



Universidad de las Ciencias Informáticas

Facultad 6

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

**TÍTULO: Herramienta para la gestión de la información de los pozos
de sondeo**

Autor

Yanier García Ferié

Tutor

Ing: José Carlos Santiesteban Rojas

Ciudad de la Habana, 23 de Junio de 2011

Año 53 de la Revolución

Declaración de Autoría

Declaro que yo Yanier García Ferié soy el único autor de este trabajo, y autorizo a la Universidad de las Ciencias Informáticas (UCI) y a la Facultad (6) para que hagan uso que estimen pertinente con este trabajo.

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

Yanier García Ferié
Autor

Ing. José Carlos Santiesteban Rojas.
Tutor

Datos de Contacto.

TUTOR: Ing. José Carlos Santiesteban Rojas.

Graduado en la Universidad de las Ciencias Informáticas en el año 2008, con 3 años de trabajo en esta institución. Instructor, perteneciente al Departamento de Técnicas de Programación, se desempeña como jefe de asignatura de Programación 4 y Programación 5. Presta servicios a la producción en el centro de desarrollo GEySED, como programador en el proyecto de Minería. Posee una experiencia de 5 años en la producción.

Dedicatoria

A mi madre, la persona más importante de mi vida, su constante abnegación y apoyo incondicional la hacen merecedora de todas mis victorias. A mis dos padres: José y Uber, por brindarme su apoyo y consejos cuando los he necesitado. Al comandante en jefe Por haber luchado incansablemente para que cada cubano tuviera la oportunidad de realizar sus sueños. A mis hermanos Yeni y Alejandro, que aunque vivamos peleando no podemos estar uno sin los otros. A mis tíos Jabao, Keyla, Melida y Luisa por brindarme su apoyo incondicional, a Yamila que me apoyó en todo momento. A todos mis amigos, que confiaron en mí y me dieron el apoyo moral que necesita una persona en momentos difíciles. A Máximo y a Chavela por sus consejos y por darme la oportunidad de ser parte de su familia. A mis vecinos que siempre se preocuparon por mí. Por último, le dedico este trabajo también a las personas que en algún momento quisieron hacerme daño, eso me hizo entender que no todo es color de rosa y que hay que ser fuerte y mirar siempre hacia adelante.

Agradecimientos:

Son muchas las personas que tuvieron que ver en este logro, pero la primera y más importante es mi mamá, ella que siempre estuvo cada vez que la necesité, por la educación que me dio, por su sacrificio incondicional.

Le agradezco grandemente a mi papá José por impulsarme siempre a estudiar y ser mejor cada día, por darme el ejemplo de ser un hombre por encima de todo y por sacrificarse cada día por la familia.

A mi papá Ulber por ser mi guía a seguir, por ayudarme y aconsejarme cuando lo necesité, por enseñarme que las cosas para lograrlas hay que lucharlas porque nadie las regala.

A mi hermana Yeni por ser la primera de los tres en tener un título e inspirarme a seguir adelante y a Ale porque aunque sea el menor me ha enseñado muchas cosas.

Le agradezco a los hermanos Pupo por soportarme cada día en su laboratorio preguntándole dudas, a Armando, a Alexaidis, a Noly, a todas las personas que me han aconsejado y que siempre creyeron en mí.

Le agradezco a mi tutor José Carlos por el profesor, tutor y amigo que es. Le agradezco a Eddy por haberme puesto de tutor a José Carlos, a Dianet por su ayuda incondicional.

Le agradezco a mi tía Melida, a Jabao y a Keyla por ser como padres para mí en estos 5 años. A Yamila por ser la gran persona que es y por permitirme contar con ella cada vez que la necesité.

Por último y no menos importante, le agradezco a Yudirenia, la persona que compartió conmigo todo en estos últimos 16 meses, la persona que aguantó mis crisis de malcriadeces, mis momentos de tristeza. Te agradezco por estar ahí para mí cuando me hacía falta hablar con alguien, cuando me hacía falta un beso y un abrazo.

A todas estas personas desde el fondo de mi corazón:

Muchísimas gracias.

Resumen

Hoy día una herramienta informática para gestionar cualquier tipo de información es muy importante debido al tiempo que se ahorra en la manipulación de esta, y convierte lo que podría ser una jornada de trabajo extenuante en un simple proceso.

El siguiente trabajo describe el desarrollo de una herramienta para la gestión de la información de pozos de sondeo, la cual contiene la implementación de una serie de funcionalidades que le permiten al usuario una más rápida manipulación de esta información.

La herramienta está desarrollada sobre tecnología libre lo cual es muy importante para el país, puesto que contará con un sistema propio y de código abierto para ajustarlo a necesidades futuras. Además evita el pago de licencias de herramientas propietarias.

Se documentó junto con la herramienta todo el proceso de desarrollo de la misma, donde se realiza una descripción de los elementos teóricos y de ingeniería utilizados. Esto servirá de apoyo a nuevos desarrolladores que necesiten hacerle modificaciones y como base para su posterior utilización.

Palabras claves: sondeo, software libre, ficheros, gestión, reporte.

Tablas:

Tabla 1: Actores del Sistema.	32
Tabla 2: Descripción del Caso de Uso Nuevo Proyecto.	35
Tabla 3: Descripción del Caso de Uso Cargar Proyecto.....	37
Tabla 4: Descripción del Caso de Uso Salvar Proyecto.	38
Tabla 5: Descripción del Caso de Uso Importar Ficheros.....	39
Tabla 6: Descripción del Caso de Uso Mostrar Tablas.....	41
Tabla 7: Descripción del Caso de Uso Gestionar Información.....	44
Tabla 8: Descripción del Caso de Uso Buscar Pozo.	45
Tabla 9: Descripción del Caso de Uso Generar Reporte.....	46

Ilustraciones:

Ilustración 1: Modelo de Dominio.....	29
Ilustración 2: Diagrama de Casos de Uso del Sistema.....	33
Ilustración 3: Esquema básico de la Arquitectura Tres Capas.....	49
Ilustración 4: Diagrama de clases persistentes.	51
Ilustración 5: Modelo de Datos.....	52
Ilustración 6: Diagrama de Clase de Análisis del Caso de Uso Nuevo Proyecto.	53
Ilustración 7: Diagrama de Clases de Diseño del Caso de Uso Nuevo Proyecto.	54
Ilustración 8: Diagrama de colaboración nuevo proyecto	55
Ilustración 9: Diagrama de despliegue.	56
Ilustración 10: Diagrama de Componentes.	57
Ilustración 11: Diagrama de Clase de Análisis del Caso de Uso Cargar Proyecto. ¡Error! Marcador no definido.	
Ilustración 12: Diagrama de Clase de Análisis del Caso de Uso Salvar Proyecto. ¡Error! Marcador no definido.	
Ilustración 13: Diagrama de Clase de Análisis del Caso de Uso Importar Ficheros. ¡Error! Marcador no definido.	
Ilustración 14: Diagrama de Clase de Análisis del Caso de Uso Mostrar Ficheros. ¡Error! Marcador no definido.	
Ilustración 15: Diagrama de Clase de Análisis del Caso de Uso Gestionar Información..... ¡Error! Marcador no definido.	
Ilustración 16: Diagrama de Clase de Análisis del Caso de Uso Buscar Pozos. ¡Error! Marcador no definido.	
Ilustración 17: Diagrama de Clases de Diseño del Caso de Uso Buscar Pozo. ¡Error! Marcador no definido.	

Ilustración 18: Diagrama de Clases de Diseño del Caso de Uso Generar Reporte. **¡Error! Marcador no definido.**

Ilustración 19: Diagrama de Clases de Diseño del Caso de Uso Gestionar Información. **¡Error! Marcador no definido.**

Ilustración 20: Diagrama de Clases de Diseño del Caso de Uso Importar Fichero. **¡Error! Marcador no definido.**

Ilustración 21: Diagrama de Clases de Diseño del Caso de Uso Importar Proyecto. **¡Error! Marcador no definido.**

Ilustración 22: Diagrama de Clases de Diseño del Caso de Uso Mostrar Ficheros. **¡Error! Marcador no definido.**

Ilustración 23: Diagrama de Clases de Diseño del Caso de Uso Salvar Proyecto. **¡Error! Marcador no definido.**

Ilustración 24: Diagrama de Colaboración del Caso de Uso Importar Proyecto. **¡Error! Marcador no definido.**

Ilustración 25: Diagrama de Colaboración del Caso de Uso Salvar Proyecto. **¡Error! Marcador no definido.**

Ilustración 26: Diagrama de Colaboración del Caso de Uso Cargar Fichero. **¡Error! Marcador no definido.**

Ilustración 27: Diagrama de Colaboración del Caso de Uso Mostrar Fichero. **¡Error! Marcador no definido.**

Ilustración 28: Diagrama de Colaboración de la sección Adicionar del Caso de Uso Gestionar Información. **¡Error! Marcador no definido.**

Ilustración 29: Diagrama de Colaboración de la sección Eliminar del Caso de Uso Gestionar Información. **¡Error! Marcador no definido.**

Ilustración 30: Diagrama de Colaboración de la sección Actualizar del Caso de Uso Gestionar Información. **¡Error! Marcador no definido.**

Ilustración 31: Diagrama de Colaboración del Caso de Uso Modelar Pozos. **¡Error! Marcador no definido.**

Ilustración 32: Diagrama de Colaboración del Caso de Uso Buscar Pozos.¡Error! Marcador no definido.

Ilustración 33: Diagrama de Colaboración del Caso de Uso Generar Reportes.¡Error! Marcador no definido.

Índice

Introducción	1
Capítulo 1: Fundamentación Teórica	5
1.1. Introducción	5
1.2. Conceptos asociados al dominio del problema.....	5
1.2.1. Minería	5
1.2.2. Mina.....	5
1.2.3. Minerales.....	6

1.2.4.	Pozos de sondeo	6
1.2.5.	Litología	6
1.2.6.	Ficheros.....	6
1.2.7.	Reportes	7
1.3.	Objeto de estudio	8
1.3.1.	Descripción general	8
1.3.1.2.	Información	9
1.3.1.3.	Gestión de la información	9
1.3.2.	Descripción actual del dominio del problema.....	10
1.3.3.	Situación Problemática.....	11
1.4.	Análisis de soluciones existentes.....	12
1.5.	Conclusiones.....	17
Capítulo 2: Tendencias y Tecnologías actuales a desarrollar.		18
2.1.	Introducción	18
2.2.	Metodología de Desarrollo de Software	18
2.2.1.	Rational Unified Process (RUP).....	18
2.3.	Lenguaje de Modelado: UML	20
2.4.	Herramienta CASE	21
2.4.1.	Visual Paradigm	21
2.5.	Leguaje de Programación	23
2.5.1.	C++	23
2.6.	Entorno de Desarrollo Integrado (IDE).....	24
2.6.1.	Qt Creator	24
2.7.	Sistema gestor de Bases de datos	26
2.7.1.	SQLite.....	26

2.8.	Conclusiones.....	27
Capítulo 3: Presentación de la solución propuesta.....		28
3.1.	Introducción	28
3.2.	Modelo de Dominio.....	28
3.2.1.	Conceptos principales del Modelo de Dominio	28
3.2.2.	Representación del Modelo de Dominio.....	29
3.3.	Especificación de los Requerimientos	29
3.3.1.	Requerimientos Funcionales (RF).....	29
3.3.2.	Requerimientos no Funcionales	30
3.4.	Descripción del Sistema.....	31
3.4.1.	Actores del Sistema	31
3.4.2.	Diagrama de Casos de Uso del Sistema.....	32
3.5.	Descripción de los Casos de Uso del Sistema.....	33
3.6.	Conclusiones.....	46
4.1.	Introducción	47
4.2.	Patrones de Diseño.....	47
4.3.	Programación por capas.....	48
4.4.	Estándares de interfaz de la aplicación.....	50
4.5.	Diseño de la Base de Datos (BD).....	50
4.6.	Diagrama de Clases del Análisis.....	52
4.7.	Diagrama de Clases del Diseño.....	53
4.8.	Diagrama de Colaboración	54
4.9.	Diagrama de despliegue	55
4.10.	Modelo de Implementación	56
4.10.1.	Diagrama de Componentes	56

4.11 Pruebas a la herramienta propuesta.....	57
4.11.1 Pruebas de caja negra:	57
4.11.2 Prueba de Caja Negra aplicada al requisito funcional Adicionar pozo.....	59
4.12 Conclusiones.....	65
Conclusiones Generales.....	66
Recomendaciones	67
Referencias Bibliográficas.....	68
Bibliografía Consultada.....	70
Anexos	¡Error! Marcador no definido.
Glosario.....	72

Introducción

Introducción

El nacimiento de la minería como actividad relativa a cualquier tipo de aprovechamiento de los recursos minerales del suelo, se remonta al propio origen del género humano. Desde entonces el hombre ha trabajado en este campo, convirtiéndolo así en el renglón económico principal de algunas naciones.

Los métodos poco desarrollados hicieron que este buscara la forma de hacer menos forzoso y más eficiente su trabajo. Con el desarrollo de las Tecnologías de la Investigación y las Comunicaciones (TIC) nació también una era de facilidades de trabajo para los obreros de la industria minera.

Diversas alternativas ofrece la industria del software minero disponible en el mercado nacional e internacional, desde soluciones orientadas a aplicaciones específicas hasta productos que brindan servicios más integrales, que abarcan y controlan diversas áreas de este negocio. Todo ello en pos de una mayor eficiencia, donde el aprovechamiento de las capacidades y los ahorros tienen un peso importante en el éxito y viabilidad de los proyectos, en un negocio cada vez más competitivo como lo es la minería. De ahí la relevancia de los productos orientados a una mejor gestión del proceso minero, a fin de contar en forma clara y amigable con los datos adecuados para la toma de decisiones.

Como consecuencia de un esfuerzo especial del Viceministerio de Geología se introduce la computación en la actividad geológica en Cuba en 1983, tomando fuerzas en la década de los años 90. Ya en la actualidad desarrolladores cubanos en conjunto con especialistas geológicos han creado herramientas para el trabajo con información geológica.

Las herramientas informáticas utilizadas por las instituciones cubanas son mayormente extranjeras y propietarias, entre ellas se encuentran GEMCOM, SURPAC, DATAMINE, MICROLIN. Esto ha contribuido al gasto de grandes sumas de dinero anualmente por parte del gobierno cubano. También se utilizan herramientas desarrolladas en el país como lo son MICRONIQ, CORTE, TIERRA y SIM, pero las mismas no satisfacen las demandas de los especialistas en esta área.

Debido a la situación existente, el país se ha trazado la meta de producir las herramientas necesarias para las estimaciones y cálculos que realizan los ingenieros mineros, al mismo nivel en el que se encuentran hoy las utilizadas en el ámbito internacional. Esta medida permitirá el máximo aprovechamiento de las minas explotadas.

La Universidad de las Ciencias Informáticas (UCI) forma parte de este plan llevado a cabo por el país. Esta universidad en los últimos dos años, impulsada por los proyectos de informatización del Banco de Datos Geológico de Cuba bajo el Programa Nacional de Informatización del Conocimiento Geológico

Introducción

(PNICG), ha madurado un departamento de Geoinformática en la facultad 6 con un amplio potencial en el desarrollo de software de las especialidades de la Geociencia.

Hay una amplia variedad de minerales en la tierra que son presentados por la naturaleza en lugares conocidos como yacimientos, que se encuentran a diferentes niveles de profundidad. Estos antes de ser explotados por el hombre deben ser estudiados y analizados para evitar riesgos que puedan causar tanto pérdidas económicas como humanas.

Existe un método muy utilizado en la minería para el estudio de estos yacimientos llamado pozos de sondeo, que no son más que pequeños hoyos en el suelo utilizados para la toma de muestras de los minerales existentes que facilitan determinar, en caso de haber estado en uso, la época en que lo estuvo y otras características que permiten establecer una excavación arqueológica de mayor envergadura.

De estos pozos de sondeo se almacena una amplia gama de información que suele ser utilizada en diversas investigaciones y tratamientos para proyectos de gran envergadura. Existen lugares en los que la gestión de dicha información se genera de forma manual y otros de forma automática.

En los casos en los que la gestión se realiza manualmente, este proceso llega a convertirse en una actividad engorrosa y compleja debido a que los datos con los cuales se trabaja son principalmente números reales y resulta trabajoso leerlos y generar reportes.

En los casos en los que se realiza de forma automática, las entidades cubanas aparte de trabajar con software propietario de los cuales la licencia resulta demasiado costosa, ya que se pagan miles de dólares para una sola computadora, también trabajan con herramientas de producción nacional, pero estas son desarrolladas a la medida, lo cual implica que solo satisfacen las necesidades de clientes específicos.

Mediante el análisis de la situación planteada anteriormente surge como **problema a resolver**: la necesidad de una herramienta que permita la gestión de la información obtenida de los pozos de sondeo sin la inversión de grandes sumas de dinero.

A raíz del problema a resolver se define como **objeto de estudio**: los procesos de gestión de la información y como **campo de acción**: los procesos de gestión de la información de los pozos de sondeo.

Introducción

La **idea a defender** plantea que: El desarrollo de una herramienta informática utilizando tecnologías libres permitirá al país gestionar eficientemente la información generada por los pozos de sondeo sin necesidad de invertir grandes sumas de dinero.

Para dar solución al problema a resolver se traza como **objetivo general**: desarrollar una herramienta para la gestión de la información de los pozos de sondeo.

Con el propósito de desarrollar el presente trabajo y dar cumplimiento al objetivo definido anteriormente se trazan las siguientes **tareas de la investigación científica**:

1. Analizar los registros de pozos de sondeo.
2. Describir el estado del arte de las herramientas para la gestión de información geológica.
3. Determinar las tecnologías y herramientas a utilizar para el desarrollo de la herramienta.
4. Realizar el análisis de la herramienta para la gestión de la información.
5. Realizar el diseño de la herramienta para la gestión de la información.
6. Implementar el prototipo funcional de la herramienta para la gestión de la información de pozos de sondeo.
7. Validar la herramienta desarrollada.

Para una correcta comprensión y realización del trabajo se hace uso de los siguientes métodos científicos:

Métodos Teóricos:

- ✓ **Histórico-Lógico:** Se empleó para el estudio de antecedentes que existen respecto a herramientas informáticas que cumplan con los requerimientos a tratar.
- ✓ **Analítico-Sintético:** Posibilitó una mayor comprensión a partir del análisis de las distintas fuentes bibliográficas consultadas, de las características que se derivan de la evaluación de la arquitectura que se le realice a todo producto de software geológico.
- ✓ **Modelación:** Permitió crear un modelo (abstracción) que explique el resultado final de este trabajo.

Introducción

Métodos Empíricos

- ✓ Entrevista: Se utilizó para realizar entrevistas a personas especializadas en el tema de la minería con el objetivo de profundizar en el tema de los pozos de sondeo.
- ✓ Observación: Se empleó para entender el funcionamiento de los procesos de gestión de la información de pozos de sondeo a través de la observación de las herramientas existentes.

El trabajo de Diploma ha sido conformado de la siguiente manera:

Capítulo 1_Fundamentación Teórica: en este capítulo se abordan los conceptos referentes al dominio del problema. También se argumenta sobre el estado actual de las herramientas existentes utilizadas en la minería para la gestión de la información de pozos de sondeo. Además se realiza una descripción acerca del problema a resolver.

Capítulo 2_Tendencias y Tecnologías actuales a desarrollar: se describe lo correspondiente a las tecnologías que se utilizan para el desarrollo de herramientas informáticas, haciendo énfasis en las herramientas para la gestión de información. Se explica el por qué de la decisión de utilizar RUP como metodología de desarrollo, UML como lenguaje de modelado, Visual Paradigm como herramienta CASE, C++ como lenguaje de programación y Qt Creator como entorno de desarrollo.

Capítulo 3_Presentación de la Solución Propuesta: se presenta el Modelo de Dominio, los requerimientos funcionales y no funcionales, y los actores y casos de uso del sistema, así como la interacción entre ellos.

Capítulo 4_Construcción de la solución propuesta: se aplica la solución propuesta anteriormente. Se definen los patrones de diseño a utilizar y cada uno de los diagramas de componentes, así como los modelos de diseño e implementación. También se describen las pruebas realizadas al sistema para comprobar que funciona correctamente.

Capítulo 1: Fundamentación Teórica

Capítulo 1: Fundamentación Teórica

1.1. Introducción

Actualmente hablar de gestión de información y no mencionar una herramienta informática que realice esta operación es sinónimo de atraso tanto laboral como tecnológico. Esto se debe al creciente auge que ha tomado la informática en la actualidad y que la mayoría de las instituciones manejan un nivel de información bastante amplio como para ser manipulado manualmente. Además, mantener la información en formato duro es correr el riesgo de que se dañe o se pierda.

En el presente capítulo se abordan los conceptos referentes al dominio del problema, los cuales son imprescindibles para el entendimiento de este trabajo. También se aborda el estado actual de las herramientas existentes utilizadas en la minería que permiten gestionar la información de pozos de sondeo. Además se realiza una descripción detallada acerca del problema existente al cual se le desea dar solución.

1.2. Conceptos asociados al dominio del problema

1.2.1. Minería

La minería es la obtención selectiva de los minerales y otros materiales de la corteza terrestre. También se denomina así a la actividad económica primaria relacionada con la extracción de elementos de los cuales se puede obtener un beneficio económico. Dependiendo del tipo de material a extraer la minería se divide en metálica y no metálica. (Florencia, 2009)

A través de la minería se puede, por un lado, obtener de manera selectiva minerales y algunos otros materiales desde la mismísima corteza terrestre y por otro lado, es también una actividad económica que despliegan muchos países del mundo a través de la cual extraen valiosos elementos de la tierra, como pueden ser el aluminio, cobre, hierro, plomo, oro, entre otros, con el objetivo de obtener una ventaja económica a partir de su comercialización.

1.2.2. Mina

Una mina es una obra resultante del conjunto de excavaciones mineras e instalaciones superficiales y subterráneas que se realizan para la investigación y la explotación de un yacimiento mineral.

También se puede decir que es un lugar subterráneo, generalmente ubicado en zonas montañosas, donde se logra hallar mediante excavaciones, a cielo abierto o por construcción de galerías, riquezas mineras. (Fingermann, 2009)

Capítulo 1: Fundamentación Teórica

Las minas son aquellos lugares subterráneos, generalmente ubicados a instancias de zonas montañosas, en los que se hallan principalmente materiales muy valiosos que pueden ser el origen de una cuantiosa riqueza.

1.2.3. Minerales

Los minerales son cuerpos naturales químicos, inorgánicos y homogéneos, que pueden hallarse en la superficie de la Tierra, formados en ella de forma espontánea. En general poseen estructura cristalina, dada por el enrejamiento de los átomos, y la composición química difiere de unos a otros. Poseen caras, o sea superficies planas, pues sus átomos se disponen en forma ordenada. Hay pocos minerales integrados por un solo elemento, como el oro y la plata; la mayoría, son compuestos químicos. (Fingermann, 2009)

1.2.4. Pozos de sondeo

Son pozos cuadrados, habitualmente de medio o un metro de lado, que se hacen en forma rápida para determinar la profundidad de la estratigrafía del terreno, la época que estuvo en uso y otras características que permiten establecer una excavación arqueológica de mayor envergadura.

1.2.5. Litología

La Litología es la parte de la Geología que trata de las rocas: el tamaño de grano, sus partículas y características físicas y químicas. La Litología es fundamental para entender cómo es el relieve, ya que dependiendo de la naturaleza de las rocas se comportarán de una manera concreta ante los empujes tectónicos, los agentes de erosión y transporte, y los diferentes climas de la Tierra. (Pastrana, 2007)

1.2.6. Ficheros

Los ficheros o archivos son unidades lógicas de almacenamiento que define el propio sistema operativo y cuyo significado está definido por su creador. Los ficheros están constituidos a nivel interno por un conjunto de registros lógicos. (Rivera, 2008)

Los ficheros almacenan dentro de ellos sus propias características: la fecha de su creación, su última modificación, sus atributos (solo lectura, etc.), su tamaño, etc. Existen dos tipos de ficheros: de texto y binario.

Capítulo 1: Fundamentación Teórica

1.2.6.1. Ficheros de texto

Los ficheros de texto son la manera que se utiliza en programación (informática) para salvar los datos (variables y valores) procesados por los programas y que estos, los datos, no tengan que ser introducidos constantemente para su posterior re-procesamiento por el programa.

Estos solo contienen ASCII estándar, denominado "Plain text" o texto plano (sin formato) en la jerga informática. No presentan problemas de presentación, aunque puede haberlo con algunos caracteres especiales si se trata de ASCII en algún dialecto de idioma distinto al del género humano. Los más notables son los ficheros read.me, lea.me, etc. Todos ellos pueden ser abiertos con un procesador de textos normal (por ejemplo el WordPad del propio Windows). (Rivera, 2008)

En los ficheros de texto la información se guarda como caracteres codificados en ASCII u otras codificaciones de texto, lo que los hace eficaces a la hora de guardar textos y números pequeños; al contrario de los ficheros binarios, que son útiles para guardar números medios y grandes. Estos archivos están compuestos de bytes que representan caracteres ordinarios como letras, números y puntuación (incluyendo espacios), también incluye algunos caracteres de control como tabulaciones y separaciones de líneas. Esta simplicidad permite que una gran variedad de programas pueda leer y editar ese contenido.

1.2.7. Reportes

Un reporte es un informe o una noticia. Este tipo de documento (que puede ser impreso, digital, audiovisual) pretende transmitir una información, aunque puede tener diversos objetivos. Existen reportes divulgativos, persuasivos y de otros tipos.

El reporte puede ser la conclusión de una investigación previa o adoptar una estructura de problema-solución en base a una serie de preguntas. En el caso de los informes impresos, el texto suele ir acompañado por gráficos, diagramas, tablas de contenido y notas al pie de página.

En el ámbito de la informática, los reportes son informes que organizan y exhiben la información contenida en una base de datos. Su función es aplicar un formato determinado a los datos para mostrarlos por medio de un diseño atractivo y que sea fácil de interpretar por los usuarios.

El reporte, de esta forma, confiere una mayor utilidad a los datos. No es lo mismo trabajar con una planilla de cálculos con 10.000 campos que con un dibujo en forma de torta que presenta dichos

Capítulo 1: Fundamentación Teórica

campos de manera gráfica. Los reportes tienen diversos niveles de complejidad, desde una lista o enumeración hasta gráficos mucho más desarrollados.

Según el programa informático, los reportes permiten la creación de etiquetas y la elaboración de facturas, entre otras tareas. (Florencia, 2009)

1.3. Objeto de estudio

1.3.1. Descripción general

La información es un agente importante en la modificación de las conductas existentes en la organización y desarrollo, y con el decursar de los años su gestión ha ocupado, cada vez más, un espacio mayor en la economía de los países a escala mundial.

La correcta gestión de la información es una herramienta fundamental para la toma de decisiones, la formación del personal, la evaluación de los productos, la determinación de los errores y el control de los procesos. El carácter intangible de la información ha hecho que muchos directivos de organizaciones no inviertan los recursos suficientes para las actividades de información. (Quiroga, 2008)

Una adecuada gestión de la información, en el contexto de una gerencia de la calidad, posibilita reducir los riesgos en la administración de la organización, como son la toma de decisiones apresuradas, tardías o inconsistentes, la entrada al mercado con productos no competitivos, que ocasionan pérdidas y reducen su competitividad. Obtener la información necesaria, con la calidad requerida, es una premisa indispensable para la supervivencia de las empresas, si se considera que las organizaciones acortan cada vez más sus ciclos estratégicos.

Es obvio que una eficiente gestión de la información, como parte de la aplicación de una política de gestión de la calidad, garantiza no sólo que las instituciones de información presten servicios eficientes, sino que la organización obtenga mayores ganancias y una mejor competitividad en el mercado.

Para el mayor entendimiento de lo explicado anteriormente se definen los siguientes conceptos:

1.3.1.1. Gestión

Capítulo 1: Fundamentación Teórica

Se dice que gestión es interactuar en todas las áreas de una empresa, organizaciones, unidades e instituciones informativas. Se entiende además a toda “actividad dirigida a obtener y asignar los recursos necesarios para el cumplimiento de los objetivos de la organización”. (Faga, 2000)

Gestión hace referencia a la acción y al efecto de gestionar o de administrar. Gestionar es realizar diligencias conducentes al logro de un negocio o de un deseo cualquiera.

El término gestión, por lo tanto, implica al conjunto de trámites que se llevan a cabo para resolver un asunto o concretar un proyecto. La gestión es también la dirección o administración de una empresa o de un negocio. Permite adicionar, eliminar y modificar información de cualquier sistema de archivos, ya sea en formato duro o digital.

1.3.1.2. Información

Se entiende por información a la agrupación de datos organizados y presentados en un contexto. También como proceso y fenómeno de manifestaciones objetivas y subjetivas, como accidente del objeto, esta relevancia se deberá a la percepción y tratamiento que le da el ser humano a la misma. La información existe en tres órdenes según Brent: como presentación, estímulos; apropiación interna del individuo y construcción negociada y validada. (Bases Teóricas y filosóficas de la Bibliografía, 1998)

Según Thompson la información es un conjunto de datos acerca de algún suceso, hecho o fenómeno, que organizados en un contexto determinado tienen su significado, cuyo propósito puede ser el de reducir la incertidumbre o incrementar el conocimiento acerca de algo.

Desde el punto de vista de la ciencia de la computación, la información es un conocimiento explícito extraído por seres vivos o sistemas expertos como resultado de la interacción con el entorno o percepciones sensibles del mismo entorno. En principio la información, a diferencia de los datos o las percepciones sensibles, tiene estructura útil que modificará las sucesivas interacciones del ente que posee dicha información con su entorno.

1.3.1.3. Gestión de la información

La gestión de la información se puede definir como el conjunto de actividades realizadas con el fin de controlar, almacenar y, posteriormente, recuperar adecuadamente la información producida, recibida o retenida por cualquier organización en el desarrollo de sus actividades. En el centro de la gestión de la

Capítulo 1: Fundamentación Teórica

información se encuentra la gestión de la documentación (la información que queda plasmada en documentos) y que puede ser de tres tipos:

- **Interna:** hace referencia a aquella documentación generada o recibida por la organización en el ejercicio de sus funciones, es decir, son documentos que surgen de la actividad diaria de esa institución. En este grupo, se encuentra tanto la documentación típicamente administrativa: la contabilidad y correspondencia como la documentación de gestión: informes, actas de reuniones, procedimientos de trabajo, y la documentación técnica que refleja la propia actividad de la organización.
- **Externa:** además de la documentación producida por la propia organización, esta y las personas que trabajan en ella necesitan, a menudo, consultar y manejar fuentes de información externas: libros, revistas, Base de Datos, Internet.
- **Pública:** es aquella documentación que la organización produce de cara al público, para comunicarles hechos, actividades, acontecimientos, por ejemplo: las memorias, los catálogos de productos y servicios, las páginas web.

La gestión de la información es imprescindible, pero sólo se convierte en conocimiento cuando los individuos la aplican para la resolución de un problema. (GESTIÓN DEL CONOCIMIENTO Y GESTIÓN DE LA INFORMACIÓN, 2001)

La gestión de la información es importante para:

- Tomar las decisiones necesarias para mejorar la gestión de prestaciones y servicios.
- Poner en práctica la planificación, implementación, supervisión y evaluación participativas.

1.3.2. Descripción actual del dominio del problema.

La sociedad cubana se encuentra actualmente en un proceso de informatización el cual es llevado a todas las áreas del país, ya sea con fines económicos, sociales o culturales. La Oficina Nacional de Recursos Minerales (ONRM) no se queda atrás en esta revolución informática encontrándose dentro del Programa Nacional de Informatización del Conocimiento Geológico (PNICG), con el principal objetivo de implementar herramientas de calidad, apropiadamente protegidas y certificadas que se rijan por estándares informáticos y normativas de la Geociencia para la conservación y administración del conocimiento geológico.

Capítulo 1: Fundamentación Teórica

Entre las diferentes ramas de la economía que aportan grandes cantidades de dinero al país se encuentra la minería, la cual está situada actualmente entre los elementos importantes que sustentan el desarrollo económico cubano. Para lograr un mayor rendimiento de esta importante rama es importante contar con las herramientas necesarias para la toma de decisiones antes de realizar excavaciones importantes.

Con el creciente auge que ha tomado la minería en la actualidad, la cantidad de información generada por este campo es muy difícil de manipular por el hombre de forma manual. Cuando las informaciones obtenidas son pequeñas son de fácil entendimiento y manipulación, pero cuando son grandes cantidades de datos como las que generan los pozos de sondeo entonces es necesario la utilización de una herramienta informática para manipularlas.

De los pozos de sondeo se derivan una serie de informaciones, las cuales se utilizan posteriormente en la toma de decisiones para excavaciones más profundas, ejemplo de ello: la excavación de minas y pozos (ya sea de agua o petróleo). Dentro de estas informaciones se encuentran como primarias las siguientes:

- **Pozo** (coordenadas x, y, z de los collares de los sondajes).
- **Muestra** (azimut e inclinación de cada muestra del sondaje).
- **Inclinometría** (campos de interés de las muestras. Leyes de los minerales de las muestras).
- **Litología** (clasificación litológica de las muestras).

Se necesita generar reportes de la información obtenida de estos pozos para tener registrado todo el proceso a llevar a cabo posteriormente. Con la inexistencia de una herramienta informática propia para realizar estas operaciones, que cumpla con todos los requisitos necesarios, se dificulta el trabajo para el análisis de los pozos de sondeo.

En la actualidad se refleja la necesidad de contar con dicha herramienta para mantener organizados y a salvo todos los datos y facilitar la consulta y gestión de la información.

1.3.3. Situación Problemática

Actualmente existe una numerosa cantidad de herramientas para la gestión de información minera desarrolladas en distintos lenguajes, la mayoría de estas herramientas no cumplen estrictamente con los requisitos requeridos por los especialistas mineros cubanos, mencionando específicamente la gestión de información de pozos de sondeo. La mayor parte de estas herramientas son propietarias, lo

Capítulo 1: Fundamentación Teórica

que significa que para utilizarlas hay que pagar una licencia (la cual resulta cara) por cada computadora donde se instalen y el país no se encuentra en condiciones para realizar dicho gasto. Mencionando también algunas de las desventajas que posee el software propietario como lo son:

- ✓ No existen aplicaciones para todas las plataformas (Windows y Mac OS).
- ✓ Imposibilidad de copia.
- ✓ Imposibilidad de modificación.
- ✓ Restricciones en el uso (marcadas por la licencia).
- ✓ Imposibilidad de redistribución.
- ✓ Por lo general suelen ser menos seguras.
- ✓ El coste de las aplicaciones es mayor.
- ✓ El soporte de la aplicación es exclusivo del propietario.
- ✓ El usuario que adquiere software propietario depende al 100% de la empresa propietaria.

Al no conocer la arquitectura de dichas herramientas no se puede reestructurar algunas funcionalidades para adaptarlas a las necesidades existentes, por lo cual el trabajo con ellas se torna difícil e insuficiente.

También se utilizan algunas herramientas desarrolladas en el país pero que son diseñadas para satisfacer necesidades específicas, y no tienen la calidad necesaria. Con el desarrollo de una herramienta propia para la gestión de la información de los pozos de sondeo se tendrá conocimiento de la arquitectura y así poder agregar más funcionalidades de ser necesario.

1.4. Análisis de soluciones existentes.

Hoy día se pueden encontrar muchas herramientas destinadas a la minería, ya sea nacional o internacional. A continuación se brindan algunas soluciones existentes:

- **Downhole Explorer**

Es un programa excepcional para ploteo y presentación de sondajes, con muchas características innovadoras. Los manuales y asistentes hacen que el programa sea extremadamente fácil de usar. Los asistentes hacen posible que un usuario inexperto pueda preparar un proyecto completo con secciones y hojas de logueo en cuestión de minutos.

Capítulo 1: Fundamentación Teórica

Downhole Explorer es independiente de la base de datos y está diseñado para trabajar con el software que usted trabaja. Cualquier cambio o adición hecho en la fuente de datos o en las tablas se refleja automáticamente en todos los ploteos.

Características generales de Downhole Explorer:

Corre en plataformas como Windows 98, ME, 2000 y XP.

Un solo programa para manejar hojas de logueo, secciones y vistas 3D.

Posee interfaces con múltiples vistas, ventanas y documentos.

Permite hipervínculos a datos y características gráficas.

Las barras de herramientas pueden ser personalizadas y ancladas.

Guarda las hojas de logueo en formatos de mapas de bits, JPEG y TIFF.

El programa es capaz de manejar cualquier formato de datos de sondaje y permite al usuario importar cualquier número de intervalos de logueo (propiedades definidas por intervalos desde-hasta) e intervalos de profundidad (propiedades definidas a profundidades específicas). Datos geofísicos, hidrogeológicos, mineralógicos y geomecánicos, pueden ser importados, combinados y mostrados con la misma facilidad de logueos geoquímicos y litológicos estándar.

Las interfaces de múltiples vistas, ventanas y documentos de Downhole Explorer le permiten adaptar la visualización a necesidades específicas. Las vistas son vinculadas permitiendo que se haga una selección en una y que se muestre en todas las demás. (Pincay, 2009)

- **Datamine**

El Datamine es un software de ingeniería orientado a las ciencias de la Tierra, que hasta ahora se ha utilizado en el sector minero en el modelamiento de yacimientos mineros de tipo metálico y no metálico, esto no impide que por la similitud de principios lógicos de ingeniería se use para los recursos de los hidrocarburos y agua.

Este software cuenta con más de 1000 sistemas en uso diario en más de 45 países a través del mundo. Los usos más comunes del sistema son; la captura y análisis de la información, exploración, geología, geoquímica, mecánica de rocas, topografía, modelamiento geológico, diseño de mina a Cielo

Capítulo 1: Fundamentación Teórica

Abierto y Subterráneas, planeamiento minero, y áreas relacionadas a los estudios ambientales. (Revista del Instituto de Investigación FIGMMG)

- **Gemcom**

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 o Whittle 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.

- ✓ **MPMS:** Solución orientada a la administración y control de los procesos en la cadena de valor productiva de una empresa minera.
- ✓ **ProdTrack:** Sistema de control de producción y balance metalúrgico a lo largo de la línea de procesos.
- ✓ **GEMS:** Suite de herramientas de aplicación a las tareas de una operación minera, que cubren las necesidades de los profesionales en todas las áreas de la ingeniería y geología.
- ✓ **GEMS-PCBC:** Suite de herramientas específicas para faenas que utilizan el método de *Block Caving* para su explotación. PCBC opera bajo ambiente GEMS, y está completamente integrado con el resto de herramientas generales de administración de datos y otras.
- ✓ **Whittle:** Sistema para optimización económica y planeamiento estratégico de minado. Herramientas para el análisis de sensibilidad, secuenciamiento óptimo, aplicación de algoritmos de Millaza y Lerch y Grossman, son parte de este sistema.
- ✓ **EQWin:** Sistema de administración y análisis de datos de medio ambiente. (Pincay, 2009)

Capítulo 1: Fundamentación Teórica

- **Vulcan**

El software minero geológico Vulcan es uno de los productos principales de Maptek. Creado en el año 1981, actualmente está disponible en la Versión 7.0. Entre sus principales características, permite su aplicación a todos los aspectos geológicos y de planificación minera; posee las herramientas necesarias para la creación del diseño, tanto para campo abierto como para bajo tierra; permite el modelamiento de yacimientos metálicos, carboníferos y de metales industriales, y entrega las herramientas para un completo desarrollo topográfico y de geomensura.

El software permite convertir los complejos datos en información visual que se reflejan en la creación de imágenes interactivas y dinámicas en 3D, facilitando su comprensión y entendimiento. Además, puede desarrollar completos estudios de factibilidad y de impacto ambiental. (Pincay, 2009)

- **Surpac**

Surpac Minex Group tiene una serie de soluciones computacionales para el trabajo geológico minero, entre las cuales se puede mencionar:

- ✓ **Surpac Vision:** Software geológico minero que cubre desde las tareas de exploración hasta la planificación de la mina. Este programa se caracteriza por tener un fácil manejo y gran potencialidad al manejar información de distintos formatos, además de poder realizar conexiones de trabajos múltiples desde internet.
- ✓ **Minesched:** Software de planificación minera de desarrollo y planificación de la producción desde el corto hasta largo plazo en minería subterránea y de cielo abierto. Este programa tiene interfaz directa con MS Project y Excel, permitiendo al planificador obtener flexibilidad y un mejor manejo del plan minero al analizar múltiples opciones en un corto tiempo.
- ✓ **Tailings:** Es un programa que permite simular el llenado de un tranque de relaves, considerando puntos de descarga y diferentes pendientes de depositado. Surpac tiene aplicación en el área geológica minera, específicamente en: Exploración geológica; Estimación de reservas; Diseño en minería open pit y subterránea; Planificación minera; Modelamiento de relaves; Levantamiento topográfico. (Pincay, 2009)

Capítulo 1: Fundamentación Teórica

➤ **MineSched**

MineSched es un programa que permite generar planificaciones a corto y a largo plazo y cumplir los objetivos de mezclado en las explotaciones mineras superficiales y subterráneas de todos los tipos y tamaños. Este es capaz de planificar sobre la base de modelos de bloques, cuadrículas y polígonos predeterminados, que han sido creados con diversos sistemas de planificación minera, incluyendo Gemcom Surpac y Gemcom Minex. El aspecto más importante de MineSched es que permite generar programas de producción que mejoran la productividad y los rendimientos más allá de lo que se lograría con una planificación manual.

Características de MineSched

➤ **Programación por objetivos:**

MineSched a través de la programación "Target scheduling" permite que el usuario fije objetivos múltiples, dadas las prioridades elegirá que bloques minar para resolver el mejor caso posible buscando el objetivo, dentro de las restricciones dadas por el usuario.

La programación Target scheduling incluye la funcionalidad de blending el cual permite optimización de mezclar y de ley, además los objetivos se pueden fijar en cocientes materiales y cálculos definidos por el usuario. Target scheduling también incluye programación basada en prioridad y nivelación automática de la producción. Precedencias de bloques, direcciones de minado, localización o ubicaciones de minado, límites de banco y polígono, dan al usuario control para alcanzar programas prácticos y óptimos.

MineSched fue desarrollado bajo la teoría de dar a los ingenieros planificadores máxima flexibilidad para la toma de decisiones.

Programar con target scheduling de MineSched se ha probado muy bien desde su lanzamiento, en términos del funcionamiento y de resultados. La programación de la mina lleva a menudo solamente minutos y la flexibilidad de permitir determinar donde minar por lo cual los resultados que reúnen los objetivos son a menudo perfectamente alcanzados.

➤ **Reportes**

Capítulo 1: Fundamentación Teórica

Se generan informes con tonelaje de producción y leyes que pueden ser obtenidas. Los informes son desarrollados en Microsoft Excel y pueden también ser generados informes sumarios por período, informes más detallados por polígono y bancos. El formato de los informes es totalmente flexible y el usuario define su formato. Las planillas se usan generalmente para ligar tablas y gráficos según la calidad del informe directamente a los resultados arrojados por MineSched, y se pueden estudiar e imprimir inmediatamente después de terminar de programar con MineSched.

Además de obtener reportes de tonelaje y ley, MineSched permite que los usuarios definan sus propias expresiones. Las expresiones definidas por el usuario pueden contener parámetros de tonelaje y ley de la producción. Esto da la capacidad para realizar, por ejemplo, los análisis financieros del plan totalmente dentro de MineSched. Externo a MineSched, el usuario puede manipular los datos de salida sin problemas.

➤ **Salidas Gráficas**

El software crea automáticamente la salida gráfica que demuestra donde ocurre la explotación minera para cada período programado, en la forma de polígonos o de sólidos tridimensionales. Los polígonos llevan sus datos asociados de tonelaje y de ley como atributo permitiendo una fácil generación de reportes de las programaciones. La secuencia de minado se puede también mostrar en pantalla animada, y cada franja (período de minado) automáticamente guardada para que el usuario defina la resolución para la exportación a otro software de presentación de multimedia. La versión 4 de MineSched puede crear automáticamente las superficies finales de cada período utilizando una técnica de grilla definida por el usuario. (Pincay, 2009)

1.5. Conclusiones

Durante la realización del presente capítulo se ha abordado la información necesaria referente a los conceptos que influyen en el entendimiento de este trabajo, y se han adquirido conocimientos acerca de los pozos de sondeo, elemento fundamental en la minería.

Se realizó un análisis sobre las condiciones en las que las instituciones mineras del país realizan la gestión de la información a los pozos de sondeo, determinando la importancia que tiene desarrollar una herramienta que satisfaga las necesidades de los geólogos cubanos.

Estudiar y comparar las soluciones existentes y analizar la situación problemática antes planteada, permitió determinar las características necesarias con las que debe cumplir la herramienta.

Capítulo 2: Tendencias y Tecnologías actuales a desarrollar.

Capítulo 2: Tendencias y Tecnologías actuales a desarrollar.

2.1. Introducción

En el presente capítulo se realiza el estudio correspondiente a las herramientas y tecnologías a utilizar en el desarrollo de la herramienta informática para la gestión de la información de los pozos de sondeo. En cada caso se analizan las fortalezas que inclinaron hacia su elección, así como las debilidades de los mismos.

2.2. Metodología de Desarrollo de Software

Una metodología de desarrollo de software es una colección de documentos referentes a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software. Estas tienen como principal fin garantizar la eficiencia, estructurar, planificar, minimizar las pérdidas de tiempo y controlar el proceso de desarrollo en sistemas de información.

Las metodologías de desarrollo surgen con el objetivo de guiar a todo el personal que interactúa en el desarrollo del software, así como garantizar un trabajo organizado y la obtención de un producto de alta calidad. En la actualidad existen dos tipos de metodologías de desarrollo del software: la metodología ágil y la metodología tradicional o robusta.

2.2.1. Rational Unified Process (RUP)

El Proceso Unificado es un proceso de desarrollo de Software. Comprende un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organización, diferentes niveles de aptitud y diferentes tamaños de proyecto. RUP está basado en componentes, lo cual quiere decir que el sistema de software en construcción está formado por componentes de software interconectados a través de interfaces bien definidas. Es un proceso que define claramente quién, cuándo, cómo y qué debe hacerse, y como su enfoque está basado en modelos, utiliza un lenguaje bien definido para tal fin, el UML.

RUP posee tres características esenciales: está dirigido por los Casos de Uso: que orientan el proyecto a la importancia para el usuario y lo que este quiere, está centrado en la arquitectura: que relaciona la toma de decisiones que indican cómo tiene que ser construido el sistema y en qué orden, y es iterativo e incremental: donde divide el proyecto en mini proyectos donde los casos de uso y la arquitectura cumplen sus objetivos de manera más depurada. (Jacobson, 2000)

La metodología RUP está dividida en cuatro fases de desarrollo del producto:

Capítulo 2: Tendencias y Tecnologías actuales a desarrollar.

- ✓ Inicio: El objetivo en esta etapa es determinar la visión del proyecto.
- ✓ Elaboración: En esta etapa el objetivo es determinar la arquitectura óptima.
- ✓ Construcción: En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial, o sea, crear el producto.
- ✓ Transición: El objetivo es llegar a obtener el release del proyecto.

RUP describe como:

- ✓ Obtener, organizar y documentar la funcionalidad y restricciones requeridas.
- ✓ Documentar y monitorear las alternativas y decisiones.

Las nociones de Casos de Uso y de Escenarios utilizadas en RUP han demostrado ser una manera excelente de capturar los requerimientos funcionales y asegurarse que direccionan el diseño, la implementación y la prueba del sistema, logrando así que el sistema satisfaga las necesidades del usuario.

Utiliza arquitecturas basadas en componentes:

- ✓ El proceso de software debe focalizarse en el desarrollo temprano de una arquitectura robusta ejecutable, antes de comprometer recursos para el desarrollo en gran escala. RUP describe como diseñar una arquitectura flexible, que se acomode a los cambios, comprensible intuitivamente y promueve una más efectiva reutilización de software. Soporta el desarrollo de software basado en componentes: módulos no triviales que completan una función clara. RUP provee un enfoque sistemático para definir una arquitectura utilizando componentes nuevos y preexistentes.

Modelación del software visualmente:

- ✓ RUP muestra cómo hacer la modelación de un software visualmente para capturar la estructura y comportamiento de arquitecturas y componentes. Las abstracciones visuales ayudan a comunicar diferentes aspectos del software; comprender los requerimientos, ver como los elementos del sistema se relacionan entre sí, mantener la consistencia entre diseño e implementación y promover una comunicación precisa. El estándar UML (Lenguaje de Modelado Unificado), creado por Rational Software, es el cimiento para una modelización visual exitosa.

Capítulo 2: Tendencias y Tecnologías actuales a desarrollar.

Para la realización de este trabajo se decidió utilizar RUP debido a que aporta todos los elementos necesarios para desarrollar aplicaciones grandes y que requieren de mucha documentación. Captura varias de las mejores prácticas en el desarrollo moderno de software en una forma que es aplicable para un amplio rango de proyectos y organizaciones. Es una guía de cómo utilizar de manera efectiva UML.

RUP permite seleccionar fácilmente el conjunto de componentes de procesos que se ajustan a las necesidades específicas del sistema. La aproximación iterativa ayuda a mitigar los riesgos en forma temprana y continua, además provee un entorno de proceso de desarrollo configurable basado en estándares; permite tener claro y accesible el proceso de desarrollo que se sigue y que este sea configurado a las necesidades de la organización y del proyecto.

2.3. Lenguaje de Modelado: UML

El lenguaje unificado de modelado o notación (UML) sirve para especificar, visualizar y documentar esquemas de sistemas de software orientado a objetos. UML no es un método de desarrollo, lo que significa que no sirve para determinar qué hacer en primer lugar o cómo diseñar el sistema, sino que simplemente le ayuda a visualizar el diseño y a hacerlo más accesible para otros. Es el estándar de descripción de esquemas de software.

UML está diseñado para su uso con software orientado a objetos, y tiene un uso limitado en otro tipo de cuestiones de programación. (Larman, 1999)

UML se compone de muchos elementos de esquematización que representan las diferentes partes de un sistema de software. Los elementos UML se utilizan para crear diagramas, que representa alguna parte o puntos de vista del sistema. Posee formas de modelar conceptos como por ejemplo las funciones del sistema, además de otras particularidades como la de escribir clases en un lenguaje determinado, esquemas de bases de datos y componentes de software reusables. Usa procesos de otras metodologías, aprovechando la experiencia de sus creadores. UML se ha convertido en el lenguaje de modelado más conocido y utilizado en la actualidad; aun cuando todavía no es un estándar oficial, con las siguientes características:

- ✓ Permite modelar sistemas utilizando técnicas orientadas a objetos (OO).
- ✓ Permite especificar todas las decisiones de análisis, diseño e implementación, construyéndose así modelos precisos, no ambiguos y completos.

Capítulo 2: Tendencias y Tecnologías actuales a desarrollar.

- ✓ Puede conectarse con lenguajes de programación (Ingeniería directa e inversa).
- ✓ Permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones, etc.).
- ✓ Cubre las cuestiones relacionadas con el tamaño, propias de los sistemas complejos y críticos.
- ✓ Existe un equilibrio entre expresividad y simplicidad, pues no es difícil de aprender ni de utilizar.
- ✓ UML es independiente del proceso, aunque para utilizarlo óptimamente se debería usar en un proceso que fuese dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental.

2.4. Herramienta CASE

Las herramientas CASE son un conjunto de programas y ayudas destinados a la asistencia de los analistas, ingenieros y desarrolladores de software para reducir el coste en términos de tiempo y dinero. Estas están presentes en el desarrollo del software en todas sus fases automatizando y facilitando todas sus tareas. La mejor razón para la creación de estas herramientas fue el incremento en la velocidad de desarrollo de los sistemas.

El concepto de CASE es muy amplio; y una buena definición genérica, que pueda abarcar esa amplitud de conceptos, sería la de considerar a la Ingeniería de Software Asistida por Computación (CASE), como la aplicación de métodos y técnicas a través de las cuales se hacen útiles a las personas, comprender las capacidades de las computadoras, por medio de programas, de procedimientos y su respectiva documentación.

2.4.1. Visual Paradigm

Visual Paradigm, es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Soporta un conjunto de lenguajes, tanto en la generación de código y como de ingeniería inversa sobre Java, C++, CORBA IDL, PHP, XML Schema, Ada y Python. Además, apoya la generación de código

Capítulo 2: Tendencias y Tecnologías actuales a desarrollar.

C#, VB, NET, código script de Flash, Delphi, Perl, Ruby, entre otros. Ingeniería Inversa también apoya clase Java, .NET, .dll y .exe, JDBC. (Canales, 2008)

Características:

- ✓ Producto de calidad.
- ✓ Soporta aplicaciones web.
- ✓ Las imágenes y reportes generados, no son de muy buena calidad.
- ✓ Varios idiomas.
- ✓ Generación de código para Java y exportación como HTML.
- ✓ Fácil de instalar y actualizar.
- ✓ Compatibilidad entre ediciones.
- ✓ Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.

Además Visual Paradigm ofrece:

- ✓ Entorno de creación de diagramas para UML 2.1
- ✓ Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- ✓ Capacidades de ingeniería directa (versión profesional) e inversa.
- ✓ Modelo y código que permanece sincronizado en todo el ciclo de desarrollo
- ✓ Disponibilidad de múltiples versiones, para cada necesidad.
- ✓ Disponibilidad de integrarse en los principales IDEs.
- ✓ Disponibilidad en múltiples plataformas.
- ✓ Generador de informes para generación de documentación

Existe otra potente herramienta CASE que es el Rational Rose, pero se decidió utilizar Visual Paradigm para la realización de la ingeniería de este trabajo porque es una herramienta multiplataforma además de ser compatible con la metodología de desarrollo que se decidió utilizar, y brinda muchas facilidades de generación de documentación lo cual es importante en el software que se está desarrollando.

Capítulo 2: Tendencias y Tecnologías actuales a desarrollar.

2.5. Leguaje de Programación

Un lenguaje de programación es un modo práctico para que los seres humanos puedan comunicarse y dar instrucciones a un equipo. Estos representan en forma simbólica y en manera de un texto los códigos que podrán ser leídos por una persona. Los lenguajes de programación facilitan la tarea de programación, ya que disponen de formas adecuadas que permiten ser leídas y escritas por personas, a su vez resultan independientes del modelo de computador a utilizar.

Está constituido por un grupo de reglas gramaticales, un grupo de símbolos utilizables, un grupo de términos con sentido único y una regla principal que resume las demás. Para que esta construcción mental sea operable en un computador debe existir otro programa que controle la validez o no de lo escrito. A este se le llama traductor.

Un lenguaje de programación puede ser de alto o bajo nivel. En los de bajo nivel las instrucciones son simples y cercanas al funcionamiento de la máquina, como por ejemplo el código máquina y el ensamblador. En los lenguajes de alto nivel hay un alto grado de abstracción y el lenguaje es más próximo a los humanos, como por ejemplo Lexico, PASCAL, Cobol o Java.

2.5.1. C++

C++ es un potente lenguaje de programación compilado y orientado a objetos que permite programar compiladores, sistemas operativos, juegos, procesadores de texto, aplicaciones de base de datos, etc. Es una extensión del lenguaje de programación C. Abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos.

Las principales características de C++ son las facilidades que proporciona para la programación orientada a objetos y para el uso de plantillas o programación genérica (*templates*). Permite trabajar tanto a alto como a bajo nivel, sin embargo es a su vez uno de los que menos automatismos trae (obliga a hacerlo casi todo manualmente al igual que C) lo que "dificulta" mucho su aprendizaje. (Stroustrup, 1998)

Características del lenguaje de programación C++:

- ✓ Compilado: más eficiente.
- ✓ Memoria automática gestionada por el RTS.
- ✓ Memoria dinámica gestionada por el programador.
- ✓ Métodos virtuales y no virtuales.

Capítulo 2: Tendencias y Tecnologías actuales a desarrollar.

- ✓ Punteros a void.

Además posee una serie de propiedades difíciles de encontrar en otros lenguajes de alto nivel:

- ✓ Posibilidad de redefinir los operadores (sobrecarga de operadores).
- ✓ Identificación de tipos en tiempo de ejecución (*RTTI*).

Paradigma: Multiparadigma: orientado a objetos, imperativo, programación genérica.

Tipo de dato: Fuerte, estático.

Implementaciones: GNU Compiler Collection, Microsoft Visual C++, Borland C++ Builder, Dev-C++, C-Free

Dialectos: ISO C++, ANSI C++ 1998, ANSI C++ 2003.

Influido por: C, Simula.

Ha influido: Ada, C#, Java, PHP, D, Perl

Cada lenguaje de programación posee características que lo hace más potente en una rama de desarrollo que otros. Para este trabajo se tuvo en cuenta lenguajes como *c#* pero el IDE a utilizar con este es propietario y lo que se quiere es desarrollar una aplicación con herramientas libres. Java es otro lenguaje que daría solución al problema a resolver pero es muy lento ya que utiliza una máquina virtual lo que hace más lento el proceso de compilación y eso no sería lo más eficiente. Se decidió utilizar *c++* porque es un lenguaje multiplataforma que a pesar de que posee un alto nivel de complejidad trabaja muy rápido a la hora de trabajar con modelaciones, lo que es una de las funciones más importantes de la herramienta a desarrollar.

2.6. Entorno de Desarrollo Integrado (IDE)

Los IDE no son más que programas compuestos por herramientas que son necesarias al programador a la hora de implementar. Puede ocurrir que el programa se dedique únicamente a un solo lenguaje de programación o a varios. Es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Los IDEs pueden ser aplicaciones por si solas o pueden ser parte de aplicaciones existentes.

2.6.1. Qt Creator

Capítulo 2: Tendencias y Tecnologías actuales a desarrollar.

Qt Creator es un IDE (entorno de desarrollo integrado) multiplataforma que se ajusta a las necesidades de los desarrolladores de Qt. Se centra en proporcionar características que ayudan a los nuevos usuarios de Qt a aprender y comenzar a desarrollar rápidamente, también aumenta la productividad de los desarrolladores con experiencia en Qt.

Qt Creator posee un depurador visual (visual debugger) para C++ consciente de la estructura de muchas clases de Qt, lo que aumenta la capacidad de mostrar los datos de Qt con claridad, además muestra la información en bruto procedente de GDB de una manera clara y concisa.

Qt Creator posee:

- ✓ Editor de código con soporte para C++, QML y ECMAScript.
- ✓ Herramientas para la rápida navegación del código.
- ✓ Resaltado de sintaxis y auto-completado de código.
- ✓ Control estático de código y estilo a medida que se escribe.
- ✓ Soporte para refactorización de código.
- ✓ Ayuda sensitiva al contexto.
- ✓ Plegado de código

Principales características de Qt Creator:

- ✓ Posee un avanzado editor de código C++.
- ✓ Además soporta los lenguajes: C#/.NET Languages (Mono), Python: PyQt y PySide, Ada, Pascal, Perl, PHP y Ruby.
- ✓ Posee también una GUI integrada y diseñador de formularios.
- ✓ Herramienta para proyectos y administración.
- ✓ Ayuda sensible al contexto integrado.
- ✓ Depurador visual.
- ✓ Resaltado y auto-completado de código.
- ✓ Soporte para refactorización de código.

Capítulo 2: Tendencias y Tecnologías actuales a desarrollar.

De los IDE existentes se decidió utilizar Qt Creator porque es un IDE potente capaz de crearle una interfaz gráfica a una aplicación realizada en C++. Es adaptable a varios lenguajes de programación y posee componentes propios que permiten un mayor desarrollo de aplicaciones.

2.7. Sistema gestor de Bases de datos

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

2.7.1. SQLite

SQLite es un sistema de gestión de bases de datos relacional contenida en una relativamente pequeña (~275 kiB)² biblioteca en C. Se diseñó para proporcionar una manera conveniente para aplicaciones utilizadas en la gestión de datos sin la sobrecarga que a menudo viene con sistemas dedicados a bases de datos relacionales. Tiene una reputación por ser muy portátil, fácil de usar, compacta, eficiente y confiable.

A diferencia de los sistemas de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo, es decir, tanto el cliente como el servidor corren juntos en el mismo proceso. Esto reduce los gastos generales relacionados con el ajuste de la red, simplifica la administración de bases de datos, y hace que sea más fácil de implementar una aplicación.

Un programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina cliente. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción.

La biblioteca implementa la mayor parte del estándar SQL-92, incluyendo transacciones de base de datos atómicas, consistencia de base de datos, aislamiento, y durabilidad (ACID), triggers y la mayor parte de las consultas complejas.

Capítulo 2: Tendencias y Tecnologías actuales a desarrollar.

2.8. Conclusiones

En este capítulo se ha realizado un estudio de las tendencias y tecnologías actuales para el desarrollo de aplicaciones informáticas, todas de alto nivel y eficiencia. Se pudo determinar que cada tecnología conlleva a un fin determinado y de su correcta elección puede depender la calidad del producto.

La combinación de C++ como lenguaje de programación, Qt Creator como IDE de desarrollo, UML como lenguaje de modelado, Visual Paradigm como herramienta CASE y SQLite como gestor de base de datos es un paso importante para el desarrollo de la herramienta en cuestión.

Capítulo 3: Presentación de la solución propuesta.

Capítulo 3: Presentación de la solución propuesta.

3.1. Introducción

En este capítulo se realiza la descripción de las principales definiciones asociadas al dominio del problema. También se especifican los requisitos funcionales y no funcionales, casos de usos que se generan a partir de los requisitos funcionales y una explicación de cada uno de ellos a través de las descripciones textuales.

3.2. Modelo de Dominio

El modelo del dominio es una representación de los conceptos u objetos del mundo real, significativos para un problema. Tiene como objetivo fundamental la descripción de las clases más importantes en el sistema y representa conceptos del mundo real, no de los componentes de software. Las clases del dominio aparecen en tres formas típicas:

- ✓ Objetos del negocio que representan cosas que se manipulan en el negocio.
- ✓ Objetos del mundo real y conceptos de los que el sistema debe hacer un seguimiento.
- ✓ Sucesos que ocurrirán o han ocurrido.

El modelo de dominio se describe mediante diagramas UML (especialmente mediante diagrama de clases). Estos diagramas muestran las clases del dominio y cómo se relacionan unas con otras mediante asociaciones. (Jacobson, y otros, 2000)

El modelo de dominio ayuda además a los usuarios, clientes, desarrolladores y otros interesados a utilizar un vocabulario común y sentar las bases del entendimiento del desarrollo. Muestra la forma en que estarán estructuradas las clases del dominio y su relación entre ellas.

Se determinó definir un modelo de dominio para el sistema a desarrollar debido a que se hace difícil definir procesos de negocio bien estructurados que permitan construir un modelo de negocio completo. Se procederá a explicar cada uno de los conceptos que forman parte del mismo. Todo ello para tener una mejor comprensión de la estructura y dinámica del sistema, los problemas actuales dentro de este e identificar las mejoras potenciales.

3.2.1. Conceptos principales del Modelo de Dominio

Pozo: es un orificio hecho sobre la corteza terrestre con una profundidad suficiente hasta alcanzar lo que se busca.

Capítulo 3: Presentación de la solución propuesta.

Información: conjunto organizado de datos que constituye un mensaje sobre cualquier fenómeno ocurrido.

Validación: revisión detallada a cada muestra para detectar errores.

Reporte: documento que contiene un conjunto de información.

Pozo: coordenadas x, y, z de los collares de los sondajes.

Muestra: Azimut e inclinación de cada muestra del sondaje.

Inclinometría: campos de interés de las muestras. Leyes de los minerales de las muestras.

Litología: clasificación litológica de las muestras.

3.2.2. Representación del Modelo de Dominio

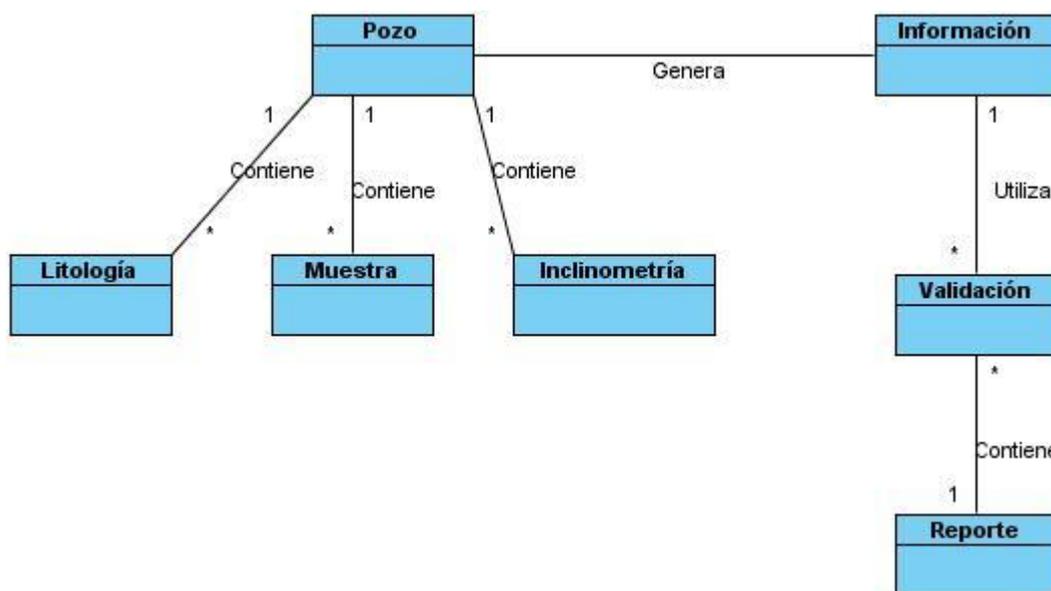


Ilustración 1: Modelo de Dominio.

3.3. Especificación de los Requerimientos

Los requisitos son condiciones o capacidades que necesita el usuario para resolver un problema o conseguir un objetivo determinado. Esta definición se extiende y se aplica a las condiciones que debe cumplir o poseer un sistema (o uno de sus componentes).

3.3.1. Requerimientos Funcionales (RF)

Capítulo 3: Presentación de la solución propuesta.

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Permiten expresar una especificación más detallada de las responsabilidades del sistema en cuestión. Se mantienen invariables, sin importarle con qué propiedades o cualidades se relacionen.

Los requerimientos funcionales con los que debe contar la herramienta para la gestión de la información de pozos de sondeo son:

RF1. Nuevo proyecto.

RF2. Importar un proyecto.

RF3. Salvar un proyecto.

RF4. Cargar información de los pozos de sondeo almacenada en ficheros de texto.

RF5. Validar información de los pozos de sondeo almacenada en ficheros de texto.

RF6. Guardar en la base de datos la información de los pozos de sondeo almacenada en ficheros.

RF7. Mostrar información de los pozos de sondeo contenida en ficheros.

RF8. Gestionar información de los pozos de sondeo.

RF8.1. Adicionar información sobre un pozo de sondeo.

RF8.2. Modificar la información almacenada de un pozo de sondeo.

RF8.3. Eliminar la información almacenada de un pozo de sondeo.

RF9. Buscar pozos dado su nombre.

RF10. Generar reportes de la información de los pozos de sondeo.

3.3.2. Requerimientos no Funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Normalmente, están vinculados a requerimientos funcionales, es decir, una vez se conozca lo que el sistema debe hacer se puede determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser:

Capítulo 3: Presentación de la solución propuesta.

- **Facilidad de uso:** el sistema debe ser fácil de usar de manera que tenga gran aceptación entre los usuarios.
- **Extensibilidad:** la nueva funcionalidad debe ser capaz de permitir la integración con otros módulos, además de permitir la inserción de cambios.
- **Apariencia o interfaz externa:** la interfaz que debe brindar la nueva funcionalidad debe ser sencilla, amigable y de rápida respuesta frente a una petición del usuario, de manera tal que agilice y facilite el trabajo con el software.
- **Mantenimiento y actualización:** debe dar facilidad de mantenimiento, y desarrollarse lo más sencilla y eficientemente posible para que en un futuro pueda ser atendido por grupos de trabajo no especializados.
- **Compatibilidad:** la nueva funcionalidad debe ser capaz de correr de manera independiente a la plataforma sobre la que es ejecutada.
- **Portabilidad:** el sistema será multiplataforma: Windows (2000, XP, Vista), Linux (Suse, Ubuntu, Debian).
- **Software:** el sistema para funcionar necesita correr sobre plataformas como: Windows (2000, XP, Vista), Linux (Suse, Ubuntu, Debian).
- **Hardware:** las PC donde se instale la herramienta deberán contar con un microprocesador Intel Pentium 4 o superior. Se debe contar con 256 Mb de RAM como mínimo. Espacio disponible en el disco duro de 50 MB como mínimo.
- **Legales:** se usarán herramientas de software libre y código abierto, o que funcionen bajo sistemas operativos representativos de código abierto.

3.4. Descripción del Sistema.

Después de haber definido los requerimientos funcionales y no funcionales con que debe contar la herramienta, se da paso a la descripción de los actores y los casos de uso y la relación entre ellos. Esta relación está representada por el diagrama de casos de uso del sistema.

3.4.1. Actores del Sistema

Se le llama actor del sistema a toda entidad externa que guarde relación con este y que le demanda una funcionalidad. El actor del sistema puede ser una persona u otro sistema o máquina. En este

Capítulo 3: Presentación de la solución propuesta.

trabajo se definió un solo actor porque la aplicación está diseñada para que un solo usuario interactúe con ella a la vez, además estará instalada en misma PC cliente, y no necesita de un administrador u otro actor que interactúe con esta.

Actores	Descripción
Geólogo	Representa al usuario que va a hacer uso del sistema, y quien tiene la responsabilidad de interactuar con todas las funcionalidades de este.

Tabla 1: Actores del Sistema.

3.4.2. Diagrama de Casos de Uso del Sistema.

Un Diagrama de Casos de Uso muestra la relación existente entre los actores y los casos de uso del sistema. También representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa.

Capítulo 3: Presentación de la solución propuesta.

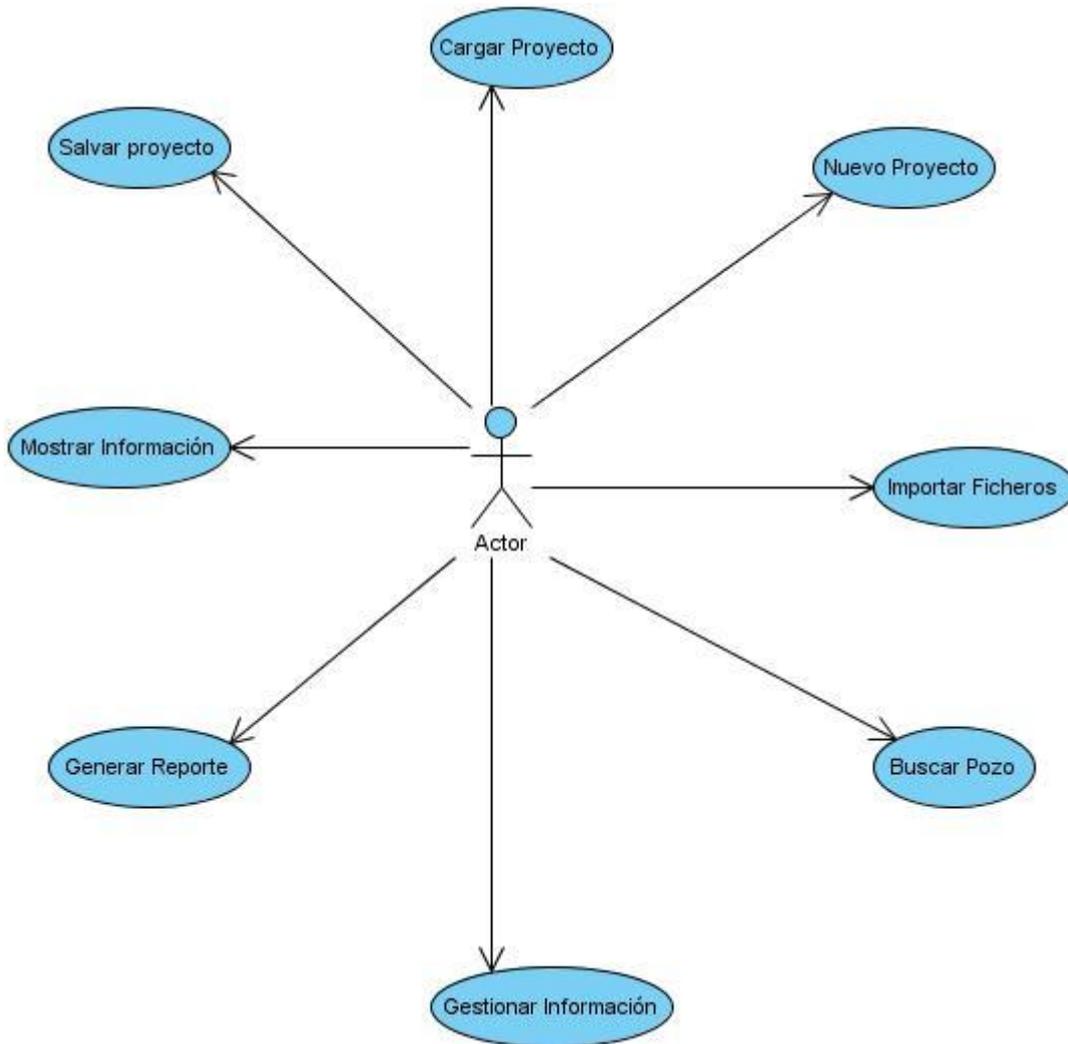


Ilustración 2: Diagrama de Casos de Uso del Sistema.

3.5. Descripción de los Casos de Uso del Sistema.

Caso de Uso:	Nuevo Proyecto
Actores:	Geólogo
Propósito	Este caso de uso se lleva a cabo con el objetivo de darle al usuario la opción de crear un nuevo proyecto.

Capítulo 3: Presentación de la solución propuesta.

Resumen:	El caso de uso se inicia cuando el usuario selecciona la opción “Nuevo Proyecto” y termina cuando el proyecto está creado.	
Precondiciones:	No necesita precondiciones.	
Referencias	RF1	
Prioridad	Crítico	
Flujo Normal de Eventos		
Acción del Actor	Respuesta del Sistema	
1- El actor selecciona la opción “Nuevo proyecto”.	2- El sistema da la opción de salvar el proyecto actual.	
3- Si el actor desea guardar el proyecto actual debe dar clic en el botón “Salvar”.	4- El sistema guarda el proyecto que estaba abierto, lo cierra y muestra una ventana para que el usuario establezca la dirección donde desea guardar el nuevo proyecto.	
5- El actor especifica la dirección donde desea guardar el proyecto y acciona el botón “Aceptar”.	6- El sistema crea el proyecto en la dirección especificada.	
Flujo alterno		
3.1- Si el actor no desea guardar el proyecto actual da clic en el botón “No”.	4.1- El sistema cierra el proyecto actual sin guardar los cambios y muestra una ventana para que el usuario establezca la dirección donde desea guardar el nuevo proyecto.	
Prototipo de Interfaz		

Capítulo 3: Presentación de la solución propuesta.

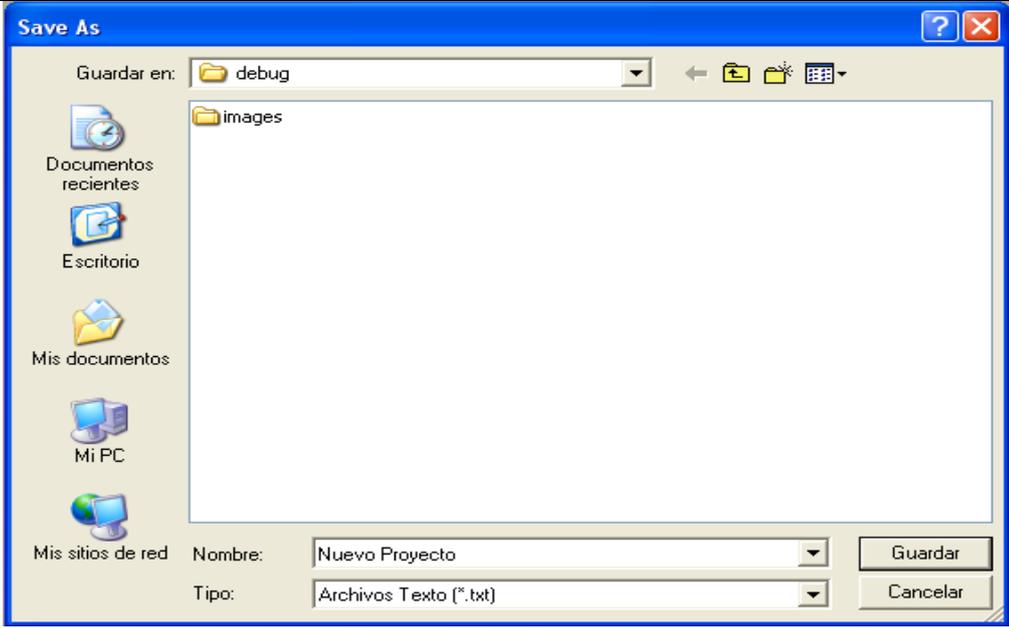
	
Poscondiciones	Se creó un nuevo proyecto

Tabla 2: Descripción del Caso de Uso Nuevo Proyecto.

Caso de Uso:	Cargar Proyecto
Actores:	Geólogo
Propósito	Este caso de uso se lleva a cabo con el objetivo de darle al usuario la opción de cargar un proyecto guardado anteriormente.
Resumen:	El caso de uso se inicia cuando el usuario selecciona la opción importar proyecto y termina cuando el proyecto haya sido importado.
Precondiciones:	Que exista un proyecto guardado.
Referencias	RF2
Prioridad	Crítico
Flujo Normal de Eventos	

Capítulo 3: Presentación de la solución propuesta.

Acción del Actor	Respuesta del Sistema
1- El actor selecciona la opción "Cargar proyecto".	2- Si existe un proyecto en ejecución el sistema lanza un mensaje preguntándole al usuario si desea guardar el proyecto actual.
3- Si el actor desea guardar el proyecto actual da clic en el botón "Sí".	4- El sistema guarda el proyecto actual y muestra una ventana para que el usuario seleccione el proyecto a cargar.
4- El actor selecciona el proyecto a cargar y da clic en el botón "Abrir".	5- El sistema carga el proyecto.
Flujo alternativo	
3.1- Si el actor no desea guardar el proyecto actual da clic en el botón "No".	4.1- El sistema cierra el proyecto en ejecución y muestra una ventana para que el usuario seleccione el proyecto a cargar.
4.1- Si el actor desea cancelar la operación Cargar Proyecto da clic en el botón "Cancelar"	5.1- El sistema cancela la operación.

Capítulo 3: Presentación de la solución propuesta.

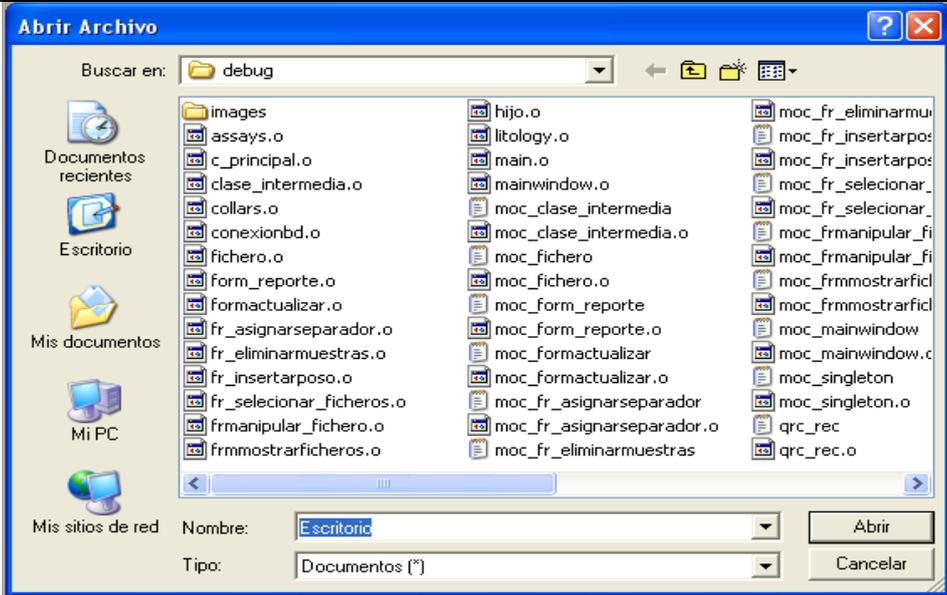
Prototipo de Interfaz	
	
Poscondiciones	Se carga el proyecto.

Tabla 3: Descripción del Caso de Uso Cargar Proyecto.

Caso de Uso:	Salvar Proyecto
Actores:	Geólogo
Propósito	Este caso de uso se lleva a cabo con el objetivo de darle al usuario la opción de salvar el proyecto en el que está trabajando.
Resumen:	El caso de uso se inicia cuando el usuario selecciona la opción Salvar y termina cuando el proyecto haya sido salvado.
Precondiciones:	Que exista un proyecto creado.
Referencias	RF3
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema

Capítulo 3: Presentación de la solución propuesta.

1- El actor selecciona la opción "Salvar".	2- El sistema salva la información en la carpeta contenedora del proyecto.
Poscondiciones	Se guarda el proyecto.

Tabla 4: Descripción del Caso de Uso Salvar Proyecto.

Caso de Uso:	Importar Ficheros	
Actores:	Geólogo	
Propósito	Este caso de uso se lleva a cabo con el objetivo de darle al usuario la opción de importar los ficheros que almacenan la información generada por los pozos de sondeo.	
Resumen:	El caso de uso se inicia cuando el usuario selecciona la opción "Importar Ficheros".	
Precondiciones:	Que exista al menos un fichero a cargar.	
Referencias	RF4, RF5, RF6	
Prioridad	Crítico	
Flujo Normal de Eventos		
Acción del Actor	Respuesta del Sistema	
1- El actor selecciona la opción "Importar Ficheros".	2- El sistema muestra un menú para que el usuario seleccione el fichero que desea importar.	
3- El actor selecciona el fichero.	4- El sistema muestra una ventana para que el usuario especifique la dirección donde está ubicado el fichero.	
5- El actor especifica la dirección del fichero y	6- El sistema carga el fichero y lo muestra en	

Capítulo 3: Presentación de la solución propuesta.

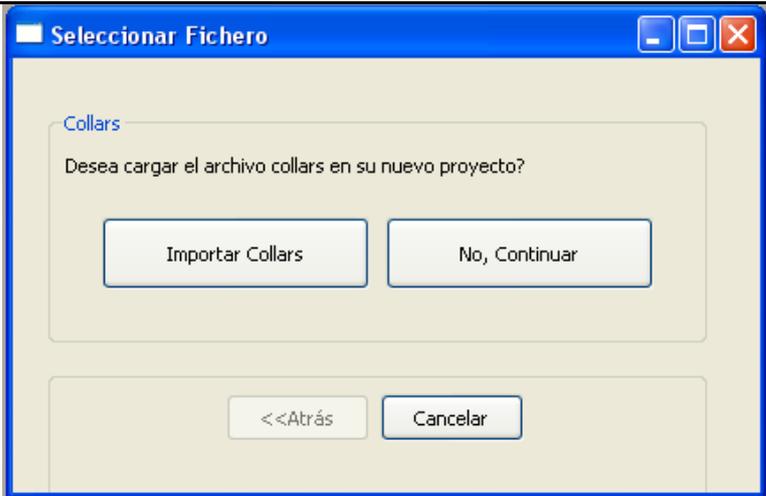
selecciona la opción "Abrir".	una ventana donde el usuario selecciona el separador que contiene el fichero importado.
7- El actor selecciona el separador y da clic en el botón "Siguiente".	8- El sistema muestra una ventana para que el usuario enlace los datos importados con cada columna del fichero.
9- El usuario enlaza los datos con las columnas y da clic en finalizar.	10- El sistema valida los datos, los inserta en la base de datos y muestra los errores del fichero.
Flujo alternativo	
7.1- Si el actor desea cancelar la operación da clic en el botón "Cancelar".	8.1- El sistema cancela la operación.
Prototipo de Interfaz	
	
Poscondiciones	Se carga el fichero.

Tabla 5: Descripción del Caso de Uso Importar Ficheros.

Caso de Uso:	Mostrar Ficheros
Actores:	Geólogo
Propósito	Este caso de uso se lleva a cabo con el objetivo de darle al usuario la opción de

Capítulo 3: Presentación de la solución propuesta.

	ver los datos contenidos en la base de datos.
Resumen:	El caso de uso se inicia cuando el usuario selecciona la opción “Mostrar Tablas”.
Precondiciones:	Que exista un proyecto en ejecución.
Referencias	RF7
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1- El actor selecciona la opción “Mostrar Tablas”.	2- El sistema muestra todas las tablas de la base de datos.
Flujo alterno	
	2.1- Si no existe ningún proyecto en ejecución el sistema muestra el mensaje “No existe ningún proyecto en ejecución”.

Capítulo 3: Presentación de la solución propuesta.

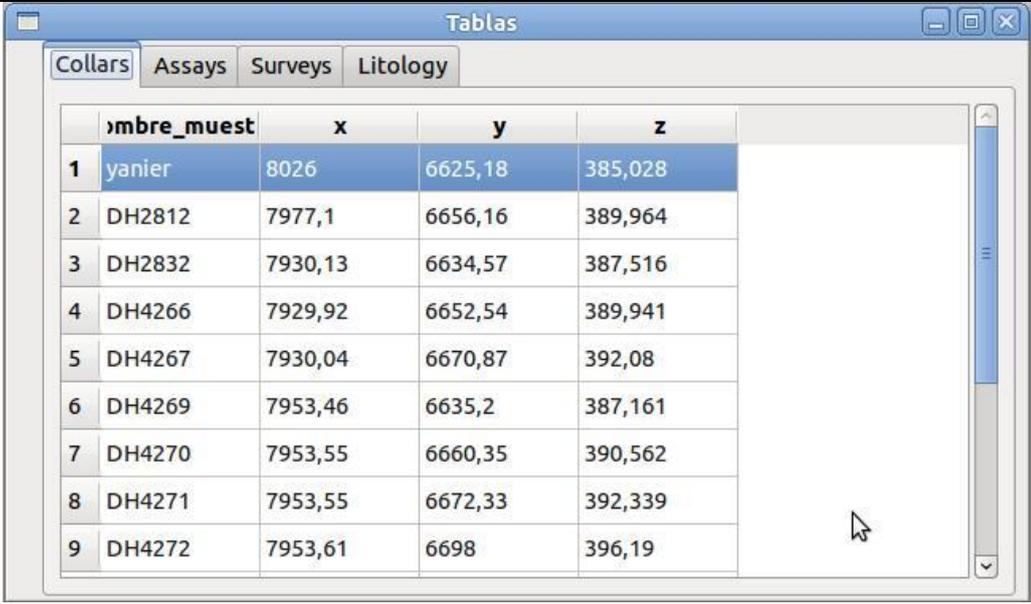
Prototipo de Interfaz	
	
Poscondiciones	Se muestran los ficheros.

Tabla 6: Descripción del Caso de Uso Mostrar Ficheros.

Caso de Uso:	Gestionar Información
Actores:	Geólogo
Propósito	Este caso de uso se lleva a cabo con el objetivo de darle al usuario la opción de adicionar, modificar o eliminar la información almacenada en la base de datos sobre los pozos de sondeo.
Resumen:	El caso de uso se inicia cuando el usuario selecciona la opción adicionar, modificar o eliminar un pozo o la información que este contiene.
Precondiciones:	Que exista al menos información sobre un pozo.
Referencias	RF5, RF6, RF8, RF8.1, RF8.2, RF8.3
Prioridad	Crítico
Flujo Normal de Eventos	

Capítulo 3: Presentación de la solución propuesta.

Acción del Actor	Respuesta del Sistema
1- El actor selecciona la opción adicionar, modificar o eliminar información de un pozo de sondeo.	2- El sistema ejecuta la opción que seleccionó el usuario (adicionar, modificar o eliminar información de un pozo de sondeo).
Sección “Adicionar”	
Acción del Actor	Respuesta del Sistema
1-El actor selecciona la acción “Adicionar muestra”.	2-El sistema muestra un formulario donde le da al usuario la opción de seleccionar la tabla en la que desea adicionar una muestra.
3- El actor selecciona la muestra.	4- El sistema muestra los campos para que el usuario adicione la información.
5- El actor escribe los datos en cada campo y da clic en el botón “Aceptar”.	6- El sistema inserta los datos en la base de datos.
Prototipo de Interfaz	
	
Sección “Modificar”	
1-El usuario da clic sobre la información que desea	2-El sistema modifica la información en la base

Capítulo 3: Presentación de la solución propuesta.

modificar y escribe la nueva información.	de datos.
Prototipo de Interfaz	
	
Sección "Eliminar"	
1-El actor selecciona la opción "Eliminar Muestra".	2-El sistema muestra una ventana con la información almacenada en la base de datos.
3- El actor selecciona la tabla de donde desea eliminar la información.	4- El sistema muestra la información contenida en la tabla seleccionada por el usuario.
4- El actor selecciona la muestra a eliminar y da clic en el botón "Eliminar".	5- El sistema muestra el mensaje "Desea eliminar la muestra + nombre de la muestra".
6- El actor da clic en el botón "Si".	7- El sistema elimina la muestra.
Flujo alternativo de la sección "Adicionar"	
5.1- El actor deja campos vacios y da clic en el botón "Aceptar".	6.1-El sistema muestra el mensaje "Faltan datos".
5.2- El actor da clic en el botón "Cancelar".	6.2- El sistema cancela la operación.
Flujo alternativo de la sección "Eliminar"	
4.1- El actor selecciona la primera fila la cual contiene el nombre de las columnas de la tabla y da clic en el botón "Eliminar".	5.1- El sistema muestra el mensaje "Debe seleccionar una muestra".
6.1- El actor después de haber seleccionado la muestra y haber ejecutado el botón "Eliminar" da	7.1- El sistema cancela la operación.

Capítulo 3: Presentación de la solución propuesta.

clic en el botón “No” del mensaje mostrado por el sistema.	
Prototipo de Interfaz	
	
Poscondiciones	Se adiciona, modifica o elimina la información.

Tabla 7: Descripción del Caso de Uso Gestionar Información.

Caso de Uso:	Buscar Pozo
Actores:	Geólogo
Propósito	Este caso de uso se lleva a cabo con el objetivo de darle al usuario la opción buscar un pozo específico sin necesidad de tener que revisar toda la lista de pozos.
Resumen:	El caso de uso se inicia cuando el usuario selecciona un pozo.
Precondiciones:	Que exista al menos un pozo en la base de datos.
Referencias	RF9
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1- El actor selecciona la opción “Tablas”.	2- El sistema muestra una ventana con la información almacenada en la base de datos y llena el combobox con el listado de los pozos

Capítulo 3: Presentación de la solución propuesta.

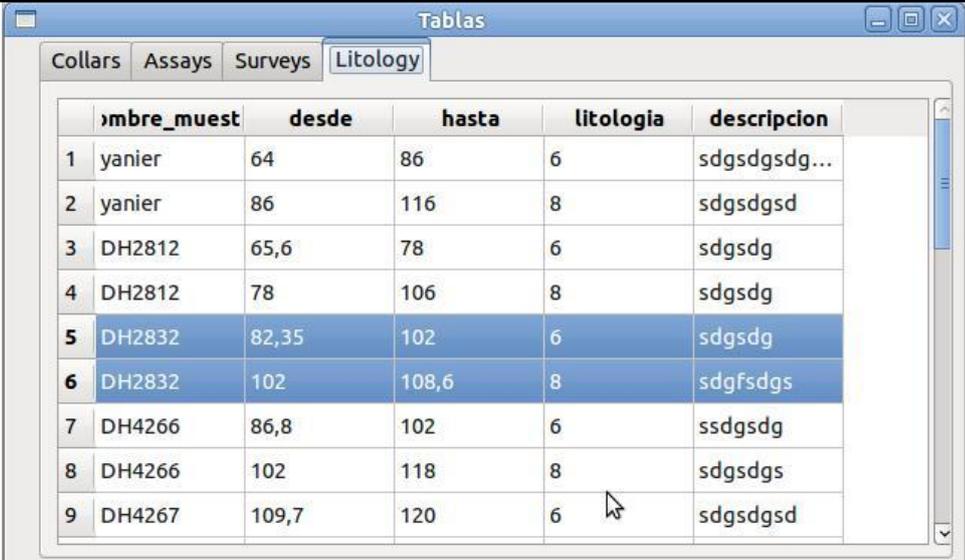
	almacenados.
3- El actor selecciona en el combobox la muestra que desea buscar.	4- El sistema marca de color azul la muestra seleccionada.
Prototipo de Interfaz	
	
Poscondiciones	Se muestra el pozo buscado.

Tabla 8: Descripción del Caso de Uso Buscar Pozo.

Caso de Uso:	Generar Reportes
Actores:	Geólogo
Propósito	Este caso de uso se lleva a cabo con el objetivo de mostrarle al usuario un resumen de la situación en la que se encuentran los pozos con los que está trabajando.
Resumen:	El caso de uso se inicia cuando el usuario selecciona la opción reportes y termina cuando se muestra la hoja de reportes de los pozos.
Precondiciones:	Que exista al menos un pozo en la base de datos al cual realizarle un análisis.
Referencias	RF10

Capítulo 3: Presentación de la solución propuesta.

Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1- El actor selecciona la opción "Reporte".	2- El sistema muestra una ventana con el reporte de los pozos almacenados en la base de datos.
Prototipo de Interfaz	
	
Poscondiciones	Se muestra la hoja de reportes de los pozos.

Tabla 9: Descripción del Caso de Uso Generar Reporte.

3.6. Conclusiones

Al no tener un negocio bien definido se tomó la decisión de realizar un modelo de dominio, el cual permitió conocer con mayor profundidad los conceptos relacionados con el problema existente. Luego de haber identificado los requerimientos funcionales quedaron definidas las funciones que el sistema debe ser capaz de ofrecerle al usuario y los casos de uso relacionados con cada requerimiento funcional. La descripción textual de los casos de uso permitió un mejor entendimiento de cómo funciona el sistema. Ya definido quién interactúa con el sistema y las funcionalidades con las que este debe cumplir se da paso al diseño de la herramienta tomando como referencia los requerimientos funcionales y no funcionales definidos anteriormente.

Capítulo 4. Construcción de la solución propuesta.

Capítulo 4. Construcción de la solución propuesta.

4.1. Introducción

Después de haber pasado por tres fases en el transcurso del desarrollo de este trabajo y haber analizado en cada una de ellas los elementos necesarios para argumentar, justificar y buscar las mejores herramientas para que el producto cumpla con todos los requerimientos definidos con buena calidad, se da paso a la construcción de la solución propuesta anteriormente. En este capítulo se definen los patrones de diseños a utilizar y cada uno de los diagramas de componentes, así como los modelos de diseño e implementación. También se describen las pruebas realizadas al sistema para comprobar que funciona correctamente.

4.2. Patrones de Diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

Para un mejor diseño de la aplicación se tuvieron en cuenta varios patrones de diseño, una vez aplicados estos patrones los diseños se hicieron más flexibles, modulares y reutilizables. Entre los patrones GRASP (patrones generales de software para asignar responsabilidades), los cuales describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones, se utilizaron los siguientes:

- Experto: para asignar una responsabilidad a la clase que tiene la información necesaria para cumplirla.
- Creador: este plantea la necesidad de asignarle a una clase la responsabilidad de crear una instancia de otra clase siempre y cuando agregue los objetos de la clase, los contenga, registre las instancias de estos objetos y los utilice específicamente.
- Alta Cohesión: que expresa que se debe asignar una responsabilidad de modo que la cohesión siga siendo alta. La cohesión no es más que la medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. (Larman, 1999)

Otros patrones de diseño utilizados fueron los GoF:

Capítulo 4. Construcción de la solución propuesta.

Patrones creacionales:

- **Fábrica Abstracta:** el cual proporciona una interfaz para la creación de familias relacionadas con los objetos o dependencias sin especificar sus clases concretas. Un ejemplo de uso de este patrón es en la creación de widgets (ventanas, scroll-bars, botones), donde se declara una fábrica abstracta `Widget`, la cual implica varias fábricas concretas de widgets adaptadas a cada situación particular (diferentes aplicaciones utilizan widgets de forma diferente) de las cuales el usuario no se da cuenta de su existencia.

Patrones estructurales:

- **Fachada:** el cual proporciona una interfaz unificada para un conjunto de interfaces o subsistemas, por lo tanto se puede afirmar que este patrón define una interfaz de alto nivel que hace más fácil el uso de los subsistemas. (Larman, 1999)

4.3. Programación por capas

La programación por capas es una técnica de ingeniería de software propia de la programación por objetos, éstos se organizan principalmente en 3 capas: la capa de presentación o frontera, la capa de lógica de negocio o control, y la capa de datos. Esta es una de las técnicas más comunes que los arquitectos de software utilizan para dividir sistemas de software complicados. (Sommerlad, y otros, 1996)

Al pensar un sistema en términos de capas, se imaginan los principales subsistemas de software ubicados de la misma forma que las capas de un pastel, donde cada capa descansa sobre la inferior. En este esquema la capa más alta utiliza varios servicios definidos por la inferior, pero esta última es inconsciente de la superior. Además, normalmente cada capa oculta las capas inferiores de las superiores a ésta. Aunque este último comportamiento algunas veces es válido obviar.

La siguiente figura muestra un esquema básico de una arquitectura siguiendo este patrón:

Capítulo 4. Construcción de la solución propuesta.



Ilustración 3: Esquema básico de la Arquitectura Tres Capas.

Las principales capas que siempre deben estar en este modelo son:

- **Capa de Presentación**

Referente a la interacción entre el usuario y el software. Puede ser tan simple como un menú basado en líneas de comando o tan complejo como una aplicación basada en formas. Su principal responsabilidad es mostrar información al usuario, interpretar los comandos de este y realizar algunas validaciones simples de los datos ingresados.

- **Capa de Negocios**

También denominada Lógica de Dominio [FOWLER], esta capa contiene la funcionalidad que implementa la aplicación. Involucra cálculos basados en la información dada por el usuario y datos almacenados y validaciones. Controla la ejecución de la capa de acceso a datos y servicios externos. Se puede diseñar la lógica en la capa de negocios para su uso directo por parte de componentes de presentación o su encapsulamiento como servicio y llamada a través de una interfaz de servicios, que coordina la conversación con los clientes del servicio e invoca cualquier flujo o componente de negocio.

- **Capa de Acceso a Datos**

Es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Existen varios patrones de arquitectura entre los que se encuentra el Modelo Vista Controlador (MVC), el cual está entre los más usados, principalmente para aplicaciones web, pero se decidió utilizar la programación por capas porque permite estructurar mucho más las aplicaciones, además de que presenta facilidades para las actualizaciones de las mismas.

Capítulo 4. Construcción de la solución propuesta.

En este aspecto la programación en capas juega un papel de suma importancia ya que le sirve al programador de guía para hacer mejoras a la aplicación sin que esto sea una tarea tediosa y desgastante, dividiendo las tareas en partes específicas para cada capa del proyecto.

4.4. Estándares de interfaz de la aplicación.

Para lograr una interfaz de usuario amigable, atractiva y funcional para el usuario final, es necesario definir principios de diseño de interfaz de usuario, pues esta es una actividad de gran importancia dentro del diseño de un software. Se debe tener en cuenta que a partir de la aceptación que tengan los usuarios depende, en gran medida el éxito de nuestro sistema.

A continuación se definen los principios a tener en cuenta en el desarrollo de la interfaz de usuario de la aplicación:

Mostrar al usuario toda la información y herramientas necesarias para cada etapa en su trabajo.

- ✓ Brindar una interfaz sencilla, de manera tal que cualquier persona con un mínimo dominio de la computación pueda trabajar con la aplicación.
- ✓ Garantizar la legibilidad de manera que exista contraste de los colores de los textos con el fondo y el tamaño de la fuente sea lo suficientemente adecuado a la vista del usuario.
- ✓ Mostrar al usuario, siempre que vaya a realizar una acción relevante sobre el sistema, un mensaje de confirmación que le permita asegurarse que es correcta la opción seleccionada.
- ✓ Los mensajes mostrados al usuario deben ser concisos y de fácil comprensión.
- ✓ Menús y etiquetas de botones deben comenzar con la palabra más importante.
- ✓ Los eventos más importantes del sistema deben ser mostrados en una barra de estado.

4.5. Diseño de la Base de Datos (BD)

En esta actividad se identifican las clases de diseño que formarán parte de la base de datos del sistema y se especifica las estructuras de la base de datos. El diseño de la base de datos tiene como propósito asegurarse de que los datos persistentes son almacenados consistente y eficientemente, así como definir el comportamiento que debe ser implementado en la base de datos.

Las bases de datos necesitan de una definición de su estructura que le permitan almacenar datos, reconocer el contenido, y recuperar la información. La estructura tiene que ser desarrollada para la necesidad de las aplicaciones que la usarán, esto puede ayudar a realizar un proceso del negocio para alcanzar un valor agregado para el cliente.

Capítulo 4. Construcción de la solución propuesta.

La base de datos diseñada para la herramienta a desarrollar es muy sencilla ya que solo se necesita almacenar la información contenida en cuatro ficheros de texto que contienen los datos generados por los pozos de sondeo.

El diagrama de clases persistentes que responde a las necesidades planteadas se muestra a continuación:

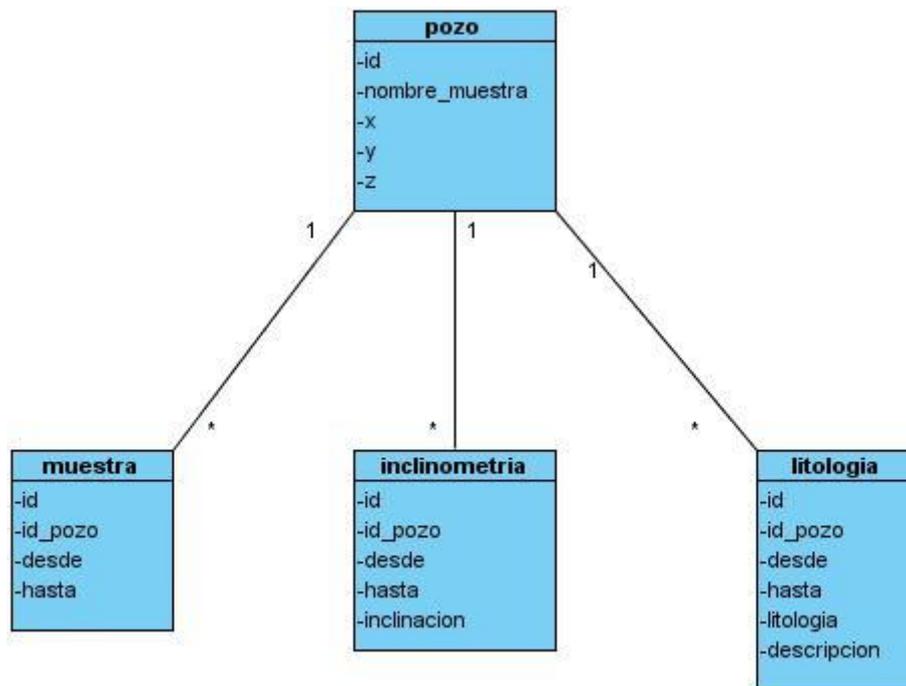


Ilustración 4: Diagrama de clases persistentes.

Una vez identificadas las clases persistentes es posible generar el modelo de datos el cual se representa en la siguiente figura:

Capítulo 4. Construcción de la solución propuesta.

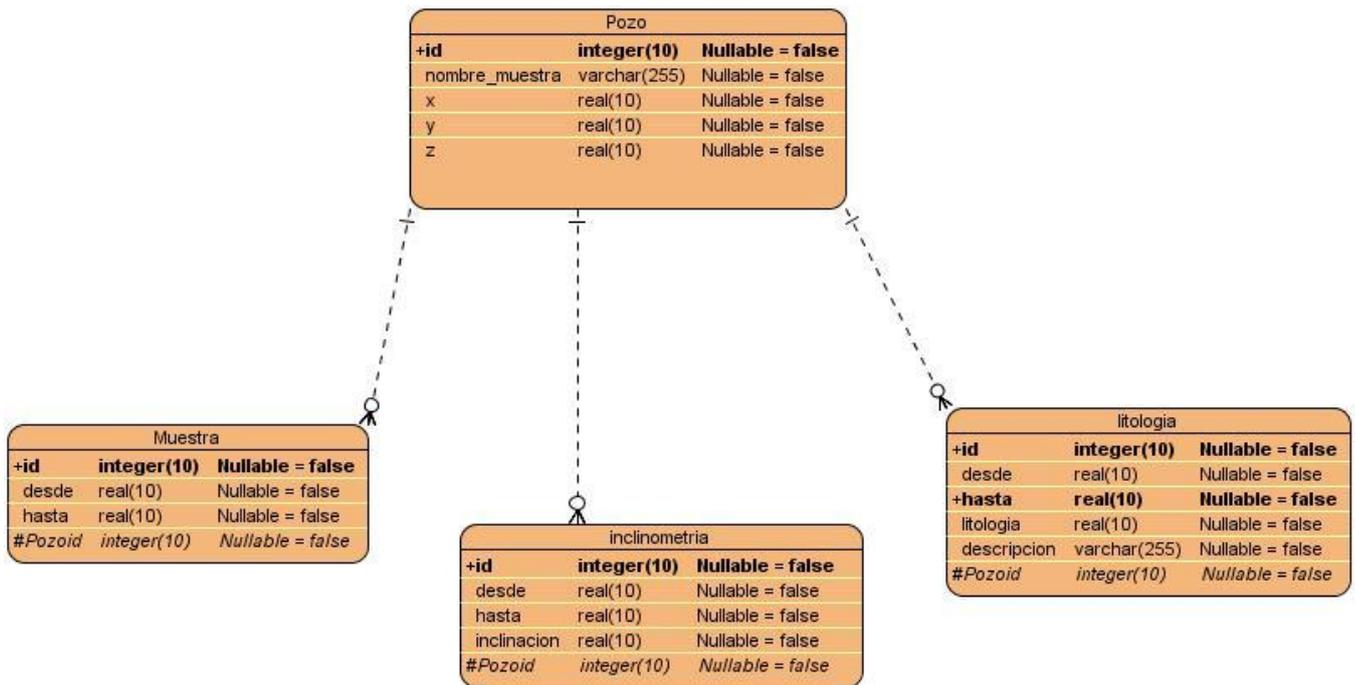


Ilustración 5: Modelo de Datos.

4.6. Diagrama de Clases del Análisis

Un diagrama de clases (**Anexo 1**) del análisis es un artefacto en el que se representan los conceptos en un dominio del problema. Representa las cosas del mundo real, no de la implementación automatizada de estas cosas.

Una clase del análisis representa una abstracción de una o varias clases y/o subsistema del diseño del sistema. UML proporciona tres estereotipos de clases estándar que podemos utilizar en el análisis: clase interfaz, clase control y clase entidad. (Jacobson, y otros, 2000)

Capítulo 4. Construcción de la solución propuesta.

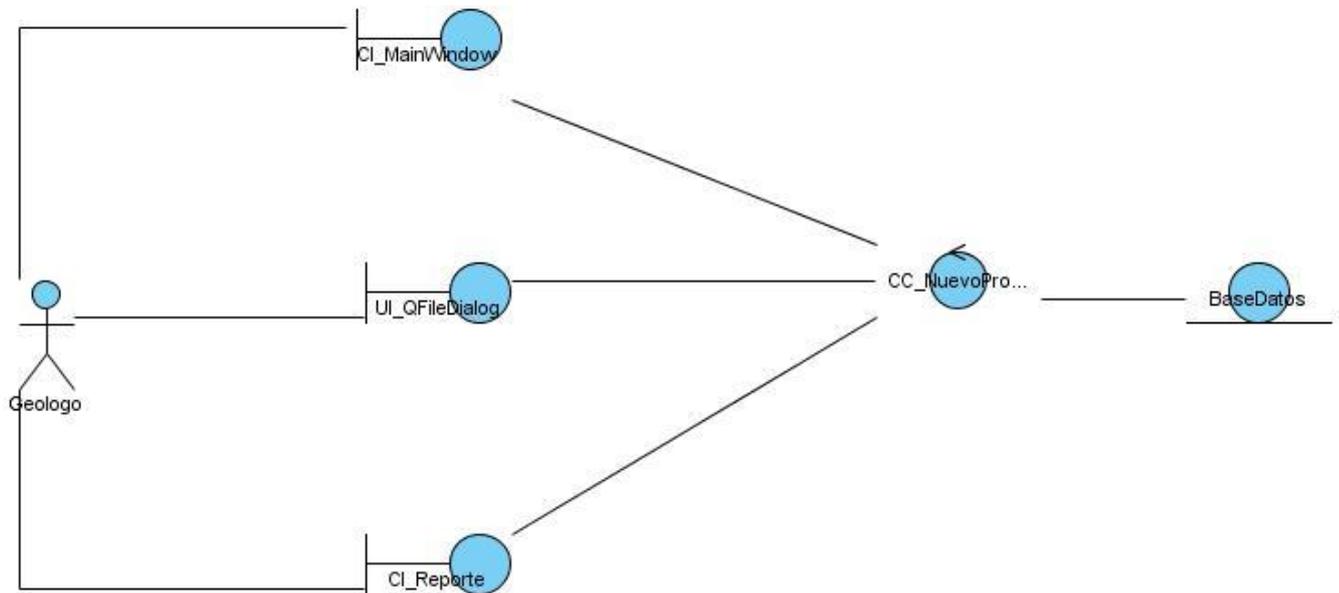


Ilustración 6: Diagrama de Clase de Análisis del Caso de Uso Nuevo Proyecto.

4.7. Diagrama de Clases del Diseño

En el diagrama de clases del diseño (**Anexo 2**) se muestran las clases participantes y sus realizaciones, así como los atributos y operaciones de cada objeto.

Las clases del diseño son una abstracción sin costuras de una clase o construcción similar en la implementación del sistema. El lenguaje utilizado para especificar una clase del diseño es lo mismo que el lenguaje de programación. Por otra parte en este modelo se hace uso de los subsistemas de diseño, que no son más que una forma de organizar los artefactos del modelo de diseño en piezas más manejables. (Jacobson, y otros, 2000)

De manera general los diagramas de clases del diseño representados funcionan de la siguiente forma: las interfaces interactúan con el controlador transmitiéndole alguna petición del usuario, este selecciona la acción correspondiente a la petición la cual se encarga de acceder a los datos necesarios para ejecutar la acción a través del modelo donde se encuentra la clase de acceso a datos; luego la capa de acceso a datos devuelve los resultados dando respuesta a la petición realizada por el usuario.

Capítulo 4. Construcción de la solución propuesta.

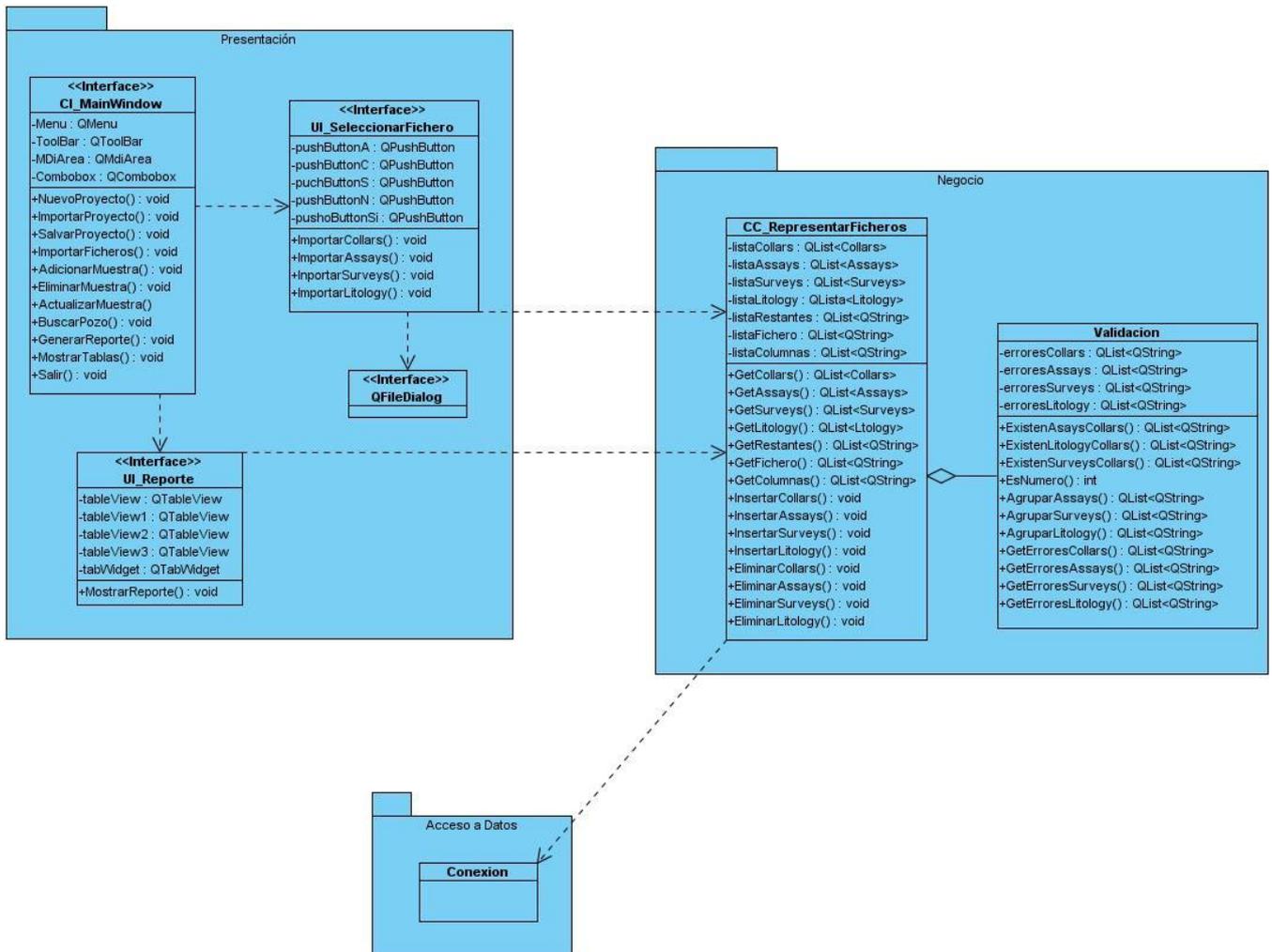


Ilustración 7: Diagrama de Clases de Diseño del Caso de Uso Nuevo Proyecto.

4.8. Diagrama de Colaboración

Un Diagrama de Colaboración (**Anexo 3**) muestra una interacción organizada basándose en los objetos que toman parte en la interacción y los enlaces entre los mismos (en cuanto a la interacción se refiere). Los Diagramas de Colaboración muestran las relaciones entre los roles de los objetos. La secuencia de los mensajes y los flujos de ejecución concurrentes deben determinarse explícitamente mediante números de secuencia.

En cuanto a la representación, un Diagrama de Colaboración muestra a una serie de objetos con los enlaces entre los mismos, y con los mensajes que se intercambian dichos objetos. Los mensajes son flechas que van junto al enlace por el que “circulan”, y con el nombre del mensaje y los parámetros (si los tiene) entre paréntesis. Cada mensaje lleva un número de secuencia que denota cuál es el

Capítulo 4. Construcción de la solución propuesta.

mensaje que le precede, excepto el mensaje que inicia el diagrama, que no lleva número de secuencia. Se pueden indicar alternativas con condiciones entre corchetes. También se puede mostrar el anidamiento de mensajes con números de secuencia como 2.1, que significa que el mensaje con número de secuencia 2 no acaba de ejecutarse hasta que no se han ejecutado todos los 2. (Jacobson, y otros, 2000)

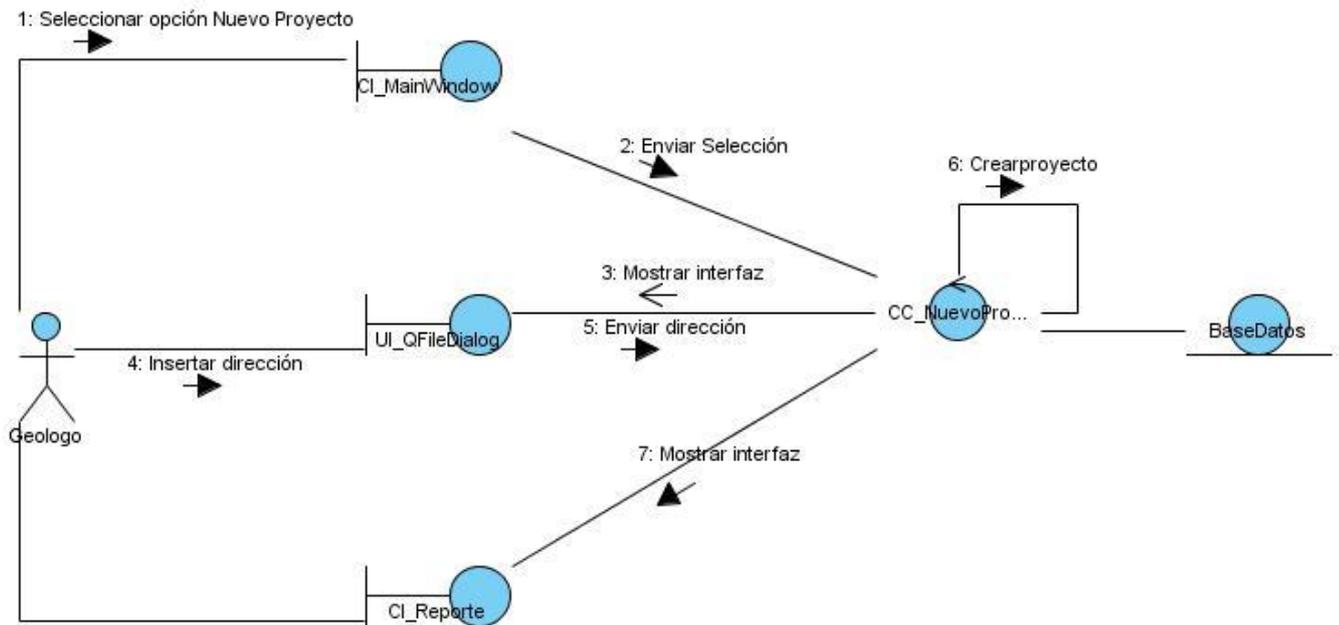


Ilustración 8: Diagrama de colaboración Nuevo proyecto

4.9. Diagrama de despliegue

En un diagrama de despliegue se muestra la distribución física de los objetos, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software (procesos y objetos que se ejecutan en ellos). A continuación se muestra el diagrama de despliegue de la solución propuesta:

Capítulo 4. Construcción de la solución propuesta.

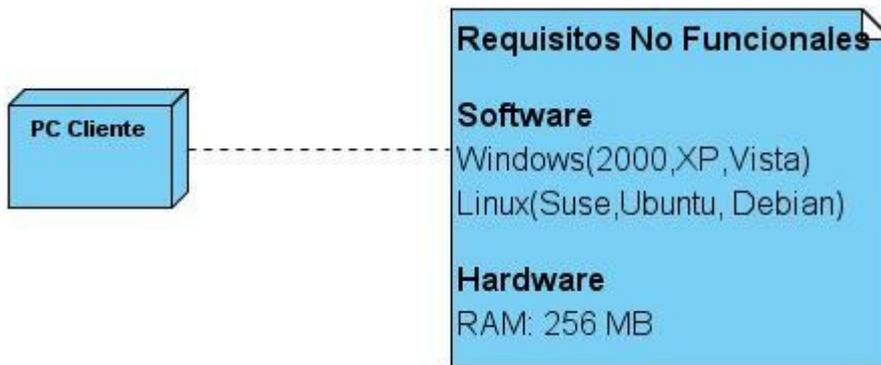


Ilustración 9: Diagrama de despliegue.

4.10. Modelo de Implementación

Un modelo de implementación consiste en una visión general de lo que tiene que ser implementado, y un apartado para cada iteración con los componentes y subsistemas a implementar durante esa iteración, así como de los resultados software que se han de obtener y el testeo que se ha de realizar sobre ellos. (Larman, 1999)

En el modelo de implementación se empieza con los resultados obtenidos en el diseño y se crean los componentes físicos de la aplicación que se traducen en ficheros de código fuente .cpp y .h debido a su implementación en el lenguaje C++.

4.10.1. Diagrama de Componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes, bibliotecas cargadas dinámicamente.

Un componente es la implementación física de un conjunto de otros elementos lógicos, como clases y colaboraciones. La relación que se establece entre componentes en un diagrama de componentes es de dependencia. Esto quiere decir que un componente necesita del otro para completar su definición. (Larman, 1999)

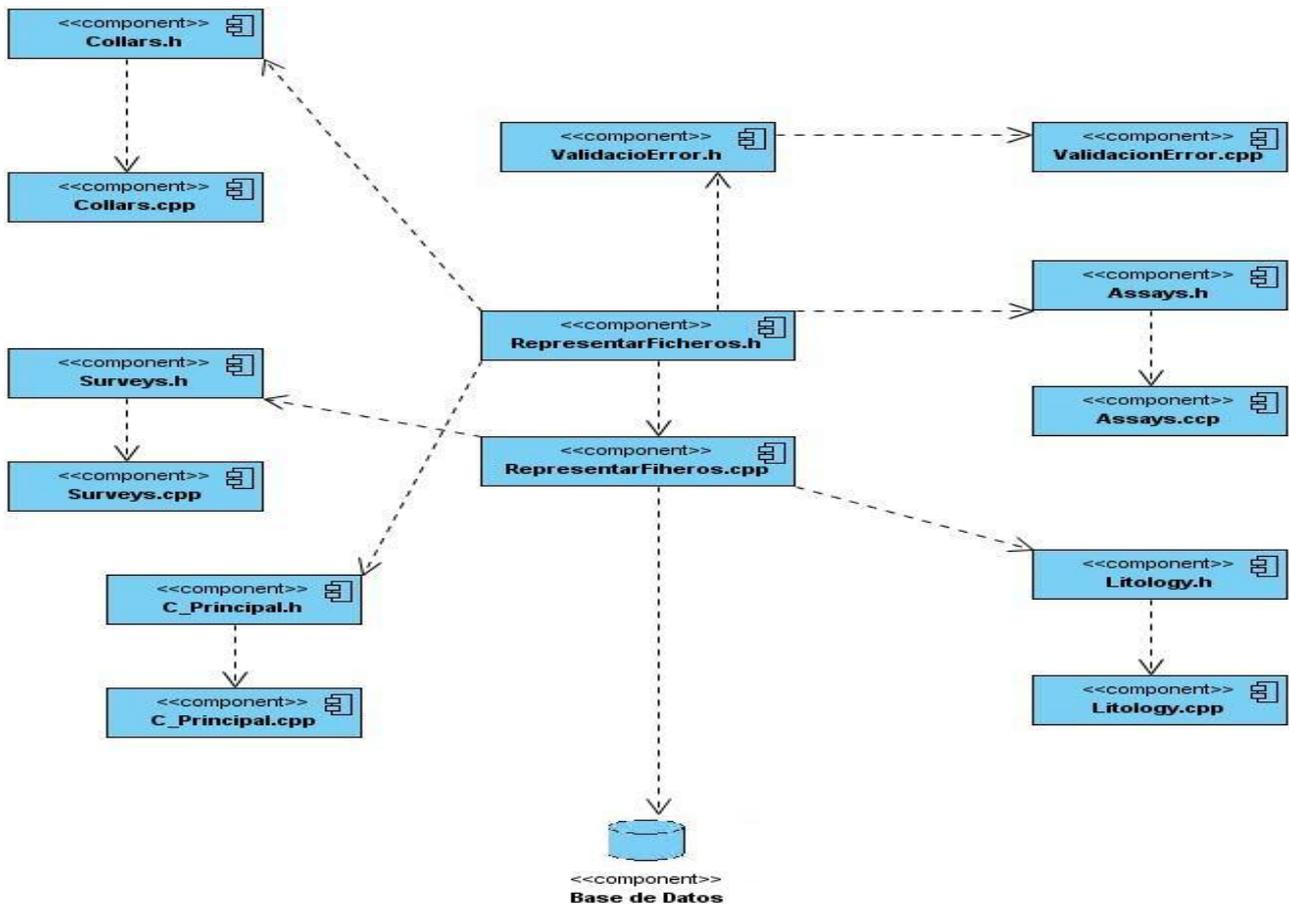


Ilustración 10: Diagrama de Componentes.

4.11 Pruebas a la herramienta propuesta

Las pruebas, por su gran importancia se llevan a cabo durante todo el ciclo de vida del producto, su mayor punto de desarrollo se encuentra en la etapa de implementación.

Dentro de las pruebas se destacan principalmente dos tipos de pruebas:

- Caja Negra.
- Caja Blanca.

4.11.1 Pruebas de caja negra:

Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información

Capítulo 4. Construcción de la solución propuesta.

externa se mantiene. Esta prueba examina algunos aspectos del modelo, fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software.

Se centran principalmente en los requisitos funcionales del *software*. Estas pruebas permiten obtener un conjunto de condiciones de entrada, que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos. (Colectivo profesores, 2010-2011)

A través de estas pruebas se pueden definir:

- ✓ Funciones incorrectas o ausentes.
- ✓ Errores de interfaz.
- ✓ Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- ✓ Errores de rendimiento.
- ✓ Errores de inicialización y terminación.

Técnicas de prueba de Caja Negra.

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están:

- ✓ Técnica de la Partición de Equivalencia: Esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del *software*.
- ✓ Técnica del Análisis de Valores Límites: Esta Técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- ✓ Técnica de Grafos de Causa-Efecto: Es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

- **Partición equivalente:**

Es un 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.

- **Análisis de valores límite:**

Los errores tienden a darse más en los límites del campo de entrada que en el centro. Por ello, se ha desarrollado el análisis de valores límites como técnica de prueba. En lugar de seleccionar cualquier

Capítulo 4. Construcción de la solución propuesta.

elemento de una clase de equivalencia, el análisis de valores límite lleva a la elección de casos de prueba en los extremos de la clase. En lugar de centrarse solamente en las condiciones de entrada, el análisis de valores límite obtiene casos de prueba también para el campo de salida.

4.11.2 Prueba de Caja Negra aplicada al requisito funcional Adicionar pozo

Se realizó un caso de prueba para cada requisito implementado. A continuación se muestra el caso de prueba realizado al requisito “Adicionar pozo” que describe cada uno de los escenarios que pueden existir ante las posibles acciones realizadas por el usuario. De manera similar se realizaron los casos de prueba a los requisitos restantes.

- **Condiciones de ejecución:**

Debe de existir un proyecto en ejecución.

Se debe seleccionar menú datos.

Se debe seleccionar adicionar muestra.

Tabla 3.1: DCP Gestionar Información

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Seleccionar Opción.	EC 1.1: Mostrar pestañas.	Se muestran tres pestañas “Adicionar”, “Eliminar” y “Actualizar” para la adición, modificación y eliminación de pozos.
SC 2: Adicionar Pozo.	EC 2.1: Selección de la opción “Adicionar”.	El sistema muestra el formulario con los campos a llenar correspondiente a esta opción: <i>Nombre, x, y, z</i> . Así como un botón “Aceptar” para adicionar el pozo y uno “Cancelar” para cancelar la operación.
	EC 2.2: Introducir datos del pozo correctamente.	El sistema guarda la información en la base de datos y muestra un mensaje para que el usuario sepa que la operación se realizó que con éxito “Pozo adicionado correctamente”.
	EC 2.3: Dejar campos en blanco.	El sistema muestra un mensaje diciéndole al usuario que faltan campos por entrar “Existen Campos en blanco”.

Capítulo 4. Construcción de la solución propuesta.

	EC: 2.4: Existencia del pozo en la base de datos.	El sistema muestra un mensaje informando al usuario sobre la existencia de ese pozo en la base de datos “El pozo existe en la base de datos”.
	EC: 2.5: Cancelar adición.	Si lo campos contienen información el sistema muestra un mensaje de confirmación “¿Está seguro que desea cancelar la operación?”, si los campos están vacios el sistema muestra el mensaje “¿Está seguro que desea salir?”. De manera que si es aceptado se cierra el formulario para adicionar pozos.
SC 3: Modificar pozo.	EC 3.1: Selección de la opción “Modificar”.	El sistema muestra el panel con un formulario que contiene los mismos datos descritos en el EC 2.1 del pozo para que el usuario modifique el/los que desea.
	EC 4.2: El usuario modifica datos correctamente.	El sistema modifica los datos del pozo y muestra el listado de pozos.
	EC 4.3: El usuario modifica el pozo por uno que existe en la base de datos.	El sistema muestra un mensaje informando al usuario sobre la existencia de ese pozo en la base de datos “El pozo existe en la base de datos”.
	EC 4.4: Cancelar la acción.	Si lo campos contienen información el sistema muestra un mensaje de confirmación “¿Está seguro que desea cancelar la operación?”, si los campos están vacios el sistema muestra el mensaje “¿Está seguro que desea salir?”. De manera que si es aceptado se cierra el formulario para adicionar pozos.
SC 4: Eliminar pozo.	EC 4.1: Eliminar datos.	El sistema muestra un mensaje de confirmación “¿Está seguro que desea eliminar el pozo x?”
	EC 4.2: Aceptar.	El sistema elimina el pozo y actualiza la lista de pozos.

Capítulo 4. Construcción de la solución propuesta.

	EC 4.3: Cancelar.	El sistema no elimina el pozo.
--	--------------------------	--------------------------------

SC 1: Seleccionar Opción

Escenario	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Mostrar pestañas.	El sistema muestra en una pantalla tres pestañas: “Adicionar”, “Modificar”, “Eliminar”.		<ul style="list-style-type: none"> Datos Adicionar

SC 2: Adicionar pozo

Escenario	Nombre	X	Y	Z	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Selección de la opción “Adicionar”.	NA	NA	NA	NA	El sistema muestra el formulario con los campos a llenar correspondiente a esta opción: Nombre, x, y, z. Así como un botón “Aceptar” para adicionar el pozo y uno “Cancelar” para cancelar la operación.		<ul style="list-style-type: none"> Datos Adicionar Pozo
Introducir datos del pozo correctamente.	V	V	V	V	El sistema guarda la información en la base de datos y muestra un mensaje para que el usuario sepa que la operación se realizó que con éxito “Pozo adicionado correctamente”.		<ul style="list-style-type: none"> Datos Adicionar Pozo Aceptar
Dejar campos en blanco.	I	V	V	V	El sistema muestra un mensaje diciéndole al usuario que faltan campos por entrar “Existen Campos		<ul style="list-style-type: none"> Datos Adicionar Pozo

Capítulo 4. Construcción de la solución propuesta.

					en blanco”.		<ul style="list-style-type: none"> • Aceptar
	V	I	V	V	El sistema muestra un mensaje diciéndole al usuario que faltan campos por entrar “El pozo se ha adicionado correctamente”.		
	V	V	I	V	El sistema muestra un mensaje diciéndole al usuario que faltan campos por entrar “El pozo se ha adicionado correctamente”.		
	V	V	V	I	El sistema muestra un mensaje diciéndole al usuario que faltan campos por entrar “El pozo se ha adicionado correctamente”.		
Existencia del pozo.	V	V	V	V	El sistema muestra un mensaje informando al usuario sobre la existencia de ese pozo en la base de datos “El pozo existe en la base de datos”.		<ul style="list-style-type: none"> • Datos • Adicionar • Pozo • Aceptar
Cancelar adición.	NA	NA	NA	NA	Si lo campos contienen información el sistema muestra un mensaje de confirmación “¿Está seguro que desea cancelar la operación?”, si los campos están vacios el sistema muestra el mensaje “¿Está seguro que desea salir?”. De manera que si es aceptado se cierra el formulario para		<ul style="list-style-type: none"> • Datos • Adicionar • Pozo • Aceptar

Capítulo 4. Construcción de la solución propuesta.

					adicionar pozos.		
--	--	--	--	--	------------------	--	--

SC 3: Modificar pozo

Escenario	Nombre	X	Y	Z	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Selección de la opción "Adicionar".	NA	NA	NA	NA	El sistema muestra el formulario con los campos a llenar correspondiente a esta opción: Nombre, x, y, z. Así como un botón "Aceptar" para adicionar el pozo y uno "Cancelar" para cancelar la operación.		<ul style="list-style-type: none"> Datos Modificar Pozo
Modificar datos del pozo correctamente.	V	V	V	V	El sistema guarda la información en la base de datos y muestra un mensaje para que el usuario sepa que la operación se realizó que con éxito "Pozo modificado correctamente".		<ul style="list-style-type: none"> Datos Modificar Pozo Aceptar
Dejar campos en blanco o introducir datos del pozo de forma incorrecta.	I	V	V	V	El sistema muestra un mensaje diciéndole al usuario que faltan campos por entrar "El pozo se ha adicionado correctamente".		<ul style="list-style-type: none"> Datos Modificar Pozo Aceptar
	V	I	V	V	El sistema muestra un mensaje diciéndole al usuario que faltan campos por entrar "El pozo se ha adicionado correctamente".		
	V	V	I	V	El sistema muestra un mensaje diciéndole al usuario que faltan campos		

Capítulo 4. Construcción de la solución propuesta.

					por entrar "El pozo se ha adicionado correctamente".		
	V	V	V	I	El sistema muestra un mensaje diciéndole al usuario que faltan campos por entrar "El pozo se ha adicionado correctamente".		
Existencia del pozo.	V	V	V	V	El sistema muestra un mensaje informando al usuario sobre la existencia de ese pozo en la base de datos "El pozo existe en la base de datos".		<ul style="list-style-type: none"> • Datos • Modificar • Pozo • Aceptar
Cancelar adición.	NA	NA	NA	NA	Si los campos contienen información el sistema muestra un mensaje de confirmación "¿Está seguro que desea cancelar la operación?", si los campos están vacíos el sistema muestra el mensaje "¿Está seguro que desea salir?". De manera que si es aceptado se cierra el formulario para adicionar pozos.		<ul style="list-style-type: none"> • Datos • Modificar • Pozo • Aceptar

SC 4: Eliminar pozo

Escenario	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Eliminar datos.	El sistema muestra un mensaje de confirmación "¿Está seguro que desea eliminarlo?".		<ul style="list-style-type: none"> • Datos • Eliminar • Pozo • Eliminar

Capítulo 4. Construcción de la solución propuesta.

Aceptar.	El sistema elimina el pozo y actualiza la lista de estos.		<ul style="list-style-type: none">• Datos• Eliminar• Pozo• Eliminar
Cancelar.	El sistema no elimina el pozo.		<ul style="list-style-type: none">• Datos• Eliminar• Pozo• Eliminar• Cerrar

4.12 Conclusiones

En este capítulo se obtiene como resultado un sistema completamente diseñado y construido bajo tecnologías libres, lo cual permite el ahorro de dinero en el pago de licencias de cada herramienta utilizada. Basados en el estilo de programación por capas se desarrollaron los modelos de clases del análisis y posteriormente del diseño, aprovechando así las muchas ventajas que brinda este importante estilo arquitectónico. Además se determinaron las clases persistentes a través de las cuales se genera el modelo de datos, para confeccionar la base de datos que permita almacenar la información.

Con las pruebas realizadas a cada requisito funcional se pudo determinar la operatividad de la herramienta, definiendo la eficiencia e inconformidades en cada funcionalidad del sistema.

Conclusiones Generales

Conclusiones Generales

Luego de haber estudiado los sistemas de gestión de informaciones mineras y las soluciones existentes, se determinó la importancia que tiene para las empresas mineras de Cuba y por ende para el país, desarrollar una herramienta que cumpla con los requerimientos definidos anteriormente. La misma permitirá tomar decisiones en menos tiempo, además del ahorro de grandes sumas de dinero invertido año tras año en soluciones informáticas propietarias.

El hecho de que la herramienta haya sido desarrollada con tecnología totalmente libre permitió la carencia de posibles conflictos y atrasos causados por problemas de licencia, además esto permite facilitar el código fuente para que otros desarrolladores cubanos actualicen el sistema posteriormente o lo adapten a necesidades propias.

Con los beneficios presentados y datos palpables en cuanto al ahorro de divisas al país, se determinó que el desarrollo de la herramienta es un paso en futuras automatizaciones de procesos relacionados con las aplicaciones informáticas desarrolladas con tecnología libre.

Recomendaciones

Recomendaciones

- ✓ Trabajar en mejoras en cuanto al diseño del sistema, teniendo en cuenta la opinión de los usuarios.
- ✓ Estudiar la posibilidad de incluir nuevas funcionalidades que por falta de tiempo no fue posible tener en cuenta, como por ejemplo brindar la posibilidad de visualizar en tercera dimensión la ubicación de un pozo.

Referencias Bibliográficas

Referencias Bibliográficas

Angel, Miguel y Rojas, Rendón. *Bases Teóricas y filosóficas de la Bibliografía.* 1998. Mexico : UNAM, 1998.

Bonaparte, Ubaldo. 2004. [En línea] 10 de Diciembre de 2004.
<http://www.frt.utn.edu.ar/sistemas/paradig...>

Baylón, Alfonso Romero. *Revista del Instituto de Investigación FIGMMG.* 13, San Marcos : s.n., Vol. 7. 1561-0888.

Canales, Roberto Mora. 2008. Informática Profesional. [En línea] 2008.
<http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=vp>.

Faga, Boberto. 2000. *Como profundizar en el análisis de sus costos para tomar mejores decisiones empresariales.* Buenos Aires : Granica, 2000.

Fingermann, Hilda. 2009. Deconceptos. [En línea] 16 de Octubre de 2009.
<http://deconceptos.com/ciencias-naturales/minerales..>

Florencia. 2009. Definicion ABC. [En línea] 3 de Diciembre de 2009. <http://www.definicionabc.com/>

Florencia. 2009. Definicion ABC. [En línea] 22 de Julio de 2009
<http://www.definicionabc.com/comunicacion/reporte.php>

Ruesta, Carlota Bustelo y Amarilla Iglesias, Raquel. *GESTIÓN DEL CONOCIMIENTO Y GESTIÓN DE LA INFORMACIÓN.* 2001. 34, 2001. 226-230.

Jacobson, Ivar, Grady, Booch y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo de Software.* Madrid : Addison Wesley, 2000. 84-7829-036-2.

Larman, Craig. 1999. *UML y Patrones Introducción al análisis y diseño orientado a objetos.* Mexico : PRENTICE HALL, 1999. 970-17-0261-1.

Pastrana, Santiago. 2007. Geografía. [En línea] 1 de Febrero de 2007.
<http://geografia.laguia2000.com/relieve/la-litologia..>

Referencias Bibliográficas

Pincay, Emily y Vázquez, Betzabé. 2009. Scribd. [En línea] Febrero de 2009. <http://www.scribd.com/doc/23846691/Trabajoooo-Software-Minero>.

Quiroga, Lourdes Aja. 2008. [En línea] 23 de Mayo de 2008. http://bvs.sld.cu/revistas/aci/vol10_5_02/aci04502.htm.

Revuelta, Manuel Bustillo. 2009. Informática aplicada a la minería. [En línea] Diciembre de 2009. <http://informaticaminera.blogspot.com/2008/08/el-mercado-de-los-software-mineros.html>.

Rivera, Javier Fernández. 2008. Aurea. [En línea] 5 de Noviembre de 2008. <http://aurea.es/wp-content/uploads/ficheros-archivos.pdf>.

Stroustrup, Bjarne. 1998. *El lenguaje de programación C++*. Madrid : Addison Wesley, 1998. 84-7829-019-2.

Thompson, Ivan. 2008. PromonegocioS.net. [En línea] Octubre de 2008. <http://www.promonegocios.net/mercadotecnia/que-es-informacion.html>.

Vila, Cristobal. 2000. Etérea. [En línea] 2000. http://www.eteraestudios.com/training_img/intro_3d/intro_3d.htm

Larman, Craig. *UML y Patrones Introducción al análisis y diseño orientado a objetos*. México: PRENTICE HALL, 1999. 970-17-0261-1.

Sobre la disciplina prueba http://eva.uci.cu/file.php/259/Curso_2010-2011/Semana_9/Conferencia_7/Materiales_Basicos/Sobre_la_disciplina_de_Prueba.pdf2010-2011

Anexos

Bibliografía Consultada

Pressman, Roger S. Ingeniería del software. Un enfoque práctico. Sexta Edición. 2005.

Visconti, Marcelo y Austillo, hernan. Fundamentos de Ingeniería de Software. Santa María. [En Línea] <http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf>

Marquina, Ernesto y Parra, José David. 2011. María Eugenia Aravelo's Blog. Categoría de Patrones. [En Línea] <http://arevalomaria.wordpress.com/2011/03/19/categorias-de-patrones/>

Sommerlad, Peter y otros. Pattern-Oriented Software Architecture: A System of Patterns. John Wiley. 1996

Vargas Del Valle, Ricardo J. y Maltés Granados, Juan P. Programación en Capas. San José, Costa Rica. 2009.

Angel, Miguel y Rojas, Rendón. *Bases Teóricas y filosóficas de la Bibliografía.* 1998. Mexico : UNAM, 1998.

Bonaparte, Ubaldo. 2004. [En línea] 10 de Diciembre de 2004. <http://www.frt.utn.edu.ar/sistemas/paradig...>

Baylón, Alfonso Romero. *Revista del Instituto de Investigación FIGMMG.* 13, San Marcos : s.n., Vol. 7. 1561-0888.

Canales, Roberto Mora. 2008. Informática Profesional. [En línea] 2008. <http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=vp>.

Faga, Boberto. 2000. *Como profundizar en el análisis de sus costos para tomar mejores decisiones empresariales.* Buenos Aires : Granica, 2000.

Fingermann, Hilda. 2009. Deconceptos. [En línea] 16 de Octubre de 2009. <http://deconceptos.com/ciencias-naturales/minerales..>

Florencia. 2009. Definicion ABC. [En línea] 3 de Diciembre de 2009. <http://www.definicionabc.com/>

Florencia. 2009. Definicion ABC. [En línea] 22 de Julio de 2009 <http://www.definicionabc.com/comunicacion/reporte.php>

Anexos

Ruesta, Carlota Bustelo y Amarilla Iglesias, Raquel. *GESTIÓN DEL CONOCIMIENTO Y GESTIÓN DE LA INFORMACIÓN.* 2001. 34, 2001. 226-230.

Jacobson, Ivar, Grady, Booch y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo de Software.* Madrid : Addison Wesley, 2000. 84-7829-036-2.

Larman, Craig. 1999. *UML y Patrones Introducción al análisis y diseño orientado a objetos.* Mexico : PRENTICE HALL, 1999. 970-17-0261-1.

Pastrana, Santiago. 2007. Geografía. [En línea] 1 de Febrero de 2007. <http://geografia.laguia2000.com/relieve/la-litologia..>

Pincay, Emily y Vázquez, Betzabé. 2009. Scribd. [En línea] Febrero de 2009. <http://www.scribd.com/doc/23846691/Trabajoooo-Software-Minero>.

Quiroga, Lourdes Aja. 2008. [En línea] 23 de Mayo de 2008. http://bvs.sld.cu/revistas/aci/vol10_5_02/aci04502.htm.

Revuelta, Manuel Bustillo. 2009. Informática aplicada a la minería. [En línea] Diciembre de 2009. <http://informaticaminera.blogspot.com/2008/08/el-mercado-de-los-software-mineros.html>.

Rivera, Javier Fernández. 2008. Aurea. [En línea] 5 de Noviembre de 2008. <http://aurea.es/wp-content/uploads/ficheros-archivos.pdf..>

Stroustrup, Bjarne. 1998. *El lenguaje de programación C++.* Madrid : Addison Wesley, 1998. 84-7829-019-2.

Thompson, Ivan. 2008. PromonegocioS.net. [En línea] Octubre de 2008. <http://www.promonegocios.net/mercadotecnia/que-es-informacion.html>.

Vila, Cristobal. 2000. Etérea. [En línea] 2000. http://www.eteraestudios.com/training_img/intro_3d/intro_3d.htm

Larman, Craig. *UML y Patrones Introducción al análisis y diseño orientado a objetos.* México: PRENTICE HALL, 1999. 970-17-0261-1.

Sobre la disciplina prueba http://eva.uci.cu/file.php/259/Curso_2010-2011/Semana_9/Conferencia_7/Materiales_Basicos/Sobre_la_disciplina_de_Prueba.pdf2010-201

Glosario

Glosario

Hardware: Conjuntos de componentes que integran la parte material de una computadora. Componentes físicos de una tecnología.

Requisitos: Condición o cualidad que debe cumplir algo o alguien.

Software: Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.

RAM: memoria de acceso aleatorio (en inglés: random - access memory) es la memoria desde donde el procesador recibe las instrucciones y guarda los resultados.

Código ASCII: siglas en inglés para American Standard Code for Information Interchange, es decir Código Americano (estadounidense) Estándar para el intercambio de Información). (Se pronuncia Aski).

Clase: es una construcción que se utiliza como un modelo (o plantilla) para crear objetos de ese tipo.

Objeto: es la unidad que en tiempo de ejecución realiza las tareas de un programa. También a un nivel más básico se define como la instancia de una clase.

C#: es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft.

Java: es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems.

Multiplataforma: es un término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas.

QML: lenguaje para el diseño de la interfaz centrada en las aplicaciones de usuario.

ECMAScript: es una especificación de lenguaje de programación publicada por ECMA International.

.NET: es un framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.

Interfaz: es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo.