Universidad de las Ciencias Informáticas

Facultad 6

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

TÍTULO: **Editor de Consultas para el Sistema de Análisis y Modelado de Yacimientos Minerales (SYAM).**

AUTOR (es): Arianna Pérez Abreu.

Humberto Alejandro Alé Pérez.

TUTOR: Lic. Maikel Castellanos Placer.

La Habana, Junio /2013

¨Año 55 de la Revolución¨

"O nosotros somos capaces de destruir con argumentos las ideas contrarias, o debemos dejar que se expresen. No es posible destruir ideas por la fuerza, porque esto bloquea cualquier desarrollo libre de la inteligencia."

Declaración de Auditoría

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

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

Arianna Pérez Abreu

_____

Firma del Autor

Humberto Alejandro Alé Pérez

_____

Firma del Autor

Maikel Castellanos Placer

_____

Firma del Tutor

Datos de Contacto

Autor (es)

Arianna Pérez Abreu
Universidad de las Ciencias Informáticas
Ciudad de la Habana, Cuba
E-mail: aabreu@estudiantes.uci.cu

Humberto Alejandro Alé Pérez
Universidad de las Ciencias Informáticas
Ciudad de la Habana, Cuba
E-mail: haale@estudiantes.uci.cu

Tutor (a):
Lic. Maikel Castellanos Placer
Universidad de las Ciencias Informáticas
Ciudad de la Habana, Cuba
E-mail: maikelcp@uci.cu.

Agradecimientos

A ti… madre adorada que me enseñaste lo malo y bueno de la vida, para que pudiese tomar las riendas de la mía. A ti...que con cada consejo, regaño o elogio, me guiaste para convertirme hoy en la mujer que soy……

A mi papá…. por mantenerte siempre conmigo en los momentos difíciles, por todo el amor, el cariño y la comprensión que has sabido darme en todos estos años, por los cuales estoy eternamente agradecida….

A mis abuelos Doris, Eladia y Vicente, por brindarme su apoyo incondicional y ocupar un lugar muy especial en mi corazón, sin ustedes no hubiese logrado llegar hasta aquí….

A mi abuelo Sergio que aunque no se encuentra físicamente entre nosotros sé que está muy orgulloso de mí….

A toda mi familia, en especial a mi prima Dunia, por confiar en mí y estimularme a lograr toda esta gran meta propuesta….

A una persona muy especial en mi vida... Emir, por vivir con él los mejores momentos, por ser mi amigo y compañero… Por ti es que mi mundo hoy es mucho mejor y mi felicidad más grande……

A mi amiga incondicional Tahimí, por estar siempre ahí, por enseñarme el valor de la amistad y apoyarme durante los momentos más difíciles vividos aquí….A mis amigas Grettel, Aida y Yanet, por compartir y disfrutar una amistad sincera……A mis amigos Renier y David…por hacer de todos estos años en la universidad algo especial…

A mi compañero de tesis Humberto…por el trabajo en equipo que realizamos y por ayudarme durante todo el desarrollo de esta investigación…….

A mi tutor, y los profesores del proyecto por su ayuda y dedicación…...

En general a todas las personas que de una forma u otra, algunos más lejos y otros más cerca me han hecho saber que puedo contar con ellos……

Arianna.

Gracias a mi madre por hacer de mi quien soy hoy, por poder contar siempre con ella…...

A mis abuelos por criarme y soportarme gran parte de mi vida, así como darme la educación que poseo……

A mi familia aquí en La Habana por acogerme estos 5 años, a tío Tati, a María Esther, a mi hermana y mis sobrinos……..

A todos mis amigos de la universidad principalmente a los de mi cuarto, a Carlos, al guajiro y al gordo Félix por siempre tener un tema de conversación y siempre estar cuando hace falta apoyo y cuando no también……...

A todos los buenos profesores que he tenido a lo largo de mi carrera, a todos los del proyecto……

A Dime por acompañarme en esta recta final……...

Y a mi compañera de tesis por hacer que todo fuera un poco más fácil……...

Humberto.

Resumen

El Centro de Geoinformática y Señales Digitales (GEYSED), se dedica a desarrollar productos, servicios y soluciones informáticas en el campo del procesamiento de Señales Digitales y la Geoinformática. Entre los productos que se conforman en el Centro se encuentra la herramienta Sistema de Análisis y Modelado de Yacimientos Minerales (SYAM), la cual cuenta a su vez con un módulo de Base de Datos (BD) al que se le pretende insertar un Editor de consultas SQL, por la necesidad de que el sistema no posee funcionalidades para la edición y gestión de datos donde el usuario pueda satisfacer sus necesidades en cuanto a la búsqueda de la información. En función de dar solución a esta problemática, se realiza un estudio de las principales soluciones existentes en este campo, haciendo énfasis en la relevancia que tiene la inclusión del editor de consultas en el módulo de BD. Se elige la metodología de desarrollo a utilizar y se presentan las principales técnicas y tecnologías para la solución del problema. Se eligen los patrones GRASP y GOF como patrones de diseño y los patrones Arquitectura n Capas y Modelo-Vista como patrones de arquitectura. Partiendo del resultado del diseño, se modela el sistema en términos de componentes y se aplican pruebas funcionales de tipo caja negra. Finalmente se obtiene como resultado del trabajo de diploma, el Editor de Consultas para el Sistema de Análisis y Modelado de Yacimientos Minerales, haciendo uso de buenos patrones y aplicando la arquitectura de referencia de la línea.

Tabla de Contenido

Índice de Figuras

Índice de Tablas

Introducción

La informática tiene un papel importante dentro del desarrollo de cualquier empresa, desde finales de los años 80 y durante los 90 se evidenció un incremento en cuanto al uso de equipos y tecnologías que les permitieran lograr: una mejor organización, aprovechamiento del tiempo e informatizar procesos y servicios. Además de brindarles a sus empleados herramientas para facilitarle su trabajo y apoyo en la toma de decisiones.

Para las empresas es importante almacenar los datos de sus recursos humanos, los productos y materiales que dispone, sus ventas, etc. Al principio estos se encontraban dentro de documentos los cuales eran archivados en grandes volúmenes, y la búsqueda de cualquier información consumía bastante tiempo a sus empleados y resultaba un proceso tedioso, además se podían deteriorar con el transcurso de los años perdiendo información valiosa para el correcto funcionamiento y desarrollo. El auge de la computación vino a resolver en gran medida estos problemas, debido a que haciendo uso de diferentes programas informáticos los trabajadores podían guardar todos los datos de interés de la organización.

En sus inicios los programas utilizaban para almacenar la información ficheros textos, en los cuales era difícil plasmar conceptos y relaciones que podían existir entre los datos y a medida que estos aumentaban la búsqueda y manipulación se complejizaba. Así surgieron al comienzo de 1960 las bases de datos jerárquicas y de red, al no ser imprescindible la ubicación de los datos en los llamados discos fijos estos permitieron el desarrollo de las mismas, brindando la posibilidad de que las estructuras de los datos (listas y árboles) pudieran ser almacenadas en dichos discos y fuera más fácil manejar por parte de los programadores.

Las Bases de Datos Jerárquicas tienen una forma particular de almacenar la información, la cual es mediante una estructura jerárquica como su nombre lo indica, estas son de gran utilidad en el uso de aplicaciones que interactúan con grandes volúmenes de datos, no siendo así su capacidad para la representación eficiente de la redundancia de los mismos, a pesar de estos las Bases de Datos de Red al ser un modelo diferente si resuelve este problema a través de la incorporación de un registro.

Debido a esto a finales de 1960 y principios de 1970 Edgar Frank Codd definió en su artículo *“Un modelo relacional de datos para grandes bancos de datos compartidos"* ***("A Relational Model of Data***

***for Large Shared Data Banks"*)** el concepto de **Modelo Relacional** y un conjunto de reglas asociadas a este para su evaluación, dando origen a las bases de datos relacionales. (A Relational Model of Data for Large Shared Data Banks , 1970)

En 1974 Donald Chamberlin, Ray Boyce y Jim Meltony, actual editor de SQL, los cuales trabajaban en los laboratorios de investigación de IBM[1] definieron un lenguaje llamado SQL (Lenguaje estructurado de consultas). Este fue creado para generar reportes y actualizar datos en este nuevo modelo relacional. En la actualidad este lenguaje es un estándar para la comunicación entre las aplicaciones y las bases de datos.

Dado el origen de las base de datos y los lenguajes para la comunicación entre ellas los Sistemas Gestores de Bases de Datos tomaron un papel fundamental, permitiendo la definición de las mismas; así como la elección de las estructuras de datos necesarios para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación. Actualmente existen varios gestores de Bases de Datos, entre los más utilizados a nivel mundial se encuentran: Oracle[2], PostgreSQL, MySQL[3], Interbase[4], Firebird[5], SQLite[6].

PostgreSQL es un Sistema Gestor de Bases de Datos de código abierto, brinda un control de concurrencia multi-versión (MVCC[7] por sus siglas en inglés) que permite trabajar con grandes

---

[1] **I**nternational **B**usiness **M**achines: Empresa multinacional estadounidense de tecnología y consultoría que fabrica y comercializa hardware y software para computadoras.

[2] Sistema de gestión de base de datos (SGBD) objeto-relacional, desarrollado por Oracle Corporation, considerado uno de los más completos.

[3] Sistema gestor de base de datos relacional (SGBDR), multihilo y multiusuario con más de seis millones de instalaciones.

[4] SGBDR relacionales desarrollado y comercializado por la compañía Borland Software Corporation, destacado por su bajo consumo de recursos, su casi nula.

[5] SGBDR relacional de código abierto, basado en la versión 6 de Interbase.

[6] SGBDR compatible con ACID de dominio público, contenida en una relativamente pequeña (~275 kiB) biblioteca escrita en C.

[7] Método para control de acceso generalmente usado por SGBDs para proporcionar acceso concurrente a los datos, y en lenguajes de programación para implementar concurrencia.

volúmenes de datos; soporta gran parte de la sintaxis SQL y cuenta con un extenso grupo de enlaces con lenguajes de programación.

Uno de los componentes dentro de los Gestores de BD son los Editores de Consultas, centrados fundamentalmente en editar procedimientos almacenados, funciones, desencadenadores y secuencias de comandos existentes, así como ayudar a desarrollar, probar y reutilizar consultas rápida y eficazmente en las principales bases de datos relacionales. Con estos se pueden crear y editar una sola consulta o juntar varias instrucciones SQL en complejos scripts SQL.

La Universidad de las Ciencias Informáticas no se queda atrás si de desarrollo se trata, e inmersos en todo lo anteriormente expuesto en la **Facultad 6**, en el departamento de **Geoinformática** se está desarrollando una herramienta informática conocida como Sistema de Análisis y Modelado de Yacimientos Minerales (SYAM) para la modelación, análisis, planificación y explotación de los recursos minerales, la cual será utilizada por geólogos y mineros de las distintas entidades geólogo-mineras del país.

El sistema cuenta con un módulo (**gmDB**) para la gestión de los datos de un determinado yacimiento mineral, utilizando como gestor de base de datos PostgreSQL y su módulo PostGIS[8] para la representación de datos espaciales como puntos, líneas, polilíneas, polígonos, etc. Estos datos son editados y consultados de manera sistemática por los usuarios en busca de información que les permita tomar decisiones en las fases mencionadas anteriormente.

En estos momentos la forma de consultarlos es accediendo a cada una de las tablas de la base de datos, a través de las funcionalidades que brinda el módulo **gmDB** del sistema; sin embargo en ciertas circunstancias los usuarios necesitan consultar estos bajo determinados criterios. Estos no se encuentran de manera explícita en las funcionalidades del módulo, debido a que los datos de un yacimiento pueden variar considerablemente con respecto a los de otro y personalizar este con las funcionalidades necesarias para trabajar con los datos de cada yacimiento resultaría en proceso muy difícil y engorroso.

[8] Módulo que añade soporte de objetos geográficos a la base de datos objeto-relacional PostgreSQL, convirtiéndola en una base de datos espacial para su utilización en Sistemas de Información Geográfica.

De acuerdo a lo anteriormente planteado se propone como **problema de la investigación:** ¿Cómo facilitarle a los usuarios del Sistema de Análisis y Modelado de Yacimientos Minerales (SYAM) la búsqueda de datos en un yacimiento mineral bajo disímiles criterios?

Para dar solución al anterior problema se propone el siguiente **objeto de estudio**: procesos de gestión de los datos geológicos en herramientas mineras, específicamente incidiendo en los procesos de gestión de los datos geológicos en la herramienta SYAM haciendo uso del lenguaje SQL, lo cual constituye su **campo de acción**.

Se propone como **objetivo general** de la presente investigación, desarrollar un editor de consultas para el sistema **SYAM** que permita al usuario acceder a los datos de un yacimiento mineral bajo determinados criterios.

Los autores consideran la siguiente **idea a defender:** Con el desarrollo del editor de consultas para el sistema **SYAM** se agilizará la búsqueda de datos en un yacimiento mineral bajo disímiles criterios, lo cual apoyará la toma de decisiones.

Tareas de la investigación:

- Caracterizar el estado del arte sobre la gestión de los datos geológicos.
- Definir las soluciones existentes a nivel nacional e internacional.
- Caracterizar la metodología de desarrollo a utilizar.
- Caracterizar las herramientas y tecnologías necesarias para la implementación.
- Elaborar la documentación técnica ingenieril correspondiente al análisis, diseño e implementación de la herramienta propuesta.
- Implementar las funcionalidades de la herramienta propuesta e integrarla al SYAM.
- Realizar pruebas para validar la implementación de la herramienta.

Métodos Científicos

Analítico-sintético: Se utiliza para el estudio a partir de fuentes bibliográficas seguras sobre la evolución y existencia de los componentes de edición dentro de las herramientas mineras, permitiendo además descomponer el problema de investigación en elementos por separado y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución de la propuesta.

Análisis Histórico-lógico: Con el objetivo de definir las tendencias actuales en el desarrollo de los componentes de edición dentro de las herramientas mineras.

Posibles resultados:

Un editor de consultas integrado al sistema **SYAM**.

La documentación técnica correspondiente al editor de consultas.

El presente trabajo de diploma estará dividido en cuatro capítulos fundamentales.

Capítulo 1. Fundamentación Teórica

En este capítulo se realiza un estudio del estado del arte, haciendo énfasis en las soluciones existentes, así como un análisis de los procesos de gestión de los datos geológicos en las herramientas mineras, además de los principales conceptos asociados al problema de investigación.

Capítulo 2. Tendencias y tecnologías actuales a desarrollar.

En este capítulo se abordan las principales herramientas a utilizar en la implementación del software, así como los lenguajes implicados durante todo el proceso de desarrollo, definiendo las características y ventajas que presentan. A través del estudio de las diferentes metodologías de desarrollo de software que existen se selecciona la más adecuada y la que más se ajusta a la solución propuesta para su posterior argumento.

Capítulo 3. Presentación de la solución propuesta.

En este capítulo se hace una descripción de los procesos actuales a través del modelo del dominio, se describen los actores, además se especifican los requisitos funcionales y no funcionales y los casos de uso del sistema, todos estos asociados con sus diagramas que los modelan respectivamente.

Capítulo 4. Construcción de la solución propuesta.

En este capítulo se describe el proceso de análisis, diseño, implementación y prueba de la solución propuesta teniendo en cuenta todo lo descrito en el capítulo anterior, los modelos de despliegue e implementación y se diseñan los casos de pruebas correspondientes para dar veracidad a la solución.

Capítulo 1.Fundamentación Teórica

En el presente capítulo se muestran diferentes puntos que permiten evidenciar la fundamentación teórica de la presente investigación. Está destinado a abarcar mediante una breve descripción los conceptos de mayor importancia asociados al dominio del problema. Se describe el objeto de estudio de la investigación haciendo énfasis en la situación problemática que dio origen a este trabajo, además de realizarse un profundo análisis de las soluciones existentes.

1.1 Proceso de Gestión de Datos Geológicos en las herramientas mineras.

Desde el surgimiento de las herramientas mineras, la necesidad de tener un mejor control y acceso a los datos geológicos se hacía cada vez más necesaria, ya que muchos de estos datos se almacenaban en ficheros y ocasionaba ciertas dificultades a la hora de manipular los mismos. Con la evolución y el avance de las tecnologías, nuevas herramientas fueron creadas así como versiones cada vez más efectivas de las ya desarrolladas.

Esto provocó que las empresas se vieran obligadas a la adquisición de estas nuevas tecnologías para su mejor funcionamiento, resolviendo en gran medida los problemas existentes respecto a los datos que se manejaban, ya que las bases de datos mejoraban de disímiles vías la forma de interactuar con la información contenida en las mismas. Siendo así y extendiéndose hasta la actualidad, varias herramientas que hoy predominan en el mercado tienen una mejor factibilidad en cuanto a la información que procesan, apoyándose estos en componentes gráficos para la gestión y edición de datos, además propiciando una mejor retroalimentación entre los usuarios y los productos.

Algunos de estos programas informáticos, no presentan una forma de manipular los datos con la flexibilidad que determinadas herramientas integradas a los programas gestores de bases de datos actuales poseen, las cuales permiten realizar consultas con parámetros que el usuario define según su comodidad y criterio para las más variadas finalidades.

En la Universidad de las Ciencias Informáticas (en lo adelante UCI) se está creando un software Sistema de Análisis y Modelado de Yacimientos Minerales (SYAM) que permita interpretar geológicamente depósitos minerales, modelar, estimar y reportar los recursos minerales. Este

software está compuesto por 8 módulos, dentro de estos se encuentra el **Módulo de Base de Datos – gmDB**, el cual tiene como objetivo principal encargarse de la administración de la información persistente del sistema en un gestor de Base de Datos. Este además contiene los principales algoritmos de validación de los datos primarios, permite crear nuevas tablas y editar la información de las mismas. Este software está diseñado para proyectos genéricos de minería, y provee a los proyectos generados de tablas básicas para su trabajo. El trabajo de minería y gestión de datos geológicos es variado y adaptable, por ende SYAM permite a los usuarios crear y/o agregar tablas al sistema, garantizando la flexibilidad y adaptabilidad. La integración de un editor de consultas con las funcionalidades adecuadas permitiría una mejor integración entre el usuario y el proyecto permitiendo al primero obtener datos entre diferentes tablas y con distintas definiciones.

1.2 Estructura de las tablas principales de la base de datos de software mineros.

Las bases de datos de las herramientas mineras están compuestas por 4 tablas principales:

- Collars: Coordenadas (X, Y, Z) de los collares de los Sondajes.
- Surveys: Azimut e Inclinación de cada muestra del Sondaje.
- Assays: Campos de interés de las muestras (leyes minerales[9] de la mena[10], definidas por el usuario).
- Lithology: Clasificación Geológica (Litológica) de las muestras.

En la tabla collar se almacenan los datos de las bocas de los pozos de muestra (id del collar, el nombre del pozo, ubicación). Cada pozo contiene ensayos (assay), estudios (survey) y litologías asociadas. Los ensayos de cada pozo contiene muestras con concentraciones de diferentes minerales, un pozo puede poseer *'n'* muestras. Los estudios poseen intervalos del pozo y registran de estos su longitud e inclinación, dado a que un pozo no es necesariamente recto en su totalidad, sino que puede variar su inclinación en varios tramos.

[9]Medida que describe el grado de concentración de recursos naturales valiosos (como los metales o minerales) disponibles en una mena.

[10]Mineral del que se puede extraer un elemento porque lo contiene en cantidad suficiente para poderlo aprovechar.

Para un software minero mostrar estas tablas tal y como se encuentran en las bases de datos no es factible, pues en la mayoría de los casos se trabaja con extensas colecciones de datos y no es adecuado que el usuario deba explorar miles de tuplas[11] para encontrar los datos que necesita.

1.3 Obtención de Datos en herramientas mineras.

Los software mineros con los que trabajan las empresas de nuestro país, actualmente muestran los datos almacenados en sus bases de datos mediante tablas generalmente, lo cual es importante a la hora de visualizar, pero aunque poseen mecanismos para que el usuario le sea más fácil navegar en la gran cantidad de datos, a veces resulta incómodo, sobre todo si no se sabe con exactitud que se busca. Por la característica privativa de este tipo de software solo se pudieron estudiar a fondo dos soluciones internacionales (Gemcom GEMS y Datamine Studio 3) con las cuales se cuenta con sus respectivos instaladores y licencias. Además de datos de una solución nacional creada hace varios años (Tierra) que es uno de los productos nacionales que más ha profundizado en el área de la minería, hay otras soluciones pero se abarca solo los datos que hay disponibles en la web.

1.3.1 Datamine Studio.

Los archivos o tablas de la base de datos son almacenados en un formato binario el cual ahorra espacio en disco y mejora la velocidad de acceso y procesamiento de los datos.Para acceder a los archivos basta con hacer doble click en los archivos mostrados en la ventana Proyecto y estos se podrán mostrar en la pantalla mediante el programa Table Editor[12]. De haber algún cambio que se desee hacer sin usar Studio 3,basta con instalar Table Editor (el programa de edición de archivos) en cualquier PC para poder ver dichos archivos. Éste programa está registado como un control de Microsoft ActiveX, por lo que puede ser incluido en cualquier entorno de scripts, se puede mostrar el editor en internet explorer usando la etiquete <OBJECT>, se debe tener el software instalado en la maquina completamente, no solo el editor de tablas (Datamine CO.ltd, 2006). Esta carácterística es propia para sistemas Windows.

[11]Función finita que mapea (asocia unívocamente) distintos valores de la BD (fila de una tabla o consulta).

[12]TableEditor: Programa incluido en la instalación de Datamine Studio 3 especializado en el manejo de las tablas del sistema.

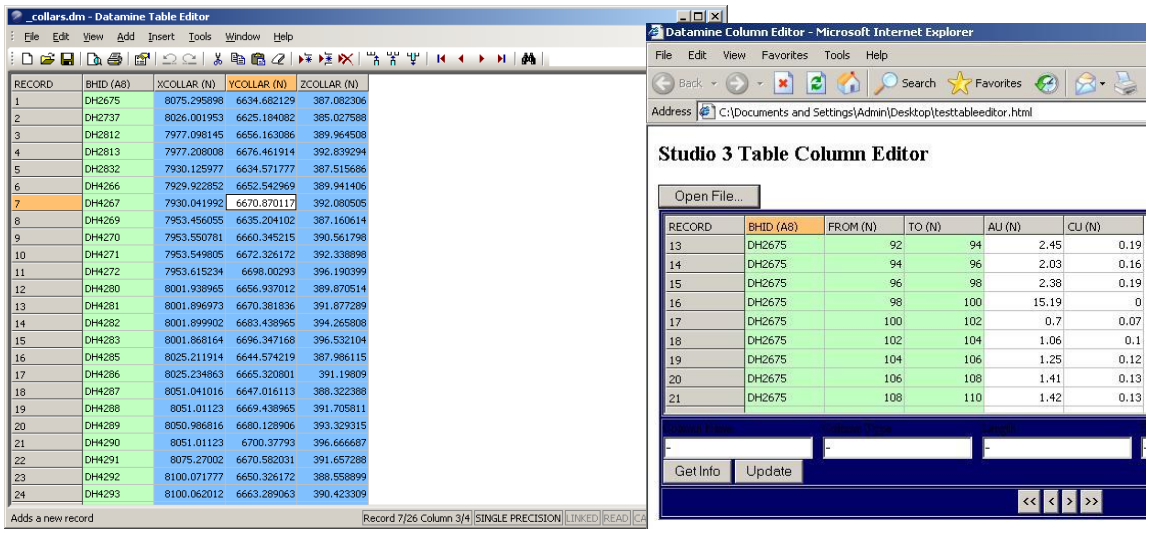


Figura 1: Editor de tablas de Datamine Studio 3

1.3.2 Gemcom GEMS.

Gemcom es una empresa que brinda un conjunto de software dirigido a empresas mineras, que ofrece una mejor planificación y producción de la información a través de una plataforma flexible, y que permite generar y controlar una única fuente de validación de datos integrados. Las soluciones integradas que desarrolla Gemcom abarcan desde las fases de exploración, evaluación de recursos, diseño de minado, optimización, planeamiento minero y control de leyes de producción, hasta la reconciliación y balance metalúrgico a lo largo de la línea de producción.

La línea de soluciones *MPMS (Mine Production Management Solutions)* es cuidadosamente diseñada y ajustada a las necesidades reales y futuras de cada uno de los clientes, incluyendo productos específicos como GEMS(cubre las necesidades de los profesionales en todas las áreas de la ingeniería y geología) o Whittle (destinado a la optimización económica y planeamiento estratégico de minado, posee herramientas para el análisis de sensibilidad y secuenciamiento óptimo) y/o herramientas de otros proveedores o desarrollos propios de cada empresa. Lo relevante es la integración real de todos los elementos involucrados en la cadena de valor, lo que posibilita una visión global confiable, con los indicadores adecuados que permiten a los ejecutivos tomar decisiones más certeras.(Unam, 2011)

Esta línea de productos están orientados a plataformas con Sistema Operativo Windows, con completa integración a sus funcionalidades y características. GEMS brinda acceso en el editor de datos a un editor de Visual Basic[13]. El editor de datos posee una apariencia como la de Microsoft Excel[14], se muestra en dos áreas verticales, la superior contiene los registros de los pozos (tabla collar) y en el area inferior se pueden apreciar pestañas como libros de Excel para cada una de las tablas, de esta manera al seleccionar un pozo en la tabla superior, sólo se mostrará para la tabla de abajo la información relativa a ese pozo (véa figura 2).

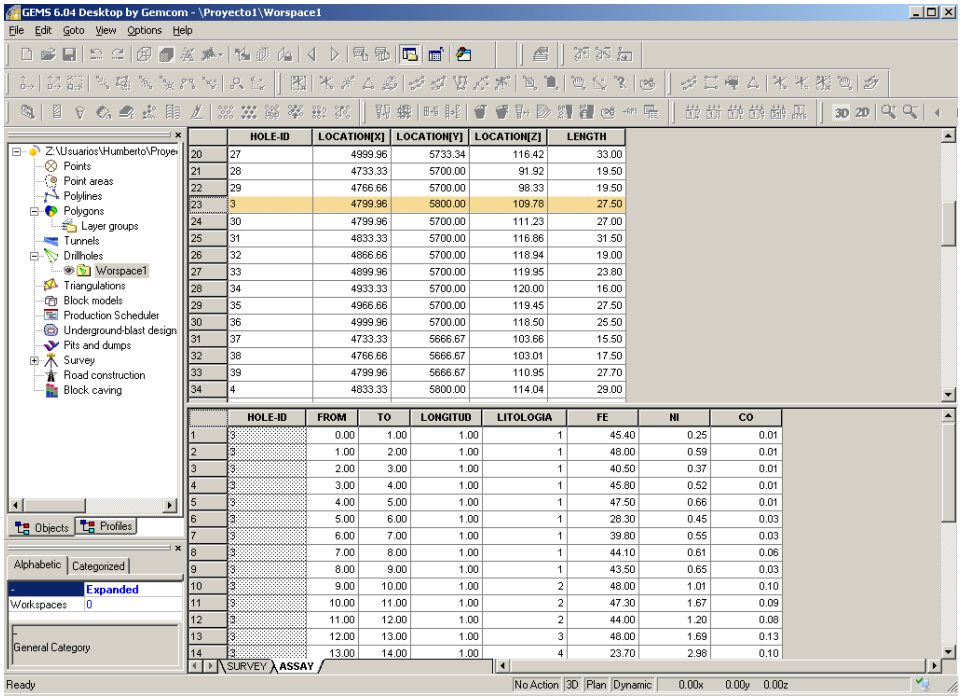


Figura 2: Editor de datos de Gemcom GEMS 6.04.

1.3.3 DataStream.

Posee una arquitectura 100% web, permitiendo su acceso por medio de un *browser standard* de Internet desde cualquier lugar en cualquier momento, contribuyendo a minimizar el tráfico por la red y evitando la compra de hardware costoso.

[13]Lenguaje de programación dirigido por eventos, desarrollado por Alan Cooper para Microsoft.

[14] Aplicación distribuida por Microsoft Office para hojas de cálculo.

Una de las principales ventajas de esta tecnología tiene que ver con la disponibilidad en tiempo real y, gracias al desarrollo de las tecnologías inalámbricas, desde cualquier hora y lugar a una fuente de información histórica sobre los equipos y sistemas, para determinar patrones de falla y mejorar los procedimientos y programas de mantenimiento que incrementen su disponibilidad a un costo cada vez menor. A su vez, cualquier funcionario de una planta minera puede disponer de manuales, planos y documentos en forma automática, ahorrando tiempo y recursos considerables. Trabaja con un sistema de base de datos Oracle (v 8i – v 9i).(UNAM, 2013)

1.3.4 Tierra.

Herramienta minera construida para sintetizar y automatizar la parte de una metodología para el pronóstico, planificación y control de la minería en yacimientos lateríticos[15] aplicada prácticamente en la Subdirección de minas de la empresa "Comandante Ernesto Guevara de la Serna". Para la realización de este software se recogió la experiencia práctica acumulada durante años de explotación de yacimientos lateríticos en Cuba, se analizó las ideas manejadas en nuestro país y el extranjero con respecto a la explotación de las lateritas que han sido publicadas y aún aquellas que han sido expuestas oralmente a los autores, se crearon nuevos algoritmos y criterios donde fue necesario. El trabajo con los datos no es tan factible como los softwares anteriormente mencionados ya que se presentan ocho opciones para resolver las necesidades del tratamiento de datos que permiten convertir las bases de datos DBF[16] a los formatos TXT que utiliza TIERRA. También se permiten accesos a toda la información y actualización de algunas de ellas; se automatizan los procesos de compactación y descompactación de archivos; se pueden hacer tratamientos de modelación matemática de algunos datos.(Lobaina, 1999)

Este software está actualmente en desuso y no posee las características necesarias para suplir las necesidades de los usuarios.

[15]La laterita es el suelo propio de las regiones cálidas, caracterizado por la pobreza en sílice y su elevado tenor en hierro y alúmina.

[16] DBF es el formato de archivo de datos utilizado originalmente por el producto dBase siendo en la actualidad el formato más comúnmente utilizado en DBMS -Sistema de Gestión de Base de Datos, para computadoras personales.

1.4 Herramientas para la gestión de los sistemas gestores de bases de datos.

Las herramientas para la gestión de los SGBD brindan una interfaz gráfica para el manejo de estos sistemas y posibilitan al usuario realizar las principales tareas con mayor facilidad y comodidad. La mayoría de los SGBD traen adjunto alguna herramienta para su diseño y administración que al final son los software con el que el usuario está más familiarizado y que poseen gran flexibilidad en el trabajo con bases de datos. Otras de estas herramientas son independientes de algún gestor y soportan varios tipos de sistemas lo que las hace aún más potentes. Las interfaces y complementos que poseen estos software son de notable calidad, por lo que se hace un estudio de algunas de ellas para ver funcionalidades que se puedan aplicar al producto SYAM.

1.4.1 Aqua Data Studio.

Aqua Data Studio es un completo entorno de desarrollo integrado (IDE[17] por sus siglas en inglés) para consultas, administración y desarrollo de bases de datos. Es compatible con la mayoría de los proveedores de bases de datos, funciona en casi todos los sistemas operativos más difundidos y está disponible en 21 lenguajes.

Está desarrollado en Java[18] y actualmente su versión más actual disponible para descarga es la 12.0.18 con posibilidad de adquirirlo en su sitio oficial para Windows, Mac, Linux, Solaris y Java Plattform[19]. Posee licencia propietaria, puede ser recibido el producto gratis por 14 días de prueba, pero luego es necesario comprar la licencia.

---

[17]IDE: Integrated Development Environment.

[18] Java es un lenguaje de programación originalmente desarrollado por James Gosling de Sun Microsystems (la cual fue adquirida por la compañía Oracle) y publicado en el 1995 como un componente fundamental de la plataforma Java de Sun Microsystems.

19Java Plattform es un entorno o plataforma de computación originaria de Sun Microsystems, capaz de ejecutar aplicaciones desarrolladas usando el lenguaje de programación Java u otros lenguajes que compilen a bytecode y un conjunto de herramientas de desarrollo.

Este software posee un avanzado analizador de consultas que permite la conexión a cualquier servidor de bases de datos para ejecutar consultas. Sus características de abreviaciones SQL, automatizado y auto completamiento ahorran tiempo a la hora de construir cláusulas SQL. Los resultados de las consultas pueden ser visualizados como hojas de cálculo de Excel o compartidos por correo electrónico con sólo un clic.

Además posee un Constructor de consultas visuales que ayuda en la construcción de consultas complejas sin tener que tener conocimientos de las cláusulas. Una amplia gama de opciones visuales se dispone para combinar cláusulas SQL como JOIN, GROUP BY con propiedades como índices, operadores, alias y ordenamiento por citar. Basado en el criterio de selección del usuario el programa construye una cláusula SQL que puede ser ejecutada para mostrar el resultado de la consulta.(AquaFold Inc., 2013)

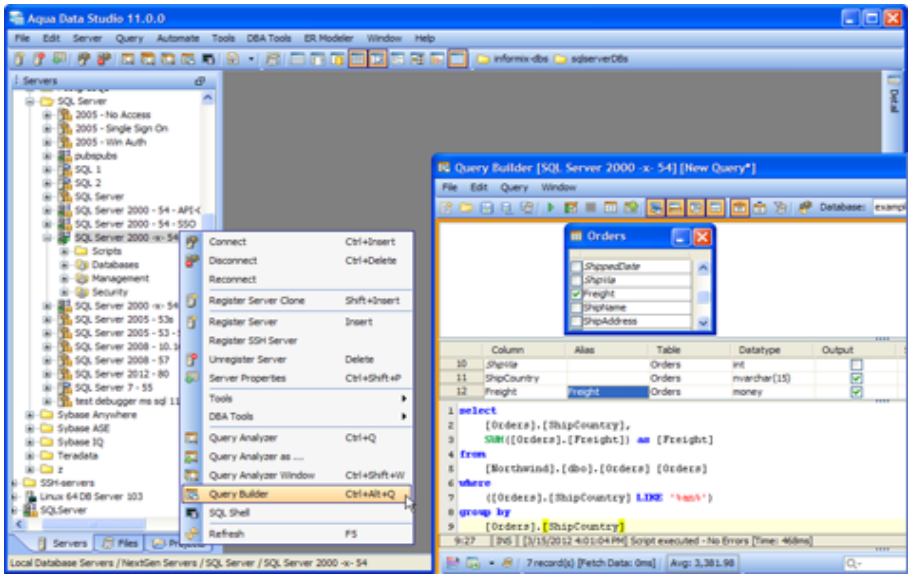


Figura 3: Constructor visual de consultas del Aqua Data Studio.

1.4.2 DBeaver.

DBeaver es un Administrador de bases de datos universal libre y de código abierto (GPL) destinado principalmente a desarrolladores y administradores de bases de datos. Se basa en la usabilidad brindando una interfaz gráfica cuidadosamente diseñada e implementada. Es multiplataforma y basado en Frameworks de código abierto, posee varias extensiones (plugins) para la mayoría de los

proveedores de bases de datos. Actualmente funciona en sistemas operativos como Windows (2000/XP/2003/Vista/7), Linux, MacOS, Solaris entre otros.

Posee ejecución de cláusulas y scripts SQL, resaltado de código (específico para cada motor de base de datos), auto completamiento e hipervínculos a metadatos en el editor SQL, editor de tablas, exportado de datos (consultas y tablas). Está desarrollado en Java y su última versión disponible para descarga desde su sitio oficial es la 2.1.0. Su principal desarrollador y fundador es Serge Rieder, el cual ha diseñado la interfaz de usuario y realizó los primeros pasos.(Rieder, 2013)

1.4.3 PgAdmin.

PgAdmin es la más popular plataforma de administración y desarrollo de código abierto para PostgreSQL. La aplicación puede ser utilizada en las plataformas con Linux, FreeBSD, Solaris, Mac OSX y Windows para la administración de cualquier servicio de PostgreSQL. Está diseñado para responder a las necesidades de los usuarios, desde escribir simples consultas SQL, hasta desarrollar complejas bases de datos.

Su interfaz gráfica soporta todas las características de PostgreSQL haciendo fácil la administración. La aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código del lado del servidor, soporte para el motor de replicación Slony-I entre otras características. Las conexiones a servidores deben ser hechas usando el TCP/IP[20] o Unix Domain Socket[21] en plataformas Unix, y para seguridad deben ser encriptados mediante SSL. No requiere de drivers adicionales para comunicarse con servidores. Es desarrollado por la comunidad de expertos de PostgreSQL alrededor del mundo y está disponible en más de una decena de lenguajes. Es software libre liberado bajo la licencia de PostgreSQL.(pgAdmin, 2013)

[20]Transmission Control Protocol/Internet Protocol: conjunto de protocolos de red en los que se basa Internet y que permiten la transmisión de datos entre computadoras.

[21] También conocido como IPC socket (inter-process communication socket) es un end point de comunicación para el intercambio de datos entre procesos en ejecución en un mismo sistema operativo.

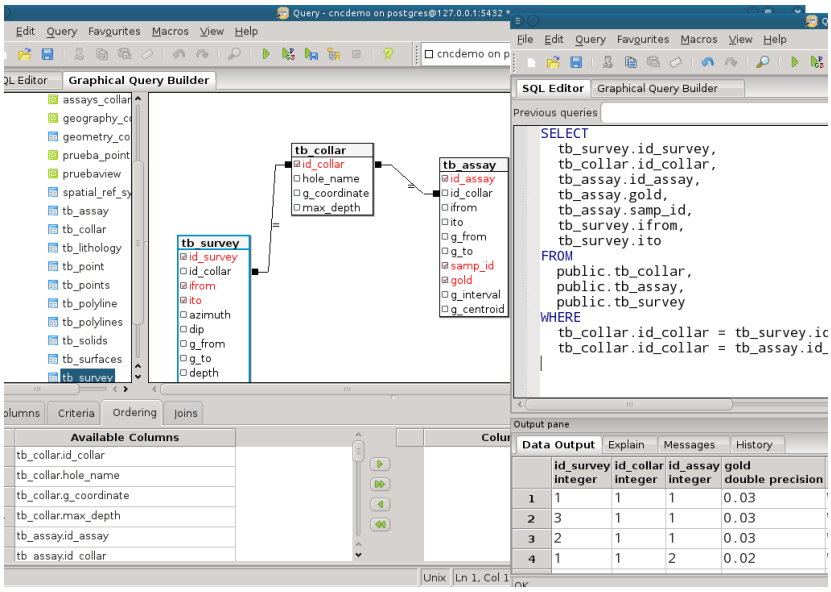


Figura 4: Constructor y editor SQL en el pgAdmin3.

Se puede apreciar que la gran mayoría de las aplicaciones multiplataforma están escritas en lenguaje JAVA por lo que se haría necesario instalar en la estación de trabajo la Máquina Virtual de Java para que dichas aplicaciones puedan ejecutarse. Se detectan algunas características principales en estas herramientas como son, resaltado de sintaxis, auto completamiento de código, y constructores de consultas gráficos para los usuarios con poco dominio del SQL.

Se distinguen dos formas básicas de realizar consultas SQL:

- Mediante escritura de cláusulas (Editores SQL).
- Selección de campos y cláusulas a utilizar (Constructores gráficos).

Esta última opción construye una consulta a partir de la selección que establece el usuario de cláusulas y relaciones.

Las primeras herramientas analizadas son de uso universal, aunque está comprobado que las más utilizadas por los usuarios son las nativas al SGBD que utilicen, como por ejemplo PgAdmin. Por lo cual se plantea realizar una aplicación lo más cercana posible en cuanto a interfaz, a esta última, buscando la familiarización de los usuarios.

Las herramientas mineras antes descritas no brindan la flexibilidad de obtención de datos que brindan los diferentes mecanismos y componentes de las herramientas para la administración de bases de datos analizadas en el sub epígrafe anterior. Por lo cual se considera que sería beneficioso presentar ambas partes, navegación básica por las tablas del sistema, además de un editor de consultas SQL que permita la obtención de datos bajo criterios personalizados que defina el usuario. Esta combinación haría de SYAM una herramienta más poderosa, flexible y atractiva.

1.5 Edición de Datos.

Las consultas SQL están definidas según criterios de varios autores como:

Según José Bengoechea Ibaceta “Una consulta es un conjunto de instrucciones en un lenguaje específico de manipulación de datos, con el que se pueden seleccionar, ordenar, filtrar, mostrar, modificar, añadir y borrar datos de una base de datos.”(Ibaceta, 2012)

Según June Jamrich Parsons “Las consultas SQL se dividen en tres elementos sencillos que especifican una acción, el nombre de una tabla de la base de datos y un conjunto de parámetros. Suele comenzar con una palabra clave de acción, o comando que especifica la operación que quiere realizar. ”(Parsons, 2008)

Según Niurka del C. Guerra Cabrera “Se utilizan consultas para ver, modificar y analizar datos de formas diferentes. También pueden utilizarse como el origen de registros para formularios, informes y páginas de acceso a datos. Es decir, es un filtrado de una tabla teniendo criterios específicos. “(Cabrera, 2010)

Estas consultas SQL se utilizan para ver, modificar y analizar datos de formas diferentes. También pueden utilizarse como el origen de registros para formularios, informes y páginas de acceso a datos. Existen varios tipos de consultas, tales como:

Las consultas de selección.

Recuperan datos de una o más tablas y muestran el conjunto de registros en una hoja de datos. También pueden ser utilizadas para agrupar datos y para calcular sumas, recuentos, promedios y otros tipos de totales.

Consulta de parámetros

Muestra un cuadro de diálogo cuando se ejecuta, que pide al usuario que escriba la información que éste desea que se utilice como criterio de la consulta. Se puede diseñar una consulta de parámetros para que solicite más de un dato de un tipo de información.

Consulta de tabla de referencias cruzadas.

Organiza un conjunto de registros para que su presentación sea más clara, mediante encabezados de fila y de columna.

Consulta de acción.

Crea una tabla nueva o modifica una existente agregando, eliminando o actualizando datos de la misma. Debido a la eficacia de las consultas de acción, sobre todo a la hora de modificar datos, es aconsejable realizar una copia de seguridad de sus datos, antes de ejecutar una consulta de acción. (Eduardo Mora Monte, 1999)

1.6 Conclusiones del capítulo.

Después de analizar exhaustivamente el contenido abordado en el actual capítulo se hizo una fundamentación de los componentes teóricos de la presente investigación, con el objetivo de esclarecer y lograr una mejor visión del problema, por lo que se concluye que:

- El estudio realizado sobre la obtención de datos en las herramientas mineras demostró la necesidad de la implementación de un componente de edición para fomentar el buen desempeño de la herramienta que actualmente se encuentra en desarrollo en la universidad
- A través del análisis y estudio de las soluciones existentes tanto a nivel nacional como internacional se sentaron las bases de la investigación y el desarrollo del problema permitiendo la definición del mismo.

Capítulo 2. Tendencias y tecnologías actuales a desarrollar

Para el desarrollo de un software es necesario tener en cuenta tecnologías y herramientas necesarias para guiar el proceso. A continuación se describen las herramientas y tecnologías usadas para la construcción de la herramienta propuesta. Además se caracterizará la metodología de desarrollo a utilizar, haciendo énfasis en sus principales características, así como la herramienta de modelado y el lenguaje de programación implicado en el desarrollo.

2.1 Metodología de Desarrollo de Software

Las metodologías son el conjunto de procedimientos, técnicas, herramientas y un soporte documental, que ayuda a los desarrolladores a realizar un nuevo software. Sin el apoyo de una metodología, nunca se llegaría a satisfacer las necesidades de los clientes para los cuales se trabaja. Se entiende por metodología de desarrollo una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software. Las metodologías de desarrollo se clasifican en ágiles y pesadas(Cockburn, 2006).

2.1.1.Proceso Unificado de Desarrollo

Proceso Unificado de Desarrollo (RUP): es una metodología de desarrollo de software que está basado en componentes e interfaces bien definidas, y junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Es un proceso que puede especializarse para una gran variedad de sistemas de software, en diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto. RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.(Software, 2009)

Como RUP es un proceso, en su modelación define como sus principales elementos:

- ✓ Trabajadores (“quién”): Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.

- ✓ Actividades (“cómo”): Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.
- ✓ Artefactos (“qué”): Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.
- ✓ Flujo de actividades (“cuándo”): Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

Características Principales de RUP.

- Unifica los mejores elementos de metodologías anteriores.
- Preparado para desarrollar grandes y complejos proyectos.
- Orientado a Objetos.
- Utiliza el UML como lenguaje de representación visual.

Principales ventajas.

- Coste del riesgo a un solo incremento.
- Reduce el riesgo de no sacar el producto en el calendario previsto.
- Acelera el ritmo de desarrollo.
- Se adapta mejor a las necesidades del cliente.

Ciclo de vida de RUP

El ciclo de vida de RUP se caracteriza por:

Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso (cómo se llevan a cabo).

Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos

del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los CU relevantes desde el punto de vista de la arquitectura. El modelo de arquitectura se representa a través de vistas en las que se incluyen los diagramas de UML.

Iterativo e Incremental: Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros.(Software, 2009)

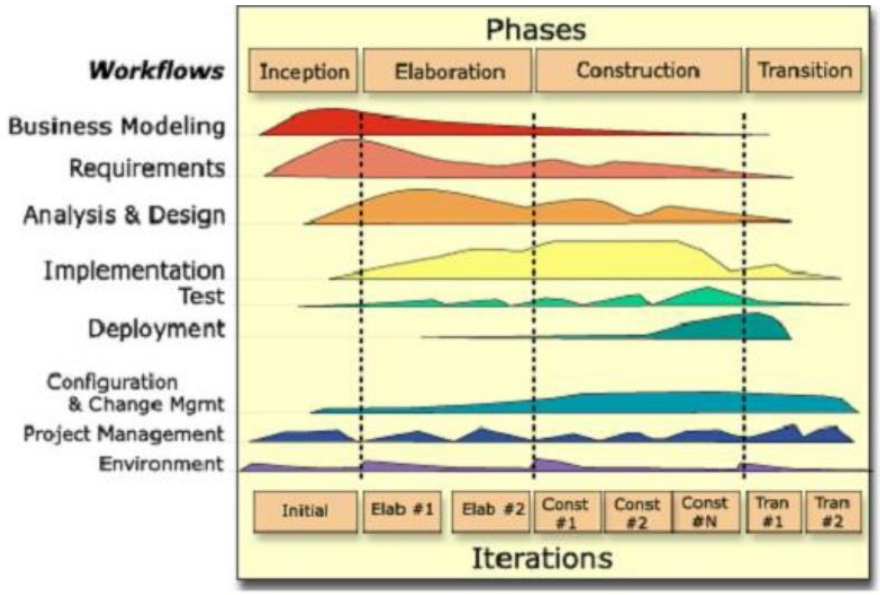


Figura 5: Ciclo de vida de RUP

2.1.2 Microsoft Solution Framework (MSF).

Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas. MSF es un compendio de las mejores prácticas en cuanto a administración de proyectos se refiere. Más que una metodología rígida de administración de proyectos, MSF es una serie de modelos que puede adaptarse a cualquier proyecto de tecnología de información.

Este modelo es parecido (pero no igual) al RUP y es propiedad de Microsoft por lo que resulta un poco caro. Se le nombra Framework debido a que sigue unas particulares filosofías:

- No existe un único proceso que pueda aplicarse a todos los requerimientos y ambientes, aunque se reconoce que siempre existe la necesidad de unas guías.
- Un Framework provee guías sin imponer demasiados detalles que se vuelvan imposibles entender o que solo apliquen a un número limitado de casos

Las características de MSF son:

- ✓ Adaptable
- ✓ Flexible
- ✓ Escalable
- ✓ Agnóstico a tecnologías

Otra de las características de MSF es que ha evolucionado con la experiencia de grupos reales de trabajo que lo ha utilizado desde 1993. Como resultado de esta experiencia las guías se han simplificado, consolidado y verificado para obtener un Framework que sea fácil de entender y adoptar.

Principios fundamentales

Los principios y conceptos de MSF siguientes ayudan a un equipo de proyecto a entregar una solución de calidad. Cada miembro del equipo debe entender y aplicar estos principios en sus interacciones con otros miembros del equipo, con su organización y con las partes interesadas. En la base de MSF hay nueve principios fundamentales:

1. Fomentar las comunicaciones abiertas. Para que el equipo sea eficaz y eficiente, usted y su equipo deben compartir los niveles adecuados de información entre los miembros del equipo y a través de la empresa.
2. Trabajar hacia una visión compartida. Tener una visión compartida capacita a los miembros del equipo y hace posible la agilidad de modo que los miembros del equipo puedan tomar decisiones informadas rápidamente en el contexto de lograr el objetivo de una perspectiva.
3. Autorización para los miembros del equipo. No solo la autorización de los miembros del equipo es una de las muchas maneras de sobrevivir en un entorno en constante cambio, sino

que los miembros del equipo también aprenden a buscar buenas soluciones creativas y a ayudarse entre sí.

4. Establecer la responsabilidad clara y responsabilidad compartida. Los miembros del equipo autorizados se sienten más responsables por sus decisiones y están dispuestos a tener responsabilidad conjunta por el proyecto. Más responsabilidad del miembro del equipo conduce a una mayor calidad.
5. Entregar el valor incremental. La entrega de valor incremental tiene dos facetas:
    1. Asegurar de que lo que se entrega tiene un valor óptimo para las partes interesadas.
    2. Determinar los incrementos óptimos en los que entregar valor o “frecuencia de entrega”.
6. Mantenerse ágil, expectante y adáptese a los cambios. Dado que el cambio puede suceder a menudo en el peor de los momentos, si se dispone de una forma ágil de controlar el cambio, se ayuda a minimizar las interrupciones comunes producidas por el cambio.
7. Invertir en calidad. Muchas organizaciones defienden la calidad, a menudo un término vagamente definido, pero carecen de los conocimientos para cuantificar la calidad.
8. Aprender de todas las experiencias. Si todos los niveles de una organización no aprenden de aquello que función anteriormente, ¿cómo pueden esperar mejorar la próxima vez? Los miembros del equipo deben entender y apreciar que el aprendizaje se da en todos los niveles:

    - A nivel de proyecto, por ejemplo, refinando un proceso para todo el proyecto
    - A nivel individual, por ejemplo, cómo interactuar mejor con otros miembros del equipo
    - A nivel de organización, como ajustar qué métricas de calidad se recopilan para cada proyecto

9. Asociarse con clientes internos y externos. Las posibilidades de éxito del proyecto aumentan cuando el cliente trabaja con el equipo de proyecto. (Software, 2009)

2.1.3 Programación Extrema (XP).

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en la

retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. (Gimson, 2012)

Los principios básicos de la programación extrema son: simplicidad, comunicación, retroalimentación y coraje.

- Simplicidad: la simplicidad es la base de la programación extrema. Se simplifica el diseño para agilizar el desarrollo y facilitar el mantenimiento. Un diseño complejo del código junto a sucesivas modificaciones por parte de diferentes desarrolladores hace que la complejidad aumente exponencialmente. Para mantener las simplicidades necesarias la refactorización del código, ésta es la manera de mantener el código simple a medida que crece. También se aplica la simplicidad en la documentación, de esta manera el código debe comentarse en su justa medida, intentando que el código esté autodocumentado.
- Comunicación: la comunicación se realiza de diferentes formas. Para los programadores el código comunica mejor mientras más simple sea. Si el código es complejo hay que esforzarse para hacerlo inteligible. El código autodocumentado es más fiable que los comentarios ya que estos últimos pronto quedan desfasados con el código a medida que es modificado. La comunicación con el cliente es fluida ya que el cliente forma parte del equipo de desarrollo. El cliente decide qué características tienen prioridad y siempre debe estar disponible para solucionar dudas.
- Retroalimentación: al estar el cliente integrado en el proyecto, su opinión sobre el estado del proyecto se conoce en tiempo real. Al realizarse ciclos muy cortos tras los cuales se muestran resultados, se minimiza el tener que rehacer partes que no cumplen con los requisitos y ayuda a los programadores a centrarse en lo que es más importante. El código también es una fuente de retroalimentación gracias a las herramientas de desarrollo. Por ejemplo, las pruebas unitarias informan sobre el estado de salud del código.
- Coraje o valentía: para los gerentes la programación en parejas puede ser difícil de aceptar, parece como si la productividad se fuese a reducir a la mitad ya que sólo la mitad de los programadores está escribiendo código. Hay que ser valiente para confiar en que la programación por parejas beneficia la calidad del código sin repercutir negativamente en la

productividad. La simplicidad es uno de los principios más difíciles de adoptar. Se requiere coraje para implementar las características que el cliente quiere ahora sin caer en la tentación de optar por un enfoque más flexible que permite futuras modificaciones. No se debe emprender el desarrollo de grandes marcos de trabajo mientras el cliente espera.(Gimson, 2012)

Proceso XP

El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos:

1. El cliente define el valor de negocio a implementar
2. El programador estima el esfuerzo necesario para su implementación
3. El programador construye ese valor
4. Vuelve al paso 1

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden.

No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse de que el sistema tenga el mayor valor de negocio posible.(Gimson, 2012)

Roles de la Programación Extrema (XP)

Según la propuesta de Beck los roles que podemos encontrar son los siguientes:

- ✓ Programador: El programador escribe las pruebas unitarias y produce el código del sistema.
- ✓ Cliente: Escribe las historias de los usuarios y las pruebas funcionales para validar su implementación. El cliente da prioridad a las historias de usuarios y decide cuál implementar en cada iteración centrándose en aportar mayor valor al negocio.
- ✓ Encargado de Pruebas: Ayuda al cliente a escribir las pruebas funcionales. Se encarga de ejecutar las pruebas con regularidad, difunde los resultados obtenidos al equipo y es el responsable de las herramientas que dan soporte a las pruebas.
- ✓ Encargado de Seguimiento: Es el que proporciona la retroalimentación al equipo. Realiza el seguimiento del proceso de cada iteración y verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado en ello para la mejora de futuras estimaciones.

- ✓ Entrenador: Es el responsable del proceso global. Se encarga de proveer guías al equipo de forma que se apliquen las prácticas XP y se vaya siguiendo el proceso correctamente.
- ✓ Consultor: Es un miembro externo del equipo con un conocimiento específico en algún tema que es necesario para el proyecto, en el que surjan problemas.
- ✓ Gestor: Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. (Gimson, 2012)

El ciclo de vida ideal de XP consta de seis fases: Exploración, Planificación de la Entrega, Iteraciones, Producción, Mantenimiento y Muerte del Proyecto:

2.1.4 Selección de la Metodología de Desarrollo.

Después del estudio realizado sobre las metodologías de desarrollo se determinó hacer uso de una metodología pesada en este caso RUP, ya que describe como controlar, rastrear y monitorear los cambios para permitir un desarrollo iterativo exitoso. Es también una guía para establecer espacios de trabajo seguros para cada desarrollador, suministrando el aislamiento de los cambios hechos en otros espacios de trabajo y controlando los cambios de todos los elementos de software (modelos, código, documentos). Teniendo en cuenta que en años posteriores no será el mismo equipo de desarrollo con el que constará el proyecto se adopta esta metodología como la más adecuada para continuar con la línea de trabajo y porque esta provee una documentación bien detallada de todo el proceso y ciclo de vida del software en caso de futuras modificaciones, no siendo así el caso de XP, radicando su mayor problema en la documentación.

2.2 El UML como soporte de la modelación de la solución propuesta.

El Lenguaje Unificado de Modelado, *(del inglés Unified Modeling Language, UML)* es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software (Larman, 2004) .El UML es utilizado por varias metodologías, una de ellas es la que será utilizada en la presente investigación, ya que guardan cierta relación porque RUP hace uso de todos los diagramas propuestos por este lenguaje, en la presente investigación se utiliza en su versión 2.0 para modelar la solución propuesta.

Dicho lenguaje es una notación unificada con la que se permite lograr un entendimiento que propicie el intercambio entre los usuarios y los desarrolladores. Se ha convertido en un estándar de la industria del software, debido a que fue impulsado por los autores (Grady Booch, Ivar Jacobson y James Rumbaugh) de los tres métodos más usados de la orientación a objetos. El UML estándar está compuesto por tres partes: bloques de construcción (tales como clases, objetos, mensajes), relaciones entre los bloques (tales como asociación, generalización) y diagramas (por ejemplo, diagrama de actividad)(Larman, 2004)

2.3 Herramientas de modelado visual.

El proceso de construcción del software requiere que el conjunto de actividades que se establecen sean organizadas y completadas de una forma eficiente y correcta, para de esta forma solucionar una serie de problemas que se generan durante el proceso de desarrollo, ejemplo de ello: insatisfacción del usuario, baja calidad del trabajo, escasa productividad, entre otros. Para atenuar el efecto de los problemas mencionados anteriormente surgen las herramientas de ayuda al desarrollo de sistemas de información, algunas de estas herramientas centran sus objetivos en mejorar la calidad como es el caso de las herramientas de Ingeniería de Software Asistida por Computadoras *(del inglés, Computer Asisted Software Engineering, CASE).*

Se puede definir a las Herramientas CASE “como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software”(Case, 2010)

Las herramientas CASE o Ingeniería de Software Asistida por Computadora son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero. Proporcionan a un ingeniero la posibilidad de automatizar actividades manuales y de mejorar su visión general de la ingeniería. Las herramientas CASE ayudan a garantizar que la calidad se diseñe antes de llegar a construir el producto. Su objetivo principal es poder mostrar al usuario, desde los primeros momentos del diseño, el aspecto que tendrá la aplicación una vez desarrollada.

2.4 Visual Paradigm Enterprise Edition.

Visual Paradigm (sus siglas en inglés VP) es una herramienta CASE que utiliza UML como lenguaje de modelado. Es un producto de calidad que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite crear todos los tipos de diagramas de clases y generar código a partir de estos.

Esta herramienta presenta diversas características, una de ellas es ser multiplataforma y gratis en su edición *Community*, otras características que permiten que sea muy usada en la actualidad son (Valdez, 2006):

- ✓ Se pueden realizar distintos tipos de diagramas, por ejemplo: Diagramas de clases, de Casos de Uso, de Secuencia y de Componentes.
- ✓ Genera el código de aplicación según el modelado de la aplicación.
- ✓ Exportación en imágenes de los diagramas que son elaborados, con formato jpg, png y svg.
- ✓ Fácil de utilizar, instalar y actualizar.
- ✓ Disponible en español.

El VP no está diseñado hacia una metodología de desarrollo en específico, pues se adapta a la mayoría de las metodologías de desarrollo existentes. Teniendo en cuenta todas las ventajas anteriores y por ser además una excelente herramienta para ser utilizada en un ambiente de software libre. Se considera Visual Paradigm en su versión 8.0 como una buena selección para modelar la aplicación propuesta.

2.5 Sistema Gestor de Base de Datos.

Un Sistema Gestor de Base de Datos *(SGBD, en inglés DBMS: Data Base Management System)* es un sistema de software que permite la definición de bases de datos; así como la elección de las estructuras de datos necesarios para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación. Un SGBD relacional es un modelo de datos que facilita a los usuarios describir los datos que serán almacenados en la base de datos junto con un grupo de operaciones para manejar los datos. Los SGBD relacionales son una herramienta efectiva

que permite a varios usuarios acceder a los datos al mismo tiempo. Brindan facilidades eficientes y un grupo de funciones con el objetivo de garantizar la confidencialidad, la calidad, la seguridad y la integridad de los datos que contienen, así como un acceso fácil y eficiente a los mismos.(Martino, 1995)

2.5.1 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tienen nada que envidiarle a otras bases de datos comerciales. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Tiene amplias características, como son:

- DBMS Objeto-Relacional: PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas, herencia, y arreglos.
- Altamente Extensible: PostgreSQL soporta operadores, funciones métodos de acceso y tipos de datos definidos por el usuario.
- Soporte SQL Comprensivo: PostgreSQL soporta la especificación SQL997 e incluye características avanzadas tales como las uniones (joins) SQL92.
- Cliente/Servidor: PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor. Esta es similar al método del Apache 1.3.x para manejar procesos. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.(Postgresql, 2010)

2.5.2 PostGIS.

Es una extensión que proporciona soporte para el manejo de datos geográficos en la base de datos relacional de PostgreSQL. PostGIS implementa las características esenciales para la especificación

SQL del *Open Geospatial Consortium*[22].PostGIS habilita a PostgreSQL para el manejo de datos espaciales, permitiéndole ser usado como un respaldo de base de datos espacial para los Sistemas de Información Geográfica (GIS por sus siglas en inglés), casi como la extensión espacial de Oracle. PostGIS implementa las siguientes características:

- Tipos de datos geométricos para puntos, polígonos, líneas, etc.
- Operadores espaciales para determinar las medidas geoespaciales como el área, distancia, longitud y perímetro.
- Operadores espaciales para determinar el conjunto de operaciones geoespaciales, como la unión, diferencia, diferencia simétrica y buffer.
- Índices espaciales de árboles R para rápidas consultas espaciales.
- Soporte para índices selectivos, para proporcionar alto desempeño en la planeación de consultas para consultas mezcladas espaciales/no espaciales.(Kamilo Cervantes, 2010)

2.5.3 PgAdmin.

Es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia Open Source. Está escrita en C++ usando la librería gráfica multiplataforma wxWidgets[23], lo que permite que se pueda usan en Linux, FreeBSD, Solaris, Mac OS X y Windows. Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y facilita enormemente la administración. La aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar scripts programados.(guia-ubuntu, 2008)

[22]El Open Geospatial Consortium (OGC) fue creado en 1994 y agrupa (en febrero de 2009) a 372 organizaciones públicas y privadas Su fin es la definición de estándares abiertos e interoperables dentro de los Sistemas de Información Geográfica y de la World Wide Web.

[23]wxWidgets: son unas bibliotecas multiplataforma y libres, para el desarrollo de interfaces gráficas programadas en lenguaje C++. Están publicadas bajo una licencia LGPL.

2.5.4 Selección del Gestor de Base de Datos.

El sistema gestor de Base de Datos seleccionado para la solución es PostgreSQL9.1.8 debido a la utilización de cliente-servidor garantiza la estabilidad del sistema, posee todas las características de una base de datos profesional y su documentación es bien detallada y organizada siendo así en la UCI hay una comunidad de PostgreSQL cubano en la que se trabaja en el desarrollo y explotación de este gestor. También se determinó el uso de su extensión PostGIS1.5.3-2 y como aplicación gráfica para el gestionar el gestor PgAdmin 1.14.0-2, debido a que la misma refleja una interfaz amigable y se ajusta a las características presentando un exitoso desarrollo.

2.6 C++ como Lenguaje de Programación.

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

Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. Existen también algunos intérpretes, como ROOT (enlace externo).Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales.

C++ está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel.

Características del lenguaje.

- ✓ Es un lenguaje de programación muy utilizado para el desarrollo de aplicaciones de escritorio.
- ✓ La posibilidad de orientar la programación a objetos permite al programador diseñar aplicaciones desde un punto de vista más cercano a la vida real; permite la reutilización del código de una forma más lógica y productiva.
- ✓ Eficiencia: Es uno de los lenguajes más rápidos en cuanto a ejecución.
- ✓ C++ contiene una extensa cantidad de librerías que evitan la re-implementación de operaciones comunes.

- ✓ Herramientas: Existen multitud de compiladores para C++ en los sistemas operativos y en todos los casos existe más de un ejemplar con licencia libre.
- ✓ Portabilidad: Es un lenguaje multiplataforma sus aplicaciones solamente tienen que ser compiladas nuevamente para adaptarse al sistema operativo, a diferencia de otros lenguajes como Java que necesita de una máquina virtual, lo que hace el proceso mucho más lento. El lenguaje está estandarizado y un mismo código fuente se puede compilar en diversas plataformas. (Mora, 2006)

2.6.1 Selección del lenguaje.

Luego del estudio realizado sobre las características del lenguaje C++ se llega a la conclusión de que este lenguaje posee las características más adecuadas para lograr cumplir los objetivos de la solución propuesta, permitiendo trabajar a alto como a bajo nivel, además de proporcionar grandes facilidades para la programación orientada a objetos. Además es el lenguaje utilizado por el proyecto para el desarrollo de la herramienta.

2.7 QtCreator como Entorno Integrado de Desarrollo.

Es un excelente entorno de desarrollo integrado *(del inglés, Integrated Development Environment, IDE)*. Fue creado por Trolltech para desarrollar aplicaciones en C++ de manera sencilla y rápida. Como su nombre lo indica, está implementado por los desarrolladores de Qt, donde este obtiene toda la ayuda que brinda el framework, es un IDE multiplataforma.(Hernández, 2011)

Entre las características fundamentales que presenta este framework se pueden mencionar:

- Editor avanzado para C++.
- Diseñador de formularios (GUI) integrado.
- Herramientas para la administración y construcción de proyectos.
- Completado automático.
- Depurador visual.

Dentro de las diferentes ventajas que presenta Qt se pueden mencionar las siguientes:

- Es multiplataforma. El mismo código que se escribe en GNU/Linux, puede ser ejecutado fácilmente en Windows XP o en Mac OSX.
- Gracias a las múltiples características que presenta, se hace muy fácil construir aplicaciones para distintos usos.

El API[24] de su biblioteca cuenta con diferentes librerías que facilitan el acceso a bases de datos mediante QSqlQuery y QSqlQueryModel, así como también para uso de XML con QXml. Se pueden utilizar las estructuras de datos tradicionales que posee el lenguaje C++. (Hernández, 2011)

2.7.1 Selección del IDE.

Se determino el uso del IDE de desarrollo QtCreator ya que constituye una buena práctica para la elaboración de aplicaciones de escritorio y da soporte al lenguaje de programación C++(seleccionado para el desarrollo). Ofrece gran compatibilidad con varios sistemas operativos, posee riquezas en sus funciones y librerías permitiendo el desarrollo de interfaces amigables y por su alto rendimiento con C++.Todas las herramientas de Qt son gratuitas por lo que el trabajo se hace mas favorable y contiene gran documentación referente al mismo desde tutoriales hasta fragmentos de código, dando la posibilidad de una mejor visión y organización al interactuar con la herramienta.

2.8 Conclusiones del capítulo.

En este capítulo describieron y abordaron haciendo una profunda investigación las herramientas, tecnologías, lenguajes y metodologías implicados en la realización de la solución, concluyendo que para el buen desarrollo se escoge como lenguaje de programación C++, usando como IDE de Desarrollo QtCreator .Como sistema gestor de base de datos PostgreSQL 9.1.8,su extensión PostGIS 1.5.3-2 y como aplicación gráfica para el gestionar, el gestor PgAdmin 1.14.0-2, además de seleccionar a RUP como metodología de desarrollo.

24 Interfaz de Programación de Aplicaciones.

Capítulo 3. Presentación de la solución propuesta.

En el presente capítulo se realizará la descripción detallada del sistema a desarrollar, identificando los actores y casos de uso, los cuales permitirán un mejor entendimiento y abstracción de las características del sistema. También se abordaran las descripciones de los requisitos funcionales y no funcionales junto con su descripción textual, además de los distintos diagramas generados durante el proceso para la comprensión del modelo del negocio.

3.1 Modelo del Dominio

El Modelo del Dominio captura los tipos de objetos más importantes en el contexto del sistema. Los objetos del dominio representan las "cosas" que existen o los eventos que suceden en el entorno en el que trabaja el sistema; con el objetivo de comprender y describir las principales clases dentro del contexto del sistema. (Ivar Jacobson, 2000)

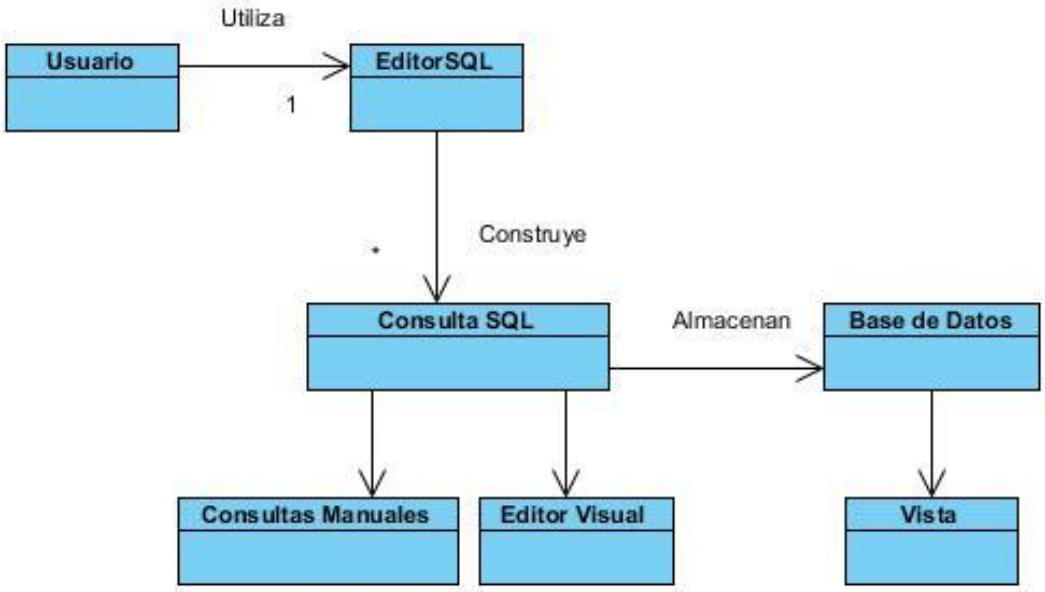


Figura 6: Diagrama de Clases del Dominio.

3.2 Requisitos del Sistema.

En todo el ciclo de vida del software la ingeniería de requisitos ayuda en gran medida al ingeniero de software a entender el problema, pues esta incluye un conjunto de tareas que conducen a aprender cuál será el impacto del software sobre el negocio, qué es lo que el cliente quiere y cómo interactuar los usuarios finales con el software.(Pressman, 2002)

De ahí la importancia de la correcta especificación de los requisitos a tener en cuenta para el éxito del sistema. Estos requisitos se pueden clasifican de la siguiente forma:

3.2.1 Requisitos Funcionales

Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer.(Sommerville, 2005)

RF 1. El sistema debe permitir al usuario del sistema “Abrir una consulta externa.”

RF 2. El sistema debe permitir al usuario del sistema “Exportar resultados”.

RF 3. El sistema debe permitir al usuario del sistema “Salvar una consulta”.

RF 4. El sistema debe permitir al usuario del sistema “Limpiar contenido de la ventana”.

RF 5. El sistema debe permitir al usuario del sistema “Visualizar errores en la consulta”.

RF 6. El sistema debe permitir al usuario del sistema “Ejecutar una consulta”.

RF 7. El sistema debe permitir al usuario del sistema “Salvar una consulta como vista”.

RF 8. El sistema debe permitir al usuario del sistema “Mostrar el Editor Visual de consultas SQL”.

RF 8.1 El sistema debe permitir al usuario del sistema “Seleccionar una tabla”.

RF 8.2 El sistema debe permitir al usuario del sistema “Seleccionar columnas”.

RF 8.3 El sistema debe permitir al usuario del sistema “Ordenar las columnas”.

RF 8.4 El sistema debe permitir al usuario del sistema “Gestionar uniones entre varias tablas del sistema”.

RF 8.5 El sistema debe permitir al usuario del sistema “Gestionar filtros entre una o varias tablas del sistema”.

RF 9 El sistema debe permitir al usuario del sistema “mostrar datos espaciales como texto”.

3.2.2 Requisitos No Funcionales

Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicio individuales del sistema. (Sommerville, 2005)

RNF#1- Usabilidad

- La herramienta a desarrollar presenta una interfaz amigable al igual que su tiempo de respuesta, por lo que el usuario debe poder realizar consultas SQL de una manera rápida y efectiva sin tener un profundo conocimiento sobre el tema, permitiendo que el sistema pueda ser utilizado sin ningún problema, además brindando al usuario una fácil interacción con el mismo.

RNF#2- Eficiencia

- En gran medida la eficiencia del sistema radica en varios aspectos tales como, la velocidad de conexión a la Base de datos y el volumen de información contenida en dicha base de datos, dando posibilidades de acceso por parte de múltiples usuarios.
- El sistema provee al usuario un tiempo de respuesta mínimo a la hora de realizar una consulta, ya sea a través del editor SQL o el editor gráfico, brindándole a este rapidez en la búsqueda de la información.

RNF#4- Portabilidad.

- Para su implementación se usaron herramientas de programación y gestión de bases de datos que son multiplataforma garantizando que pueda ser utilizado en diferentes sistemas operativos como Windows, Linux.

RNF#4- Software:

- Se requerirá como gestor de bases de datos PostgreSQL en su versión 9.1.
- El sistema debe ser multiplataforma con enfoque en tecnologías basadas en software no propietario.

RNF#5-Interfaz externa:

- La aplicación debe mostrar una interfaz sencilla y agradable para el usuario

RNF#6-Soporte

- El sistema debe estar codificado de acuerdo al estándar de codificación para C++ establecido en el proyecto y estándares de Base de Datos de SQL99.

RNF#7- Hardware:

Para las PCs clientes:

- Se requiere tengan tarjeta de red.
- Al menos 512 MB de memoria RAM.
- Se requiere al menos 40 GB de disco duro.
- microprocesador con velocidad superior a los 1.7GHz.

Para los servidores:

- Se requiere tarjeta de red.
- El Servidor de BD tenga como mínimo 2GB de RAM y 80 GB de disco duro.
- Procesador 2 GHz como mínimo.

3.3 Definición de los Casos de Uso del Sistema.

Un caso de uso es una descripción de los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso. Los personajes o entidades que participarán en un caso de uso se denominan actores. En el contexto de ingeniería del software, un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas.(Jacobson, 1992)

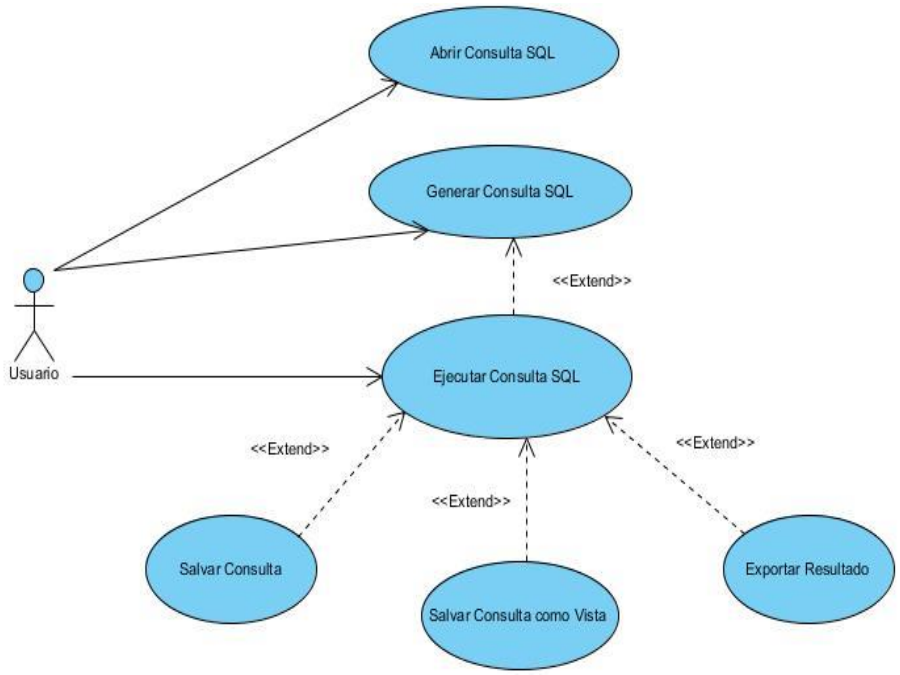


Figura 7: Diagrama de Casos de Uso del Sistema.

3.4 Descripción de los actores del Sistema.

Tabla 1: Descripción de los actores del Sistema

Actor	Descripción
Usuario	Es el encargado de ejecutar los casos de uso del Sistema

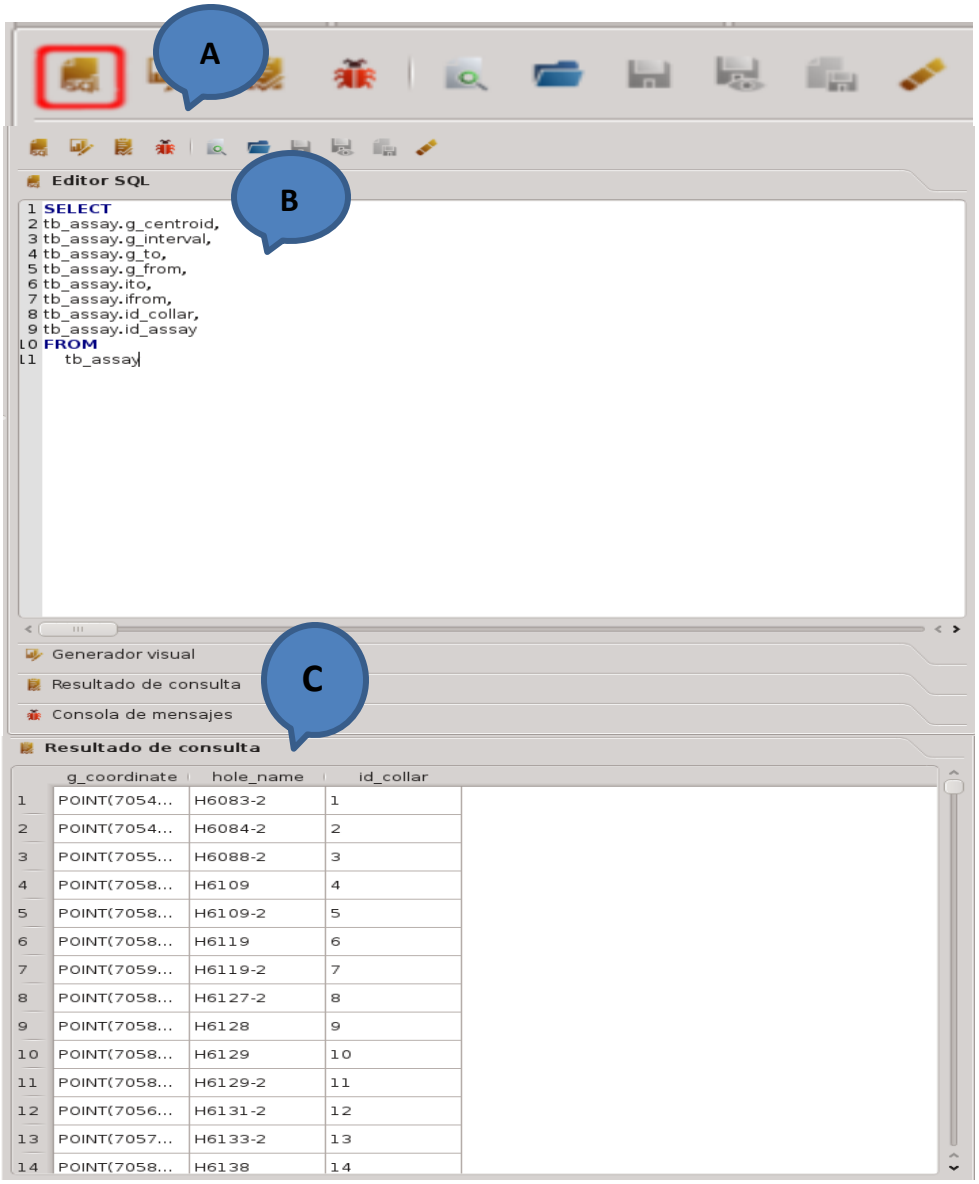
3.5 Descripción de los Casos de Uso del Sistema.

Tabla 2: Caso de Uso: "Ejecutar Consulta SQL"

Caso de uso	Ejecutar Consulta SQL
Actores	Usuario
Resumen	El caso de uso se inicia cuando el usuario selecciona el editor de consultas, el sistema muestra la ventana de la aplicación para que el usuario escriba la consulta. El usuario escribe la consulta y selecciona la opción consultar. El sistema muestra una tabla con el resultado de la consulta terminando el caso de uso.
Precondiciones	El sistema debe tener conexión con la base de datos.
Referencias	
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso se inicia cuando el usuario hace clic izquierdo en la opción del editor de consultas. Ver interfaz A.	2. El sistema muestra la ventana de la aplicación. Ver interfaz B
3. El usuario escribe la consulta en el editor SQL.	
4. El usuario presiona el botón consultar.	5. El sistema muestra en una tabla el resultado de la consulta. Ver interfaz C

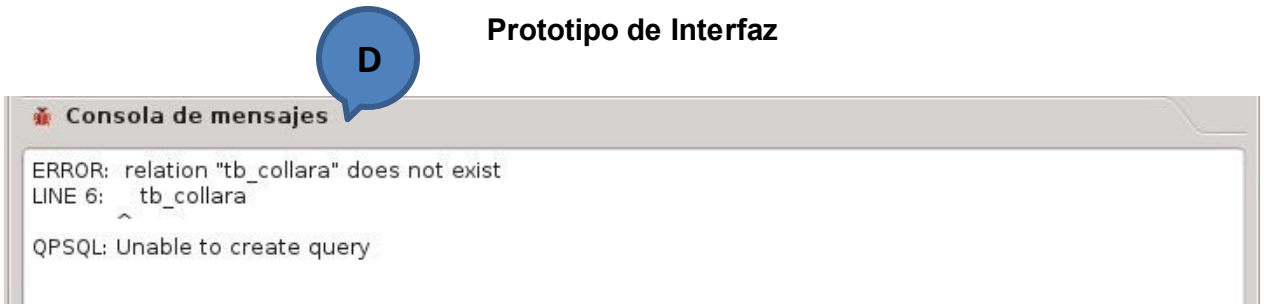
Prototipo de Interfaz



Flujo Alterno	
	1. El sistema muestra la consola de Mensajes con los errores de la consulta y se regresa al

	paso 3. Ver interfaz D.

Prototipo de Interfaz



3.6 Conclusiones del capítulo.

En el presente capítulo se describió la solución propuesta haciendo uso a través del modelado del dominio para un esclarecimiento del sistema, permitiendo una mejor definición del mismo. Se obtuvieron los requisitos funcionales y no funcionales, así como la creación de los diagramas persistentes y las descripciones de los casos de uso, concluyendo que:

- Los requisitos funcionales y no funcionales brindan una breve descripción de las características que presenta el sistema, brindando la posibilidad de la confección del Diagrama de Casos de Uso.
- Los requisitos funcionales permiten mostrar de forma gráfica la información del sistema, para un mejor entendimiento de lo que se pretende desarrollar.
- Los requisitos no funcionales permiten mostrar la información detallada sobre la tecnología a desarrollar.
- Las descripciones de los casos de uso permiten descifrar la manera de cómo es que desea el cliente que quede estructurado el sistema en términos de negocio, posibilita además el trabajo de los desarrolladores.

Capítulo 4. Construcción de la solución propuesta.

En el presente capítulo se abordarán las soluciones a los flujos de trabajo de análisis, diseño, implementación y prueba, reflejando cada uno de estos a través de los artefactos llevados a cabo en la construcción del software mediante los diagramas correspondientes. Se describen además los patrones de diseño y arquitectura presentes en el desarrollo de la solución para la obtención de una mejor visión del diseño y posteriormente de la implementación del sistema se realizaran los diseños de casos de prueba, con el objetivo de validar la solución y verificar las necesidades del cliente.

4.1 Modelo de Análisis.

Un modelo de análisis se describe utilizando el lenguaje de los desarrolladores, y puede introducir un mayor formalismo y ser utilizado para razonar sobre los funcionamientos internos del sistema. También ofrece una especificación más precisa de los requisitos, de modo que facilita su comprensión, su preparación, su modificación, y en general su mantenimiento.(Booch, y otros, 2002)

4.2 Diagrama de clases del análisis.

Modelo conceptual temprano que describe las características y comportamiento comunes de un conjunto de cosas que existen en el sistema. Se indica que es conceptual pues pospone todos los elementos de diseño ya que no considera posibles tecnologías a emplear en el desarrollo del software. Constituyen un prototipo de las futuras clases que darán vida al mismo.(Booch, Grady; Jacobson, Ivar; Rumbaugh, James, 2000). Las clases del Análisis están siempre identificadas con uno de los tres estereotipos siguientes:

Clases de interfaz: Modelan las partes del sistema que dependen de sus actores, lo cual implica que clasifican y reúnen los requisitos en los límites del sistema. Por tanto, un cambio en la interfaz de usuario o en un interfaz de comunicaciones queda normalmente aislado en una o más clases de interfaz.

Clases de entidad: Reflejan la información de un modo que beneficia a los desarrolladores al diseñar e implementar el sistema, incluyendo su soporte de persistencia.

Clases de control: Representan coordinación, secuencia, transacciones, y control de otros objetos y se usan con frecuencia para encapsular el control de los casos de uso.(Booch, y otros, 2002)

En este epígrafe se llevará a cabo la representación de los principales diagramas de clases del análisis.

Figura 8: Diagrama de Clases del Análisis "Ejecutar Consulta SQL".

Figura 9: Diagrama de Clases del Análisis "Generar Consulta SQL".

4.3 Diagramas de Interacción.

Los diagramas de interacción son diagramas que describen como grupos de objetos colaboran para conseguir algún fin. Estos diagramas muestran objetos, así como los mensajes que se pasan entre ellos dentro de un caso de uso. Los diagramas de interacción capturan el comportamiento de los casos de uso.(Pressman, 2002)

4.3.1 Diagramas de Colaboración.

Se utilizan para mostrar cómo colaboran los objetos entre sí para realizar un proceso.Se representa la secuencia de las operaciones a realizar entre los objetos, y los parámetros pasados en los métodos invocados por éstos.(Areba, 2001)

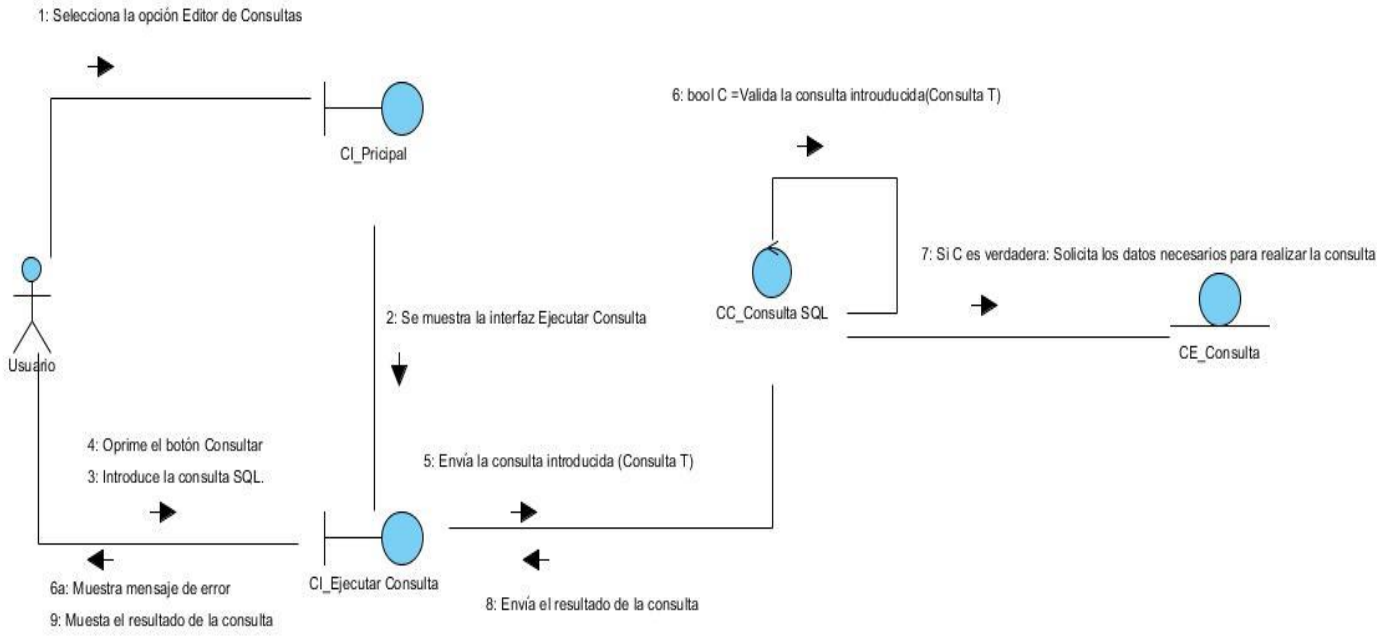


Figura 10: Diagrama de Colaboración "Ejecutar Consulta SQL".

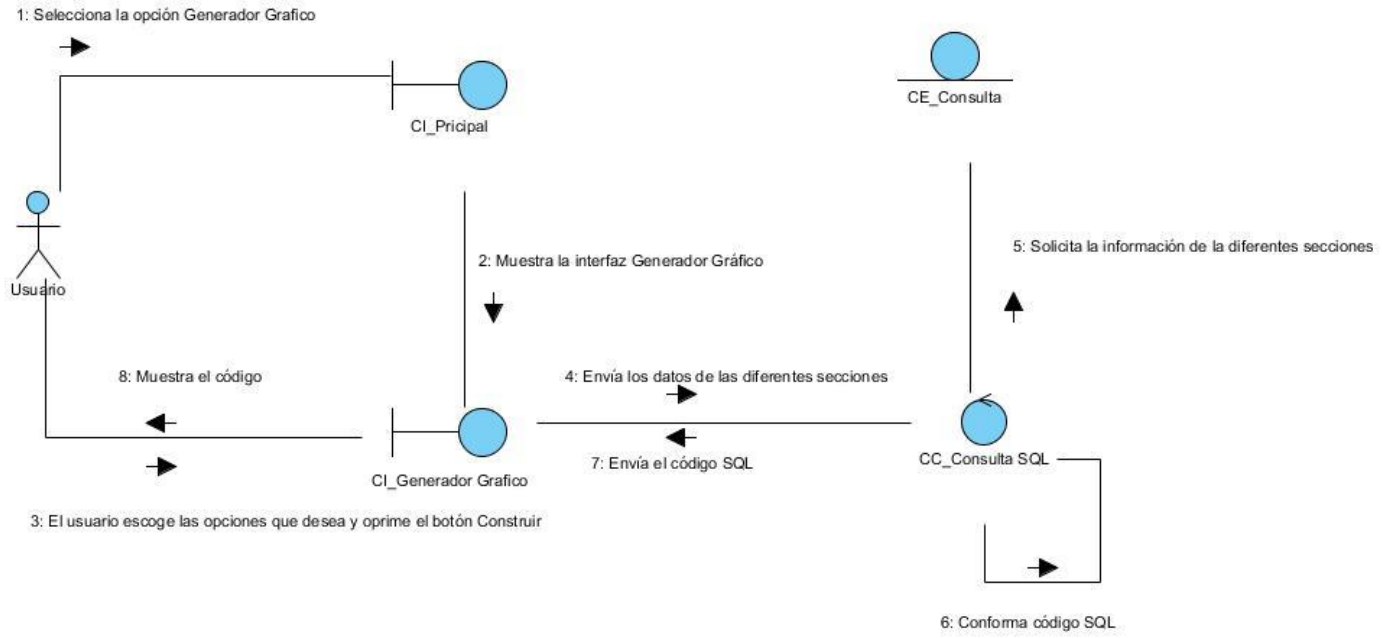


Figura 11: Diagrama de Colaboración "Generar Consulta SQL".

4.3.2 Diagramas de Secuencia.

Un diagrama de secuencia muestra una interacción que está organizada como una secuencia temporal. En particular, muestra los objetos que participan en la interacción mediante sus líneas de vida y los mensajes que intercambian, organizados en forma de una secuencia temporal.(Quintana, 2010)

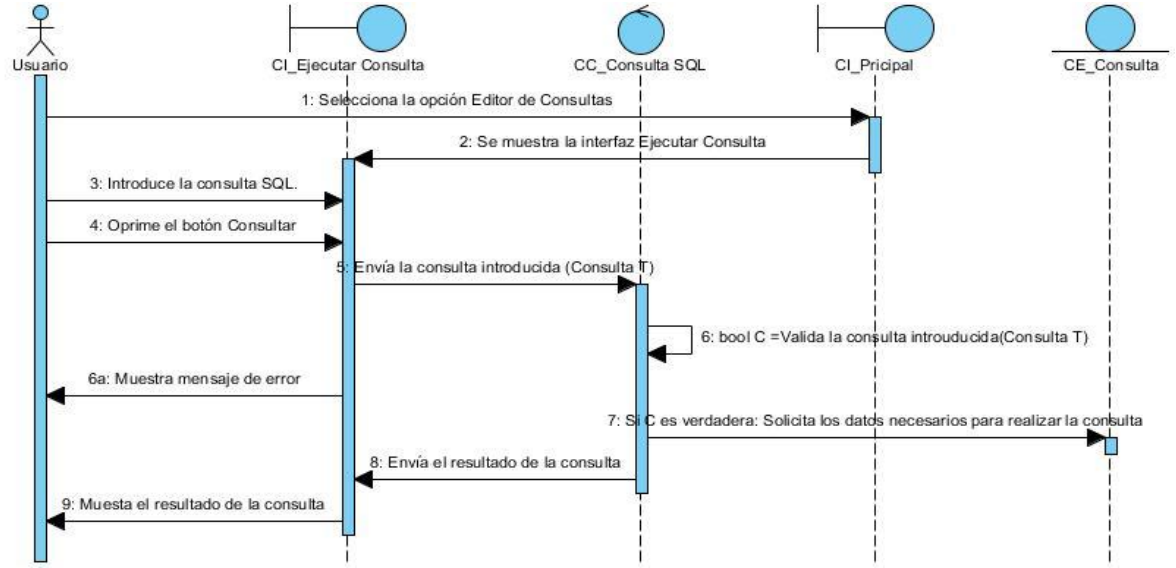


Figura 12: Diagrama de Secuencia "Ejecutar Consulta SQL".

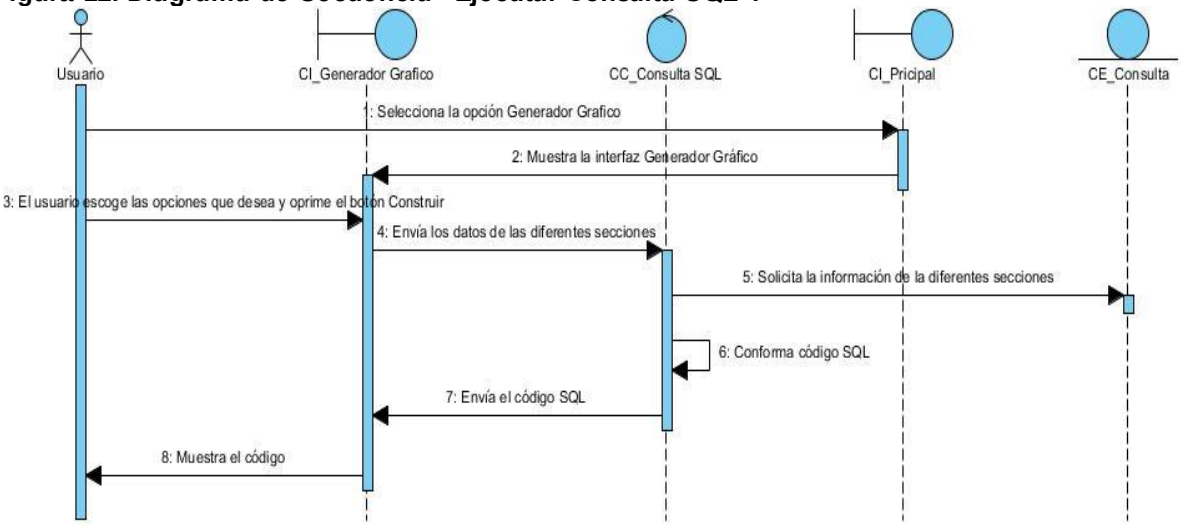


Figura 13: Diagrama de Colaboración "Generar Consulta SQL".

4.4 Diseño.

El diseño tiene el propósito de formular los modelos que se centran en los requisitos no funcionales y en el dominio de la solución y que prepara para la implementación y prueba del sistema. Pretende crear un plano del modelo de implementación, por lo que el grueso del esfuerzo está en las últimas iteraciones de elaboración y las primeras de construcción.

La Disciplina de Diseño tiene como propósito:

- Adquirir una comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia y tecnologías de interfaz de usuario.
- Crear una entrada apropiada y un punto de partida para actividades de implementación, capturando los requisitos o subsistemas individuales, interfaces y clases.
- Descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo.
- Capturar las interfaces entre los subsistemas antes en el ciclo de vida del software, lo cual es muy útil cuando utilizamos interfaces como elementos de sincronización entre diferentes equipos de desarrollo.(Gamma, 1995)

4.4.1 Patrones.

Un patrón se define como una solución probada con éxito que aparece una y otra vez ante determinado tipo de problema en un contexto dado. Los patrones se definen por un nombre, un problema, una solución y las consecuencias de su aplicación. Este define una posible solución correcta para un problema de diseño dentro de un contexto dado, describiendo las cualidades invariantes de todas las soluciones.

Los patrones se categorizan según la escala o nivel de abstracción, sin embargo cada una de las categorías de patrones define un mismo nivel de abstracción o escala de aplicabilidad.(Pressman, 2006)

4.4.1.1 Patrones arquitectónicos.

Los patrones de arquitectura de software son patrones de diseño de software que constituyen una vía en la solución de problemas de arquitectura de software. Los mismos poseen un nivel de abstracción mucho mayor que los patrones de diseño. Además brindan una descripción de los elementos y el tipo de relación que tienen, así como las restricciones para su uso. Los patrones se especifican describiendo los componentes, con sus responsabilidades, relaciones, y las formas en que colaboran.(Buschmann F, y otros, 1996)

4.4.1.2 Modelo Vista.

El modelo se comunica con una fuente de datos, que proporciona una interfaz para los otros componentes en la arquitectura. La naturaleza de la comunicación depende del tipo de fuente de datos, y la forma en que el modelo se implementa.

La vista obtiene índices modelo del modelo, que son referencias a elementos de datos. Mediante el suministro de los índices de modelo para el modelo, la vista puede recuperar los elementos de datos de la fuente de datos.

Modelo: Esta clase define una interfaz que se utiliza por las opiniones y los delegados de acceso a datos. Los datos en sí no tienen que ser almacenados en el modelo, sino que puede ser mantenida en una estructura de datos o repositorio proporcionada por una clase separada, un archivo, una base de datos, o algún otro componente de la aplicación.

Qt proporciona algunos modelos preparados que se pueden utilizar para manejar elementos de datos:

- QStringListModel: Se utiliza para almacenar una simple lista de artículos QString.
- QFileSystemModel: Proporciona información sobre los archivos y directorios del sistema de archivos local.
- QSqlQueryModel, QSqlTableModel y QSqlRelationalTableModel: Se utilizan para acceder a bases de datos utilizando las convenciones modelo / vista.

Vistas: Implementaciones completas se proporcionan para los diferentes tipos de vistas:

- QListView: Muestra una lista de elementos.

- QTableView: Muestra los datos de un modelo en una tabla.
- QTreeView: Muestra los elementos del modelo de datos en una lista jerárquica. Cada una de estas clases se basa en la clase base abstracta QAbstractItemView.

4.4.1.3 Arquitectura en Capas.

La arquitectura en capas es una organización jerárquica tal que cada capa proporciona servicios a la capa inmediata superior y se sirve de las prestaciones que le brinda la inmediata inferior. Las ventajas del estilo en capas son muchas, primero que nada, el estilo soporta un diseño basado en niveles de abstracción crecientes, lo cual a su vez permite a los programadores la participación de un problema complejo en una secuencia de pasos incrementales. En segundo lugar admite optimizaciones y refinamientos. Proporciona además amplia reutilización, lo que se convierte en uno de los fuertes de esta arquitectura.

En el caso de la Arquitectura en Capas se aplica estructurando el trabajo en dos capas, una de ellas representa la interfaz de comunicación conteniendo además toda la lógica del negocio y la otra capa es la de acceso a datos.

4.4.1.4 Arquitecturas orientadas a objetos

Los componentes de un sistema encapsulan los datos y las operaciones que se deben realizar para manipular los datos. La comunicación y la coordinación entre componentes se consiguen a través del paso de mensajes.

Características

- En este estilo los componentes son los objetos, o instancias de tipos de datos abstractos.
- Estos objetos son de un tipo de componente denominado manager porque es responsable por preservar la integridad de un recurso.
- Los objetos interactúan a través de invocaciones a procedimientos y funciones.

Aspectos importantes:

Un objeto es responsable de preservar la integridad de su representación (usualmente manteniendo algún invariante). La representación se oculta a otros objetos.(Software, 2009)

En el caso de la Arquitectura Orientada a Objetos sirve de guía para entender mejor la programación orientada a objetos siendo aplicada en la biblioteca según todos sus conceptos.

4.4.2 Patrones de diseño.

Un patrón de diseño constituye una solución estándar para un problema común de programación en el desarrollo del software. Además es una técnica muy eficaz para flexibilizar el código haciéndolo satisfacer ciertos criterios, así como permite una manera más práctica de describir ciertos aspectos de la organización de un programa.(Gamma, 1995)

4.4.2.1 Patrones GOF.

- De Creación:

Singleton (instancia única) garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Se utiliza en el objeto de base de datos que es una instancia única para el proyecto en uso.

- Estructurales:

Facade (Fachada): Provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema. Se utiliza al unificar las interfaces en una sola ventana, manejando esta varias interfaces (mensajes, datos, consulta, etc.)

- De Comportamiento:

Observer (Observador): Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él. Se hace uso de este en el generador visual, al actualizar las columnas, automáticamente se actualizan en sus respectivas interfaces, esto se logra con el mecanismo de señales y ranura que implementa Qt.

4.4.2.2 Patrones GRASP.

Bajo Acoplamiento: Debe haber pocas dependencias entre las clases. Si todas las clases dependen de todas ¿cuánto software podemos extraer de un modo independiente y reutilizarlo en otro proyecto? El gestor de consultas es completamente independiente del módulo de base de datos, solo utilizando la conexión a la base de datos que está en uso, las clases poseen mínimas dependencias de otras.

Alta cohesión: Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. Cada clase realiza una función especializada que va de acuerdo con el rol asignado.

Creador: Se asigna la responsabilidad de que una clase B cree un Objeto de la clase A solamente cuando:

- ✓ B contiene a A.
- ✓ B es una agregación (o composición) de A.
- ✓ B almacena a A.
- ✓ B tiene los datos de inicialización de A (datos que requiere su constructor).
- ✓ B usa a A.

4.5 Diagrama de Clases del Diseño.

Los diagramas de clases se utilizan para modelar la visión estática de un sistema. Esta visión soporta los requisitos funcionales del sistema, en concreto, los servicios que el sistema debería proporcionar a sus usuarios finales. Normalmente contienen: clases, interfaces y relaciones entre ellas: de asociación, de dependencia y/o generalización.

Los diagramas de clases también pueden contener a paquetes o subsistemas, que se usan para agrupar elementos del modelo en partes más grandes (por ejemplo, paquetes que, a su vez contienen a varios diagramas de clases)(Pressman, 2005)

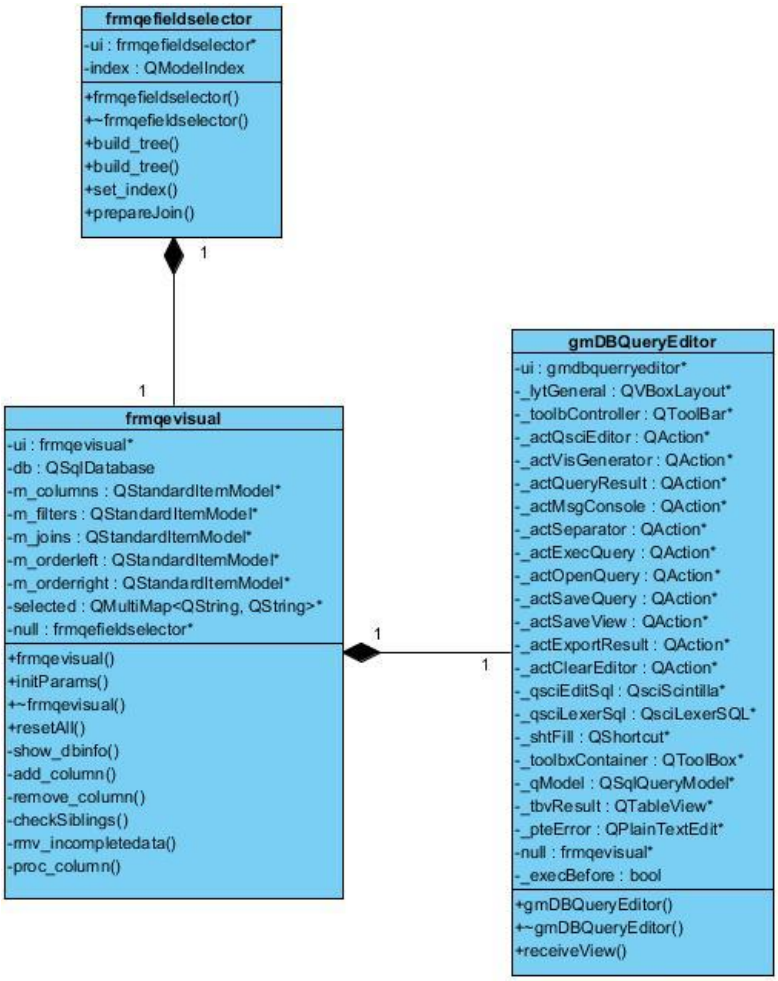


Figura 14: Diagrama de Clases del Diseño.

4.6 Implementación.

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes. Describe también la organización de los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados y la dependencia entre estos.

Esta disciplina explica cómo desarrollar, organizar, realizar pruebas de unidad e integrar los componentes implementados basándose en las especificaciones del diseño.

La finalidad de la implementación es:

- ✓ Definir la organización del código, en términos de los subsistemas de implementación, organizados en capas.
- ✓ Implementar los elementos de diseño en términos de los elementos de implementación (archivos de origen, binarios, programas ejecutables y otros).
- ✓ Probar y desarrollar componentes como unidades.
- ✓ Integrar los resultados producidos por los implementadores individuales, o equipos en un sistema ejecutable. (Corp, IBM, 2006)

4.6.1 Modelo de Despliegue.

El modelo de despliegue es utilizado para capturar los elementos de configuración del procesamiento y las conexiones entre esos elementos. También se utiliza para visualizar la distribución de los componentes de software en los nodos físicos. El mismo está compuesto por:

- ✓ Nodos: Elementos de procesamiento con al menos un procesador, memoria, y posiblemente otros dispositivos.
- ✓ Dispositivos: Nodos estereotipados sin capacidad de procesamiento en el nivel de abstracción que se modela.
- ✓ Conectores: Expresan el tipo de conector o protocolo utilizado entre el resto de los elementos del modelo. (Gamma, 1995)

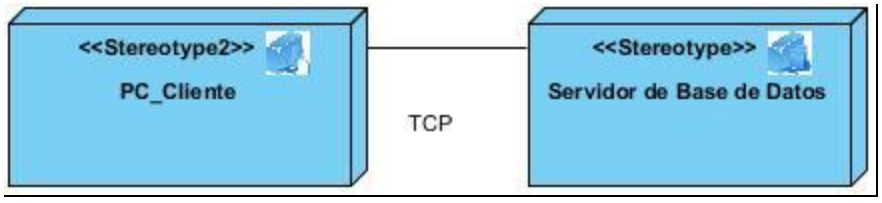


Figura15: Diagrama de Despliegue.

4.6.2 Diagrama de Componentes.

Contiene los elementos de arquitectura física necesarios para determinar los componentes e interfaces a desarrollar en el resto del ciclo de vida.(Areba, 2001). Un diagrama de componentes es

utilizado para estructurar el modelo de implementación en términos de subsistemas y modelar la vista estática de un sistema, describiendo sus elementos físicos y las relaciones entre estos.

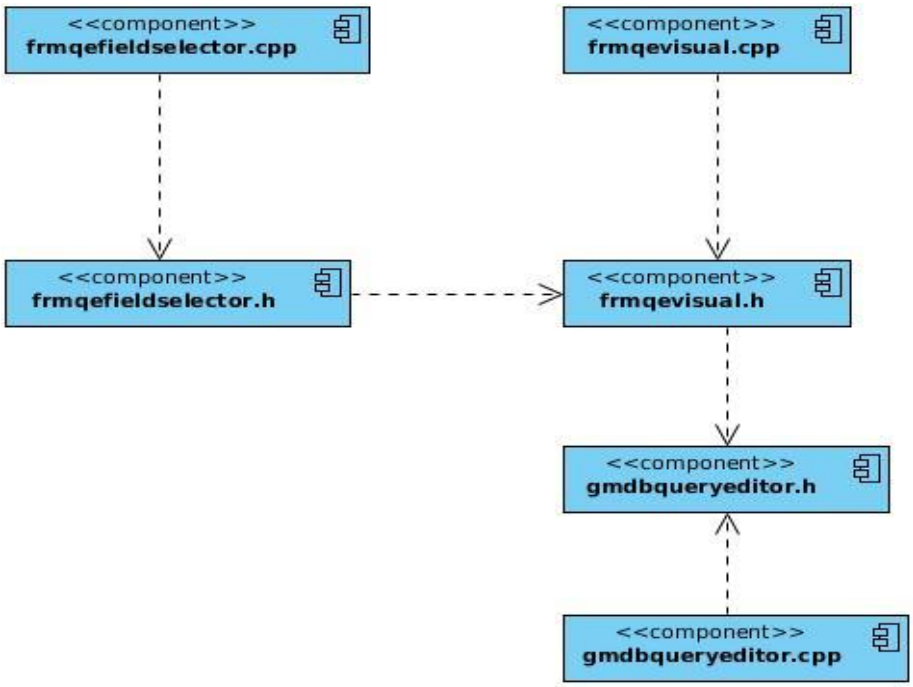


Figura 16: Diagrama de Componentes "Código Fuente".

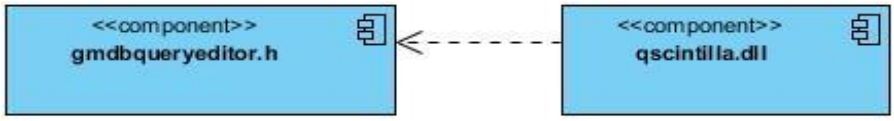


Figura 17: Diagrama de Componentes "Código Ejecutable".

4.7 Prueba.

La prueba del software es un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y la codificación. La prueba requiere que se descarten las ideas preconcebidas sobre la "corrección" del software que se acaba de desarrollar y se supere cualquier conflicto de intereses que aparezca cuando se detecten errores.(Arechavala, 2001)

RUP define cuatro niveles de prueba: unidad, integración, sistema y aceptación, dentro de los cuales se encuentran diferentes tipos de pruebas, como son:

- Pruebas de unidad: se realizan durante la fase de construcción, específicamente en el flujo de trabajo de implementación; las cuales se basan en probar los componentes implementados como unidades individuales. Las pruebas de unidad están divididas en dos grupos, pruebas de caja blanca y pruebas de caja negra.
- Pruebas de integración: se llevan a cabo durante la fase de construcción, las mismas involucran a un número creciente de módulos y terminan probando el sistema como conjunto. Estas pruebas se pueden plantear desde un punto de vista estructural o funcional. Verifican que los componentes interaccionan entre sí de un modo apropiado después de haber sido integrados en el sistema. Se toman como casos de prueba los casos de uso del diseño. Para ello se utiliza el diagrama de secuencia correspondiente y se diseñan combinaciones de entrada y salida del sistema que lleven a distintas utilizaciones de las clases y en consecuencia de los componentes, que participan en el diagrama.
- Pruebas de sistema: prueban que el sistema funciona globalmente de forma correcta. Cada prueba del sistema prueba combinaciones de casos de uso bajo condiciones diferentes. Se prueba el sistema como un todo probando casos de uso unos detrás de otros y si es posible, en paralelo. En este nivel existen una gran variedad de pruebas que se utilizan según el interés que se tenga respecto al funcionamiento del software.
- Pruebas de aceptación: se realizan para permitir que el cliente valide y verifique todos los requisitos pactados. Estas pruebas las realiza el usuario final en lugar del responsable del desarrollo del sistema. El cliente es quien impone los requisitos pues quien mejor que él para dar fe de su satisfacción.

4.7.1 Pruebas Unitarias o de unidad.

Las pruebas unitarias aíslan cada parte del programa y muestran que las partes individuales son correctas. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. La idea es escribir casos de prueba para cada función no trivial o método en el módulo de forma que cada caso sea independiente del resto.

Estas pruebas aisladas proporcionan ventajas básicas:

- Fomentan el cambio: las pruebas unitarias facilitan que el programador cambie el código para mejorar su estructura (lo que se ha dado en llamar refactorización), puesto que permiten hacer pruebas sobre los cambios y así asegurarse de que los nuevos cambios no han introducido errores.
- Simplifica la integración: puesto que permiten llegar a la fase de integración con un grado alto de seguridad de que el código está funcionando correctamente. De esta manera se facilitan las pruebas de integración.
- Los errores están más acotados y son más fáciles de localizar: dado que se prueban pequeñas porciones del software.

Para la realización de las pruebas unitarias las más utilizadas son las pruebas de Caja Blanca y Caja Negra.

4.7.1.1 Pruebas de Caja Negra.

Las pruebas de Caja negra se llevan a cabo sobre la interfaz del software. El objetivo es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene.(Booch, Grady; Jacobson, Ivar; Rumbaugh, James, 2000)

La prueba de caja negra intenta encontrar errores de las siguientes categorías:

- Funciones incorrecta o ausente.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

Partición equivalente: Método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

Una clase de equivalencia representa un conjunto de estados válidos o no válidos para condiciones de entrada. Típicamente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica. El objetivo de partición equivalente es reducir el posible conjunto de casos de prueba en uno más pequeño, un conjunto manejable que evalúe bien el software. Se toma un riesgo porque se escoge no probar todo. Así que se necesita tener mucho cuidado al escoger las clases. La partición equivalente es subjetiva. Dos probadores quienes prueban un programa complejo pueden llegar a diferentes conjuntos de particiones.

En el diseño de casos de prueba para partición equivalente se procede en dos pasos:

1. Se identifican las clases de equivalencia. Las clases de equivalencia son identificadas tomando cada condición de entrada (generalmente una oración o una frase en la especificación) y repartiéndola en dos o más grupos.

Es de notar que dos tipos de clases de equivalencia están identificados: las clases de equivalencia válidas representan entradas válidas al programa, y las clases de equivalencia inválidas que representan el resto de los estados posibles de la condición (es decir, valores erróneos de la entrada).

2. Se define los casos de prueba. El segundo paso es el uso de las clases de equivalencia para identificar los casos de prueba. El proceso es como sigue: se asigna un número único a cada clase de equivalencia. Hasta que todas las clases de equivalencia válidas han sido cubiertas por los casos de prueba, se escribe un nuevo caso de prueba que cubra la clase de equivalencia válida. Y por último hasta que los casos de prueba hayan cubierto todas las clases de equivalencia inválidas, se escribe un caso de la prueba que cubra una, y solamente una, de las clases de equivalencia inválidas descubiertas. (Yailen Rondón Calzadilla, 2010)

4.7.2 Diseños de Caso de Pruebas.

Caso de Uso: "Ejecutar Consulta SQL".

Descripción General.

El caso de uso se inicia cuando el usuario selecciona el editor de consultas, el sistema muestra la ventana de la aplicación para que el usuario escriba la consulta. El usuario escribe la consulta y

selecciona la opción consultar. El sistema muestra una tabla con el resultado de la consulta terminando el caso de uso.

Condiciones de Ejecución.

El sistema debe tener conexión con la Base de Datos.

Tabla 3: DCP: Caso de Uso "Ejecutar Consulta SQL"

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC1: "Ejecutar Consulta SQL"	EC 1.1: "Ejecutar Consulta SQL con éxito "	El usuario selecciona el editor de consultas para realizar una consulta. El sistema muestra la ventana de la aplicación para que el usuario escriba la consulta que desea. El sistema muestra una tabla con el resultado de la consulta realizada terminando el caso de uso.	Editor de Consultas para el Sistema de Análisis y Modelado de Yacimientos Minerales (SYAM). 1. Página Principal 2. El usuario selecciona el Editor de Consultas. 3. El sistema muestra la ventana de la aplicación. 4. El usuario escribe la consulta en el Editor SQL. 5. Da clic en el botón

			Consultar. 6. El sistema muestra una tabla con el resultado de la consulta.
	EC 1.2: "Ejecutar Consulta SQL sin éxito "	El usuario no realiza la consulta correctamente. El sistema muestra la consola de Mensajes con los errores de la consulta y se regresa al paso 3 del escenario 1.1	Editor de Consultas para el Sistema de Análisis y Modelado de Yacimientos Minerales (SYAM). 1. Página Principal 2. El usuario selecciona el Editor de Consultas. 3. El sistema muestra la ventana de la aplicación. 4. El usuario escribe la consulta. 5. Da clic en el botón Consultar. 6. El sistema muestra la consola de Mensajes con

			los errores de la consulta

Descripción de las Variables.

Tabla 4: Descripción de las Variables "CU: Ejecutar Consulta SQL"

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Consulta	Campo de texto	NO	El usuario escribe la consulta en la ventana del

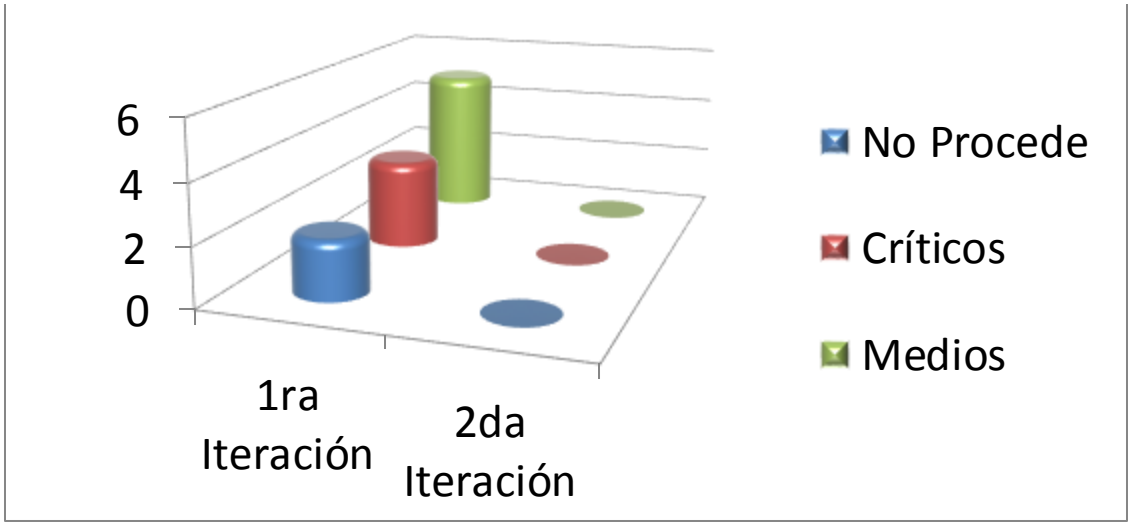
Matriz de Datos.

Tabla 5: Matriz de Datos "CU: Ejecutar Consulta SQL"

ID del escenario	Escenario	V1	Respuesta del Sistema	Resultado de la Prueba
EC 1.1:	Ejecutar Consulta SQL con éxito.	V	Ejecuta la consulta correctamente	Satisfactorio
EC 1.2:	Ejecutar Consulta SQL sin éxito.	N/A	Muestra la consola de Mensajes con los errores de la consulta	Satisfactorio

4.8 Resultados de las Pruebas.

Durante la primera iteración de pruebas realizadas se detectaron 10 no conformidades. De estas dos no procedían, y las restantes se detectaron que afectaban el sistema 3 de forma crítica y 5 de forma media.



Después de realizarse la segunda iteración se pudo comprobar que el 100 % de las no conformidades fueron resueltas satisfactoriamente y no se detectó ninguna nueva, siendo así el correcto funcionamiento de la herramienta.

4.9 Conclusiones del capítulo.

En el presente capítulo se representaron los diagramas correspondientes a los flujos de trabajo análisis, diseño, implementación y prueba, concluyendo que:

- Para un buen desarrollo de la solución se especificaron y abordaron los patrones de diseño y arquitectura permitiendo una descripción de los elementos y el tipo de relación que tienen, así como las restricciones a para su uso, además de constituir una solución estándar para problemas comunes de programación en el desarrollo del software.
- Se plasmaron los mecanismos de implementación acordes con el análisis, el diseño y el diagrama de despliegue del sistema
- Se ejecutaron los diferentes tipos de pruebas especificados para validar el correcto funcionamiento de la aplicación utilizando pruebas de tipo caja negra.
- Se realizaron los casos de prueba, basados en casos de uso que sirvieron como documentos de apoyo y guía al grupo de calidad que llevó a cabo el proceso de liberación.

Conclusiones Generales.

Una vez terminada la investigación es posible afirmar que los objetivos trazados para el desarrollo de la misma fueron cumplidos

- Se realizó un estudio sobre las herramientas y tecnologías que podían aportar elementos al desarrollo de la solución propuesta, esto brindo la posibilidad de obtener como resultado final una herramienta con las características y funcionalidades necesarias para el uso de los usuarios a los cuales está destinado.
- A través del estudio minucioso de la edición y consulta de datos en las diferentes herramientas integradas a los gestores de Base de Datos y las integradas a Software Mineros, facilitó las pautas para una correcta visión de lo que se quería implementar, resultado lo esperado.
- La metodología adecuada permitió la correcta obtención de los artefactos y diagramas generados en todas las fases, así como la realización del análisis y diseño de la aplicación.
- El editor de consultas SQL se implementó de acuerdo a los requisitos funcionales u no funcionales identificados erradicándose los problemas existentes plasmados a través de la situación problemática.
- Las pruebas de caja negra realizadas a la aplicación permitieron detectar un conjunto de no conformidades, las cuales fueron erradicadas.

Recomendaciones.

Partiendo de la experiencia acumulada durante el proceso de construcción de la herramienta y de los resultados obtenidos en la presente investigación se recomienda:

- Incluir en el generador visual una interfaz que permita dibujar las tablas y hacer las uniones mediante arrastrado y soltado.
- Incluir un visualizador 3d en el resultado de la consulta.

Trabajos citados

AquaFold Inc. 2013. AquaFold. [En línea] 2013. [Citado el: 18 de 3 de 2013.] http://www.aquafold.com/aquadatastudio.html.

Areba, Jesús Barranco de. 2001. Metodología Del Análisis Estructurado de Sistemas. Madrid : s.n., 2001.

Arechavala, Yolanda Gonzalez. 2001. Calidad de Software 1. 2001.

Booch, Grady, Jacobson, Ivar y Rumbaugh, James. 2002. El Proceso Unificado de Desarrollo de Software. Madrid : s.n., 2002.

Booch, Grady; Jacobson, Ivar; Rumbaugh, James. 2000. El Proceso Unificado de Desarrollo de Software. Madrid : s.n., 2000.

Buschmann F, Meunier, R, Rohnert y Sons, . s.l. : John Wiley &. 1996. Pattern-Oriented Software Architecture: A System of. 1996.

Cabrera, MsC. Niurka del C. Guerra. 2010. Diseño y creación de bases de datos. Capítulo 9: Creación de consultas. Algunos elementos del Lenguaje SQL (1/4). 2010.

Case, Herramientas. 2010. Herramientas Case. 2010. Capítulo 1: Herramientas Case. Capítulo 1: Herramientas Case. [En línea] 2010. http://es.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.

Cockburn, Alistair. 2006. Agile Software Development. 2006.

Corp, IBM. 2006. Rational Software Architect. 2006.

Datamine CO.ltd. 2006. Ayuda del Editor de Tablas. The Datamine Library. 2006. 040806.

Eduardo Mora Monte, VV Staff,Marta E. Zorrilla Pantaleón. 1999. Iniciación a las Bases de Datos con Access 2002. 1999.

Gamma, Erich. 1995. Design Patterns: Elements of Reusable Object-Oriented Software. 1995.

Gimson, Lic. Loraine. 2012. Metodologías ágiles y desarrollo basado en conocimientos. 2012.

Godet, Michel. 2007. Caja de herramientas . 2007.

guia-ubuntu. 2008. guia-ubuntu. [En línea] 2008. http://www.guia-ubuntu.com/index.php?title=PgAdmin_III.

Hernández, Frank Emilio. 2011. Desarrollo del componente: manejador de datos del servidosde streaming distribuido allfrys. La Habana : s.n., 2011.

Ibaceta, José Bengoechea. 2012. Microsoft Access : diseño de aplicaciones sencillas de bases de datos. 2012.

Ivar Jacobson, Grady Booch y James Rumbaugh. 2000. El Proceso Unificado de Desarrollo de Software. Madrid : Addison Wesley, 2000.

Jacobson, Ivar, Rumbaugh, James y Booch, Grady. 2000. El proceso unificado de desarrollo: Addison Wesley Capítulso 9. 2000.

Jacobson, P.Jonsson, M. Christerson and G.Overgaard. 1992. Ingeniería de Software orientada a Objetos- Un acercamiento a través de lod casos de uso. 1992.

Kamilo Cervantes. 2010. Scribd. [En línea] 2010. http://es.scribd.com/doc/45240510/Servidor-de-PostGIS.

Larman, Craig. 1999.UML y Patrones Introducción al análisis y diseño orientado a objetos. México : PRENTICE HALL, 1999.

Lobaina, Arístides Alejandro Legrá. 1999. Metodología para el pronóstico, planificación y control integral de la minería en yacimientos lateríticos. 1999.

Martínez, Rafael. 2010. Sobre PostgrSQL. [En línea] 2 de Octubre de 2010. [Citado el: 6 de Febrero de 2013.] http://www.postgresql.org.es/sobre_postgresql.

Martino, E. A Bertino y. 1995. Sistemas de bases de datos orientadas a objetos. s.l. : Ediciones Díaz de Santos. 1995.

Parsons, June Jamrich. 2008. Conceptos de Computación: Nuevas Perspectivas. decima edicion. 2008.

Pérez, Liset. 2008. Estartegia para el desarrollo de requisitos funcionales de software en proyectos productivos de la UCI. Ciudad de la Habana : s.n., 2008.

pgAdmin. 2009. pgAdmin PostgreSQL tools. [En línea] 26 de Marzo de 2009. [Citado el: 7 de Febrero de 2013.] http://www.pgadmin.org/licence.php.

Pressman, Roger. 2008. Capítulo 8: Modelado del Análisis. Ingeniería de Software. New York : s.n., 2008.

Pressman, Roger S. 2005. Ingeniería del Software. 2005.

Pressman, Roger S. Ingeniería de Software. Un enfoque práctico. Ingeniería de Software. Un enfoque práctico. s.l. : Mc Graw Hill.

Pressman, Roger. 2010. Software Engineering: A Practitioner´s Approach. New York : s.n., 2010.

Quintana, Abelardo López. 2010. Diseño e implementación del módulo de Gestión de indicadores del proyecto del Ministerio del Poder Popular de la Comunicación y la Información. Cuidad Habana : s.n., 2010.

Rieder, Serge. 2013. Dbeaver Universal Database Manager. [En línea] 2013. [Citado el: 18 de 3 de 2013.] http://dbeaver.jkiss.org/.

Software, Arquitectura de. 2009. Arquitectura de Software. [En línea] 2009.

Sommerville, Ian. 2005. Ingeniería de software. 2005.

Unam. 2011. Unam. [En línea] 2011. [En línea] 18 de agosto de 2011. [Citado el: 9 de enero de 2013.] http://www.unam.edu.pe/webminas/index.php?option=com_content&view=article&id=81:softwares-mineros..

UNAM, Universidad Nacional de Moquegua. 2013. Universidad Nacional de Moquegua Universitas Universitatis. [En línea] 2013. [Citado el: 5 de 3 de 2013.] http://www.unam.edu.pe/webminas/index.php?option=com_content&view=article&id=81:softwares-mineros..

Valdez, Alfonso Altamirano. 2006. Comparativos de Entornos de Desarrollo Integrados. 2006.

Yailen Rondón Calzadilla, Yaelsy Tamayo Echemendia. 2010. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas: Estrategia de pruebas de software para el proyecto Fuerza de Trabajo Calificada. 2010.

Bibliografía

Álvarez, Gonzálo. 1999. ¿ Qué es Java ? Características del lenguaje. [En línea] 1999. [Citado el: 8 de Enero de 2013.] http://www.iec.csic.es/criptonomicon/java/quesjava.html.

Álvarez, Marcos Manuel. 2012. Tecnologías de la Información. 2012.

AquaFold Inc. 2013. AquaFold. [En línea] 2013. [Citado el: 18 de 3 de 2013.] http://www.aquafold.com/aquadatastudio.html.

Areba, Jesús Barranco de. 2001. Metodología Del Análisis Estructurado de Sistemas. Madrid : s.n., 2001.

Arechavala, Yolanda Gonzalez. 2001. Calidad de Software 1. 2001.

Beck, Kent. 2000. Extreme Programming Explained. Madrid : Addison Wesley. 2000. 201-61641-6.

Booch, Grady, Jacobson, Ivar y Rumbaugh, James. 2002. El Proceso Unificado de Desarrollo de Software. Madrid : s.n., 2002.

Booch, Grady; Jacobson, Ivar; Rumbaugh, James. 2000. El Proceso Unificado de Desarrollo de Software. Madrid : s.n., 2000.

Buschmann F, Meunier, R, Rohnert y Sons, . s.l. : John Wiley &. 1996. Pattern-Oriented Software Architecture: A System of. 1996.

Cabrera, MsC. Niurka del C. Guerra. 2010. Diseño y creación de bases de datos. Capítulo 9: Creación de consultas. Algunos elementos del Lenguaje SQL (1/4). 2010.

Carrillo, Isaías. 2008. Metodología del Desarrollo del Software. 2008.

Case, Herramientas. 2010. Herramientas Case. 2010. Capítulo 1: Herramientas Case. Capítulo 1: Herramientas Case. [En línea] 2010. http://es.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.

Cockburn, Alistair. 2006. Agile Software Development. 2006.

Corp, IBM. 2006. Rational Software Architect. 2006.

Costal, Dolores. 2009. Introducción al diseño de bases de datos. 2009.

Datamine CO.ltd. 2006. Ayuda del Editor de Tablas. The Datamine Library. 2006. 040806.

De Nobrega, María. 2012. Herramientas CASE: Rational Rose. [En línea] 2012. [Citado el: 10 de Diciembre de 2012.] http://curso_sin2.blogia.com/2005/060401-herramientas-case-rational-rose.-por-maria-de-nobre.

Eduardo Mora Monte, VV Staff,Marta E. Zorrilla Pantaleón. 1999. Iniciación a las Bases de Datos con Access 2002. 1999.

Escofet, Carme Martín. El lenguaje SQL. P06/M2109/02149.

Escuela Politécnica Superior. 2009. Patrones de Diseño. 2009.

Gamma, Erich. 1995. Design Patterns: Elements of Reusable Object-Oriented Software. 1995.

García, Javier. 1998. Aprenda C++ como si estuviera en primero. España : s.n., 1998.

Gimson, Lic. Loraine. 2012. Metodologías ágiles y desarrollo basado en conocimientos. 2012.

Godet, Michel. 2007. Caja de herramientas . 2007.

guia-ubuntu. 2008. guia-ubuntu. [En línea] 2008. http://www.guia-ubuntu.com/index.php?title=PgAdmin_III.

Ibaceta, José Bengoechea. 2012. Microsoft Access : diseño de aplicaciones sencillas de bases de datos. 2012.

Isidro Ramos Salavert, María Dolores Lozano Pérez. 2000. Ingeniería Del Software Y Bases de Datos: Tendencias Actuales. 2000. pág. 62.

Ivar Jacobson, Grady Booch y James Rumbaugh. 2000. El Proceso Unificado de Desarrollo de Software. Madrid : Addison Wesley, 2000.

J.J. Gutierrez, M.J. Escalona, M.Mejias, J.Torres. 2012. Sistema de Programacion Extrema. [En línea] 2012. http://www.lsi.us.es/~javierj/investigacion_ficheros/PSISEXTREMA.pdf.

Jacobson, Ivar, Rumbaugh, James y Booch, Grady. 2000. El proceso unificado de desarrollo: Addison Wesley Capítulso 9. 2000.

Jacobson, P.Jonsson, M. Christerson and G.Overgaard. 1992. Ingeniería de Software orientada a Objetos- Un acercamiento a través de lod casos de uso. 1992.

Javier García de Jalón. 2000. Aprenda Java como si estuviera en primero,Ciudad de la Habana. [En línea] 2000. http://www.tecnun.es/asignaturas/Informat1/AyudaInf/aprendainf/Java/Java2.pdf.

Kamilo Cervantes. 2010. Scribd. [En línea] 2010. http://es.scribd.com/doc/45240510/Servidor-de-PostGIS.

Lanzillota, Analia. 2007. [En línea] 2007. [Citado el: 16 de 11 de 2010.] http://www.mastermagazine.info/termino/6333.php.

Larman, Craig. 1999.UML y Patrones Introducción al análisis y diseño orientado a objetos. México : PRENTICE HALL, 1999.

Lobaina, Arístides Alejandro Legrá. 1999. Metodología para el pronóstico, planificación y control integral de la minería en yacimientos lateríticos. 1999.

Martínez, Rafael. 2010. Sobre PostgrSQL. [En línea] 2 de Octubre de 2010. [Citado el: 6 de Febrero de 2013.] http://www.postgresql.org.es/sobre_postgresql.

Martino, E. A Bertino y. 1995. Sistemas de bases de datos orientadas a objetos. s.l. : Ediciones Díaz de Santos. 1995.

Marvin, David. 2008. Definición de lenguaje de programación. Tipos. Ejemplos. [En línea] 16 de Octubre de 2008. [Citado el: 11 de Diciembre de 2012.] http://catedraprogramacion.foroactivos.net/t83-definicion-de-lenguaje-de-programacion-tipos-.

Mireles, Gabriel Alberto García. 2004. Introducción a los casos de uso. Sonora : s.n., 2004.

Murillo, Félix. 1999. Herramientas CASE. 1999.

Parsons, June Jamrich. 2008. Conceptos de Computación: Nuevas Perspectivas. decima edicion. 2008.

Pérez, Liset. 2008. Estartegia para el desarrollo de requisitos funcionales de software en proyectos productivos de la UCI. Ciudad de la Habana : s.n., 2008.

pgAdmin. 2009. pgAdmin PostgreSQL tools. [En línea] 26 de Marzo de 2009. [Citado el: 7 de Febrero de 2013.] http://www.pgadmin.org/licence.php.

Postgresql. 2010. postgresql. [En línea] 2 de Octubre de 2010. [Citado el: 24 de mayo de 2013.] http://www.postgresql.org.es/sobre_postgresql.

Pressman, Roger. 2008. Capítulo 8: Modelado del Análisis. Ingeniería de Software. New York : s.n., 2008.

Pressman, Roger S. 2005. Ingeniería del Software. 2005.

Pressman, Roger. 2010. Software Engineering: A Practitioner´s Approach. New York : s.n., 2010.

Reynoso, Carlos. 2004. Introducción a la arquitectura de software. Buenos Aires : s.n., 2004.

Rieder, Serge. 2013. Dbeaver Universal Database Manager. [En línea] 2013. [Citado el: 18 de 3 de 2013.] http://dbeaver.jkiss.org/.

Roberth G. Figueroa, Camilo J. Solís, Armando A. Cabrera. Metodologias Tradicionales Vs Metodologias Agiles. Valencia : Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación.

Rosas, Gonzalo. 2011. Modelado del Negocio con UML. 2011.

Rumbaugh, James. 2004. El proceso unificado de desarrollo de software. La Habana : Félix Varela, 2004.

Software, Arquitectura de. 2009. Arquitectura de Software. [En línea] 2009.

Software, Colectivo de Autores Asignatura Ingeniería de. 2011. Entorno Virtual de Aprendizaje. [En línea] 2011. [Citado el: 15 de 2 de 2011.] http://eva.uci.cu/mod/resource/view.php?id=35381.

Software, Laboratorio Nacional de Calidad del. 2009. Ingeniería del Software: Metodologías y ciclos de vida. 2009.

Unam. 2011. Unam. [En línea] 2011. [En línea] 18 de agosto de 2011. [Citado el: 9 de enero de 2013.] http://www.unam.edu.pe/webminas/index.php?option=com_content&view=article&id=81:softwares-mineros..

UNAM, Universidad Nacional de Moquegua. 2013. Universidad Nacional de Moquegua Universitas Universitatis. [En línea] 2013. [Citado el: 5 de 3 de 2013.] http://www.unam.edu.pe/webminas/index.php?option=com_content&view=article&id=81:softwares-mineros..

Valdez, Alfonso Altamirano. 2006. Comparativos de Entornos de Desarrollo Integrados. 2006.

Yailen Rondón Calzadilla, Yaelsy Tamayo Echemendia. 2010. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas: Estrategia de pruebas de software para el proyecto Fuerza de Trabajo Calificada. 2010.