduction to Software Architecture* School of Computer Science. Pittsburgh, Carnegie Mellon University, 1994.37. p.

F.J ARIZA LÓPEZ, C. P. R. *Las componentes de la calidad del dato geográfico.*, 2000. [2011]. Disponible en: http://www.mappinginteractivo.com/plantilla-ante.asp?id_articulo=341

FOUNDATION, C. *Proceedings: Asia Pacific Software Engineering Conference and International Computer Science Conference : December 2-5, 1997, Hong Kong*. PRESS, I. C. S., 2008.

GAGNE, S. G. *Fundamentos de sistemas operativos.Parte No 5: Protección y Seguridad*. Séptima. 1999. p.

GARCERANT, I. *Modelo de Dominio*
Tecnología y Synergix.Visión de Synergix de los Sistemas de Información y la Ingeniería del Software, 2008. [2011]. Disponible en: http://synergix.wordpress.com/?s=modelo+de+dominio

GONZÁLEZ, A. P. *Las nuevas tecnologías en la formación ocupacional: retos y posibilidades.* Sevilla, 1996. 195-226 p.

GRACIA, J. *Patrones de diseño. Diseño de Software Orientado a Objetos*, 2005. [2011]. Disponible en: http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php

GUERRERO, R. M. S. *Estudio y puesta en marcha de una infraestructura de gestión de identidad federada basada en SAML 2.0*. Ingeniería de telecomunicación. Madrid, Universidad CARLOS III. Escuela Politécnica Superior., 2009.179. p.

INFORMÁTICA, S. D. P. E. O. D. *Estándares para el uso de herramientas de desarrollo y plataformas de aplicaciones web.Versión 3*, 2006.

Informe sobre el Desarrollo Mundial de las Telecomunicaciones/TIC, 2006.Evaluación de las TIC para el desarrollo económico y social. Unión Internacional de Telecomunicaciones 2008. [2010]. Disponible en: http://www.itu.int/es/pages/default.aspx

ING. MIGUEL SÁNCHEZ, I. B. J., ING. FRANCISCO L. GUTIÉRREZ. *Estudio del control de Acceso en Sistemas Colaborativos*. *Sociedad de Ingeniería de Software y Tecnologías de Desarrollo de Software*, 2007.55-62.

INTEGRATOR, C. A. V. S. *¿Qué es un Sistema Gestor de Bases de Datos o SGBD?* , 2009. [2011]. Disponible en: http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/

IVAR JACOBSON, G. B., JAMES RUMBAUGH *El Proceso Unificado de Desarrollo de Software*. Addison Wesley, 2000. 464 p. 84-7829-036-2

JAVIER GARCÍA DE JALÓN , J. I. R., JOSÉ MARÍA SARRIEGUI , ALFONSO BRAZÁLEZ. *Aprenda C++ como si estuviera en primero*, Escuela Superior de Ingenieros Industriales de San Sebastián. Universidad de Navarra, 1998.

LAFUENTE, G. J. *UML: Unified Modeling Languaje,* 2001.

LARMAN, C. *Applying UML and Pattern. An Introduction to Object-Oriented Analysis and Design*, Prentice Hall, 1999.

MACEDO, T. M. *AsegurarTe. Consultora en Seguridad de la Información*, 2010. [2010]. Disponible en: https://sites.google.com/site/seguridadenredessociales/seguridad-informatica

MCCLURE, C. L. *CASE is software automation*. PRENTICE HALL, Universidad de California, 2008.

ORALLO, E. H. *El Lenguaje Unificado de Modelado (UML)*, Paraninfo Thomson Learning, 2002.

PETER A. BURROUGH, R. A. M. O. U. P. *Principles of Geographical Information Systems*. *Data Models and Axioms*, OXFORD UNIVERSITY PRESS, 1998.

PRESSMAN, R. S. *Ingeniería de Software . Un enfoque práctico*. MCGRAW-HILL, 2005.

REYNOSO, C. B. *Introducción a la Arquitectura de Software*, Marzo 2004.

ROQUE, J. M. *Sistema de Gestión de Datos Geológicos. Módulo: Registro Minero. Rol: Implementador*. La Habana, Universidad de las Ciencias Informáticas, 2010.63. p.

SILVA, D. J. L. B. *Aplicación de Sistemas de Información Geográfica en Cuba.*: *Mapping Interactivo. Revista Internacional de Ciencias de la Tierra.*, 2005.

TEDESCHI, N. *¿Qué es un Patrón de Diseño?* , MSDN Microsoft, 2011. [2011]. Disponible en: http://msdn.microsoft.com/es-es/library/bb972240.aspx

WESTFALEN, H. B. L. N.-. *Telemática de datos geográficos . La situación europea y Alemana GEO-DATA TELEMATICS*, Mapping Interactivo. Revista Internacional de Ciencias de la Tierra, 1995. [2011]. Disponible en: http://www.mappinginteractivo.com/plantilla-ante.asp?id_articulo=995

BIBLIOGRAFÍA

1. DR. HUMBERTO GARCÍA RODRÍGUEZ, M. M. P. I., ING. MAGDELIS MORENO ORTEGA. Las nuevas tecnologías de la información y su impacto en la formación de los recursos humanos. GestioPolis.com, 2002.
2. Seguridad Informática. Disponible en: https://sites.google.com/site/seguridadenredessociales/seguridad-informatica.
3. REDES, R. R. L. C. D. E. E. Seguridad Informática. ¿Qué, por qué y para qué?, Ciberhábitat. Ciudad de la Informática, 2002. [2011]. Disponible en: http://www.inegi.gob.mx/inegi/contenidos/espanol/ciberhabitat/museo/cerquita/redes/seguridad/intro.htm
4. DEFINICIÓN.ORG. Glosario, [2011]. Disponible en: http://www.definicion.org/diccionario/204
5. COMPANY, A. Visual Paradigm for UML (Enterprise Edition) 8.0,Apple Inc., 2011. [2011]. Disponible en: http://www.apple.com/downloads/macosx/development_tools/visualparadigmforumlenterpriseedition.html
6. WELICKI, L. El Patrón Singleton MSDN, 2011.
7. RAVI S. SANDHU, E. J. C., HAL L. FEINSTEIN, CHARLES E. YOUMAN Role-Based Access Control Models IEEE Computer, 1996, 29: 38-47.
8. KODITUWAKKU, S. R. The administrator object pattern for Role-Based Access Control. Internacional Journal of Engineering Science and Technology. Sri Lanka University of Peradeniya, 2010. 2:7880-7884.
9. CHANDRAMOULI, R. Application of XML Tools for Enterprise-Wide RBAC Implementation Tasks. National Institute of Standards and Technology Gaithersburg. p.
10. SEGUEL, D. C. TIERRA, SENTIDO Y TERRITORIO: LA ECUACIÓN GEOSEMÁNTICA, REVISTA AAINTELIGENCIA.Proyecto Aurora Australis: Inteligencia de Emergencia para la Era de la Informacion, 2008.

GLOSARIO DE TÉRMINOS

Artefacto: Es una información que es utilizada o producida mediante un proceso de desarrollo de software.

CASE: (Computer Aided Software Engineering).Constituye una herramienta que ayuda al ingeniero de software a desarrollar y mantener software.

Componente de software: Se define típicamente como algo que puede ser utilizado como una caja negra, en donde se tiene de manera externa una especificación general, la cual es independiente de la especificación interna

Geo-referenciación: Neologismo[20] que refiere al posicionamiento con el que se define la localización de un objeto espacial (representado mediante punto, vector, área, volumen) en un sistema de coordenadas y datum[21] determinado. Posee una definición tecnocientífica, aplicada a la existencia de las cosas en un espacio físico, mediante el establecimiento de relaciones entre las imágenes de ráster o vector sobre una proyección geográfica o sistema de coordenadas.

Paradigma: Procede del griego *paradeigma,* que significa “ejemplo” o “modelo”. En principio, se aplicaba a la gramática (para definir su uso en cierto contexto) y a la retórica (para referirse a una parábola o fábula). A partir de la década del ´60, comenzó a utilizarse para definir a un modelo o patrón en cualquier disciplina científica o contexto epistemológico.

Plugins: Un software plug-in es un complemento para un programa que añade funcionalidad a la misma.

PostGIS: Es un módulo que añade soporte para objetos geográficos a la base de datos objeto-relacional PostgreSQL. PostGIS "habilita espacialmente" el servidor PostgreSQL, permitiendo que sea utilizado como una base de datos de backend espacial para los SIG.

[20] Se define como una palabra nueva que aparece en una lengua, ya sea procedente de otra lengua o de nueva creación.

[21] El término datum se aplica en varias áreas de estudio y trabajo específicamente cuando se hace una relación hacia alguna geometría de referencia importante, sea ésta una línea, un plano o una superficie (plana o curva).

Software: Es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación. Equipamiento lógico o soporte lógico de una computadora digital; comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos, que son llamados hardware. Los componentes lógicos incluyen, entre muchos otros, las aplicaciones informáticas; tales como el procesador de textos, que permite al usuario realizar todas las tareas concernientes a la edición de textos; el software de sistema, tal como el sistema operativo, que, básicamente, permite al resto de los programas funcionar adecuadamente, facilitando también la interacción entre los componentes físicos y el resto de las aplicaciones, y proporcionando una interfaz para el usuario.

Software libre: Es la denominación del software que respeta la libertad de los usuarios sobre su producto adquirido y, por tanto, una vez obtenido puede ser usado, copiado, estudiado, cambiado y redistribuido libremente.

SourceForge: Software de colaboración para la administración de desarrollos. Provee una portada para un amplio rango de servicios para el ciclo de vida de desarrollo de software e integra un amplio número de aplicaciones de software libre (tales como PostgreSQL y CVS). Es una central de desarrollos de software que controla y gestiona varios proyectos de software libre y actúa como un repositorio de código fuente.

Workflow: Es el estudio de los aspectos operacionales de una actividad de trabajo: cómo se estructuran las tareas, cómo se realizan, cuál es su orden correlativo, cómo se sincronizan, cómo fluye la información que soporta las tareas y cómo se le hace seguimiento al cumplimiento de las tareas.