Instituto Superior Minero Metalúrgico de Moa

"Dr. Antonio Núñez Jiménez"

Facultad Geología y Minas

Departamento de Informática

TRABAJO DE DIPLOMA

En opción al Título de Ingeniería Informática

Automatización de Metodologías Geomecánicas de Clasificación de Macizos Rocosos.

Autora: Yulidelmis Rodríguez Jorge.

Tutores: MsC. Roilber Lambert Sánchez.

Dr. Armando Cuesta Recio.

Moa, 2012

DECLARACIÓN DE AUTORÍA

Yo, Yulidelmis Rodríguez Jorge declaro que soy la única autora de este trabajo y autorizo al Instituto Superior Minero Metalúrgico de Moa "Dr. Antonio Núñez Jiménez" y al departamento de Minas para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmamos la presente a los _____ días del mes de _____ del 2012.

Yulidelmis Rodríguez Jorge

_____

Firma del autor.

Msc. Roilber Lamber Sánchez

_____

Firma del tutor.

Dr. Armando Cuesta Recio.

_____

Firma del tutor.

Para triunfar en la vida, no es importante llegar primero. Para triunfar simplemente hay que llegar, levantándose cada vez que se cae en el camino.

Rabindranath Tagore

Dedicatoria

A mi Madre y mi Padre

Por su amor sin límites, confianza y apoyo, por creer siempre en mi y haberme dado la vida. Ningún sueño es posible sin ustedes.

Agradecimientos

Cada persona que ha estado a mi lado en este camino se merece un espacio en esta página, pero más que eso, tienen un espacio en mi corazón y siempre lo tendrán:

A mi hermana Yusy, por existir, por ser tan especial y porque su amor me da fuerzas para no rendirme.

A Mami, por ser mi ejemplo a seguir siempre, por su perseverancia, su dedicación, por enseñarme a ser la mujer que soy.

A Papi, por su apoyo, por su esfuerzo, por haberme educado en el camino de la verdad y el amor.

A mi esposo Yoander, por su amor sincero, por ser mi protector, guía y amigo, por su incondicionalidad y por hacerme tan feliz.

A mi tutor Roilber, por haber confiado en mí, por su paciencia, apoyo y dedicación.

A Mayi, Zenia, Rosy, porque han formado parte de cada momento de mi vida, los buenos y los malos, porque sé que siempre podré contar con ustedes.

A mis amigas Dayi, Yenia, Keily, Nela, Yanelis, por haber llegado a mi vida para quedarse, por tantos momentos inolvidables, por las risas y el llanto, por su ayuda, su cariño y porqué sé que aunque sea en la distancia, nuestra amistad será para siempre. Las quiero un mundo y otro.

A mi abuelita Rubí, por mimarme, por cuidarme, por ser mi segunda madre.

A Zoila e Isabel, por su preocupación, por hacerme sentir que nunca estaré sola.

A todos los que quiero y me apoyaron de una forma u otra

Muchas gracias.

Resumen

En este trabajo, se desarrolló un sistema informático que permite clasificar mediante Metodologías Geomecánicas los Macizos Rocosos; como parte de la investigación se sostuvieron encuentros con especialistas de esta disciplina, donde se debatieron experiencias, problemas y deficiencias existentes en esta temática. Luego de un análisis efectuado a los sistemas existentes vinculados al campo de acción se determinó como una necesidad, el desarrollo de una aplicación multiplataforma para estos fines, tomando como base los presupuestos teóricos de la minería en Cuba, su desarrollo histórico, su situación actual, y una valoración sobre las diferentes clasificaciones geomecánica. En el desarrollo del software fue utilizado el método de Proyección Estereográfica para la obtención del número de familias de grietas, necesario a la hora de desarrollar el cálculo de las metodologías. Fue utilizado además el lenguaje UML para la construcción de modelos de datos físico y lógico. Por último, el estudio de factibilidad del sistema realizado por el método COCOMO II mediante el modelo Post Arquitectura, arrojó el costo y factibilidad de la aplicación en CUP, determinando que el sistema reporta ahorros considerables.

Abstract

In this work, we developed a computer system that allows Methodologies classify by Geomechanical Rock Mass, as part of the investigation was held meetings with specialists in this discipline, which discussed experiences, problems and existing deficiencies in this area. After an analysis of existing systems linked to the field of action was determined as a necessity, an application development multiplatform for this purpose, based on the theoretical assumptions of mining in Cuba, its historical development, current situation, and an evaluation of the different classifications geomechanics. In developing the software was used stereographic projection method to obtain the number of families of cracks, when necessary to develop the calculation methodologies. It was also UML language for modeling physical and logical data. Finally, the system feasibility study performed by the method using the model COCOMO II Post Architecture, showed the cost and feasibility of the CUP application, determining that the system reports considerable savings.

Índice

Índice de Tablas

Anexo # 3: Descripciones de Tablas que Conforman Modelo Físico de Datos

Índice de Figuras

Introducción

El mundo de hoy se caracteriza por el desarrollo acelerado de las tecnologías. Ningún centro, empresa o negocio podrá tener éxito si no es capaz de adaptarse a los diferentes cambios. En tal sentido se aprovechan los beneficios de esta nueva era, en la que la informática rige cada vez más el proceso de desarrollo y evolución del ser humano. Además, permite automatizar cada uno de los procesos donde el hombre se ve involucrado, simplificándole con esto, disímiles gastos económicos y sociales.

El avance tecnológico en informática ha mejorado y sigue desarrollándose aun más desde su aparición. Ha llegado incluso a que los software[1] informáticos aumenten sus prestaciones, su accesibilidad de forma vertiginosa y que cada día sea mayor la cantidad herramientas informáticas, capaces de solucionar problemas relacionados con diversas ramas de la ciencia. Dentro de ella la minería, quien se define como la obtención selectiva de los minerales y otros materiales de la corteza terrestre. La misma a su vez, ocupa un lugar esencial para el desarrollo de cualquier país que cuente con reservas minerales significativas. Para la realización de procesos mineros existen metodologías geomecánicas de cálculo cuya automatización facilitaría las excavaciones subterráneas y operaciones a cielo abierto a partir de la clasificación de los macizos rocosos.

Cuba, en los últimos años, ha dado pasos positivos en torno al perfeccionamiento de la actividad minera. Los trabajos relacionados con la disminución del impacto ambiental ocasionado por la minería, el ahorro de insumos energéticos y recursos no renovables utilizados por las entidades mineras, forman parte de estrategias establecidas para alcanzar la explotación sostenible de los recursos minerales. Aún así, la actividad de la minería en Cuba no escapa a los efectos que provoca el bloqueo económico. Debido a esa injusta medida, el país no puede adquirir equipos y software de tecnología de punta en el extranjero, porque sólo son fabricados por compañías norteamericanas.

El Instituto Superior Minero Metalúrgico de Moa (ISMMM) no está exento a esta realidad. Siendo una institución formadora de mineros, no cuenta con un software

[1] Programa.

propio para la Automatización de las Metodologías Geomecánicas de Clasificación de los Macizos Rocosos capaz de facilitar el trabajo de los especialistas de esta rama de la economía. A través de la utilización de esta nueva herramienta se evitan los cálculos manuales, los cuales interfieren con el tiempo de realización de los proyectos, así como el costo y la calidad de los mismos. De esta forma se le proporciona la oportunidad de obtener resultados certeros y realizar análisis profundos sobre los datos que se obtengan.

En correspondencia con lo anteriormente expuesto se plantea como **Problema Científico:**

"La necesidad de una aplicación multiplataforma para la Automatización de Metodologías Geomecánicas de Clasificación de Macizos Rocosos para el Departamento de Minas del ISMMM."

Este problema se enmarca en el **Objeto de estudio:** Sistemas Informáticos para la Clasificación de Macizos Rocosos mediante las Metodologías Geomecánicas.

El **Objetivo general** de esta investigación es elaborar una aplicación multiplataforma para la Automatización de Metodologías Geomecánicas de Clasificación de Macizos Rocosos. Específicamente se investigará en el siguiente **campo de acción:** Proceso de Automatización de Metodologías Geomecánicas de Clasificación de Macizos Rocosos en el Instituto Superior Minero Metalúrgico de Moa "Dr. Antonio Núñez Jiménez".

Objetivos específicos:

1. Establecer el estado del arte sobre la información disponible referida al objeto de estudio.

2. Seleccionar las herramientas a utilizar para el diseño e implementación del sistema.

3. Desarrollar un software multiplataforma para la Automatización de Metodologías Geomecánicas de Clasificación de Macizos Rocosos.

4. Implantar el sistema para la Automatización de Metodologías Geomecánicas de Clasificación de Macizos Rocosos.

En la presente indagación científica se perfila la siguiente **idea a defender:** con la realización de un sistema multiplataforma para la Automatización de Metodologías Geomecánicas de Clasificación de Macizos Rocosos se logrará una mejor gestión de los datos obtenidos y por consiguiente el desarrollo de las excavaciones subterráneas tendrá un menor coste y mayor calidad.

Con el propósito de darle cumplimiento a los objetivos trazados se realizaron **tareas de investigación** que se relacionan a continuación**:**

- Definir detalladamente del proceso a automatizar.
- Hacer un estudio de viabilidad de la automatización del proceso.
- Realizar búsquedas y recopilación de información concerniente al objeto de estudio.
- Investigar sobre el método de proyección estereográfica para el cálculo del número de familia de grietas.
- Recopilar información sobre las metodologías de ingeniería de software.
- Seleccionar las herramientas adecuadas para el desarrollo de la aplicación.
- Realizar la ingeniería de software siguiendo la metodología seleccionada.
- Desarrollar la aplicación de escritorio utilizando como lenguaje de programación java.
- Realizar pruebas previas para verificar la eficiencia del sistema.
- Determinar la factibilidad y sostenibilidad del sistema.

Para darle cumplimiento a estas tareas los **métodos de investigación** empleados fueron los **teóricos** y **empíricos**. Entre los **métodos empíricos** se pueden citar la entrevista y el análisis de documentos para la recopilación de la información. Mediante el análisis de la documentación y del proceso se supo cual era la funcionalidad del mismo para su posterior gestión; la entrevista posibilitó conocer en detalles cuales eran las necesidades de los directivos para la informatización de la gestión de solicitudes de servicios informáticos y además para determinar los requerimientos funcionales que debe cumplir el sistema a desarrollar.

Los **métodos teóricos** utilizados fueron: el método histórico y lógico para la búsqueda de los antecedentes del software. El análisis y síntesis para la recopilación y el

procesamiento de la información obtenida en los métodos empíricos y arribar a las conclusiones de la investigación; y el hipotético deductivo para la elaboración de la idea a defender y su verificación. Mediante la modelación se realizó el estudio de la Automatización de Metodologías Geomecánicas de Clasificación de Macizos Rocosos.

Las etapas de la investigación son:

- El estudio de las distintas metodologías de desarrollo de software, de las herramientas, tecnologías y patrones arquitectónicos para hacer una selección de las mejores, y utilizarlos en la confección del software de acuerdo a sus características.
- La contextualización del problema, enmarcando los objetivos específicos requeridos por el cliente.
- Análisis y diseño del software con todas las especificaciones requeridas.
- Realización del estudio de factibilidad.

Capítulo I: “Fundamentación Teórica”, en este capítulo se hace una reseña histórica del surgimiento de la minería en Cuba y su desarrollo hasta nuestros días. Se realiza una valoración sobre las diferentes clasificaciones geomecánicas. También se realiza un estudio de las tendencias y tecnologías actuales que se emplearán para el desarrollo de la aplicación, fundamentando el uso de cada herramienta escogida.

Capítulo II: “Modelo de Dominio”, en este se aborda la descripción de un modelo del dominio, explicando las razones por la que se escogió. Se definen los requisitos funcionales y no funcionales, las entidades y conceptos principales y se especifican las funcionalidades que va a concebir el sistema.

Capítulo III: “Diseño e Implementación”, es aquí donde se definen los actores, se realizan las descripciones de los casos de uso del sistema, se muestra el diagrama de clases del diseño, es definida la arquitectura que se emplea. Se construyen los principales diagramas empleados para el desarrollo del software. Se muestra el Modelo lógico de datos que genera el Modelo físico, se detallan, además, las principales tablas que conforman este último. Por último, se precisan los principios del diseño y se explica, cómo el sistema afrontará el tratamiento de errores.

Capítulo IV: “Estudio de Factibilidad”, se realiza el estudio de factibilidad del sistema por el método COCOMO II mediante el modelo Post Arquitectura, se presentan demás los beneficios tangibles e intangibles obtenidos con la implantación de la aplicación y se realiza un análisis de costos y beneficios para expresar la factibilidad.

Capítulo I. Fundamentación Teórica

1.1- Introducción

En el presente capítulo se describen los principales aspectos y conceptos de relevancia que han sido objeto de análisis a lo largo de la investigación. Se realiza un acercamiento al estado en el que se encuentra actualmente la Automatización de Metodologías Geomecánicas de Macizos Rocosos a diferentes escalas, así como las principales tendencias, plataformas de desarrollo, tecnologías, metodologías y herramientas que hicieron posible la realización del presente trabajo.

1.2- La Minería en Cuba

La minería no llegó a constituir una actividad de primera importancia para la economía cubana ni siquiera en la primera mitad del siglo XX. El mayor auge en la producción minera en la etapa anterior a 1959 se alcanzó en los periodos de confrontación bélicas cuando se incentivaba la producción minera en Cuba vinculadas a las guerras mundiales y a la de Corea. Existía desconocimiento del potencial minero del territorio nacional y una participación casi nula de inversionistas cubanos. Al triunfar la Revolución en 1959 es que se toma la decisión de establecer un programa encaminado a precisar y desarrollar el potencial minero del país (NEIRA, 2009).

En 1961 se constituye el Ministerio de Industrias, existiendo solo dos geólogos cubanos, por lo que se requería un importante proceso de preparación de condiciones en ese sentido. El 1975 solo se conocía, desde el punto de vista geológico, el 5% del territorio nacional. Para finales de la década del ochenta se logró elevar el grado de conocimiento del potencial minero desde el 5 hasta el 50%, lo cual se logró con la colaboración de la Antigua Unión Soviética y países del CAME[2]. A partir de la creación del Servicio Geológico Nacional se comenzó un trabajo sistemático encaminado a precisar las características geológicas del país y revelar la presencia de yacimientos de minerales.

[2] Consejo de Ayuda Mutua Económica

Cuba cuenta con las principales reservas de níquel del mundo, lo que ha permitido desarrollar una industria que está representada por tres plantas. En 1943, durante la II Guerra Mundial, una empresa de Estados Unidos construyó en Nicaro, al este de la bahía de Nipe, la primera planta de níquel en la isla. Se trató de un centro industrial moderno, para explotar los yacimientos de Pinares de Mayarí. En 1955, la *Freeport Sulphur Corporation*, inició la construcción en Moa de la segunda planta cubana para la explotación del níquel. Con el triunfo de la Revolución, los técnicos y especialistas estadounidenses abandonaron este combinado, único en el mundo por su forma de operación, y llevaron consigo la documentación sobre la tecnología de esa industria.

Una década más tarde, se reparó la planta de Nicaro y se emprendió la construcción de otra, la Ernesto Che Guevara, en Punta Gorda, con capacidad para producir 30 mil toneladas anuales del metal. La primera fase de esta inversión, concluyó en 1984, y se emprendió la construcción de un cuarto combinado en Las Camariocas, 10 kilómetros al este de Moa, que fue necesario cancelar en la década del 90 tras el derrumbe del campo socialista europeo.

1.2.1- Situación actual de la minería en Cuba

Cuba, aunque no tan desarrollada como otras industrias posee importantes minas, principalmente las de níquel (34,4% de las reservas mundiales), cobalto y cobre, entre otras. Los principales yacimientos de níquel se encuentran en el municipio de Moa, provincia de Holguín y en la provincia de Guantánamo (aunque en menor escala). Este producto de hecho se ha convertido en una importante base económica cubana.

El cobalto es otro mineral extraído en el oriente cubano, aunque también es extraído en provincias como Villa Clara. Cuba cuenta con el 26% de las reservas mundiales (segunda mayor) produce aproximadamente el 10% de este mineral a nivel mundial y la mayor parte la exporta a China. Respecto a este asunto, Cuba firmó acuerdos con Canadá. Al igual que con el níquel, se encuentra cooperando con China y explorando nuevas reservas de este mineral en el norte del oriente cubano. Cuba también produce 400.000 toneladas anuales de acero en las industrias de La Habana y Las Tunas. Por su situación geográfica, Cuba extrae sales marinas del mar Caribe. Ha

hecho de ellas un nuevo producto, que es exportado al mercado internacional y empleado en el consumo.

1.3- Clasificación Geomecánica de las Rocas

Existen diferentes tipos de roca, cada una de las cuales tienen sus propias características y propiedades físicas. Existen también, diferentes situaciones que requieren el uso de fortificación adicional para consolidar los estratos de la roca, afirmar los bloques y prevenir la caída de estas. Previo a la construcción de una labor subterránea, se realiza un estudio preliminar de la geología del terreno mediante sondajes (muestras de perforación diamantina), mapeos geológicos y otros, es físicamente imposible detectar completamente las condiciones en que se encuentran los diversos elementos de un cuerpo tan complicado como es el macizo rocoso.

En la mayoría de los casos, el macizo rocoso aparece como un conjunto ensamblado de bloques irregulares, separados por discontinuidades geológicas como fracturas o fallas y, por ello la caracterización geomecánica de los Macizos Rocosos es compleja; pues debe incluir tanto las propiedades de la matriz rocosa así como de las discontinuidades. En resumen, el diseño de una excavación subterránea, que es una estructura de gran complejidad, es en gran medida el diseño de los sistemas de fortificación. Por lo tanto, el objetivo principal del diseño de los sistemas de refuerzo para las excavaciones subterráneas, es de ayudar al macizo rocoso a soportarse, es decir, básicamente están orientados a controlar la caída de rocas, que es el tipo de inestabilidad que se manifiesta de varias maneras (Avila, 2010).

Controlar los riesgos de accidentes a personas, equipos y pérdidas de materiales (producto de la inestabilidad que presenta una labor durante su abertura), constituye una preocupación primordial que debe ser considerada en la planificación de las labores mineras.

El diseño de sostenimiento de terrenos es un campo especializado, y es fundamentalmente diferente del diseño de otras estructuras civiles. Por lo tanto el procedimiento de diseño para el sostenimiento de terrenos tiene que ser adaptado a cada situación. Las razones son los hechos siguientes:

- Los “materiales utilizados” son altamente variable.
- Hay limitaciones severas en lo que se puede proporcionar la información por medio de investigaciones geológicas.
- Existen limitaciones en exactitud y la importancia de parámetros probados del material de la roca.
- Existen limitaciones severas en el cálculo y los métodos para modelar el sistema de sostenimiento.
- El comportamiento de aberturas es dependiente del tiempo, y también influenciado por los cambios en filtraciones de agua.
- Incompatibilidad entre el tiempo necesario para las pruebas de los parámetros, para los cálculos y modelos, comparados al tiempo disponible.

1.3.1- Metodologías para la clasificación Geomecánica de los Macizos Rocosos

Para el diseño de un túnel primero se debe realizar un estudio geológico – geotécnico del sector donde se lo proyecta, en esto la mecánica de rocas juega un papel fundamental en la clasificación del macizo rocoso e incluso estableciendo un pre diseño con los elementos necesarios para el sostenimiento del túnel en función a la altura de carga (zona de aflojamiento) después de la excavación, con estos datos ya se podría estimar el costo de la obra tunelera lo cual resulta muy útil para poder ver su viabilidad de esta alternativa. En los túneles y taludes rocosos, los mecanismos de inestabilidad son controlados por el grado de alteración y por las anisotropías existentes en el macizo, tales como la estratificación, juntas, fallas, cuya relación con los mecanismos de inestabilización es regida por los siguientes factores:

- Distribución espacial de las discontinuidades, relación entre su posición (rumbo y buzamiento) con la dirección del túnel. Siendo este el más importante a considerarse en el trazo de entrada y salida del túnel.

- ❖ Presencia y naturaleza de los materiales de relleno de las discontinuidades.
- ❖ Irregularidades en las superficies de las discontinuidades.
- ❖ Rotura y movimientos anteriores.

Existen muchos métodos útiles para poder clasificar un macizo rocoso, entre ellos como propuesta del Departamento de Minería del ISMMM se escogieron para ser automatizados dos métodos elaborados por autores conocidos mundialmente en el campo de la mecánica de rocas estos son: *Barton* (1974) y *Bieniawski* (1973, 1989), además del método propuesto por el Dr. Armando Cuesta Recio (2010), docente en el Instituto Superior Minero Metalúrgico:

- Clasificación de la Masa Rocosa, Bieniawski, (*Rock Mass Rating*) **RMR**

Es una clasificación que ha sido utilizada en África del Sur y fue desarrollada principalmente a partir de excavaciones subterráneas mineras. La evaluación de calidad de macizos rocosos *Rock Mass Ratting* (RMR) es realizada mediante la atribución de valores a los cinco parámetros que intervienen:

- ✓ Resistencia a la compresión a la roca alterada, *Bieniawski* emplea la clasificación de la resistencia a la compresión uniaxial de la roca que proponen, *Deere* y *Millar*, como alternativa se podrá utilizar la “Clasificación de carga de punta”, para cualquier tipo de roca, excepto la muy frágil.
- ✓ RQD, índice de calidad de la roca según *Deere* y *Miller*.
- ✓ Espaciamiento de las discontinuidades, es decir de las fallas, planos de estratificación y otros planos de debilidad.
- ✓ Condiciones físicas y geométricas de las discontinuidades, este parámetro toma en cuenta la separación o abertura de las fisuras, su continuidad, la rugosidad de su superficie, el estado de las paredes (duras o blandas), y la presencia de relleno en las discontinuidades.
- ✓ Presencia de agua subterránea, se intenta medir la influencia del flujo de las aguas subterráneas sobre la estabilidad de las excavaciones en función del caudal que penetra en la excavación, y de la relación entre la presión

del agua en las discontinuidades y el esfuerzo principal. *Bieniawski* reconoció que cada parámetro no contribuye necesariamente de igual manera al comportamiento del macizo. Por ejemplo un RQD de 90 y una resistencia a la compresión uniaxial de 2000 Kg/cm2 parecerían indicar una roca de calidad excelente, pero una infiltración grande en esa misma roca puede cambiar radicalmente esta opinión.

- Índice de la Calidad del Túnel, Barton, (*Tunnel Quality Index*) **Q**

Basándose en una gran cantidad de casos tipo de estabilidad en excavaciones subterráneas, el *Norgerian Geotechnical Institute* (N.G.I.), propuso un índice para determinar la calidad del macizo rocoso en túneles y taludes. El valor numérico de éste índice Q se define por:

Q = (RQD / JN) (JR / JA) (Jw / SRF)

Esta clasificación utiliza seis parámetros para definir la clase de macizo:

1. RQD, índice de calidad de la roca.
2. Jn, índice del número de familias de fracturas
3. Jr, índice de rugosidades en las fracturas
4. Ja, índice de alteración de las paredes de las fracturas
5. Jw, índice del caudal afluente
6. SRF, índice del estado de tensión del macizo

PARAMETRO	DESCRIPCIÓN	VALOR	OBSERVACIONES
RQD	A.- Muy mal	0 a 25	Cuando RQD se reporta o es medido como menor a 10, se lo otorga un valor nominal de 10.
	B.- Mala	25 a 50	
	C.- Regular	50 a 75	
	D.- Buena	75 a 90	
	E.- Excelente	90 a 100	

Figura 1.1. Valoración de los parámetros Q (según Barton)

- (Método para cuando el agua afecta una excavación). Armando Cuesta Recio (2011)

En la identificación en una excavación subterránea de las áreas que requieren tratamiento para controlar las filtraciones, se debe realizar un estudio integral en toda la traza de la misma y zona de influencia de los principales aspectos que inciden en los procesos de circulación de agua y su control. Con el propósito de racionalizar el estudio y realizar un análisis la susceptibilidad de la excavación a inestabilidad e inundación por la presencia de agua, en la primera etapa del procedimiento se identificarán las áreas susceptibles por la acción de agua. El procedimiento se divide en cuatro etapas:

A: - Zonificación en la traza de la excavación de áreas de susceptibilidad por la presencia de agua.

B: - Caracterización de las zonas susceptibles por las presencia de agua en el área de influencia de la excavación subterráneas.

C: - Selección de la técnica para el control de las filtraciones de agua a las excavaciones subterráneas.

D: - Elección de la técnica o sistema más adecuada para controlar el agua.

1.3.2- Método de la Proyección Estereográfica

La proyección estereográfica es un sistema de representación gráfico en el cual se proyecta la superficie de una esfera sobre un plano mediante haces de rectas que pasan por un punto, o foco. El plano de proyección es tangente a la esfera, o paralelo a éste, y el foco es el punto de la esfera diametralmente opuesto al punto de tangencia del plano con la esfera. La superficie que puede representar es mayor que un hemisferio. El rasgo más característico es que la escala aumenta a medida que nos alejamos del centro. En su proyección polar los meridianos son líneas rectas, y los paralelos son círculos concéntricos. En la proyección ecuatorial sólo son líneas rectas el ecuador y el meridiano central (Hürlimann, 2010).

Para una mejor comprensión de este método se muestra un gráfico a continuación con la representación en un plano de la Proyección Estereográfica.

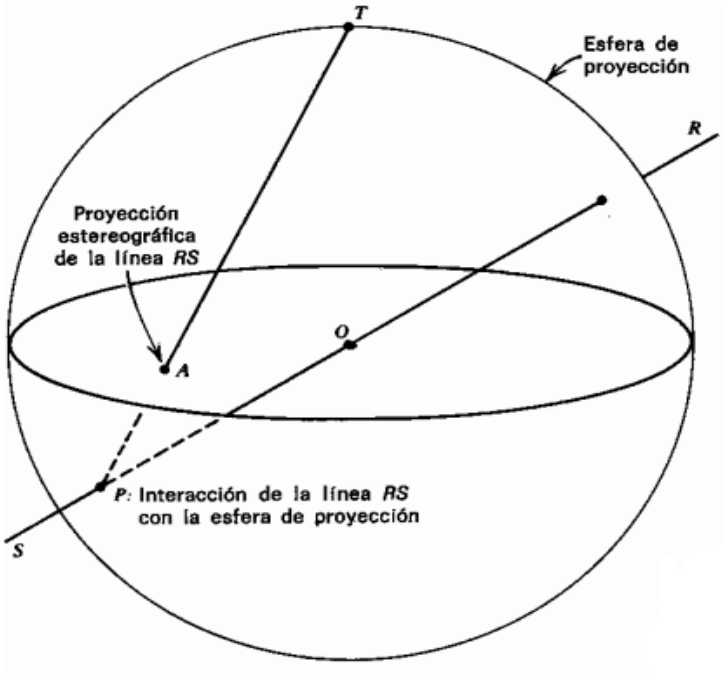

Figura 1.2. Método de proyección estereográfica

Se construye una esfera centrada en algún punto “O” de la traza de afloramiento de un plano geológico inclinado. El plano y su prolongación cortarán la esfera según un círculo máximo o círculo mayor. Este método será utilizado en el sistema para el cálculo del número de familia de grietas, dato necesario a la hora de calcular alguna de las metodologías automatizada en el sistema.

1.4- Antecedentes

El software “ContExt” fue creado para el control de excavaciones. Este permite el análisis de la susceptibilidad de la excavación a inestabilidad e inundación por la presencia de agua, además de la selección de la técnica para el control de las filtraciones de agua a las excavaciones subterráneas.

Inconvenientes del Software “ContExt”:

1. No es un software multiplataforma.
2. Utiliza ficheros para el almacenamiento de los datos.

3. No cuenta con una ayuda explícita para el trabajo con el software.
4. No permite el cálculo del número de familia de grietas, necesario para la Clasificación Geomecánica.

1.4.2- Sistemas automatizados existentes vinculados al campo de acción

1.4.2.1- Datastream

Los sistemas EAM (Software para la Gestión de Activos Corporativos) fueron desarrollados originalmente para solucionar los requerimientos de industrias extremadamente intensivas en capital (Assets), como es la minería. El mantenimiento es un área importante a considerar en toda industria, ya que representa un porcentaje considerable de los costos, como por ejemplo, un 30% a 35% en la minería. Datastream 7i es una solución para la gestión de activos que ayuda a las empresas a incrementar el retorno sobre la inversión (ROI) durante todo su ciclo de vida. Posee una arquitectura 100% web, permitiendo su acceso por medio de un browser standard de Internet desde cualquier lugar en cualquier momento, contribuyendo a minimizar el tráfico por la red y evitando la compra de hardware costoso.

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

La tecnología Mobile, incluida en la solución Datastream 7i, permite a las empresas mineras tener un acceso remoto y sin limitaciones a todos los datos vitales para administrar el ciclo de vida de sus activos, optimizando el intercambio de información y el flujo de trabajo, lo que constituye un factor clave para mejorar su eficiencia operacional e incrementar las ganancias y retorno de la inversión ROI.

1.4.2.2- Datamine

Sin considerar algunas mínimas diferencias existentes entre algunos sectores productivos mineros, que se rigen por condiciones muy particulares, todo negocio productivo asociado al área minera está constituido por las siguientes áreas: Proceso productivo-Logística de apoyo al proceso productivo-Finanzas. Hoy en día los grandes sistemas informáticos están siendo sometidos al mayor y más complejo de los requerimientos, que proviene directamente de los profesionales del área minera: soluciones efectivas a todas sus necesidades.

El análisis del círculo de planeamiento minero y la solución global propuesta por Datamine comenzó con un riguroso análisis del mercado minero, que luego se fundamentó en la mejora continua de toda la línea de productos y en el desarrollo de todas aquellas tecnologías que permitieran cubrir todas las áreas del negocio minero, en cualquiera de sus variantes. De esta forma, las alternativas propuestas por la compañía para suministrar soluciones efectivas a estos exigentes requerimientos cuentan con todas las características fundamentales para asegurar el éxito del negocio minero, tales como integralidad, Versatilidad, Potencialidad y Confiabilidad, permitiendo un trabajo rápido, eficiente, que considera el riesgo asociado, generando resultados auditables y repetibles a lo largo del tiempo.

1.4.2.3- 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 *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.

1.4.2.4- Dips

Este software está diseñado para el análisis interactivo de orientaciones geológicas. Es capaz de proporcionar al usuario mediante el método de la proyección estereográfica las ubicaciones de los polos en un plano. Permite además analizar y visualizar datos estructurales siguiendo las mismas técnicas usadas en estereoscopios manuales. Además, tiene muchas características computacionales, tales como contorno de estadística de la agrupación de orientación, el cálculo del promedio de orientación y la confianza, la variabilidad del clúster, y el análisis de atributo de rasgo cualitativo y cuantitativo.

1.4.2.5- 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.

Características de estos sistemas:

1. Son reconocidos mundialmente dentro del entorno minero.

2. Son software propietario y su adquisición es costosa.

3. Empleados para proyectos mineros de alto nivel.

4. Manejan grandes cantidades de datos, más de los necesarios para la automatización de metodologías geomecánicas de macizos rocosos.

Por tales razones es preciso desarrollar un software multiplataforma para la Automatización de Metodologías Geomecánicas de Clasificación de Macizos Rocosos, capaz de proporcionar un resultado certero sin necesidad de cargar datos innecesarios para el usuario, haciéndolo de esta forma más óptimo y contribuir además con la economía del país.

1.5- Herramientas para el desarrollo de aplicaciones

1.5.1- Lenguajes de programación

Un lenguaje de programación es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana (Mark, 2010).

Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

En el transcurrir de los años y en la medida en que la tecnología ha ido avanzado, han venido surgiendo diferentes lenguajes de programación, cada uno con características y objetivos específicos distintos pero todos con la misma finalidad, la comunicación hombre-máquina a través de una estructura sintáctica similar al lenguaje común utilizado en la vida diaria, como por ejemplo la utilización de métodos numéricos para resolver problemas.

Dentro de los lenguajes de programación más utilizados para el desarrollo de aplicaciones de escritorio están C, C++, C#, Java, Visual Basic, entre otros.

1.5.2- Java

El lenguaje de programación Java, fue diseñado por la compañía Sun Microsystems Inc, con el propósito de crear un lenguaje que pudiera funcionar en redes computacionales heterogéneas (redes de computadoras formadas por más de un tipo de computadora, ya sean PC, MAC's, estaciones de trabajo, etc.), y que fuera independiente de la plataforma en la que se vaya a ejecutar. Esto significa que un programa de Java puede ejecutarse en cualquier máquina o plataforma.

La sintaxis del lenguaje heredó características de C y C++, adoptando una muy similar a la del C++. Actualmente, dentro de los lenguajes populares es uno de los mejores en cuanto a definición, debido a que goza de total independencia del implementador del lenguaje y de sus clases auxiliares. Proporciona los tipos de datos primitivos similares a los de C++, proporciona todas las estructuras contenedoras "clásicas". Tiene cuatro niveles de empaquetamiento: variables y funciones, al igual que los lenguajes anteriores y otros dos propios de él, denominados: clases y paquetes.

Este lenguaje cuenta con una interfaz orientada a objetos para acceder de un modo portable a cualquier base de datos, promoviendo la portabilidad. Es poseedor de una extensa biblioteca utilitaria y la portabilidad alcanzada es cualitativamente superior a la que se puede obtener con los lenguajes C/C++.

Desde sus inicios se concibió como un lenguaje muy fácil de comprender y utilizar, sin que esto signifique que su aprendizaje sea trivial.

1.5.3- C++

C++ es un lenguaje de programación orientado a objetos que toma la base del lenguaje C y le agrega la capacidad de abstraer tipos como en Smalltalk. Proporciona soporte a clases, objetos, herencia, es decir, todo lo característico de un lenguaje orientado a objetos. Estas cualidades facilitan la reutilización de código, consiguiendo con ello un ahorro de trabajo y tiempo. Una de sus principales características está dada en el soporte de plantillas o programación genérica (templates). Posee un entorno de desarrollo simple, flexible y potente al mismo tiempo, especializado en la creación de proyectos Windows. Tiene un compilador rápido y brinda la posibilidad de obtener un ejecutable que puede ser utilizado aún cuando el programa no se encuentre instalado en la computadora.

1.5.4- Delphi

La plataforma de implementación (IDE) Delphi está diseñada para la programación de alto nivel y de propósito general. Tiene como basamento el lenguaje Pascal. A pesar de contar con un editor de HTML, XML y Servicios y Aplicaciones Web, su principal uso está dado por la realización de aplicaciones de escritorio. Es un software propietario. Sirve de interfaz para generar entornos gráficos, aplicaciones multimedia, Internet, gestión de datos, de sistemas, de redes, etcétera. Permite intercambiar, transformar y manipular documentos y datos XML.

En la construcción de aplicaciones nativas Windows, Delphi permite una fácil creación y reutilización de DLLs, controles COM, así como el desarrollo de licencias comerciales para ventas de software profesional. Construye y utiliza Windows COM, COM+, ActiveX y automatización de objetos.

1.5.5- C

El lenguaje C es creado entre los años 1972-1973 por un equipo de programadores de los antiguos Laboratorios Bell de AT&T, como un lenguaje de propósito general. Fue considerado por mucho tiempo un buen ejemplo de un lenguaje consistente y sin

ambigüedades notorias, especialmente entre otros de su misma época. Sin embargo sus propios creadores reconocen en él ciertos inconvenientes en la notación, que aunque no es estrictamente un error y puede ser evitable, originan confusiones y uno de los principales problemas son los aspectos que son dejados a criterio del implementador, por ejemplo el tamaño de los tipos de datos, lo que puede convertirse en problemas de portabilidad. Sin embargo fue uno de los primeros "casos de éxito" de portabilidad gracias a estándares abiertos.

El único aspecto cuestionable radica en que muy pocas de estas librerías están estandarizadas del mismo modo que el lenguaje. Por ejemplo, el estándar de lenguaje C actualmente incluye una librería (conocida como la "librería estándar") que permite realizar una gran cantidad de tareas esenciales (mostrar un mensaje, grabar en un archivo de disco, calcular funciones trigonométricas, procesar cadenas de caracteres, etc.) pero que para muchas aplicaciones modernas, resulta francamente insuficiente.

1.6- Entornos de desarrollo integrados (IDE)

Un entorno de desarrollo integrado, llamado también IDE (sigla en inglés de *Integrated Development Environment*), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien poder utilizarse para varios.

Un IDE 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 sí solas o pueden ser parte de aplicaciones existentes (Wikipedia, 2009).

1.6.1- NetBeans

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo, es un producto libre y gratuito sin restricciones de uso. NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo.

Sun MicroSystems fundó el proyecto de código abierto en junio de 2000 y continúa siendo el patrocinador principal de los proyectos.

La plataforma permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en esta plataforma pueden ser extendidas fácilmente por otros desarrolladores de software. Por tales características NetBeans cumple con los requisitos necesarios para ser utilizado en el desarrollo de la aplicación.

1.7- Bases de Datos Embebidas

Cuando hablamos de bases de datos, normalmente nos imaginamos un gran servidor, con grandes cantidades de información en su interior. Ahora bien, también existen bases de datos minúsculas para entornos donde el espacio, tanto de almacenaje como de proceso. Una Base de Datos Embebida o Empotrada es aquella que no inicia un servicio en nuestra máquina independiente de la aplicación, pudiéndose enlazar directamente a nuestro código fuente o bien utilizarse en forma de librería.

Normalmente las bases de datos embebidas comparten una serie de características comunes:

- ✓ Su pequeño tamaño**.**
- ✓ El "small footprint" en términos de tamaño de código añadido a nuestra aplicación.
- ✓ Recursos que consumen.

Una base de datos embebida es un ficherito que colocas en la carpeta de tu aplicación y que corre dentro de tu proceso, siendo únicamente accesible por éste. Es decir, sólo la aplicación que lanza la base de datos embebida puede acceder a sus tablas. (Vega, 2008)

Estas bases de datos pueden estar contenidas exclusivamente en memoria (y al cerrar la aplicación vuelcan las tablas a disco) o en disco, siendo en el primer caso mucho más rápidas, como era de esperar. Muchas de estas bases de datos disponen de librerías e interfaces para acceder a ellas desde Java (mediante JDBC) o .NET (en ADO.NET), incluso algunas pueden ejecutarse también en modo cliente-servidor (como un MySQL u Oracle). Dentro de este grupo de bases de datos se encuentran HSQLDB, SQLite, Firebird, Apache Derby entre otras.

1.7.1- Firebird

Firebird es un sistema de administración de base de datos relacional (Lenguaje consultas: SQL) de código abierto, basado en la versión 6 de Interbase, cuyo código fue liberado por Borland en 2000. Su código fue reescrito de C a C++. Los objetivos de la Fundación FirebirdSQL son:

- Apoyar y lograr el avance del manejador de base de datos relacional Firebird.
- Proveer los mecanismos e infraestructura no comerciales para aceptar y administrar los fondos recaudados, e invertir tales fondos para promover el esfuerzo del desarrollo de esta base de datos.
- Fomentar la cooperación y la afiliación de individuos, organizaciones sin fines de lucro y compañías comerciales involucradas o que estén planeando estar involucradas en el desarrollo, apoyo y promoción de los proyectos de software de Firebird y sus productos y actividades asociadas.

1.7.3- Apache Derby

Apache Derby es un sistema gestor de base de datos relacional escrito en Java que puede ser empotrado en aplicaciones Java y utilizado para procesos de transacciones online. Tiene un tamaño de 2 MB de espacio en disco. Inicialmente distribuido como IBM Cloudscape, Apache Derby es un proyecto open source licenciado bajo la Apache 2.0 *License.* Apache Derby tiene su origen en la empresa *Cloudscape Inc*, en Oakland, California, que se fundó en 1996 para desarrollar una tecnología de base de datos para Java. La primera versión del motor de base de datos, que por entonces se llamó JBMS, tuvo lugar en 1997. Posteriormente el

producto fue renombrado como *Cloudscape* y aparecieron versiones nuevas cada seis meses. En 1999 *Informix Software*, adquirió *Cloudscape, Inc*. En 2001 IBM[3] adquirió los activos de *Informix Software*, incluyendo *Cloudscape*. El motor de base de datos fue renombrado a IBM *Cloudscape* y continuaron apareciendo versiones, enfocadas principalmente a usos embebidos en productos Java de IBM y *middleware* (Wikipedia, 2009).

Características

- APIs para JDBC y SQL. Soporta todas las características de SQL92 y la mayoría de SQL99. La sintaxis SQL usada proviene de IBM DB2.
- Su código mide alrededor de 2000KB comprimido.
- Soporta cifrado completo, roles y permisos. Además posee SQL SCHEMAS para separar la información en una única base de datos y control completo de usuarios.
- Soporta internamente cifrado y compresión.
- Trae soporte multilenguaje y localizaciones específicas.
- A partir de la versión 10.4 trae un sistema simple de replicación maestro-esclavo.
- Transacciones y recuperación ante errores.
- Posee tres productos asociados a la marca:
    - *Derby Embedded Database Engine*: El motor propiamente dicho.
    - *Derby Network Server:* Permite convertir *Derby* en una base de datos que sigue el modelo cliente-servidor tradicional.
    - *Database Utilities:* Un paquete de utilidades.

Ventajas

✓ Está basado en java, usa los estándares de JDBC y SQL.

[3] El mayor fabricante de ordenadores del mundo. IBM es el inventor del PC y con sus sistemas medios (AS/400) y sus grandes ordenadores (mainframes) han revolucionado el mundo de la empresa. Es también uno de los mayores vendedores de software, servicios y equipos de comunicaciones

- ✓ Tiene la opción del manejador de base de datos empotrado, embebido, encajado.
- ✓ Soporta la arquitectura cliente/servidor.
- ✓ Flexibilidad Manejo de Datos Complejos.
- ✓ BDOO-Ajusta al espacio necesario y elimina espacio desperdiciado.
- ✓ Manipulación de Objetos complejos en forma rápida y ágil.

1.8- Fundamentación de las herramientas a Utilizar

1.8.1- ¿Porqué utilizar Java?

Cuba en los últimos años ha experimentado cambios importantes en cuanto al desarrollo del uso de las tecnologías, quienes a su vez han influido sobre manera en las políticas que se llevaban a cabo en el país. Por tal motivo se ha tomado como estrategia la migración paso a paso de los sistemas que se utilizan en el país hacia software libre, para ello los nuevos sistemas que si desarrollen y los ya existentes deben cumplir con estas características. Una solución que han adoptado hoy muchos desarrolladores cubanos es el hacer software en el lenguaje de programación Java capaz de funcionar en varios entornos.

Teniendo en cuenta el conocimiento que se ha adquirido durante el estudio de la carrera y siendo la política de nuestro centro utilizar el lenguaje de programación Java para desarrollar aplicaciones de escritorio, se propone la utilización de este lenguaje para la implementación del sistema.

Otras características que hacen de este un lenguaje ideal para el diseño de esta aplicación son:

Simple: Elimina la complejidad de los lenguajes como "C" y da paso al contexto de los lenguajes modernos orientados a objetos. Orientado a Objetos. La filosofía de programación orientada a objetos es diferente a la programación convencional.

Familiar: Como la mayoría de los programadores están acostumbrados a programar en C o en C++, la sintaxis de Java es muy similar al de estos.

Robusto: El sistema de Java maneja la memoria de la computadora por ti. No te tienes que preocupar por apuntadores, memoria que no se esté utilizando, etc. Java realiza todo esto sin necesidad de que uno se lo indique.

Seguro: El sistema de Java tiene ciertas políticas que evitan se puedan codificar virus con este lenguaje. Existen muchas restricciones, especialmente para los applets, que limitan lo que se puede y no puede hacer con los recursos críticos de una computadora.

Orientada a Objeto: Java está orientado a objetos, ya que un buen diseño conlleva a componentes reutilizables, extensibles y sostenible. Esto permite controlar los cambios que puedan producirse a lo largo del tiempo, dado que el trabajo principal de estos objetos es intercambiar mensajes entre sí.

Portable: Como el código compilado de Java (conocido como byte code) es interpretado, un programa compilado de Java puede ser utilizado por cualquier computadora que tenga implementado el interprete de Java.

Independiente a la arquitectura: Al compilar un programa en Java, el código resultante un tipo de código binario conocido como *byte code*.

Este código es interpretado por diferentes computadoras de igual manera, solamente hay que implementar un intérprete para cada plataforma. De esa manera Java logra ser un lenguaje que no depende de una arquitectura computacional definida.

Multithreaded: Un lenguaje que soporta multiples *threads* es un lenguaje que puede ejecutar diferentes líneas de código al mismo tiempo.

Interpretado. Java corre en máquina virtual, por lo tanto es interpretado.

Dinámico: Java no requiere que compiles todas las clases de un programa para que este funcione. Si realizas una modificación a una clase Java se encarga de realizar un *Dynamic Bynding* o un *Dynamic Loading* para encontrar las clases.

1.8.2- ¿Porqué utilizar Apache Derby?

Apache Derby es un sistema gestor de base de datos relacional escrito en Java que puede ser empotrado en aplicaciones Java y utilizado para procesos de

transacciones online, es además una fuente abierta de pura Java *Database Management System* (DBMS). Para el sistema a implementar Derby es el DBMS más conveniente porque está libremente disponible, responde a las necesidades del sistema, integra perfectamente con *NetBeans* y a su vez ejecuta en todas las plataformas de este, sin dejar de mencionar es mucho más simple de instalar y administrar que los tradicionales DBMS.

1.8.3- Embarcadero ER/Studio 8.0.0

Es una herramienta de modelado de datos, se usa para el diseño y la construcción lógica y física de bases de datos. Su ambiente es de gran alcance y multinivel. ER/Studio se diseña para hacer más fácil de entender el estado actual de los datos de las empresas. Simple y fácil al usuario, ayuda a organizaciones para tomar decisiones en cómo resolver embotellamientos de los datos, elimina redundancia y alcanza en última instancia usos de más alta calidad que entreguen datos más eficientes y exactos a la empresa (Wikipedia, 2011).

Ventajas: Si se está comenzando un nuevo diseño o está manteniendo una base de datos existente, ER/Studio se combina con las características para ayudarle a conseguir el trabajo hecho con eficacia.

Con el potencial y la facilidad de empleo de ER/Studio's que modela el ambiente, será productivo rápidamente y podrá casi demostrar resultados inmediatamente después de la instalación.

Diagramas: La creación de diagramas es clara y rápida. Tiene la posibilidad de realizar diagramas con desempeño rápido. También es posible cambiar el estilo de las líneas, los colores, tipos de letra, niveles de acercamiento, y modelos de despliegue. Es posible crear subvistas para separar y manejar áreas importantes. ER/Studio automáticamente mantiene todas las dependencias entre subvistas y el diagrama completo. El Explorer Navigation facilita el trabajo hasta con los diagramas más grandes. Si se está trabajando con un modelo largo de Datos, ER/Studio ofrece un aumento en la ayuda y fácil navegación en sus modelos. La Apreciación global (overview). Se usa el browser Explorer para encontrar y seleccionar entidades. Un solo clic inmediatamente enfoca una ventana de diagrama.

Ventajas:

- Soporta el proceso de diseño interactivo inherente en el ciclo de vida de la aplicación.
- ER/Studio puede documentar automáticamente un diagrama entero, generando un conjunto integrado de reportes HTML sofisticados que múltiples usuarios pueden compartir en Internet.
- Calidad de presentación en los reportes. Además de los reportes de HTML, ER/Studio puede generar reportes de alta calidad con un formato de texto amplio que está disponible para presentaciones profesionales.

Soporta metodología de Yourdon, con diagramas relación-entidad y modelos IDEF1.

1.9- Metodologías para el desarrollo de Sistemas Informáticos

Las metodologías de desarrollo constituyen una colección de documentos formales referentes a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software. La finalidad de una metodología de desarrollo es garantizar la eficacia y la eficiencia en el proceso de generación de software (Cuaresma, 2008).

1.9.1- Metodologías tradicionales

Las metodologías tradicionales son también denominadas metodologías clásicas, las cuales están guiadas por una fuerte planificación durante todo el proceso de desarrollo basadas en normas provenientes de estándares seguidos por el entorno de desarrollo, las mismas generan un gran número de artefactos[4] que registran lo que se ha desarrollado en las distintas etapas del software. Además estas metodologías ofrecen una cierta resistencia a los cambios, siguen un proceso controlado por numerosas políticas y contemplan un contrato prefijado sobre las funcionalidades que deberá poseer la solución informática dada. Cuentan a su vez con la peculiaridad de que el cliente interactúa con el equipo de desarrollo mediante reuniones en las cuales se evalúan los adelantos del software.

[4] Documentos, diagramas, modelos, etc.

1.9.3- Proceso Unificado de Desarrollo del Software (RUP)

RUP (*Rational Unified Process*), en español Proceso Unificado de Desarrollo del Software, es un proceso que contiene un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software. Es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, tipos de organizaciones, niveles de actitud y tamaños de proyecto. Está basado en componentes, lo cual quiere decir que el sistema software en construcción está formado por componentes software interconectados a través de interfaces bien definidas. Utiliza el UML (Unified Modeling Lenguaje, UML) para realizar todos los artefactos de un sistema software, lenguaje que permite la modelación de sistemas con tecnología orientada a objetos.

El proceso de desarrollo RUP aplica varias de las mejores prácticas en el desarrollo moderno de software en una forma que se adapta a un amplio rango de proyectos y de organizaciones. Provee a cada miembro del equipo, un fácil acceso a una base de conocimiento con guías, plantillas y herramientas para todas las actividades críticas del desarrollo de software.

Esta metodología permite que todos los integrantes de un equipo de trabajo, conozcan y compartan el proceso de desarrollo, una base de conocimientos y los distintos modelos de cómo desarrollar el software utilizando un lenguaje de modelado común: UML.

RUP a su vez provee un enfoque estructurado para realizar tareas y responsabilidades en una organización de desarrollo. Su principal objetivo es asegurar la producción de software de alta calidad, que cumpla las necesidades de sus usuarios finales, que sea realizado en las fechas acordadas y con el presupuesto disponible.

1.9.3.1- Características principales de RUP

Dirigido por casos de uso:

Es necesario saber que un caso de uso es un fragmento de funcionalidad del sistema que proporciona un resultado de valor a un usuario. Son estos los que modelan los

requerimientos funcionales del sistema. En su conjunto constituyen el modelo de casos de uso. A partir de aquí estos guían el proceso de desarrollo (diseño, implementación, y prueba). Basándose en los casos de uso los desarrolladores crean una serie de modelos de diseño e implementación que llevan a cabo estos. De este modo no solo inician el proceso de desarrollo sino que le proporcionan un hilo conductor, avanza a través de una serie de flujos de trabajo que tienen su punto de partida en ellos.

Centrado en la arquitectura:

La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los CU relevantes desde el punto de vista de la arquitectura. Los casos de uso y la arquitectura están profundamente relacionados.

Los casos de uso deben encajar en la arquitectura, y a su vez la arquitectura debe permitir el desarrollo de todos los casos de uso requeridos, actualmente y a futuro.

Iterativo e Incremental:

La alta complejidad de los sistemas actuales hace que sea factible dividir el proceso de desarrollo en varios mini-proyectos. Cada uno de estos mini-proyecto se les denomina iteración y pueden o no representar un incremento en el grado de terminación del producto completo. En cada iteración los desarrolladores seleccionan un grupo de casos de uso, los cuales se diseñan, implementan y prueban. La planificación de iteraciones hace que se reduzcan los riesgos de los costes de un solo incremento, no sacar al mercado un producto en el tiempo previsto, mantener la motivación del equipo pues puede ver avances claros a corto plazo y que el desarrollo pueda adaptarse a los cambios en los requisitos.

1.9.3.2- Fases de RUP

RUP divide el proceso en 4 fases que son detalladas a continuación:

Inicio: Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.

Elaboración: Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos (funcionales y no funcionales) identificados de acuerdo al alcance definido.

Construcción o Implementación: Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene una o varias versiones del producto que han pasado las pruebas. Se ponen estas versiones a consideración de un subconjunto de usuarios.

Transición o Prueba: La versión ya está lista para su instalación en las condiciones reales. Puede implicar reparación de errores.

Ventajas de RUP:

- Reconoce que las necesidades del usuario y sus requerimientos no se pueden definir completamente al principio.
- Permite evaluar tempranamente los riesgos en lugar de descubrir problemas en la integración final del sistema.
- Acelera el ritmo del esfuerzo de desarrollo en su totalidad debido a que los desarrolladores trabajan para obtener resultados claros a corto plazo.
- Distribuye la carga de trabajo a lo largo del tiempo del proyecto ya que todas las disciplinas colaboran en cada iteración.
- Facilita la reutilización del código teniendo en cuenta que se realizan revisiones en las primeras iteraciones.

1.9.4- Fundamentación de la metodología a utilizar

Después de realizar un análisis e investigación de estas metodologías, se selecciona RUP para el desarrollo del software, porque es la más completa y abarcadora, pues como señalan algunos autores, las otras metodologías son casos particulares de esta RUP es adaptable, según el tipo de proyecto, así serán las características del mismo, haciéndose énfasis en aquellos flujos de trabajo durante la vida del software, que reporten más importancia y sean indispensables. RUP permite trazarse planes de riesgos y pruebas durante el ciclo de vida. Contiene artefactos que son diseñados durante las diferentes fases, los cuales describen detalladamente las características del software desde que se realiza el análisis del problema hasta la entrega final del producto. Además constituye la metodología ideal para el desarrollo del software ya que esta permite una vez concluido el mismo continuar desarrollando otros módulos a partir de la información que se genera, tan útil y precisa para aumentar otras potencialidades al sistema.

1.10- Herramientas de modelado

Las herramientas de modelado fueron creadas con el fin de desarrollar programas, utilizando técnicas de diseño y metodologías bien definidas, soportadas por herramientas automatizadas. Suelen llamarse herramientas CASE, significado de las siglas: *Computer Aided Software Engineering*, en español Ingeniería de Software Asistida por Computadora. Es un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación.

Una herramienta CASE es un instrumento que permite desarrollar soluciones informáticas abarcando todo el ciclo de vida de un software[5], por lo que constituye un apoyo para los desarrolladores del mismo.

1.10.1- Visual Paradigm

Visual Paradigm para UML (VP-UML) es una herramienta de diseño (UML), además constituye una herramienta CASE (*Computer Aided Software Engineering*) en

[5] Programa.

español, ingeniería de software asistida por computadora, diseñada para la ayuda al desarrollo de software. Es una herramienta UML profesional que soporta el ciclo de vida completo de desarrollo de un programa: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Se integra con las siguientes herramientas Java: *JBuilder, NetBeans, JDeveloper*, *Weblogic* Está disponible en varias ediciones, cada una destinada a unas necesidades: *Enterprise, professional, community, estándar* y *modeler*. El Visual Paradigm es un software que cuenta con una licencia gratuita y una comercial, de esta forma es posible la utilización de la versión gratuita para el desarrollo de la Ingeniería de este software.

Características de Visual Paradigm:

- ✓ Licencia: gratuita y comercial.
- ✓ Producto de calidad.
- ✓ Soporta aplicaciones web.
- ✓ Varios idiomas.
- ✓ Fácil de instalar y actualizar.
- ✓ Compatibilidad entre ediciones.

1.10.2. Rational Rose

Rational Rose es una herramienta CASE que propone la utilización de cuatro tipos de modelos para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del mismo. Permite crear y refinar estas vistas estableciendo de esta forma un modelo completo que representa el dominio del problema y de las características del sistema.

Rational Rose utiliza un proceso de desarrollo iterativo controlado. Cada iteración comienza con una primera aproximación del análisis, diseño e implementación para identificar los riesgos del diseño, los cuales se utilizan para conducir la iteración,

primero se identifican los riesgos y después se prueba la aplicación para que estos se hagan mínimos. Cuando la implementación pasa todas las pruebas que se determinan en el proceso, esta se revisa y se añaden los elementos modificados al modelo de análisis y diseño.

Características

- Capacidad para proporcionar el desarrollo iterativo.
- Mantiene la consistencia de los modelos del sistema.
- Chequeo de la sintaxis UML.
- Generación de código a partir de los modelos.
- Ingeniería Inversa (crear modelo a partir código).
- No es gratuito, se debe hacer un previo pago para poder adquirir el producto.
- Admite la integración con otras herramientas de desarrollo.

1.10.3. Fundamentación de la herramienta de modelado a utilizar

Para el modelado del sistema se utiliza como herramienta CASE a *Visual Paradigm* en su versión 6.4, puesto que es una herramienta potente y distribuida bajo una licencia gratuita, es fácil de instalar y actualizar permitiendo una excelente compatibilidad entre ediciones, lo que facilita el trabajo de modelado de la aplicación.

Conclusiones del capítulo

En este capítulo se abordan los conceptos fundamentales asociados al dominio del problema relacionados con el objeto de estudio y el campo de acción. Se realizó un análisis sobre la minería en Cuba, y sobre las metodologías geométricas de Barton, Bieniawski, y Cuesta. Luego de abordar las metodologías así como tecnologías empleadas para el desarrollo de la herramienta informática se escogieron las que presentan mayor ventaja con respecto a las características de nuestro sistema, entre ellas se encuentran: Java como lenguaje de programación, RUP como metodología de desarrollo, Visual Paradigm como herramienta de modelación visual para el análisis y diseño de sistemas basados en objetos, y Apache Derby como sistema gestor de bases de datos.

Capítulo II Modelo de Dominio y Requerimientos

2.1- Introducción al capítulo

El modelado de dominio contribuye a la comprensión del contexto del sistema, así como de sus requisitos, el cual captura los tipos más importantes de objetos del dominio para representar las “cosas” que existen o los eventos que suceden en el entorno en el que se trabaja. Todo ello ayuda a los usuarios a entender el contexto en el que va dirigido el sistema. Es decir la selección de un modelo de dominio puede utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño del sistema, en el cual se precisa los requisitos funcionales y no funcionales además se definen las principales entidades del dominio.

2.2- ¿Por qué Modelo de Dominio?

El Modelo de Dominio es una representación visual estática del entorno real objeto del proyecto. Es decir, un diagrama con los objetos que existen (reales) relacionados con el proyecto que vamos a acometer y las relaciones que hay entre ellos. Se llama "de Dominio" para distinguirlo del Modelo de Negocio Sin embargo, el Modelo de Dominio se centra en una parte del negocio, la relacionada con el ámbito del proyecto. En este contexto el término "dominio" representa una parte del "negocio" (Larman, 2008).

Para el desarrollo de la investigación, se tiene como punto de partida el estudio del software “ConExc”, existente en el ISMM, que aunque no está siendo utilizado es el precedente del sistema a automatizar, por ello, el propósito de esta investigación no está enfocado a entender los procesos de negocio, sino en comprender los conocimientos asociados con el sistema similar al que se está desarrollando, además del aporte de ideas de algunos expertos del dominio para alcanzar el perfeccionamiento del sistema.

El objetivo del modelado del dominio es contribuir a la comprensión del contexto del Sistema, así como a la comprensión de los requisitos, el cual captura los tipos más importantes de objetos de dominio para representar las cosas que existen o los

eventos que suceden en el entorno en el que se trabaja. Todo esto ayuda a los usuarios a entender el contexto en el que va dirigido el sistema.

2.2.1- Definición de las entidades y conceptos fundamentales

Las entidades y conceptos utilizados para el desarrollo del modelo de dominio son:

Minero: es la persona que se encarga de excavar minas para extraer minerales. Las principales ocupaciones de un minero incluyen taladrar la roca con picos y palas o utilizando herramientas eléctricas para extraer el mineral, apuntalar los túneles con soportes de madera para impedir su derrumbe, desplegar vías para el transporte de la piedra o cargar el mineral en vagonetas para su transporte al exterior.

Clasificación Geomecánica: aporta los índices de calidad relacionados con parámetros geomecánicos del macizo, sostenimiento de túneles y taludes y excavabilidad. Describe las características y propiedades de la matriz rocosa, de las discontinuidades y de los parámetros globales del macizo rocoso, proporciona los parámetros requeridos por las distintas clasificaciones (Emérito, 2009).

Clasificación Bieniawski: Permite hacer una clasificación de las rocas 'in situ' y estimar el tiempo de mantenimiento y longitud de un vano. Se utiliza usualmente en la construcción de túneles, de taludes y de cimentaciones. Consta de un índice de calidad RMR (*Rock Mass Ratting*), independiente de la estructura, y de un factor de corrección (Rose, 2007).

Clasificación Barton: cataloga los macizos rocosos según el denominado índice de calidad Q, basado en los seis parámetros siguientes: *Rock Quality Designation* (RQD), Número de familias de juntas (Jn), Rugosidad de las juntas (Jr), Meteorización de las juntas (Ja), Presencia de agua (Jw), S.R.F. *Stress Reduction Factor* (SRF) (Rose, 2007).

Clasificación Cuesta: realiza un estudio integral en toda la traza de la misma y zona de influencia de los principales aspectos que inciden en los procesos de circulación de agua y su control. Con el propósito de racionalizar el estudio y realizar un análisis la susceptibilidad de la excavación a inestabilidad e inundación por la presencia de agua (Cuesta, 2011).

Excavación: proceso de análisis de las estratigrafías naturales y antrópicas que se sedimentan en un determinado lugar. El proceso de excavación consiste en remover los depósitos en el orden inverso a cómo se han ido formando. Por este motivo es preciso comprender en todo momento durante una excavación: 1. los límites y la naturaleza de los depósitos que configuran la estratificación; 2. los procesos formativos que se han dado lugar a estos depósitos; 3. el orden o la secuencia relativa con la que se han formado los depósitos.

Propuesta de Sostenimiento: depende de la calidad del macizo rocoso y del tipo de roca. Esta es la que determinará el tipo de excavación que deberá realizarse (Emérito, 2009).

2.2.2- Representación del diagrama de clases del Modelo del Dominio

Un modelo de dominio sirve como una clara representación de la estructura conceptual del dominio del problema y por lo tanto tiene un valor incalculable para asegurar que todos los interesados están alineados en el alcance y significado de los conceptos para el dominio del problema.

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 seguimiento.

· Sucesos que ocurrirán o han ocurrido.

El modelo de dominio se representa fundamentalmente por diagramas de clases en UML. Su objetivo es comprender y describir las clases más importantes dentro del contexto del sistema.

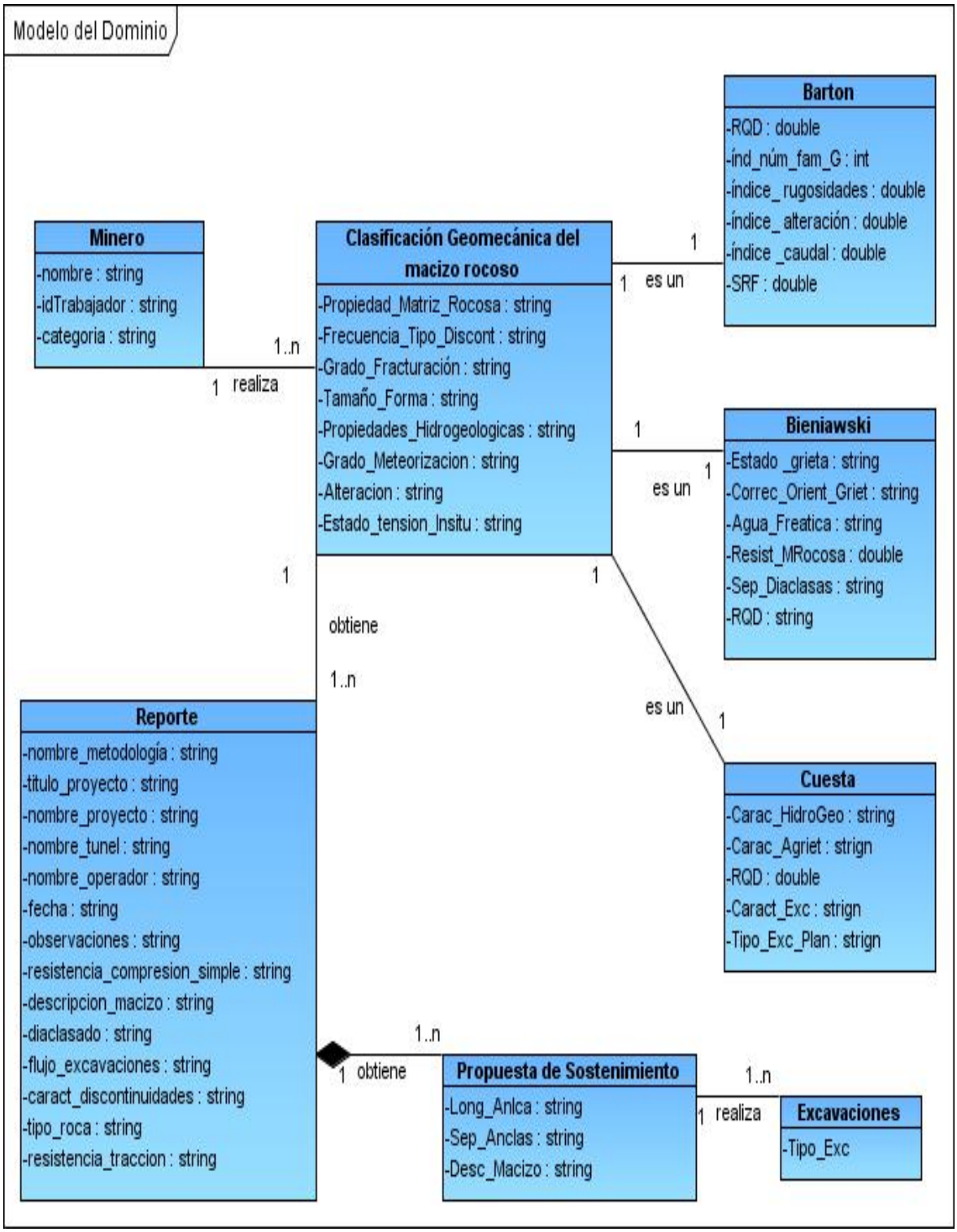

Figura 2.1. Diagrama del Modelo del Dominio

2.2.3- Requerimientos

Cualquier metodología de Análisis y Diseño para el desarrollo de sistemas tiene como punto de partida la captura de requisitos, obtenidos por los analistas en interacción con los usuarios, que más tarde serán analizados y plasmados en herramientas propias de cada metodología de manera que cubran las expectativas de los usuarios y que se ajusten a las tendencias actuales de desarrollo de aplicaciones (Karl, 2008).

2.2.3.1- Requerimientos funcionales

Un requisito funcional define el comportamiento interno del software: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que muestran cómo los casos de uso serán llevados a la práctica. Estos a su vez constituyen las condiciones o capacidades que el sistema debe cumplir, no alteran la funcionalidad del producto, se mantienen invariables sin importar las propiedades o cualidades con las que se relacione este (Karl, 2008).

El sistema en desarrollo debe ser capaz de:

RF-1. Crear Proyecto

RF-2. Guardar Proyecto

RF-3. Abrir Proyecto

RF-4. Gestionar Azimut_Buzamiento

4.1 Insertar Azimut y Buzamiento

4.2 Modificar Azimut y Buzamiento

4.3 Eliminar Azimut y Buzamiento

RF-5. Gestionar Metodologías

5.1 Crear Metodología

5.2 Eliminar Metodología

RF-6. Gestionar Proyección Estereográfica

5.1 Crear Proyección Estereográfica

5.2 Eliminar Proyección Estereográfica

RF-7. Graficar Proyección Estereográfica

RF-8. Calcular Familia de grietas

RF-9. Calcular Metodología Barton

RF-10. Calcular Metodología Bieniawski

RF-11. Calcular Metodología Cuesta

RF-12. Mostrar Nomograma Barton

RF-13. Mostrar reporte Barton

RF-14. Mostrar reporte Bieniawski

RF-15. Mostrar reporte Cuesta

RF-16. Imprimir reporte

RF-17. Cerrar proyecto

2.2.3.2- Requerimientos no funcionales

Los Requisitos no funcionales son propiedades o cualidades que el producto debe tener. Un requisito no funcional especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos (Karl, 2008). En la mayoría de los casos, estos son esenciales en el éxito del producto para que los clientes y los usuarios puedan valorar las características no funcionales que posee, pues si se conoce que cumple con toda la funcionalidad requerida, las propiedades no funcionales como cuán usable, seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación. Están vinculados con los requerimientos funcionales, pues cuando se conoce lo que el sistema debe hacer podemos determinar cómo ha de comportarse, qué cualidades tendrá o cuán rápido podrá ser.

Apariencia e interfaz externa:

Los usuarios pueden ser personas que no estén familiarizadas con herramientas informáticas, por lo que se requiere una interfaz sencilla, con la mayor simplicidad

posible para lograr que el sistema sea de fácil entendimiento, de apariencia profesional e intuitiva. Debe ser formal, deberán usarse colores claros y formatos de textos legibles que mejoren la visibilidad del usuario. Desde una perspectiva más amplia del diseño visual de la aplicación, debe mantener una coherencia y estilo común entre todas las ventanas, proporcionando una consistencia visual a la aplicación. La interfaz debe utilizar preferentemente conceptos que sean manejados y les resulten familiares a los usuarios para que les sea fácil el uso de la herramienta y su aprendizaje.

Requerimiento de rendimiento:

El sistema propuesto pretende manejar el procesamiento de datos con una buena eficiencia, precisión y disponibilidad para que el éxito del producto se vea reflejado en la capacidad de la velocidad con que se manipula la información. Se realiza el tratamiento de errores para evitar un posible bloqueo de la aplicación. De suma importancia es además la precisión de toda la información almacenada para la ejecución posterior de posibles Excavaciones Subterráneas.

Requerimiento de confiabilidad:

La seguridad de los datos es garantizada por la base de datos embebida Apache Derby, escogido para el desarrollo de la aplicación, quien ofrece una garantía de integridad en los datos. Es el indicado para el software ya que no se manejan altos volúmenes de información.

Requerimiento de software:

Mínimos Recomendados:

- ✓ Sistema Operativo: Familia Windows superior a Windows Millenium, Linux, entre otros.
- ✓ Apache Derby como gestor de base de datos relacional.
- ✓ Maquina Virtual 6.0 (JDK).
- ✓ Herramienta para reportes: iReport.

Requerimiento de hardware:

Mínimos Recomendados:

- ✓ Pentium II con 128 MB de RAM y un microprocesador a 300 MHz, 6 Gb de disco duro.

Requerimiento de diseño e implementación:

Para la construcción del sistema se requiere:

- ✓ Lenguaje de programación para la implementación: Java.
- ✓ Como artefactos para el diseño se usan los que propone RUP apoyado en el estándar de notaciones de UML.
- ✓ Como gestor de base de datos Apache Derby.

Requerimiento de soporte:

El proceso de instalación y configuración del sistema será realizado por los usuarios que utilicen el Software, para ello deberán recibir un adiestramiento correspondiente para el trabajo con el software, luego de culminada la realización de mismo se realizarán pruebas de rendimiento para garantizar su eficiencia y se brindará un curso de instalación, configuración y mantenimiento del producto.

Requerimiento de usabilidad:

El sistema será usado principalmente por mineros o geólogos que necesiten utilizar las Metodologías Geomecánicas de clasificación de los macizos rocosos para proyectos de excavaciones subterráneas o determinar el Número de Familia de Grietas mediante la Proyección Estereográfica.

Requerimiento de portabilidad:

El producto de Software podrá ser usado bajo cualquier sistema operativo, Windows, Linux y otros.

Conclusiones del capítulo

En el capítulo se argumentó la utilización del Modelo de Dominio, se definieron las entidades utilizadas en el Diagrama del Modelo del Dominio, además se obtuvieron los requisitos funcionales, y no funcionales del software. Los que serán la base para el diseño del diagrama de caso de uso del sistema para una mejor compresión de este.

Capítulo 3. Diseño e implementación del Sistema

3.1- Introducción al capítulo

En este capítulo se detalla la construcción de la solución propuesta, para ello se presenta el diagrama de clases por casos de uso y se realiza una descripción detallada para cada uno. Se mencionan los principios que se tienen en cuenta para el diseño de la aplicación, entre los que se pueden mencionar estándares de interfaz, concepción general de la ayuda y el tratamiento de errores. Se presenta el diagrama de clases de diseño, el diagrama de clases persistentes, utilizando este último para generar el modelo físico de datos con el uso de la herramienta Visual Paradigm. Se especifica también, la arquitectura empleada en la confección de este producto y son mostrados los diagramas de secuencia. Por otro lado, se ofrece una representación gráfica del modelo de despliegue y el de componente.

3.2- Identificación de los actores del sistema a automatizar

Los actores representan los usuarios del sistema e incluyen otros sistemas que puedan interactuar con él, es decir, representan terceros fuera del sistema que colaboran con este. Una vez identificados, cada actor desempeñará un papel bien definido, y en el contexto de ese papel debe tener interacciones útiles con el sistema, además se tiene delimitado el entorno externo del mismo. (Pressman, 2005)

Tabla 3.1: Actores del sistema

ACTOR	JUSTIFICACIÓN
Usuario	Especialista en Minería o Geología que accede al sistema teniendo todos los permisos en él, puede obtener información sobre los Macizos Rocosos mediante el cálculo por las metodologías Barton, Bieniawski y Cuesta, así como visualizar los reportes de cada uno de estos cálculos, además de graficar y seleccionar el número de familia de grietas.

3.3- Diagrama de caso de uso del sistema

El artefacto fundamental del flujo de trabajo de requerimientos es el diagrama de caso de uso que es una especie de diagrama de comportamiento. Los diagramas de casos de uso son importantes para modelar el comportamiento de un sistema, un subsistema o una clase, además se utilizan para visualizar, especificar, y documentar el comportamiento de un elemento. Los casos de uso pueden presentar varios tipos de relaciones: generalización/especialización[6], include[7] y extend[8]; los actores solo pueden presentar la relación de generalización/especialización; es importante resaltar que un caso de uso puede ser inicializado por un solo actor.

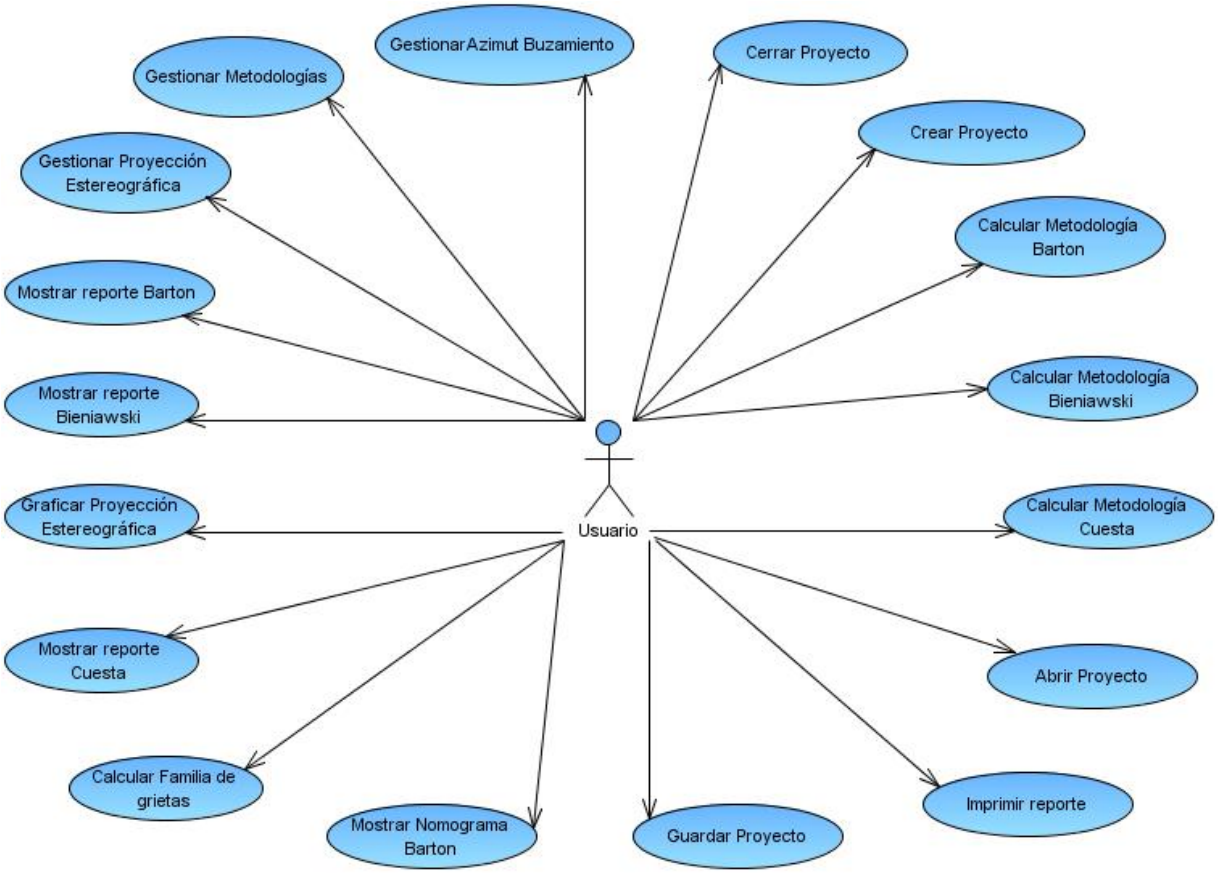

Figura 3.1: Diagrama de caso de uso del sistema

[6] Se comparten estructuras, propósitos y comportamientos.

[7] El caso de uso de inclusión se inserta explícitamente dentro del comportamiento definido para el caso de uso base.

[8] El caso de uso de extensión tiene un comportamiento opcional u optativo, puede ser usado o no por caso de uso base.

3.3.1- Descripciones de los casos de uso del sistema

Tabla 3.2: Descripción del caso de uso "Gestionar Metodología"

Caso de uso:	Gestionar Metodología
Actor:	Usuario (inicia)
Propósito:	Permite insertar y eliminar formularios de las diferentes metodologías Barton, Bieniawski, Cuesta.
Resumen:	El caso de uso inicia cuando el usuario avanzado escoge insertar o eliminar una metodología, finaliza cuando el sistema realiza una de las operaciones seleccionadas anteriormente.
Referencia:	RF-1
Precondiciones:	Para la inserción: No debe existir ese proyecto. Para la eliminación: El proyecto debe estar registrado.
Poscondiciones:	Son guardados los cambios efectuado en la base de datos.
Requisitos Especiales	-

Las restantes descripciones de los casos de uso se encuentran en: **Anexo 1**

3.4- Diagrama de clases del diseño

Este describe las clases que componen el sistema con un nivel de detalle mayor. Incluye los atributos particulares de cada clase y las relaciones existentes entre ellas. Se crean para cumplimentar los requisitos funcionales y no funcionales, teniendo en cuenta la tecnología en la cual se implementará el diseño.

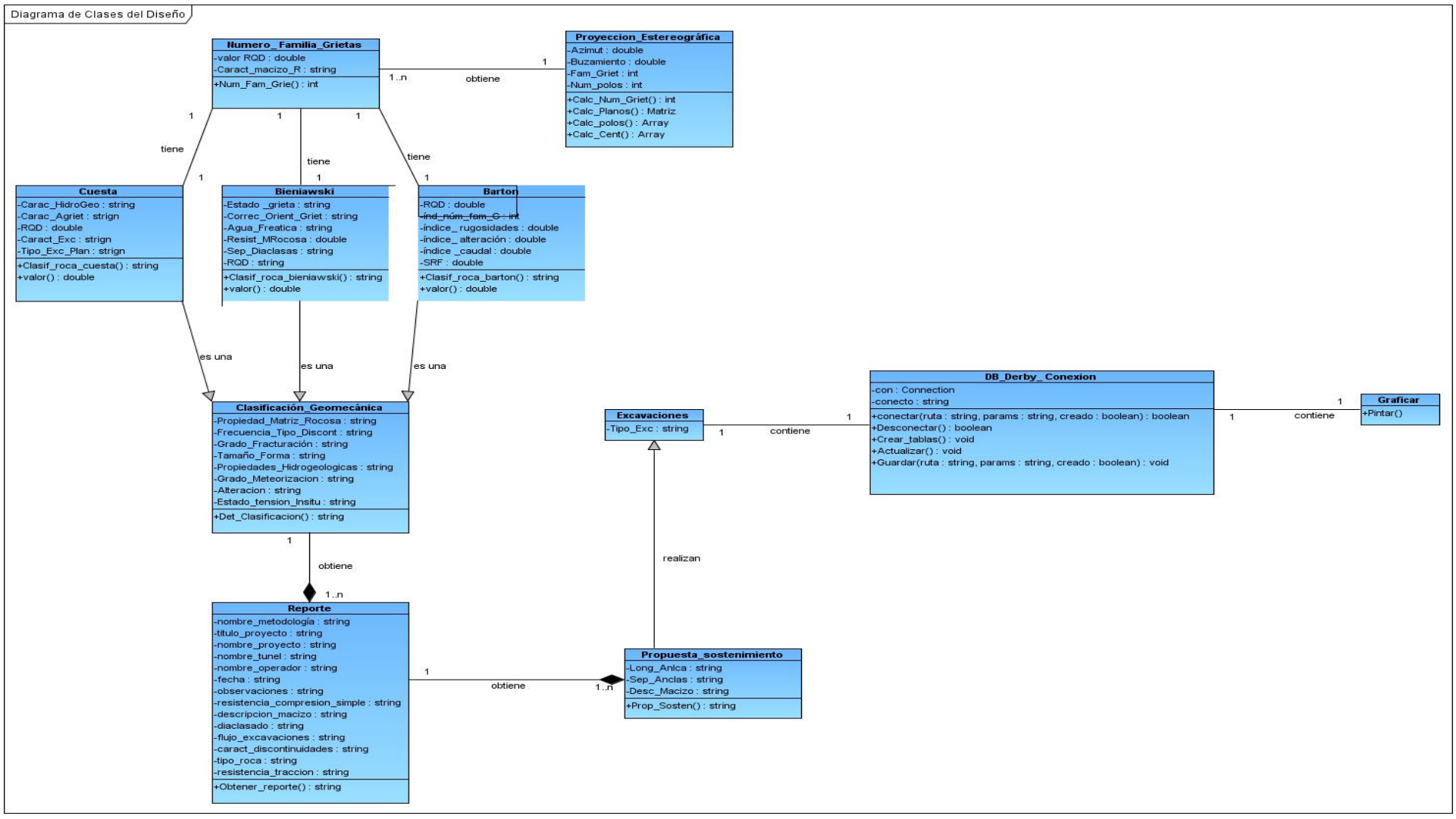

Figura 3.2: Diagrama de clases del diseño

3.4.1- Descripción de las clases del diseño

A continuación se describe cada clase representada en el diagrama anterior, Se define el tipo de clase que es cada una (entidad o controladora), las funciones que realizan, los atributos y métodos que la conforman.

Tabla 3.3: Descripción de la clase DB_Derby_ Conexion

Nombre de la clase	DB_Derby_ Conexion
Tipo de clase:	Entidad
Resumen: Es la clase encargada de relacionar la aplicación con la base de datos embebida .Permite conectar, guardar, y desconectar, crear tablas y actualizar la base de datos Apache Derby.	
Atributos	**Tipo**
con	Connection
conecto	String
Responsabilidades	
Nombre.	**Descripción.**
Conectar (ruta: String, Params: String, creado: boolean)	Permite realizar la conexión de la base de datos.
Desconectar()	Este método se encarga de desconectar la base de datos.
crearTablas()	Es el encargado de crear las tablas en la base de datos.
Guardar (ruta: String, Params: String, creado: boolean	Este método se encarga guardar la base de datos.
Actualizar()	Este método se encarga actualizar la base de datos.

Las descripciones de las demás clases se encuentran en él: **Anexo 2**

3.5- Principios del Diseño

El diseño, sea cual sea el objeto del mismo, tiene que basarse en el usuario, y en nuestro caso estamos hablando de estudiantes, profesores, u otros usuarios que necesiten interactuar con el sistema; muchos de ellos sin una preparación para utilizar herramientas informáticas. Para ello, este sistema utiliza ciertos principios que garantizan la usabilidad en los diseños para aplicaciones de escritorio.

Visibilidad del estado del sistema: La aplicación debe mantener siempre informado al usuario del estado del sistema así como de los caminos que este pueda tomar con una retroalimentación visual apropiada en un tiempo razonable. El sistema ofrecerá al usuario una respuesta que le indique lo que está sucediendo en cada una de las operaciones que realiza.

Control y libertad del usuario: La interfaz debe ser diseñada de tal manera que el control de la interacción con el sistema lo tenga el usuario de manera que interactúe directamente con los objetos de la pantalla. De esta forma, este, se sentirá más cómodo y no se sentirá un módulo más de la aplicación. Esto se consigue cuando el usuario manipula los objetos como si fueran objetos físicos.

Consistencia y estándares: Una buena interfaz contribuye al aumento de la productividad si es consistente en todos los diálogos que desarrolla, basándose en el conocimiento que el usuario ha adquirido con otras aplicaciones y en la aplicación propia. Se debe mantener la consistencia en todas las aplicaciones relacionadas. Deberán implementar las mismas reglas de diseño para mantener la consistencia en toda la interacción.

Prevención de errores: El mejor tratamiento de los errores es prevenirlos con un buen diseño de los diálogos desde el primer momento en que ocurren, minimizando los riesgos de que puedan ocurrir. Se debe realizar un buen diseño de mensajes de error que den la posibilidad al usuario de retraerse antes de que se realice la acción y se comprometan los datos.

Estética y diseño minimalista: Los diálogos no deben contener información que sea irrelevante para la tarea que está realizando el usuario. Debe ser una interfaz simple, fácil de aprender y de usar y con fácil acceso a las funcionalidades que ofrece la aplicación. La información extra no necesaria disminuye la visibilidad al usuario causando errores en la interacción y distrayendo al usuario en la realización de la tarea.

Seguidamente se representan las interfaces gráficas, diseñadas para la aplicación final, a ellas se adjunta un resumen con algunas de las funcionalidades del sistema.

3.5.1- Interfaz de usuario

Atendiendo a las necesidades del cliente, el sistema tiene una interfaz clara, su diseño se concibió con la utilización de colores tenues, e imágenes y palabras relacionadas con el entorno de los usuarios, existe poca visualización de imágenes lo que permite que sea más eficiente a la hora en que se cargue y que el rendimiento sea mayor, la interfaz es además sencilla y amigable, facilitando la navegación por el software.

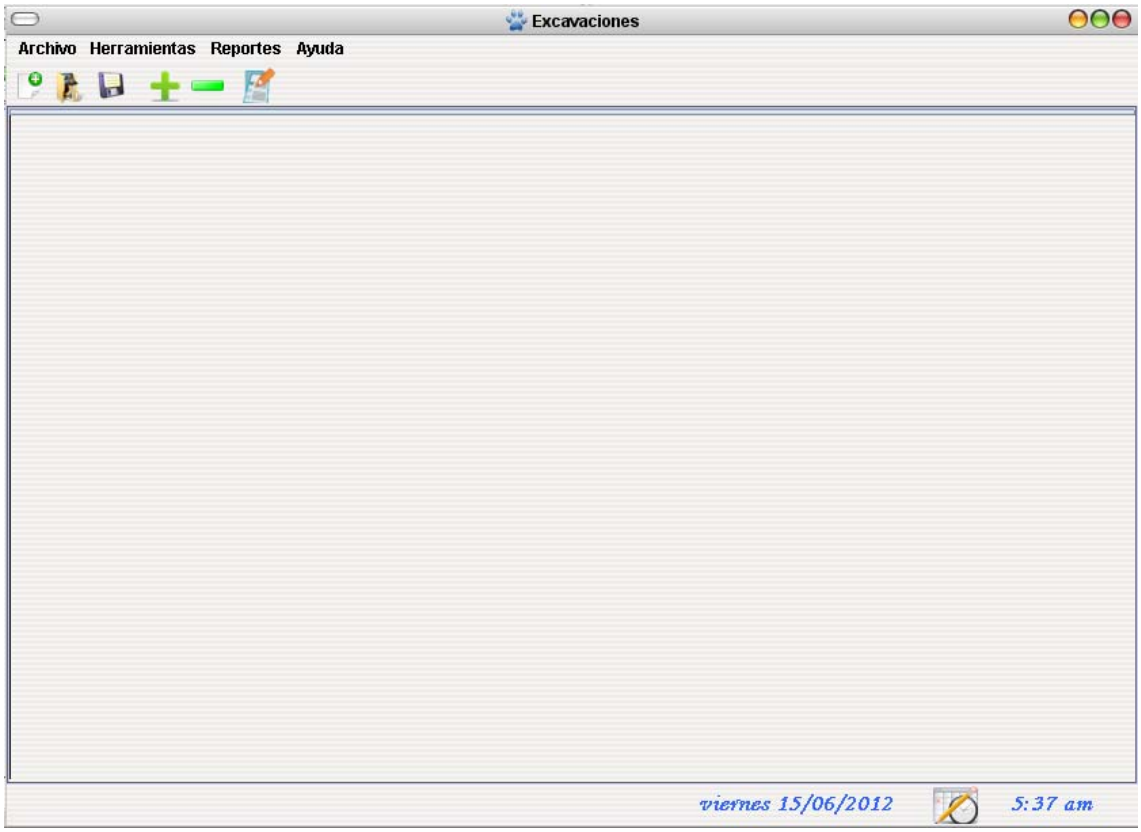

Figura 3.3: Interfaz principal del software

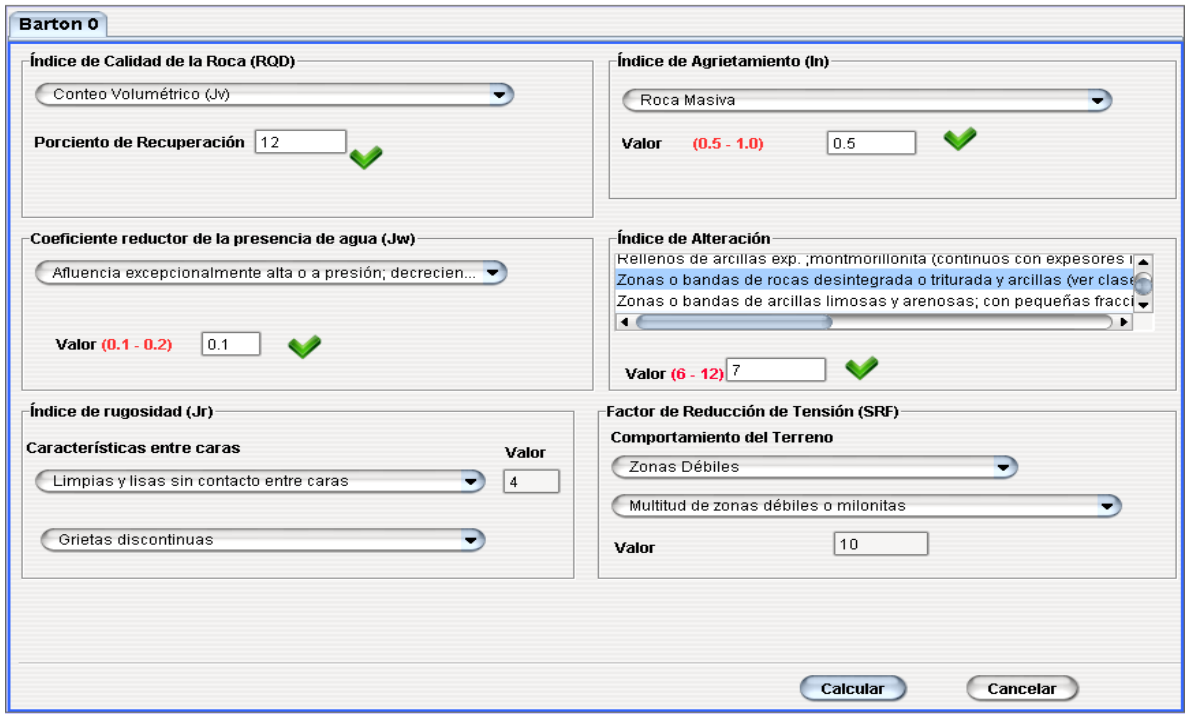

Figura 3.4: Interfaz de usuario para el caso de uso "Calcular Barton"

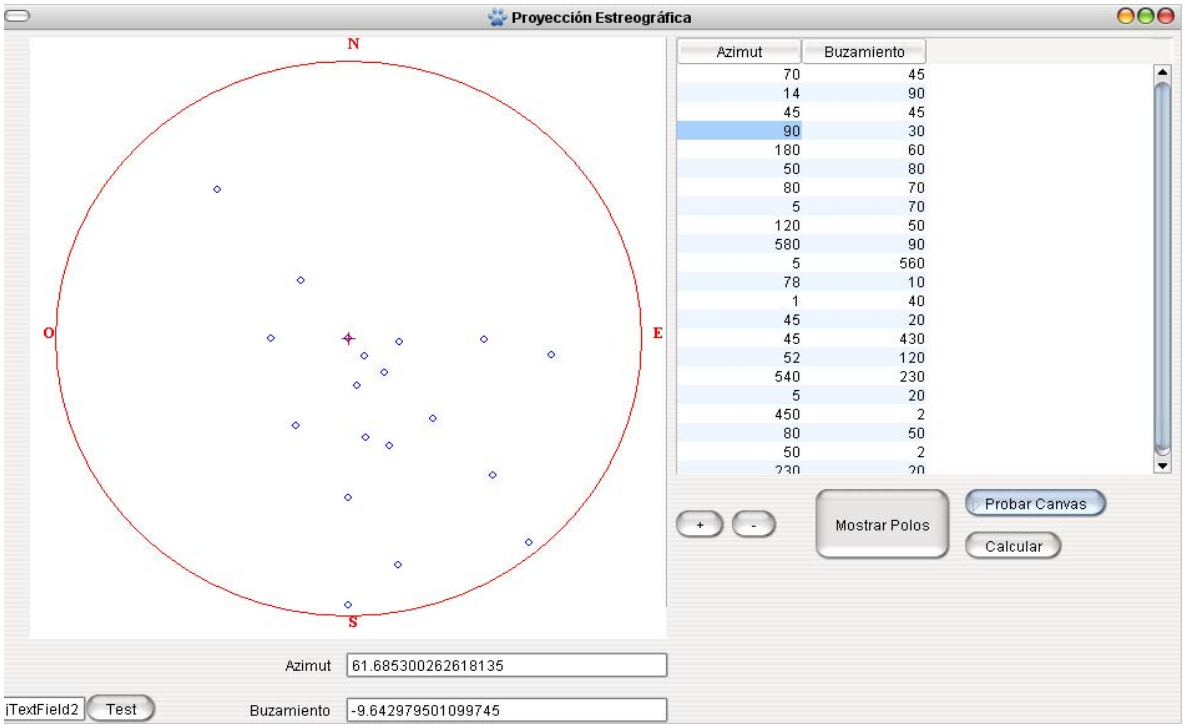

Figura 3.5: Interfaz de usuario para el caso de uso "Graficar Proyección Estereográfica"

3.5.2- Tratamiento de errores

Para que un sistema sea confiable, este debe ser capaz de detectar la mayor cantidad de errores que sea posible y explicarle al usuario dónde es que se ha cometido el error. Los errores más frecuentes cometidos por los usuarios son: al insertar un elemento que ha sido insertado anteriormente, cuando van a eliminar o modificar uno que no está registrado en la base de datos o al realizar una de las acciones anteriores se dejan campos vacíos. Para estos, el sistema utiliza las excepciones try y catch dándole posibilidades al usuario de rectificar su falta. Los mensajes mostrados a los usuarios son breves y orientadores, ayudando en alguna medida a estos a rectificar las causas que le dieron origen.

Figura 3.6: Tratamiento de errores

3.6- Diseño de la base de datos

La base de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso, pueden clasificarse de varias maneras, de acuerdo al contexto que se esté manejando, la utilidad de las mismas o las necesidades que satisfagan. La base de datos utilizada en el sistema, es embebida y se describe a continuación (Pérez, 2010).

3.6.1- Modelo lógico de datos

En el diagrama de clases persistentes mostrado en la **Figura 3.6**, se aprecian las relaciones de asociación entre las clases y se incluyen las llaves primarias de cada clase.

3.6.2- Modelo físico de datos

Este modelo, mostrado en la **Figura 3.7**, describe la estructura lógica de la información que será almacenada en la base de datos. Es generado a partir del diagrama de clases persistentes mediante la herramienta Embarcadero.

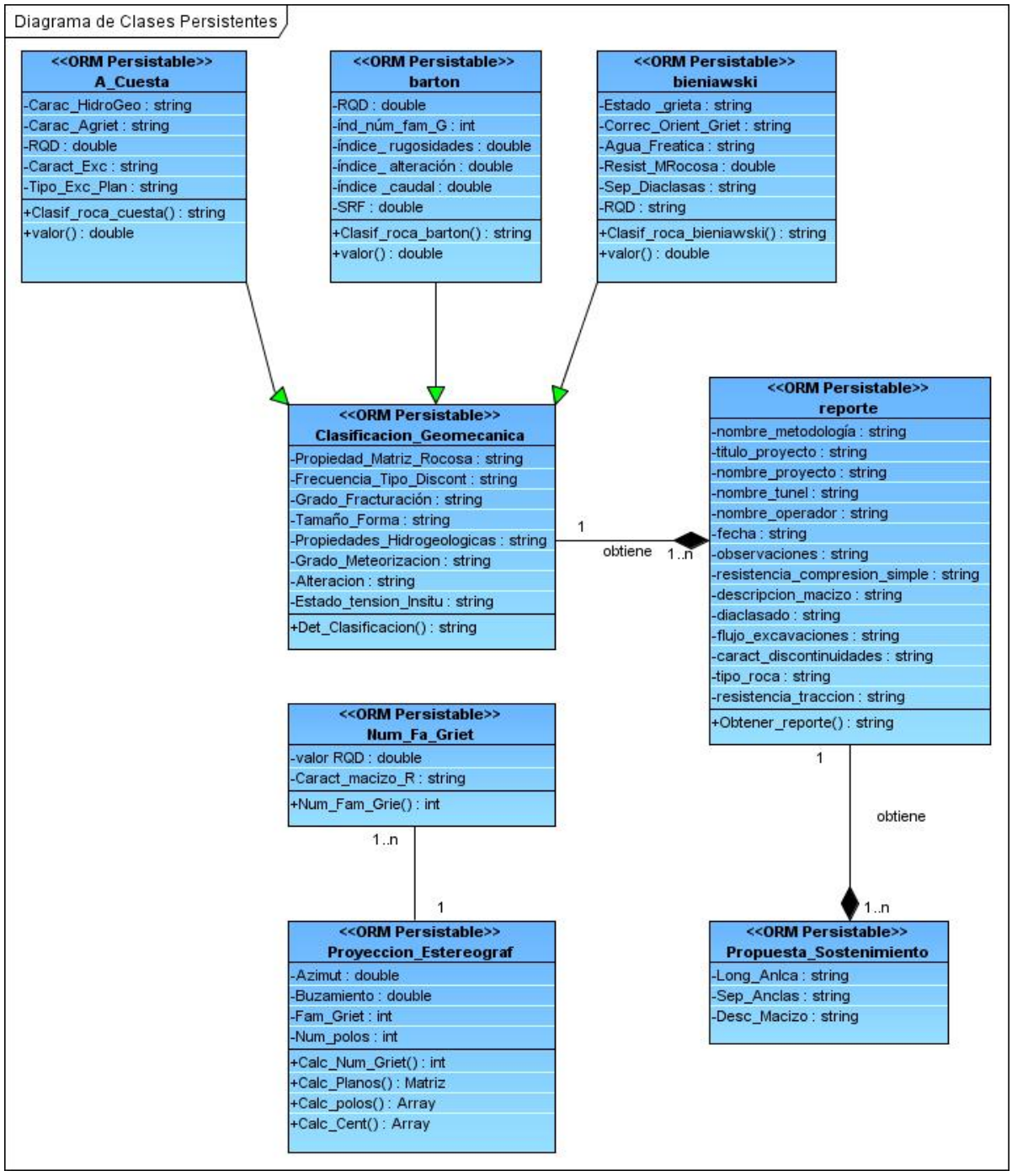

Figura 3.7: Diagrama de Clases Persistentes

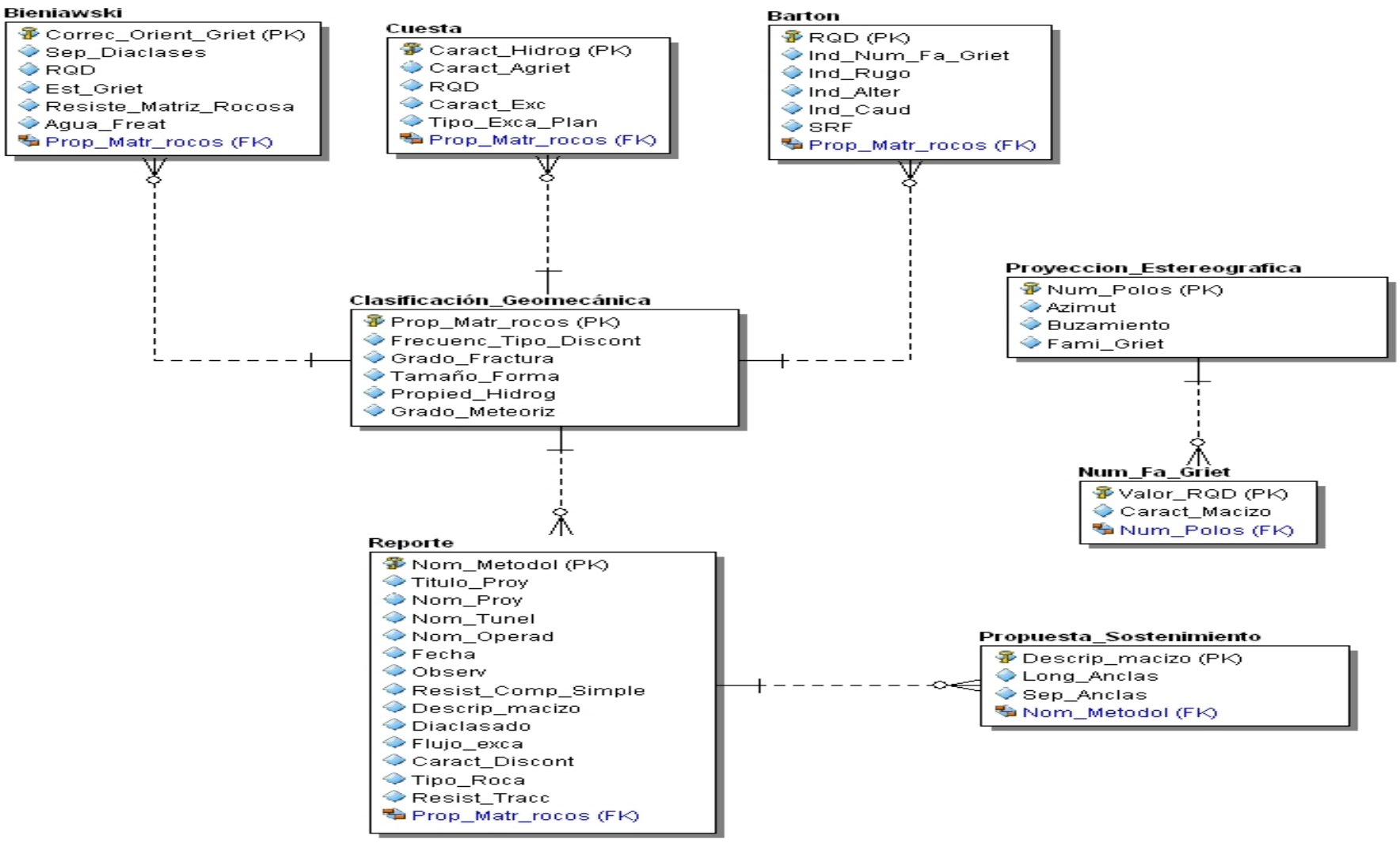

Figura 3.8: Modelo Físico de Datos

3.6.2.1- Descripción de las tablas del modelo físico

Seguidamente son descritas las tablas que conforman el modelo físico de datos.

Tabla 3.4: Descripción de la tabla reporte

Nombre de la tabla	T_reporte	
Descripción.	Muestra los valores del cálculo de las metodologías Barton, Bieniawski y A_Cuesta.	
Atributos	**Tipo**	**Descripción de los atributos**
nombre_metodología	string	Nombre de la metodología.
titulo_proyecto	string	Título del proyecto
nombre_tunel	string	Nombre del túnel donde va a efectuarse la excavación.
nombre_operador	string	Nombre del operador que toma los datos en el campo
fecha	string	Fecha en la que se realiza el reporte.
observaciones	string	Detalles de relevancia sobre la excavación.
resistencia_compresion_simple	string	Datos sobre la resistencia de compresión simple.
descripcion_macizo	string	Descripción del macizo rocoso.
diaclasado	string	Valor del diaclasado.
flujo_excavaciones	string	Valor del flujo de excavaciones
caract_discontinuidades	string	Características del las discontinuidades.
tipo_roca	string	Tipos de roca de la excavación.

resistencia_traccion	string	Detalles de la resistencia a tracción.

3.7- Definición de la arquitectura

3.7.1- Patrones Arquitectónicos

Los patrones arquitectónicos, son patrones de diseño de software que ofrecen soluciones a problemas de arquitectura de software en ingeniería de software. Dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones (Buschmann, 2007). El lenguaje de programación JAVA, seleccionado anteriormente, implementa a su vez el patrón arquitectónico MVC, es por ello que adoptamos esta arquitectura para el desarrollo de la propuesta de solución. Modelo Vista Controlador (MVC). Es un patrón de diseño de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

- Modelo: Representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- Vista: Presenta el modelo en un formato adecuado, como en una página Web que le permite al usuario interactuar con ella, usualmente un elemento de interfaz de usuario.
- Controlador: Responde a eventos, usualmente acciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) logrando un mantenimiento más rápido y sencillo de las aplicaciones. Ejemplo, para el caso de la web, si se fuera a mostrar una misma aplicación en un navegador estándar, como en un navegador de un dispositivo móvil, sólo es necesario crear una vista nueva por cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los

detalles del protocolo utilizado para las peticiones (Aplicación de escritorio, HTTP, consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación.

Ventajas del Modelo Vista Controlador

- La separación del Modelo de la Vista, es decir, separa los datos de la representación visual de los mismos.
- Crea independencia de funcionamiento.
- Facilita el mantenimiento en caso de errores.
- Permite el escalamiento de la aplicación en caso de ser requerido.

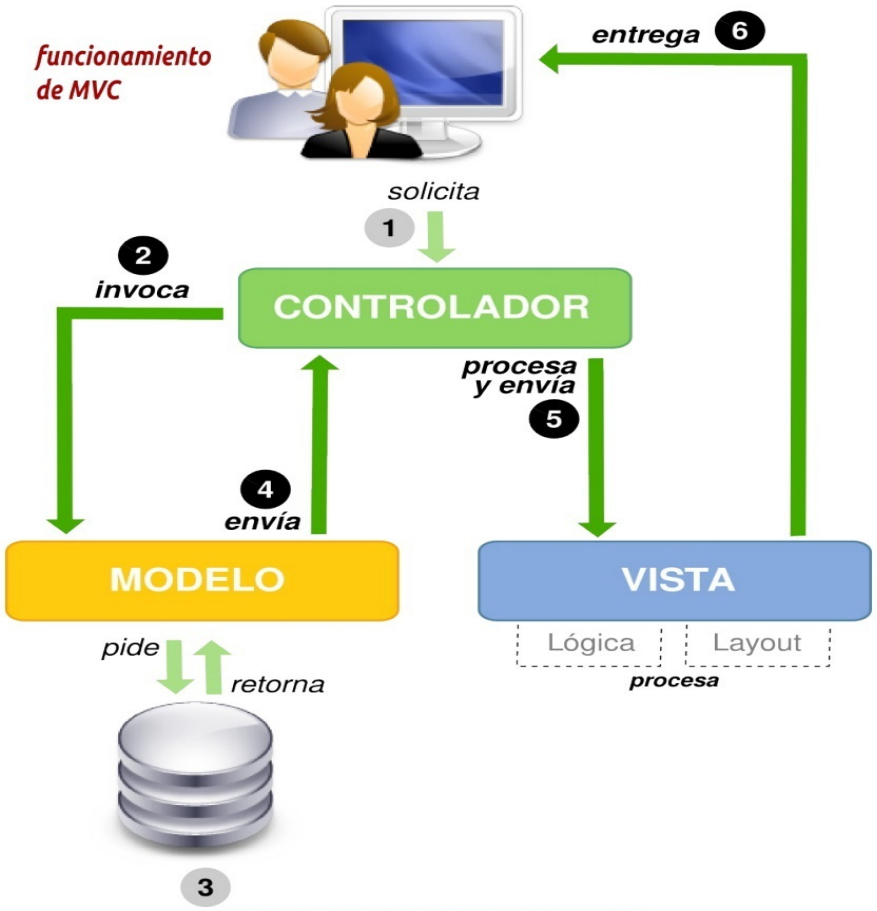

Figura 3.9: Funcionamiento del patrón MVC

3.8- Diagrama de secuencia

El diagrama de secuencia es un tipo de diagrama usado para modelar interacción entre objetos en un sistema, muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso, además contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes intercambiados entre los objetos.

En la figura 3.9, se aprecia el diagrama de secuencia del caso de uso “Gestionar Azimut Buzamiento”, los demás se encuentran en el **Anexo 4.**

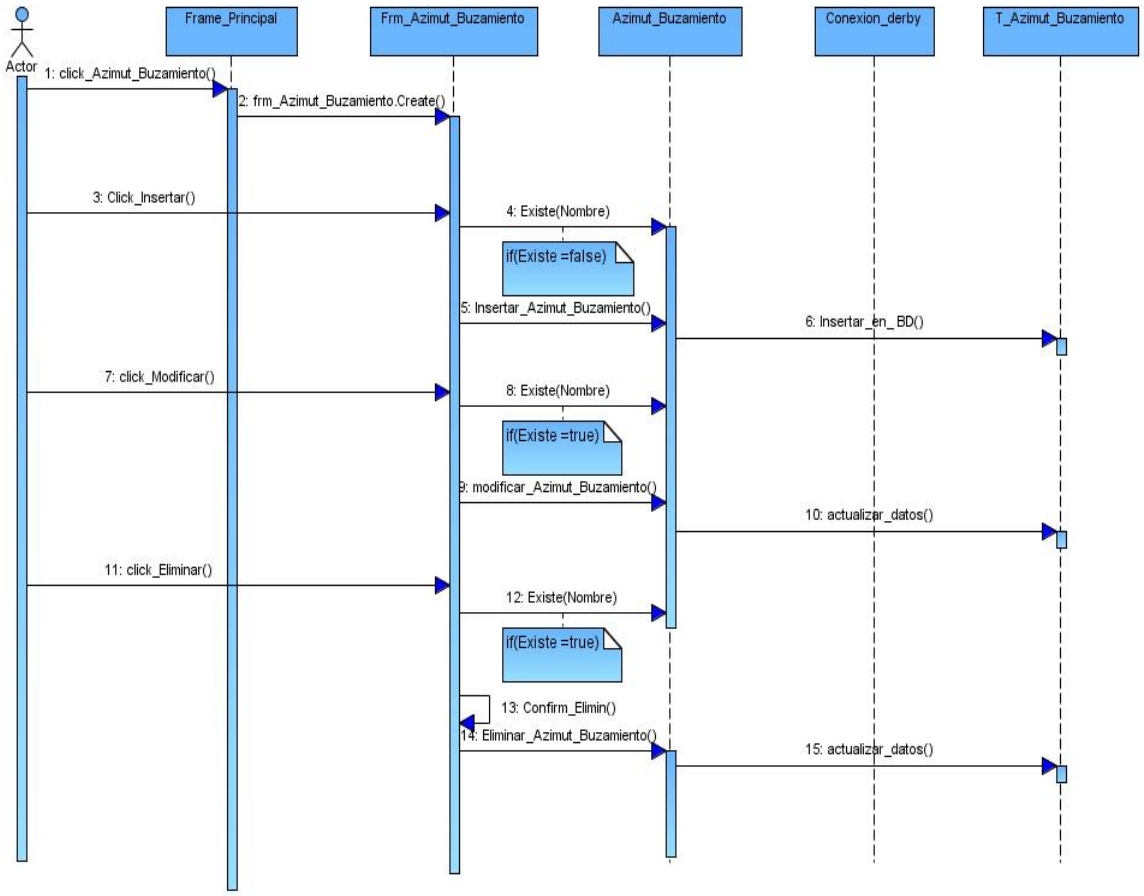

Figura 3.10: Diagrama de Secuencia Caso de uso “Gestionar Azimut Buzamiento”

3.9- Modelo de despliegue

El Diagrama de Despliegue es un tipo de diagrama del Lenguaje Unificado de Modelado que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Los elementos usados por este tipo de diagrama son nodos (representados como un prisma), componentes (representados como una caja rectangular con dos protuberancias del lado izquierdo) y asociaciones. La mayoría de las veces el modelado de la vista de despliegue implica modelar la topología del hardware sobre el que se ejecuta el sistema, este diagrama ha sido diseñado para modelar muchos de los aspectos hardware de un sistema a un nivel suficiente para que un ingeniero software pueda especificar la plataforma sobre la que se ejecuta el software del sistema.

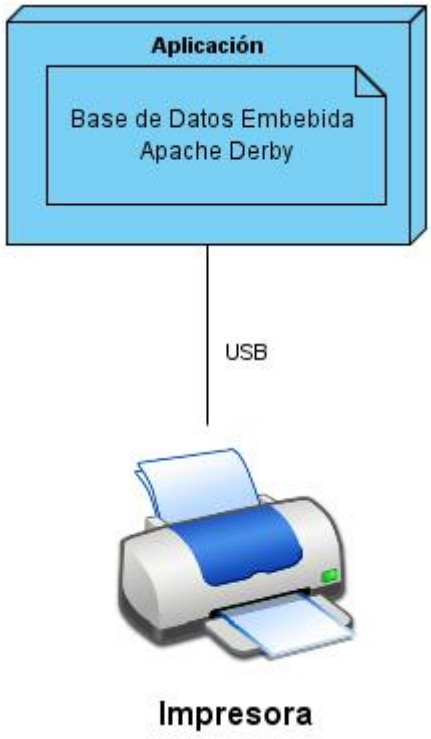

Figura 3.11: Diagrama del despliegue

3.10- Diagrama de Componente

Estos diagramas representan cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables, o paquetes, pueden ser usados para modelar y documentar cualquier arquitectura de sistema.

Debido a que los diagramas de componentes son más parecidos a los diagramas de casos de usos, éstos son utilizados para modelar la vista estática y dinámica de un sistema. Muestra la organización y las dependencias entre un conjunto de componentes. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. Cada diagrama describe un apartado del sistema. En él se situarán librerías, tablas, archivos, ejecutables y documentos que formen parte del sistema. Uno de los usos principales es que puede servir para ver qué componentes pueden compartirse entre sistemas o entre diferentes partes de un sistema.

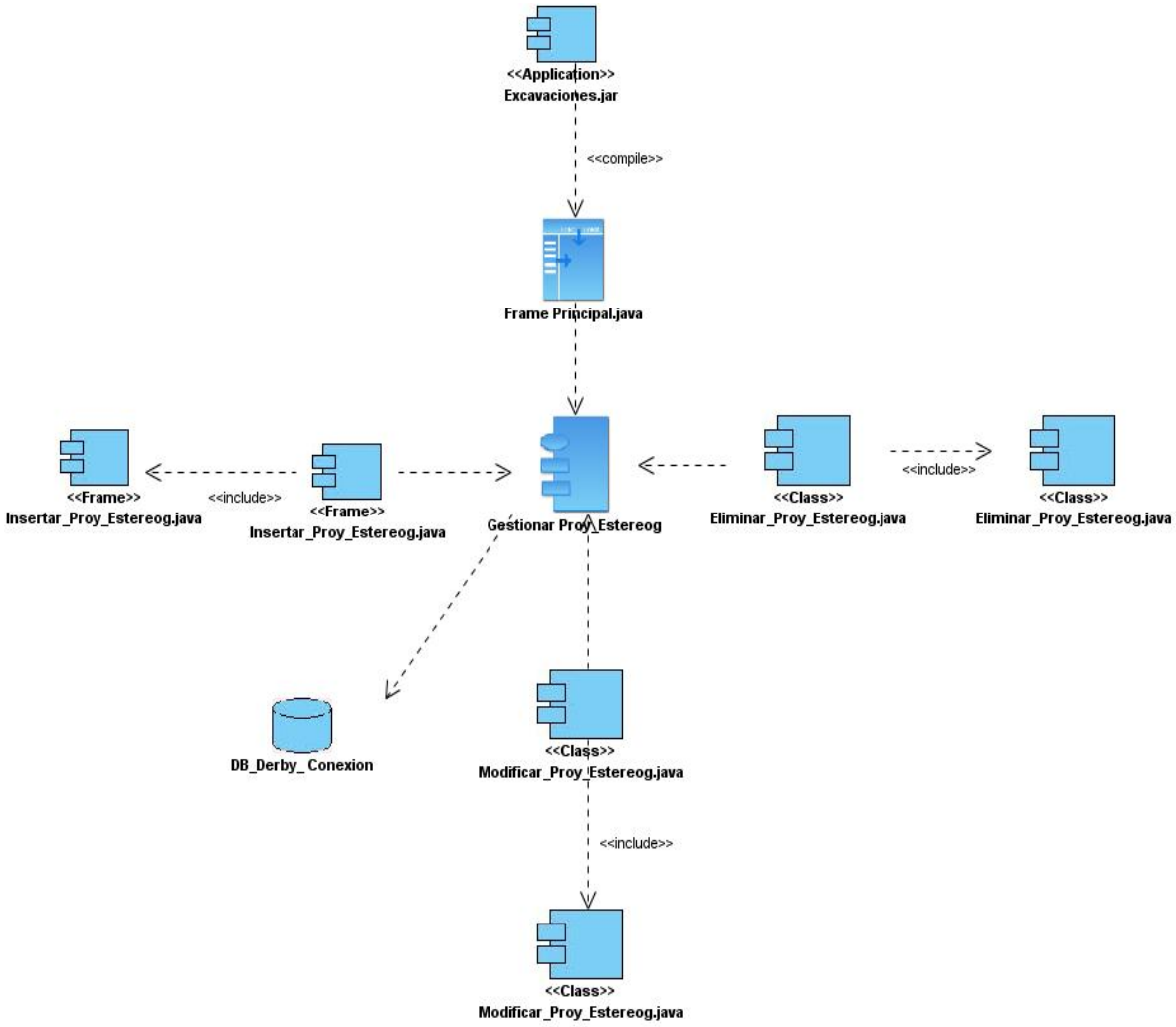

Figura 3.12: Diagrama de componente CU "Gestionar Proyección Estereográfica"

Conclusiones del capítulo

En este capítulo se comenzó a desarrollar la propuesta de solución, obteniéndose a partir del análisis, un listado con las funciones que debe tener el sistema, que se representaron mediante un Modelo de Dominio, y finalmente se describieron paso a paso todas las acciones de los actores del sistema con los casos de uso con los que interactúan. Además se marcan pautas importantes de la construcción del sistema, llegándose al modelo de datos que soportará todo el manejo de información de la aplicación, e incluso a una representación de la arquitectura de componentes de la misma. Se definió un modelo de tres capas que proporciona amplias ventajas en su desarrollo.

Capítulo IV: Estudio de Factibilidad

Para llevar a cabo la realización de un producto informático, se hace necesario realizar una planificación del mismo, esto es, un temprano estudio de los gastos económicos en que se incurrirá, los recursos humanos de que se dispone, así como el tiempo que demorará la culminación del mismo, para luego, en dependencia de los resultados arrojados por este estudio, poder llegar a conclusiones efectivas en cuanto a la factibilidad de producción de dicho producto. Existen diversas teorías y técnicas de estimación de esfuerzo (coste) en las que se puede apoyar la planificación, dentro de las cuales se destaca el COCOMO (Constructive Cost Model), que es en la que se basa el presente estudio. Se abordarán además en este capítulo otros aspectos de vital importancia como los son los beneficios tangibles e intangibles que reporta la construcción del presente sistema, y el análisis de costo – beneficio que permitirá sacar conclusiones más acertadas en cuanto a la factibilidad del producto informático.

4.1- COCOMO II.

El COCOMO surgió para medir y calcular el coste y el tiempo de un determinado proyecto basándose fundamentalmente en las líneas de código y algunas constantes. COCOMO II es un modelo que permite estimar el coste, esfuerzo y tiempo cuando se planifica una nueva actividad de desarrollo de software.

Este método posee tres modelos denominados **Composición de Aplicación, Diseño Temprano y Post Arquitectura**, cada uno orientado a sectores específicos del mercado de desarrollo de software y a las distintas etapas.

El **Modelo Composición de Aplicación**, es utilizado en los proyectos que se construyen a partir de componentes pre-empaquetados. En este caso, se emplean puntos objetos para estimar el tamaño del software, lo cual está acorde al nivel de información que generalmente se tiene en la etapa de planificación, y el nivel de precisión requerido en la estimación de proyectos de esta naturaleza.

El **Modelo Diseño Temprano** se emplea en las primeras etapas del desarrollo en la que se evalúan las alternativas de hardware y software de un proyecto. En esta etapa se tiene poca información, lo que concuerda con el uso de puntos de función, para estimar tamaño y el uso de un número reducido de factores de costo.

El **Modelo de Post Arquitectura** se aplica en la etapa de desarrollo, después que se define la arquitectura del sistema y en la etapa de mantenimiento. Es el más detallado.

El Modelo que se utiliza es el Post Arquitectura, escogido entre los demás, porque brinda valores más exactos y se realiza en una fase avanzada en el desarrollo.

Pasos a seguir en el método Post Arquitectura:

1. Obtener los puntos de función (UFP).
    a. Identificación de las características.
    b. Clasificación.
    c. Ponderación aplicando pesos.
2. Estimar la cantidad de instrucciones fuente. (SLOC).
    a. Utilizar tabla de lenguajes.
3. Aplicar las fórmulas de Bohem.
    a. Obtener esfuerzo (PM) y tiempo (TDEV).
    b. Costo del Proyecto

4.2- Desarrollo del método Post Arquitectura

4.2.1- Paso 1: Obtener los puntos de función

Seguidamente se muestran las funcionalidades del sistema, las cuales se agrupan en: Entradas externas, Salidas externas, Consultas, Ficheros lógicos internos y Ficheros de interfaz externa y se clasifican según su nivel de complejidad en: Simple, Media y Compleja.

4.2.1.1- Entradas Externas

Entradas externas	Cantidad de ficheros	Cantidad de elementos de datos	Clasificación (Simple, Media, Compleja)
Insertar Proyecto	2	4	**Simple**
Eliminar Proyecto	2	1	**Simple**
Insertar Proyección Estereográfica	1	2	**Simple**
Eliminar Proyección Estereográfica	1	2	**Simple**
Calcular Barton	2	14+	**Media**
Calcular Bieniawski	1	15+	**Compleja**
Calcular Cuesta	4	15+	**Compleja**
Calcular Número Familia Grietas	2	2	**Simple**
Cantidad de ficheros			
Simple	**Media**		**Compleja**
5	**1**		**2**

Tabla 4.1: Entradas Externas

4.2.1.2- Salidas externas

Salidas externas	Cantidad de ficheros	Cantidad de elementos de datos	Clasificación (Simple, Media, Compleja)
Mostrar Reporte Barton	2	14+	**Media**
Mostrar Reporte Bieniawski	1	15+	**Compleja**
Mostrar Reporte Cuesta	4	15+	**Compleja**
Mostrar Reporte Proyección Estereográfica	2	2+	**Simple**
Mostrar Nomograma Barton	1	2	**Simple**
Cantidad de ficheros			
Simple	**Media**	**Compleja**	
2	**1**	**2**	

Tabla 4.2: Salidas externas

4.2.1.3- Ficheros Lógicos Internos

Ficheros lógicos internos	Cantidad de records	Cantidad de elementos de datos	Clasificación (Simple, Media, Compleja)
Barton	1	14+	**Simple**
Bieniawski	1	15+	**Simple**
Cuesta	1	15+	**Simple**
Proyección Estereográfica	1	2	**Simple**
Cantidad de ficheros			
Simple	**Media**	**Compleja**	

4	**0**	**0**

Tabla 4.3: Ficheros Lógicos Internos

4.2.1.4- Puntos de función desajustados

Elementos	Simple		Media		Compleja		total
	No.	XPeso	No.	XPeso	No.	XPeso	
Entradas Externas	7	3	1	4	2	6	**37**
Salidas Externas	2	3	1	5	2	7	**25**
Ficheros Lógicos Internos	4	7	0	10	0	15	**28**
Total	**13**		**2**		**4**		
Puntos de Función Desajustados (UFP) Total							**90**

Tabla 4.3: *Puntos de función desajustados*

R/ El total de puntos de función desajustados (UFP) que se obtuvo es de 90.

4.2.2- Paso 2. Estimar la cantidad de instrucciones fuente. (SLOC)

Para el cálculo de las instrucciones fuentes (SLOC) se utiliza la ecuación:

SLOC= UFP * ratio. **(1)**

Donde:

UFP: Total de puntos de función desajustados, (calculados en el paso anterior).

Ratio: constante para las SLOC de cada lenguaje de programación en este caso tiene un valor para Java de 53.

Luego:

SLOC: Líneas de código fuente.

KSLOC: Miles de línea de código.

Sustituyendo los valores UFP = 90 y ratio =53 en la ecuación **(1)** tendríamos:

SLOC= 4770

Obtenido el valor de SLOC calculamos KSLOC mediante la ecuación **(2).**

KSLOC = SLOC/1000. **(2)**

= 4770 /1000

= 4,77

R/ EL valor obtenido de KSLOC es de 4,77.

4.2.3- Paso 3: Aplicando las fórmulas de Bohem

Una vez realizado el cálculo de la cantidad de instrucciones fuentes se procede a calcular el esfuerzo (PM)**.**

- **Cálculo del esfuerzo (PM).**

$PM_{NS} = A \times (Size)^{E} \times \prod_{i=1}^{n} EM_i$ **(3)**

Donde:

$E = B + 0{,}01 \times \sum_{j=1}^{5} SF_j$ **(4)**

Se conoce que:

A = 2,94 (constante).

B = 0.91 (constante).

Size: Tamaño estimado KSLOC.

EMi: Multiplicadores de Esfuerzo.

SF_j : Factores de escala.

Los valores de los factores de escala y los 17 multiplicadores de esfuerzo se muestran en la **Tabla 4.4** y **Tabla 4.5** respectivamente.

Tabla 4.4: Factores escala

Factor de escala	Valor	Justificación
PREC (Precedencia)	3,72	Se considera con algunos aspectos novedosos para el desarrollador.
FLEX (Flexibilidad)	2,03	Hubo cierto grado de acuerdo con los requerimientos pre-establecidos y con las interfaces externas.
RESL (Riesgos)	2,83	Se consideraron muchos de los riesgos críticos y se establecieron hitos para resolverlos.
TEAM(Cohesión del Equipo)	0,00	No se requiere más de un desarrollador para la aplicación.
PMAT (Madurez de las capacidades)	4,68	Se estima un nivel de madurez normal para el desarrollo de la aplicación.

Tabla 4.5: Valores específicos de los multiplicadores de esfuerzo

No.	Multiplicadores	Valor	Justificación
1	**RELY**	1,10	El cliente exige un producto que tenga un grado de fiabilidad muy alto.
2	**DATA**	0,90	El tamaño de la base de datos es pequeño porque tiene solamente cuatro tablas.
3	**CPLX**	1,00	La complejidad del producto es normal porque a pesar de tener una base de datos pequeña los gráficos le dan un poco de complejidad al producto.
4	**RUSE**	1,15	Es muy alta la reusabilidad de código en la implementación de la aplicación.

5	**DOCU**	1,00	La documentación es normal, por la cantidad de artefactos que se deben documentar.
6	**TIME**	1,00	Las exigencias de la capacidad en ejecución son normales.
7	**STOR**	1,00	Las exigencias de almacenamiento del sistema son normales.
8	**PVOL**	0,87	Es poco volátil porque no se modifica en prolongados períodos de tiempo.
9	**ACAP**	0,85	Se considera que la capacidad del analista es elevada.
10	**PCAP**	1,00	Se considera que la capacidad del programador es normal.
11	**PCON**	1,29	La volatilidad de la persona es muy baja.
12	**APEX**	1,10	Existe una experiencia previa limitada en el área de la aplicación.
13	**PLEX**	0,85	Existe un conocimiento muy alto en aplicación NetBeans.
14	**LTEX**	1,00	Existe una experiencia normal con las herramientas utilizadas.
15	**TOOL**	0,78	Hay variedad en la utilización e herramientas de software por lo tanto es elevado el uso de las mismas.
16	**SITE**	1,22	No existe desarrollo de la aplicación en localidades distribuidas.
17	**SCED**	1,00	Las exigencias sobre el calendario son normales, como cualquier producto de software, entre más rápido sea el software en

		brindar una respuesta más satisfecho queda el cliente.

Como resultado obtenemos:

$\sum_{j=1}^{5} SF_j$ = 13,86. y $\prod_{i=1}^{n} EM_i$ = 1,00

Sustituyendo el valor de $\sum_{j=1}^{5} SF_j$ en la ecuación **(4)** obtendríamos E:

E = 0,91 +0,01 x 13,86

= 1,0486

Nos quedaría sustituir E, en la ecuación **(3)**:

PM_{NS} = 2,94 x (4,77) ^{1,0486} x 1,00

= 15,1300 hombres-mes

R/ Para realizar el software en un mes se necesita el esfuerzo de 15.13 hombres.

- **Cálculo de tiempo de desarrollo (TDEV).**

$TDEV_{NS} = C \times (PM_{NS})^{F}$ **(5)**

Donde:

$F = D + 0,2 \times 0,01 \times \sum_{j=1}^{5} SF_j$ **(6)**

= D + 0,2 x (E-B)

Se conoce que:

C= 3,67 y D=0,28. (Valores constantes)

Calculamos F y Tiempo de desarrollo.

F = 0,30772

$TDEV_{NS} = 3,67 * 15,1300^{0,30772}$

= 8,4669 meses.

R/ El tiempo de duración del proyecto sería de 8,4669 meses aproximadamente.

- **Cantidad de hombres**

Una vez calculado el tiempo de desarrollo y el esfuerzo, se calcula la cantidad de personas (CH) que son necesarias para el desarrollo del software. En la tabla 4.7 se muestran los valores para el cálculo.

CH = PM/TDEV **(7)**

= 15,1300 / 8,4669

= 1,7869

Tabla 4.6: Cantidad de hombres

Siglas	Indicador	Valor o fórmula
CH	**Cantidad de hombres por mes**	**PM/TDEV**
PM	**Esfuerzo.**	**15,1300**
TEDV	**Tiempo de desarrollo**	**8,4669**

Son necesarias 2 personas para realizar el software en 8 meses y como 1 persona es la que trabaja, se reajustan los cálculos para este valor:

CH * = 1 persona

TEDV = PM / CH*

= 15,1300 / 1

= 15,1300 meses.

R/ Son necesarias 15,13 meses para que una persona desarrolle el software.

- **Costos del Proyecto.**

  El costo lo calculamos por la ecuación siguiente:

  C = CHM * PM **(8)**

  Se conoce que:

  C: Costo del proyecto

  SP: Salario promedio.

  CHM: Costo por hombre-mes

  Donde:

  CHM =1* SP. **(9)**

  Calculando obtenemos:

  CHM = 1*365 = $ 365,00

  C = 365,00 * 15,1300 = $ 5 522,45 CUP.

R/ El costo real del proyecto es de $ 5 522,45 CUP.

4.2.4- Beneficios tangibles e intangibles

Con el desarrollo de esta aplicación se obtiene una herramienta informática, capaz de realizar la gestión de datos obtenidos en una investigación minera de forma rápida y eficiente, agilizando este proceso y el de la Clasificación Geomecánica de Macizos Rocosos. Se logra además que las excavaciones subterráneas se desarrollen con un alto grado de precisión, alcanzando así la explotación sostenible de los recursos minerales.

Se ahorran materiales básicos como papel, lápiz e instrumentos de medición. Se reduce un alto por ciento de cometer errores humanos. La información mostrada las metodologías y el gráfico de proyección estereográfica, ya no será necesario utilizar cálculos manuales para la obtención de estos resultados. Se incrementa el conocimiento de los recursos existentes.

Se gana en velocidad y precisión a la hora de obtener los resultados.

4.2.5- Análisis de costos y beneficios

El costo de desarrollo del sistema está valorado en $ 5 522,45 CUP según los estudios de factibilidad realizados a través del modelo COCOMO II.

Este producto que se obtuvo como resultado de la investigación, no aporta ingresos de forma directa, pero el tiempo que tardan en desarrollarse los procesos que se automatizaron se reducen en un gran por ciento, y ese tiempo ganado puede ser aprovechado por el minero para emplearlo en otras tareas de la minería. Con la implantación del sistema, el departamento de geología-minería necesitará como mínimo una computadora, por lo que se incrementará un gasto de energía eléctrica. Pero a esa pérdida, le contrarresta la ventaja que propicia el uso de la herramienta (ahorro de materiales básicos, tiempo y esfuerzo). Una vez implantado, se eliminarán en un gran porciento los errores humanos.

Desde el punto de vista social-humanista, con el desarrollo del sistema se le facilitará en gran medida la realización de las actividades laborales al ingeniero en minas, pudiendo realizarlas con mayor rapidez y aumentando la confiabilidad en los datos. Se dan facilidades al usuario de realizar búsquedas de datos en tiempos mínimos y un conjunto de validaciones que evitan incurrir errores no deseados. Los tiempos de respuestas del sistema son mínimos. No se afecta la plantilla de los trabajadores, el sistema no pretende suplantar al trabajador sino agilizar su trabajo y hacerlo más eficiente.

Se entregará un manual de usuario donde el este podrá consultar con detalle el funcionamiento del sistema. Se espera que sean satisfechas las necesidades y expectativas de los clientes finales, debido a que se ha establecido una estrecha comunicación entre los desarrolladores de la aplicación y los usuarios finales.

Conclusiones del Capítulo

El estudio de Factibilidad desarrollado en este capítulo ha permitido calcular matemáticamente el comportamiento del desarrollo del proyecto por lo que es de suma importancia en la planificación y toma de decisiones concernientes al mismo. Además el análisis no solo se llevó a las particularidades del proceso de desarrollo, sino que también se tomaron en cuenta las condiciones con las que cuenta el usuario para la implantación y correcto funcionamiento del sistema.

Conclusiones Generales

Con el desarrollo del este proyecto se dio cumplimiento a los objetivos propuestos en esta investigación, arribándose a las siguientes conclusiones:

- Se desarrolló el software multiplataforma “Excavaciones” que permite la automatización del proceso de Clasificación Geomecánica de Macizos Rocosos.
- Se hizo un análisis crítico de los sistemas similares existentes determinándose que es más factible el empleo de una aplicación multiplataforma para la clasificación de macizos rocosos desarrollada dentro del país, lo que favorece el desarrollo de las políticas de migración a software libre trazadas.
- Se determinó el empleo de Java como lenguaje de programación, Apache Derby como Sistema de Gestión de Base de Datos; herramientas libres, que posibilitaron la realización de un software multiplataforma.
- Se realizó la Ingeniería de Software, basado en la metodología RUP, obteniéndose la arquitectura del sistema, lo que facilitó la realización del software, y a su vez demuestra las amplias potencialidades y ventajas que brinda esta metodología en la realización de aplicaciones de escritorio.
- Se analizaron los costos y beneficios, arribando a un costo total de $5.522,45 CUP, con lo cual se redujo de forma considerable los gastos que implicaban los cálculos manuales, para la realización de excavaciones subterráneas y otras operaciones a cielo abierto.

Recomendaciones

Aunque mucho se ha avanzado con el desarrollo de este trabajo, es necesario recomendar aspectos importantes para la Automatización Metodologías Geomecánicas de Clasificación de Macizos Rocosos. A partir de los resultados obtenidos de la presente investigación se recomienda con la necesidad de proponer:

- Utilizar el sistema propuesto como apoyo al departamento de Minería para el cálculo de las Metodologías Geomecánicas de Clasificación de Macizos Rocosos en el ISMMM.
- Continuar con el estudio para poder implementar otras de las metodologías de clasificación geomecánica que sean importantes en el estudio de la minería.
- Presentar el resultado obtenido en diferentes eventos.

Referencias Bibliográficas

(1). **AMERICATI**. *Ventajas y Desventajas: Comparación de los Lenguajes C, C++ y Java.* [En línea] 2006. [Citado el: 26 de Febrero de 2012.] http://www.americati.com.

(2). **Arquitectura**. *Programación por capas.* [En línea] [Citado el: 15 de Marzo de 2012.] http://es.wikipedia.org/wiki/Programacion_por_capas.

(3). **Avila.** [En línea] 2010. [Citado el: 15 de 10 de 2011.]

www.castem.com.pe/clasificacion-de-rocas.pdf.

(4). **Buschmann.** *Pattern-Oriented Software Architecture.* s.l. : John Wiley & Sons, 2007.

(5). ***Bases de Datos Empotradas (Embedded)***. [En línea] [Citado el: 20 de Enero de 2012.] http://techerald.com/page/base-de-datos-empotradas-embedded.html.

(6). **Bases de datos embebidas**. *Berkeley DB (BDB).* [En línea] [Citado el: 25 de Abril de 2012.] http://ktulu.com.ar/blog/2010/09/29/bases-de-datos-embebidas/.

(7). ***Conoce Firebird en 2 minutos***. [En línea] [Citado el: 25 de Abril de 2012.] http://www.firebirdnews.org/docs/fb2min_es.html.

(8). **Cuaresma, Sergi Blanco.** *Metodologías de desarrollo.* [En línea] [Citado el: 10 de Noviembre de 2011.] http://www.marblestation.com/?p=644.

(9). **Cuesta, Armando.** *Control de Excavaciones.* s.l. : ISMMM, 2011.

(10). **EcuRed**. *Proceso Unificado de Desarrollo.* [En línea] [Citado el: 15 de Noviembre de 2011.] http://www.ecured.cu/index.php/Proceso_Unificado_de_Desarrollo#Fases.

(11). ***EmbarcaderoER/Studio***. [En línea] [Citado el: 10 de Abril de 2012.]

http://bureaudeprensa.com/es/view.php?bn=bureaudeprensa_softwarre&key=1153755965

(12). **Hall, C.** *Drying technollogy An International Journal.* New York : s.n., 1983.

(13). **HURLIMANN.** [En línea] 2010. [Citado el: 14 de Enero de 2012.] http://www2.etcg.upc.edu/asg/geologia/pdf/pract2.pdf.

(14). **Intro Ingenieria Software**. *Microsoft Solution Framework 3.0 (MSF).* [En línea] [Citado el: 20 de Noviembre de 2011.] http://scruz334.blogspot.es/i2007-11/.

(15). **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El proceso Unificado de Desarrollo de Software.* Madrid : Addison Wesley : s.n., 2000. 84-7829-036-2.

(16). **Karl.** *Software Requirements.* s.l. : Microsoft Press, 2008.

(17). **LARMAN, C.** *UML y patrones. Introducción al análisis y diseño orientado a objeto.3 ed.* La Habana : Félix Varela, 2004.507 p.

(18). **LEGRÁ LOBAINA, A. y RAMÓN SILVA, O.** *La investigación Científica: Conceptos y Reflexiones.*

(19). **Larman, Craig.** *Applying UML and Patterns.* s.l. : Prentice Hall, 2008.

(20). **MARK.** *Learning Python, Fourth Edition.* s.l. : O'Reilly Media, 2010.

(21). ***NetBeans***. [En línea] [Citado el: 15 de Abril de 2012.]

http://es.wikipedia.org/wiki/NetBeans.

(22). **NEIRA, T.** La Industria Minera en Cuba. *Revista AreaMinera.* [En línea] 2009. [Citado el: 3 de Noviembre de 2011.] http//www.areaminera.com/contenidos/entrevistas.

(23). **Pérez, Damián.** Maestros del Web. [En línea] 2010. [Citado el: 24 de 5 de 2012.] http://www.maestrosdelweb.com/principiantes/¿que-son-las-bases-de-datos/.

(24). **Wikipedia**. *Apache Derby.* [En línea] [Citado el: 25 de Febrero de 2012.]

http://es.wikipedia.org/wiki/ApacheDerby.

(25). **Wikipedia.** [En línea] 2009. [Citado el: 26 de 1 de 2012.]

http://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado.

(26). **wordReference.** [En línea] 2012. [Citado el: 5 de 4 de 2012.]

http://www.wordreference.com/definicion/excavacion.

(27). **Vega.** El blog de Advl. [En línea] 2008. [Citado el: 6 de 2 de 2012.]

http://www.albertodevega.es/index.php/bases-de-datos-embebidas?blog=1.

(28). **Z, Emérito.** [En línea] 2009. [Citado el: 5 de 12 de 2011.]

http://es.wikipedia.org/wiki/Clasificación_geomecánica.

(29). **ZUKOWISKI, J.** *Programación Java2 J2SE 1.4.* La Habana : Félix Varela, 2006. Volumen 1.

Anexos

Anexo #1: Descripción de los casos de uso.

Tabla1. Descripción del caso de uso "Gestionar Metodología"

Caso de uso:	Gestionar Metodología
Actor:	Usuario (inicia)
Propósito:	Permite insertar y eliminar un proyectos.
Resumen:	El caso de uso inicia cuando el usuario avanzado escoge insertar o eliminar un proyecto. finaliza cuando el sistema realiza una de las operaciones seleccionadas anteriormente.
Referencia:	RF-1
Precondiciones:	Para la inserción: No debe existir ese proyecto. Para la eliminación: El proyecto debe estar registrado.
Poscondiciones:	Son guardados los cambios efectuado en la base de datos.
Requisitos Especiales	-

Tabla2. Descripción del caso de uso "Guardar Proyecto"

Caso de uso:	Guardar base de datos
Actor:	Usuario (inicia)
Propósito:	Permitir al usuario guardar el proyecto en el directorio que el desee.
Resumen:	El caso de uso inicia cuando el usuario crea un nuevo proyecto y el sistema muestra un dialogo para guardarlo en el directorio que escoja el usuario, o cuando ya existe un proyecto y el usuario escoge la opción guardar y el sistema

	muestra un dialogo para guardarlo en el directorio que escoja.
Referencia:	RF-2
Precondiciones:	Cuando se guarda con un nombre existente el sistema muestra la opción de reemplazar en este caso el sistema borra la carpeta existente y la reemplaza por otra del mismo nombre, en caso de no reemplazar el sistema te da la opción de no guardar con el mismo nombre y en caso de cancelar el sistema cerrará el diálogo y no hará nada.
Poscondiciones:	La base de datos será guardada con el nombre y en el directorio que especifique el usuario.
Requisitos Especiales	-

Tabla3. Descripción del caso de uso "Abrir Proyecto"

Caso de uso:	Abrir Proyecto
Actor:	Usuario (inicia)
Propósito:	Abrir el proyecto que el usuario desee y tener acceso a los proyectos de las diferentes metodologías.
Resumen:	El caso de uso inicia cuando el usuario escoge la opción abrir y el sistema muestra un diálogo para abrirlo desde el directorio que se encuentra, luego el sistema carga todos los proyectos que estén en esa base de datos.
Referencia:	RF-3
Precondiciones:	Cuando se intenta abrir un archivo con el formato no especificado para el sistema, el mismo muestra un cartel de error y termina.

Poscondiciones:	El sistema carga el proyecto escogido.
Requisitos Especiales	-

Tabla 4. Descripción del caso de uso "Gestionar Azimut_Buzamiento"

Caso de uso:	Gestionar Azimut_Buzamiento
Actor:	Usuario (inicia)
Propósito:	Permite que al usuario insertar, modificar y eliminar el Azimut_Buzamiento
Resumen:	El caso de uso inicia cuando el usuario necesita utilizar la proyección estereográfica, introduce, modifica o elimina los valor de Azimut y Buzamiento mediante un formulario.
Referencia:	RF-4
Precondiciones:	Para la inserción: El usuario debe haber entrado los valores numéricos Para la modificación: El usuario puede cambiar los valores que considere. Para la eliminación: El usuario puede eliminar los valores que considere innecesarios.
Poscondiciones:	Para la inserción: el sistema valida los datos insertados. Para la modificación: el sistema modifica los valores correspondientes. Para la eliminación: El elimina los valores del sistema.
Requisitos Especiales	-

Tabla5. Descripción del caso de uso "Graficar Proyección_Esterográfica"

Caso de uso:	Graficar Proyección_Estereográfica
Actor:	Usuario (inicia)
Propósito:	Graficar Proyección_Estereográfica
Resumen:	El caso de uso inicia cuando el usuario elige la opción Graficar, el sistema muestra los polos graficados con los valores de Azimut y Buzamiento culminando así este caso de uso.
Referencia:	RF-5
Precondiciones:	El usuario debe haber introducido valores de Azimut y Buzamiento.
Poscondiciones:	-
Requisitos Especiales	-

Tabla 6. Descripción del caso de uso "Calcular familia de grietas"

Caso de uso:	Calcular Familia de grietas
Actor:	Usuario (inicia)
Propósito:	Calcular familia de grietas
Resumen:	El caso de uso inicia cuando el sistema grafica, el usuario selecciona las áreas para conocer el número de familia de grietas que existen, el sistema devuelve el valor en el formulario y termina el caso de uso.
Referencia:	RF-6
Precondiciones:	El usuario debe haber seleccionado las áreas.
Poscondiciones:	El número de familia de grietas ha sido calculado y el sistema muestra el resultado y lo guarda en la Base de Datos.
Requisitos Especiales	-

Tabla 7. Descripción del caso de uso "Calcular metodología Barton"

Caso de uso:	Calcular metodología Barton
Actor:	Usuario (inicia)
Propósito:	Permite que el usuario calcule la calidad del macizo rocoso mediante la metodología de Barton.
Resumen:	El caso de uso inicia cuando el usuario escoge la metodología Barton e introduce los datos para el cálculo y luego presiona Calcular.
Referencia:	RF-7
Precondiciones:	El usuario debe haber entrado los valores dentro de los rangos correspondientes y no dejar campos vacíos
Poscondiciones:	El valor ha sido calculado y el sistema muestra el resultado mediante un gráfico y lo guarda en la Base de Datos.
Requisitos Especiales	-

Tabla 8. Descripción del caso de uso "Calcular metodología Bieniawski"

Caso de uso:	Calcular metodología Bieniawski
Actor:	Usuario (inicia)
Propósito:	Permite que el usuario calcule la calidad del macizo rocoso mediante la metodología de Bieniawski.
Resumen:	El caso de uso inicia cuando el usuario escoge la metodología Bieniawski e introduce los datos para el cálculo y luego presiona calcular.
Referencia:	RF-8
Precondiciones:	El usuario debe haber entrado los valores dentro de los rangos correspondientes y no

	dejar campos vacíos
Poscondiciones:	El valor ha sido calculado y el sistema muestra el resultado y lo guarda en la Base de datos.
Requisitos Especiales	-

Tabla 9. Descripción del caso de uso "Calcular metodología Cuesta"

Caso de uso:	Calcular metodología Cuesta
Actor:	Usuario (inicia)
Propósito:	Permite que el usuario calcule la calidad del macizo rocoso mediante la metodología de Cuesta.
Resumen:	El caso de uso inicia cuando el usuario escoge la metodología Cuesta e introduce los datos para el cálculo y luego presiona calcular.
Referencia:	RF-9
Precondiciones:	El usuario debe haber entrado los valores dentro de los rangos correspondientes y no dejar campos vacíos
Poscondiciones:	El valor ha sido calculado y el sistema muestra el resultado y lo guarda en la base de datos.
Requisitos Especiales	-

Tabla 10. Descripción del caso de uso "Mostrar Nomograma Barton"

Caso de uso:	Mostrar Nomograma Barton
Actor:	Usuario (inicia)
Propósito:	Mostrar el nomograma de Barton para que el usuario escoja el punto en el cual quiere

	conocer la calidad del macizo.
Resumen:	El caso de uso inicia cuando el usuario calcula la metodología de Barton, el sistema muestra el nomograma para que usuario elija el lugar en el gráfico donde quiere conocer la calidad del macizo.
Referencia:	RF-10
Precondiciones:	El usuario debe haber calculado la metodología de Barton.
Poscondiciones:	-
Requisitos Especiales	-

Tabla 11. Descripción del caso de uso "Mostrar reporte Barton"

Caso de uso:	Mostrar reporte Barton
Actor:	Usuario (inicia)
Propósito:	Mostrar el reporte de Barton que se obtiene a partir del nomograma.
Resumen:	El caso de uso inicia cuando el usuario escoge el punto en el nomograma el sistema muestra un botón para visualizar el reporte. El usuario da click sobre el botón y se visualiza el reporte.
Referencia:	RF-11
Precondiciones:	El usuario debe haber escogido el punto en el nomograma.
Poscondiciones:	-
Requisitos Especiales	-

Tabla 12. Descripción del caso de uso "Mostrar reporte Bieniawski"

Caso de uso:	Mostrar reporte Bieniawski
Actor:	Usuario (inicia)
Propósito:	Mostrar el reporte de Bieniawski
Resumen:	El caso de uso inicia cuando el usuario calcula la metodología de Bieniawski, el sistema visualiza el reporte con los datos calculados.
Referencia:	RF-12
Precondiciones:	El usuario debe haber calculado la metodología de Bieniawski.
Poscondiciones:	-
Requisitos Especiales	-

Tabla 13. Descripción del caso de uso "Mostrar reporte Cuesta"

Caso de uso:	Mostrar reporte Cuesta
Actor:	Usuario (inicia)
Propósito:	Mostrar el reporte de Cuesta
Resumen:	El caso de uso inicia cuando el usuario calcula la metodología de Cuesta, el sistema visualiza el reporte con los datos calculados.
Referencia:	RF-13
Precondiciones:	El usuario debe haber calculado la metodología de Cuesta.
Poscondiciones:	-
Requisitos Especiales	-

Tabla 14. Descripción del caso de uso "Imprimir reporte"

Caso de uso:	Imprimir reporte
Actor:	Usuario (inicia)
Propósito:	Imprimir el reporte de Barton, Bieniawski, Cuesta o Proyección_Estereográfica.
Resumen:	El caso de uso inicia cuando el sistema muestra un reporte en una ventana del IReport, y el usuario selecciona la opción imprimir.
Referencia:	RF-14
Precondiciones:	-
Poscondiciones:	-
Requisitos Especiales	-

Tabla 15. Descripción del caso de uso "Cerrar Proyecto"

Caso de uso:	Cerrar Proyecto
Actor:	Usuario (inicia)
Propósito:	Cerrar la conexión con la base de datos.
Resumen:	El caso de uso inicia cuando el usuario desea cerrar su base de datos, el sistema cierra el proyecto y guarda los datos.
Referencia:	RF-15
Precondiciones:	-
Poscondiciones:	-
Requisitos Especiales	-

Tabla 16. Descripción del caso de uso "Gestionar Metodología"

Caso de uso:	Gestionar Gestionar Metodología
Actor:	Usuario (inicia)
Propósito:	Permite al usuario insertar o eliminar proyectos para las diferentes metodologías.
Resumen:	El caso de uso inicia cuando el usuario necesita crear proyectos para las diferentes metodologías, introduce o elimina algún proyecto mediante un formulario.
Referencia:	RF-4
Precondiciones:	Para la inserción: El usuario debe haber entrado un nombre para el nuevo proyecto. Para la eliminación: El usuario puede eliminar los valores que considere innecesarios.
Poscondiciones:	Para la inserción: el sistema valida los datos insertados. Para la eliminación: El elimina los valores del sistema.
Requisitos Especiales	-

Tabla 17. Descripción del caso de uso "Gestionar Proyección Estereográfica"

Caso de uso:	Gestionar Proyección Estereográfica
Actor:	Usuario (inicia)
Propósito:	Permite al usuario insertar, o eliminar los proyectos de Proyección Estereográfica.
Resumen:	El caso de uso inicia cuando el usuario necesita utilizar la proyección estereográfica, introduce o elimina los datos del nuevo proyecto de proyección estereográfica mediante un formulario.
Referencia:	RF-4

Precondiciones:	Para la inserción: El usuario debe haber entrado el nombre del proyecto. Para la eliminación: El usuario puede eliminar los valores que considere innecesarios.
Poscondiciones:	Para la inserción: el sistema valida los datos insertados. Para la eliminación: El elimina los valores del sistema.
Requisitos Especiales	-

Anexo 2: Descripción de las clases del Diseño.

Tabla 1: Descripción de la clase Numero_familia_Grietas

Nombre de la clase	Numero_familia_Grietas.
Tipo de clase:	Entidad
Resumen: esta clase permite calcular en número de familia de grietas.	
Atributos	**Tipo**
valor RQD	double
Caract_macizo_R	string
Responsabilidades	
Nombre.	**Descripción.**
Num_Fam_Grie()	Este método se encarga de calcular el número de familia de grietas

Tabla 2: Descripción de la clase Barton

Nombre de la clase	Barton
Tipo de clase:	Entidad
Resumen: esta clase permite calcular la calidad del macizo mediante el método de Barton.	
Atributos	**Tipo**
RQD	double
índ_núm_fam_G	int
índice_ rugosidades	double
índice_ alteración	double

índice _caudal	double
SRF	double
Responsabilidades	
Nombre.	**Descripción.**
Clasif_roca_barton() :	Permite clasificar según Barton un macizo rocoso.
+valor()	Este método de devuelve el valor del cálculo de la metodología de Barton.

Tabla 3: Descripción de la clase Cuesta

Nombre de la clase	**Cuesta**
Tipo de clase:	Entidad
Resumen: esta clase permite calcular la calidad del macizo mediante el método de Cuesta.	
Atributos	**Tipo**
Carac_HidroGeo	string
Carac_Agriet	string
RQD	double
Caract_Exc	string
Tipo_Exc_Plan	string
Responsabilidades	
Nombre.	**Descripción.**
Clasif_roca_Cuesta()	Permite clasificar según Cuesta un macizo

	rocoso.
+valor()	
Nombre de la clase	Cuesta
Tipo de clase:	Entidad

Tabla 4: Descripción de la clase Clasificacion_Geomecanica

Nombre de la clase	**Clasificacion_Geomecanica.**
Tipo de clase:	Controladora
Resumen: esta clase permite clasificar la calidad del macizo mediante los métodos geomecánicos.	
Atributos	**Tipo**
Propiedad_Matriz_Rocosa	string
Frecuencia_Tipo_Discont	string
Grado_Fracturación	string
Tamaño_Forma	string
Propiedades_Hidrogeologicas	string
Grado_Meteorizacion	string
Alteracion	string
Estado_tension_Insitu	string
Responsabilidades	
Nombre.	**Descripción.**
+Det_Clasificacion()	Permite determinar el tipo de clasificación por la que va a ser

calculado el macizo.

Tabla 5: Descripción de la clase Reporte

Nombre de la clase	Reporte
Tipo de clase:	Entidad
Resumen: Permite crear y obtener los reportes de cada metodología, resultado obtenido del cálculo de las mismas.	
Atributos	**Tipo**
nombre_metodología	string
titulo_proyecto	string
nombre_proyecto	string
nombre_tunel	string
nombre_operador	string
fecha	string
observaciones	string
resistencia_compresion_simple	string
descripcion_macizo	string
diaclasado	string
flujo_excavaciones	string
caract_discontinuidades	string
tipo_roca	string
resistencia_traccion	string
Responsabilidades	

Nombre.	**Descripción.**
Obtener_reporte()	Permite obtener los reportes de cada metodología.

Tabla 6: Descripción de la clase Propuesta_sostenimiento

Nombre de la clase	**Propuesta_sostenimiento**
Tipo de clase:	Entidad
Resumen: Permite obtener las propuestas de sostenimiento, según la calidad del macizo que se obtiene con cálculo de las metodologías.	
Atributos	**Tipo**
Long_Anlca	string
Sep_Anclas	string
Desc_Macizo	string
Responsabilidades	
Nombre.	**Descripción.**
Prop_Sosten()	Permite obtener la propuesta de sostenimiento para el túnel.

Tabla 7: Descripción de la clase Excavaciones

Nombre de la clase	**Excavaciones**
Tipo de clase:	Controladora
Resumen: Permite cargar todos los datos de las Clasificaciones Geomecánicas.	

Atributos	Tipo
Tipo_Exc	string

Tabla 8: Descripción de la clase DB_Derby_ Conexion

Nombre de la clase	DB_Derby_ Conexion
Tipo de clase:	Entidad
Resumen: Es la clase encargada de relacionar la aplicación con la base de datos embebida. Permite conectar, guardar, y desconectar, crear tablas y actualizar la base de datos Apache Derby.	
Atributos	**Tipo**
con	Connection
conecto	String
Responsabilidades	
Nombre.	**Descripción.**
Conectar (ruta: String, Params: String, creado: boolean)	Permite realizar la conexión de la base de datos.
Desconectar()	Este método se encarga de desconectar la base de datos.
crearTablas()	Es el encargado de crear las tablas en la base de datos.
Guardar (ruta: String, Params: String, creado: boolean	Este método se encarga guardar la base de datos.
Actualizar()	Este método se encarga actualizar la base de datos.

Tabla 9: Descripción de la clase Proyeccion_Estereográfica

Nombre de la clase	Proyeccion_Estereográfica
Tipo de clase:	Entidad
Resumen: Esta clase es la encargada de gestionar todo lo relacionado con la ubicación de polos en el plano, y seleccionarlos para que el usuario determine el número de familia de grietas.	
Atributos	**Tipo**
Azimut	double
Buzamiento	double
Fam_Griet	int
Num_polos	int
Responsabilidades	
Nombre.	**Descripción.**
Calc_Num_Griet()	Este método se encarga de calcular el número de familia de grietas.
Calc_Planos()	Este método se encarga de calcular los planos en los que se ubicarán los polos
Calc_polos()	Este método se encarga de calcular mediante la conversión de ángulos los polos que se ubicarán en el gráfico
Calc_Cent()	Este método se encarga de calcular los centroides de los polos.

Tabla 10: Descripción de la clase Bieniawski

Nombre de la clase	Bieniawski
Tipo de clase:	Entidad
Resumen: esta clase permite calcular la calidad del macizo mediante el método de Bieniawski.	
Atributos	**Tipo**
Estado _grieta	string
Correc_Orient_Griet	string
Agua_Freatica	string
Resist_MRocosa	double
Sep_Diaclasas	string
RQD	string
Responsabilidades	
Nombre.	**Descripción.**
Clasif_roca_bieniawski()	Permite clasificar según Bieniawski un macizo rocoso.
+valor()	Este método de devuelve el valor del cálculo de la metodología de Bieniawski.

Tabla 11: Descripción de la clase Graficar

Nombre de la clase	Graficar
Tipo de clase:	Entidad
Resumen: Esta clase es la encargada de graficar los polos, que el usuario debe agrupar para determinar el Número de Familia de Grietas.	
Atributos	**Tipo**
Azimut	double
Buzamiento	double
Responsabilidades	
Nombre.	**Descripción.**
Pintar()	Este método se encarga de graficar los polos mediante los valores de Azimut y Buzamiento.

Anexo # 3: Descripciones de las Tablas que Conforman el Modelo Físico de Datos

Tabla 1: Descripción de la tabla T_A_Cuesta

Nombre de la tabla	T_A_Cuesta	
Descripción.	Calcula y almacena los datos y valores escogidos por el usuario para la clasificación del macizo.	
Atributos	**Tipo**	**Descripción de los atributos.**
Carac_HidroGeo	string	Resultado de las características hidrogeológicas.
Carac_Agriet	string	Resultado de las características del agrietamiento
RQD	double	Índice de calidad de la Roca.
Caract_Exc	string	Resultado de las características de la excavación.
Tipo_Exc_Plan	string	Tipo de excavación y plan de servicio.

Tabla 2: Descripción de la tabla T_barton

Nombre de la tabla	T_barton	
Descripción.	Calcula y almacena los datos y valores escogidos por el usuario para la clasificación del macizo.	
Atributos	**Tipo**	**Descripción de los atributos.**
RQD	string	Descripción del Índice de calidad de la Roca.
índ_núm_fam_G	double	Índice de calidad de la Roca.
índice_ rugosidades	double	Resultado del índice de rugosidades.

índice_ alteración	double	Resultado del índice de alteración
índice _caudal	double	Resultado del índice de caudal.
SRF	double	Resultado del Factor de reducción de tensión.

Tabla 3: Descripción de la tabla T_bieniawski

Nombre de la tabla	**T_bieniawski**	
Descripción.	almacena los datos y valores escogidos por el usuario para la clasificación del macizo.	
Atributos	**Tipo**	**Descripción de los atributos**
Estado _grieta	string	Resultado del estado de la grieta.
Correc_Orient_Griet	string	Corrección de orientación de la grieta
Agua_Freatica	string	Resultado del agua Freática
Resist_MRocosa	double	Resultado de resistencia de la matriz rocosa.
Sep_Diaclasas	string	Separación entre diaclasas
RQD	string	Índice de calidad de la Roca

Tabla 4: Descripción de la tabla reporte

Nombre de la tabla	**T_reporte**	
Descripción.	Muestra los valores del cálculo de las metodologías Barton, Beniawski y A_Cuesta.	
Atributos	**Tipo**	**Descripción de los atributos**
nombre_metodología	string	Nombre de la metodología.
titulo_proyecto	string	Título del proyecto

nombre_tunel	string	Nombre del túnel donde va a efectuarse la excavación.
nombre_operador	string	Nombre del operador que toma los datos en el campo
fecha	string	Fecha en la que se realiza el reporte.
observaciones	string	Detalles de relevancia sobre la excavación.
resistencia_compresion_simple	string	Datos sobre la resistencia de compresión simple.
descripcion_macizo	string	Descripción del macizo rocoso.
diaclasado	string	Valor del diaclasado.
flujo_excavaciones	string	Valor del flujo de excavaciones
caract_discontinuidades	string	Características del las discontinuidades.
tipo_roca	string	Tipos de roca de la excavación.
resistencia_traccion	string	Detalles de la resistencia a tracción.

Tabla 5: Descripción de la tabla T_Propuesta_Sostenimiento

Nombre de la tabla	T_Propuesta_Sostenimiento	
Descripción.	Muestra las propuestas de sostenimiento que se obtienen a partir del reporte.	
Atributos	**Tipo**	**Descripción de los atributos**
Long_Anlca	string	Longitud de las anclas
Sep_Anclas	string	Separación entre las anclas
Desc_Macizo	string	Descripción del macizo.

Tabla 6: Descripción de la tabla T_Num_Fa_Griet

Nombre de la tabla	T_Num_Fa_Griet	
Descripción.	Muestra el número de familia de grietas que se obtienen con la selección de los polos.	
Atributos	**Tipo**	**Descripción de los atributos**
valor RQD	double	Índice de calidad de la roca.
Caract_macizo_R	string	Características del macizo rocoso.

Tabla 7: Descripción de la tabla T_Proyeccion_Estereograf

Nombre de la tabla	T_Proyeccion_Estereograf	
Descripción.	Almacenar los datos de la proyección estereográfica.	
Atributos	**Tipo**	**Descripción de los atributos**
Azimut	double	Valor del Azimut
Buzamiento	double	Valor del Buzamiento
Fam_Griet	int	Número de familia de grietas.
Num_polos	int	Número polos.

Tabla 8: Descripción de la tabla T_Clasificacion_Geomecanica

Nombre de la tabla	T_Clasificacion_Geomecanica	
Descripción.	Almacenar datos generales sobre la clasificación Geomecánica.	
Atributos	**Tipo**	**Descripción de los atributos**
Propiedad_Matriz_Rocosa	string	Propiedades de la matriz rocosa
Frecuencia_Tipo_Discont	string	Valor de la frecuencia del tipo de

		discontinuidades.
Grado_Fracturación	string	Grado de fracturación del macizo
Tamaño_Forma	string	Tamaño de la forma.
Propiedades_Hidrogeologicas	string	Propiedades hidrogeológicas del macizo.
Grado_Meteorizacion	string	Grado de meteorización del macizo.
Alteracion	string	Valor de la alteración.
Estado_tension_Insitu	string	Estado de tensión del macizo.

Anexo # 4: Diagramas de Secuencia de los Casos de Uso del Sistema

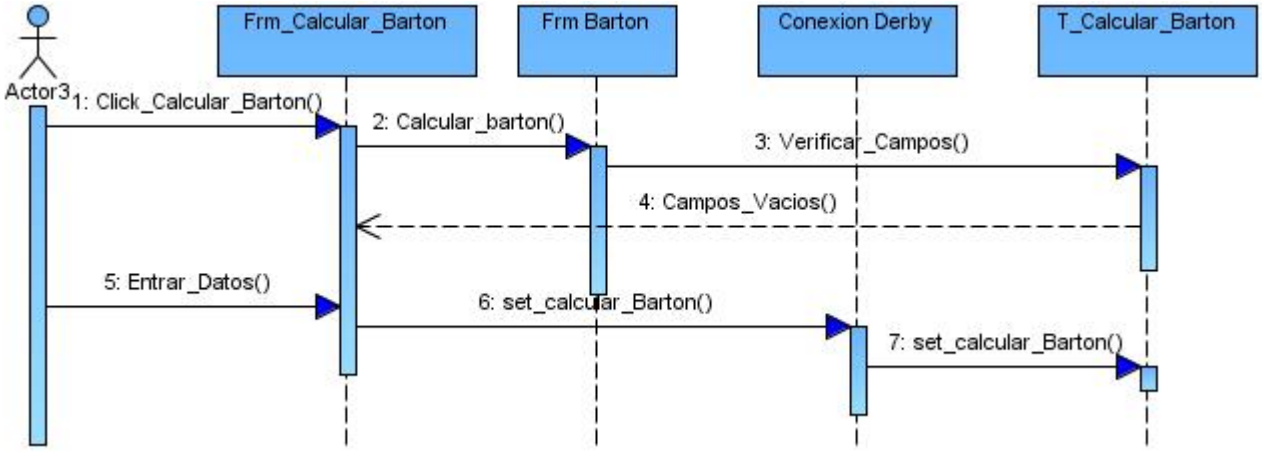

Figura 1. Diagrama de secuencia del CU "Calcular Barton"

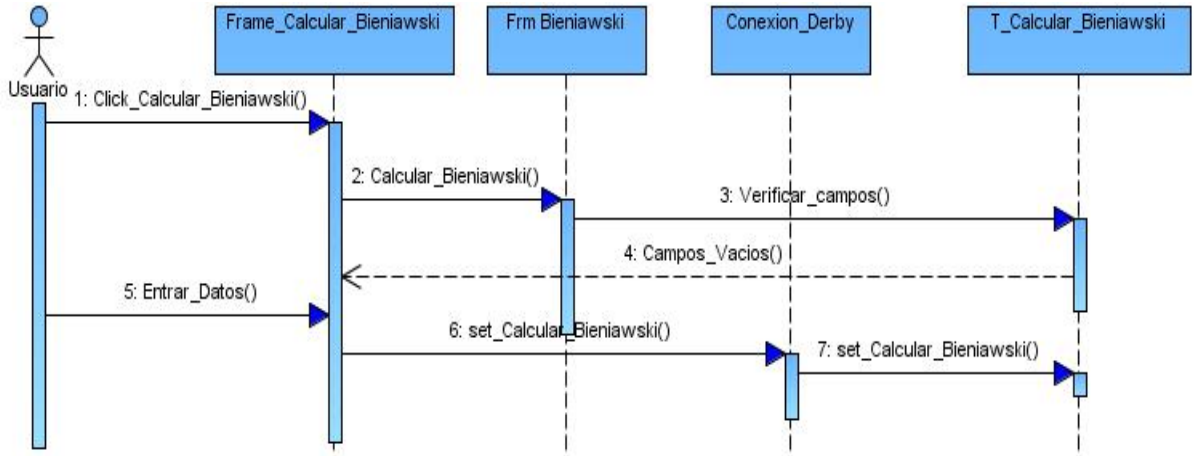

Figura 2. Diagrama de secuencia del CU "Calcular Bieniawski"

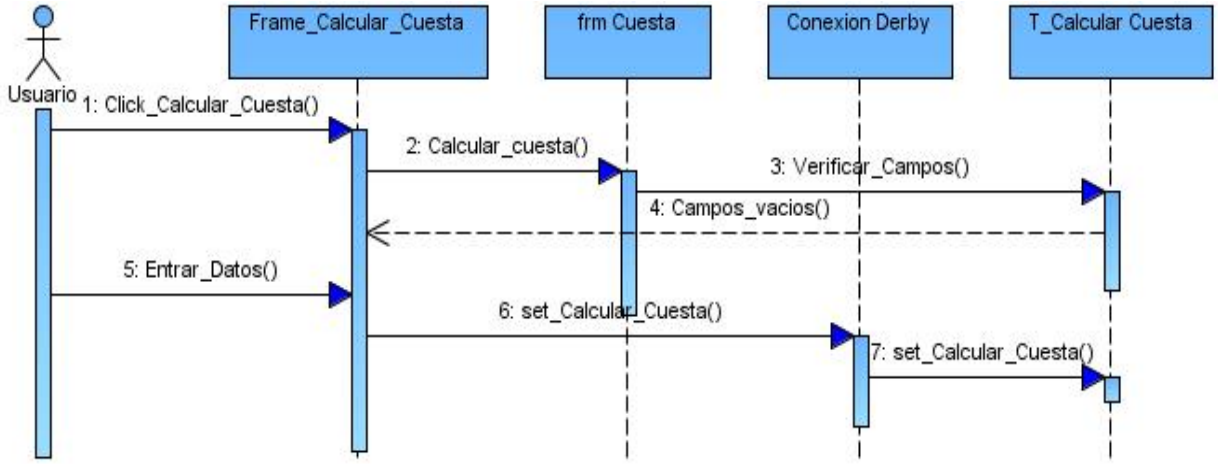

Figura 3. Diagrama de secuencia del CU "Calcular Cuesta"

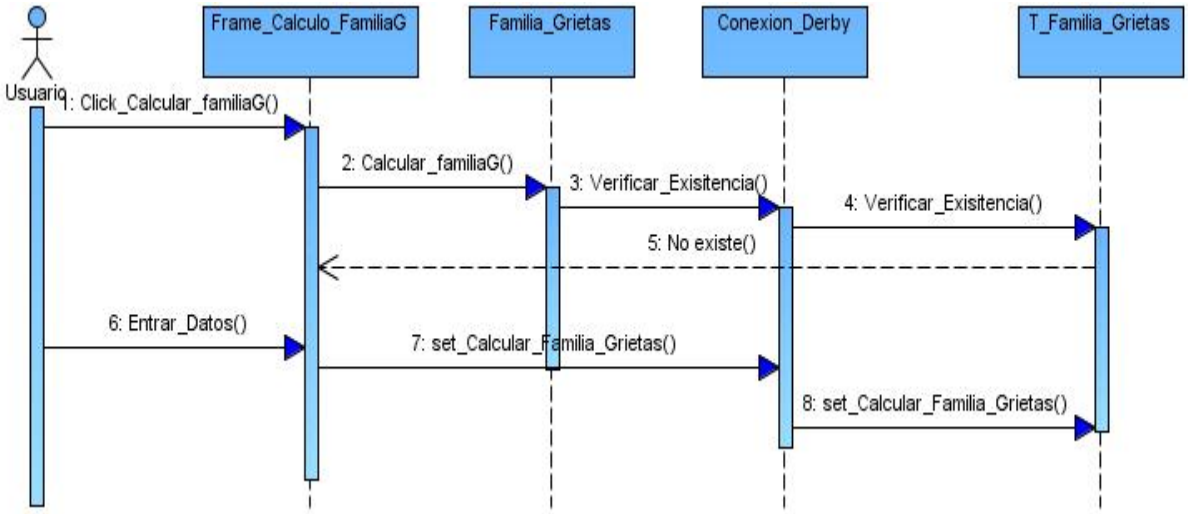

Figura 4. Diagrama de secuencia del CU "Calcular_Familia_griet"

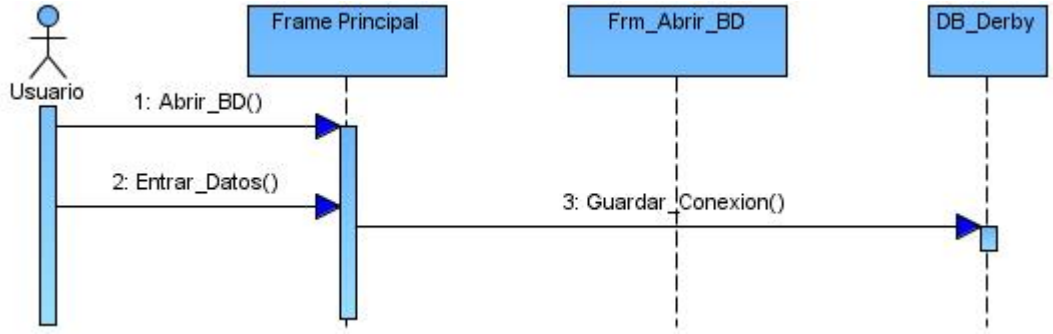

Figura 5. Diagrama de secuencia del CU "Abrir Proyecto"

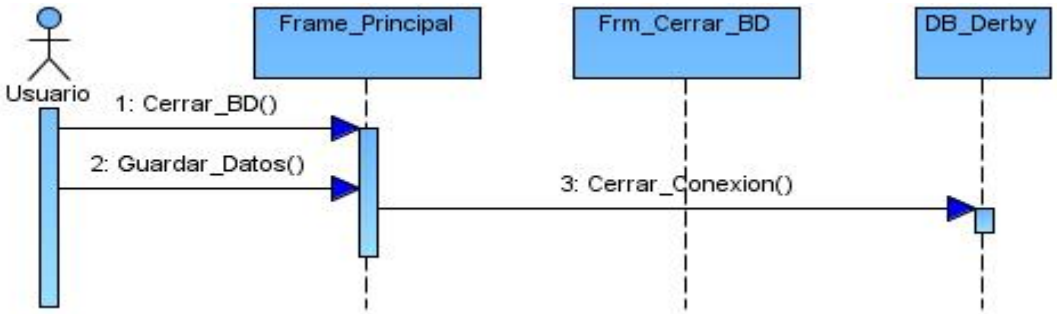

Figura 6. Diagrama de secuencia del CU "Cerrar Proyecto"

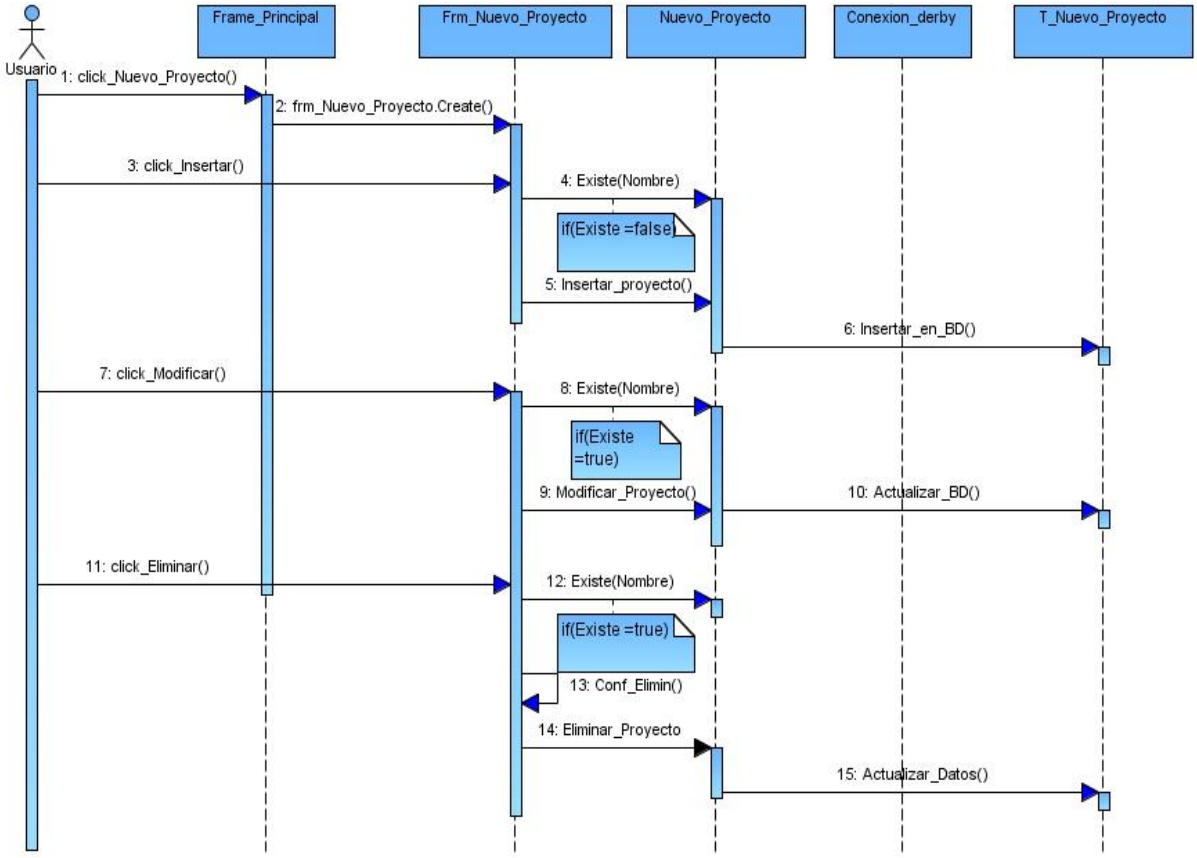

Figura 7. Diagrama de secuencia del CU "Gestionar_Metodología"

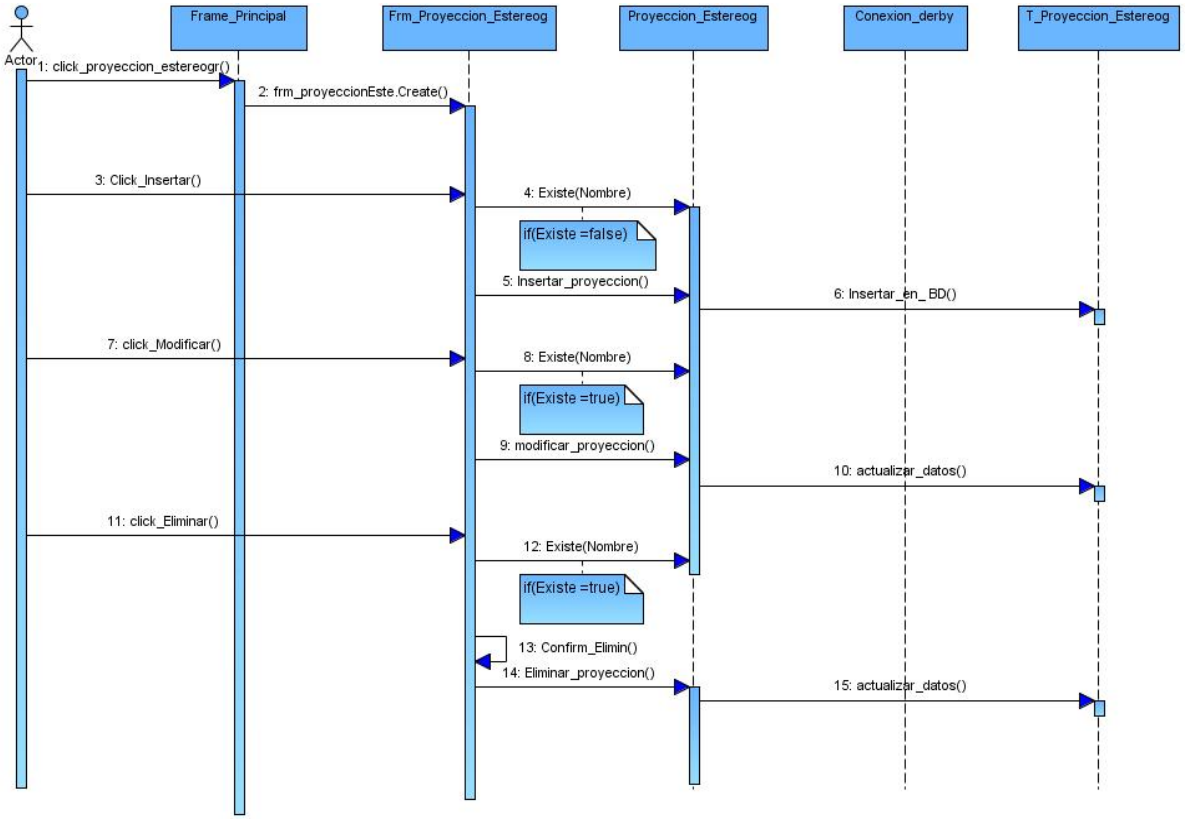

Figura 8. Diagrama de secuencia del CU "Graficar_Proy_estereog"

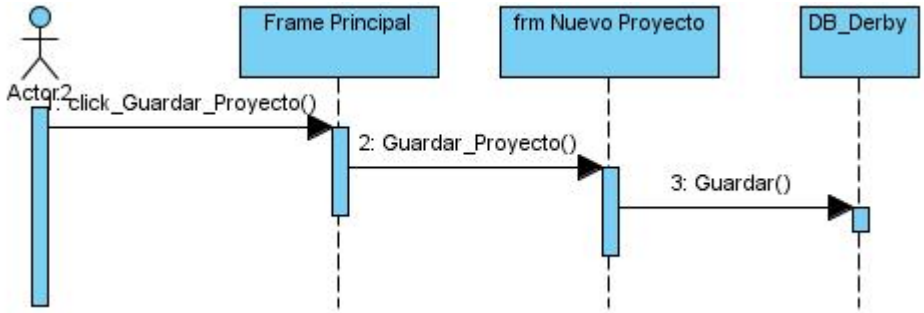

Figura 9. Diagrama de secuencia del CU "Guardar Proyecto"

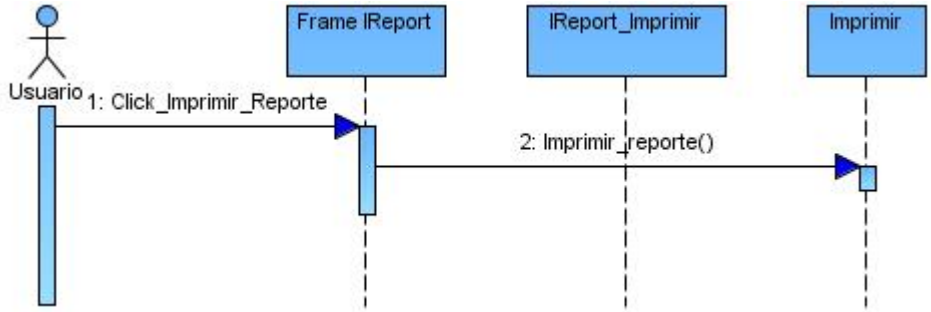

Figura 10. Diagrama de secuencia del CU "Imprimir_Report"

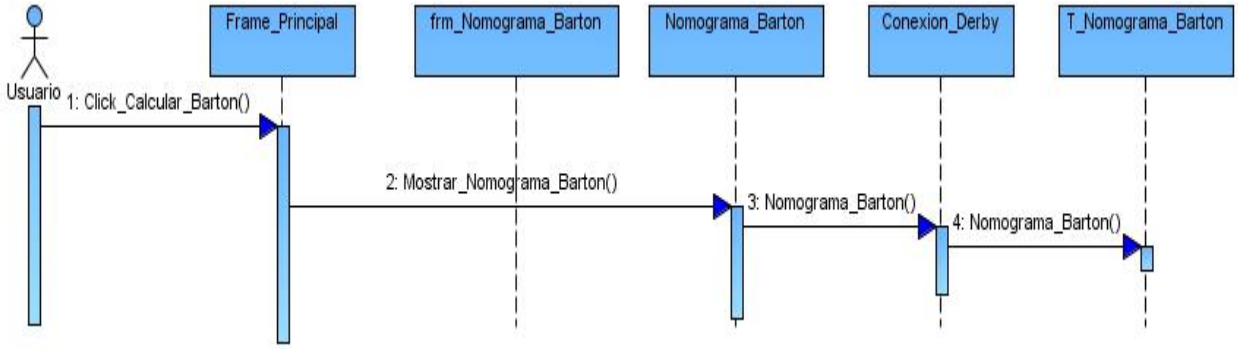

Figura 11. Diagrama de secuencia del CU "Mostrar_Nomogr_Barton"

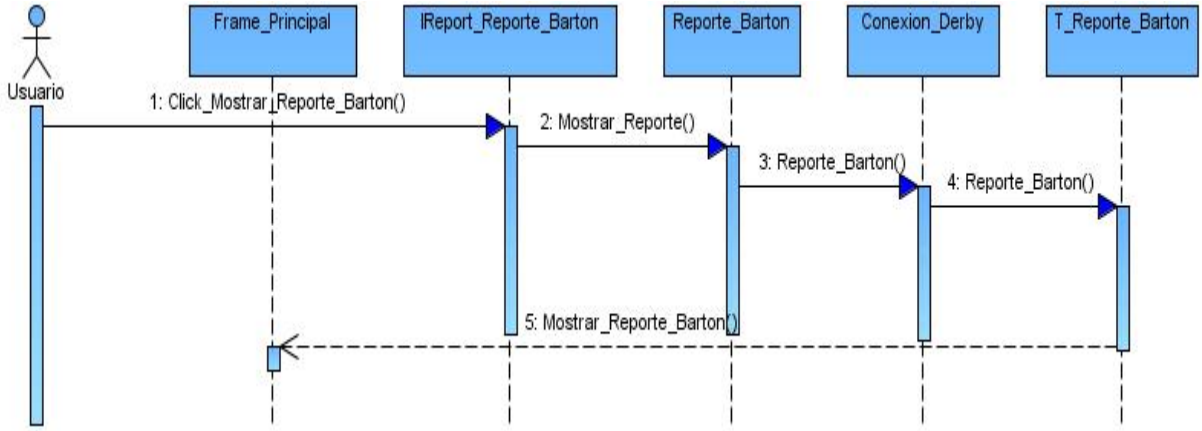

Figura 12. Diagrama de secuencia del CU "Mostrar_reporte_Barton"

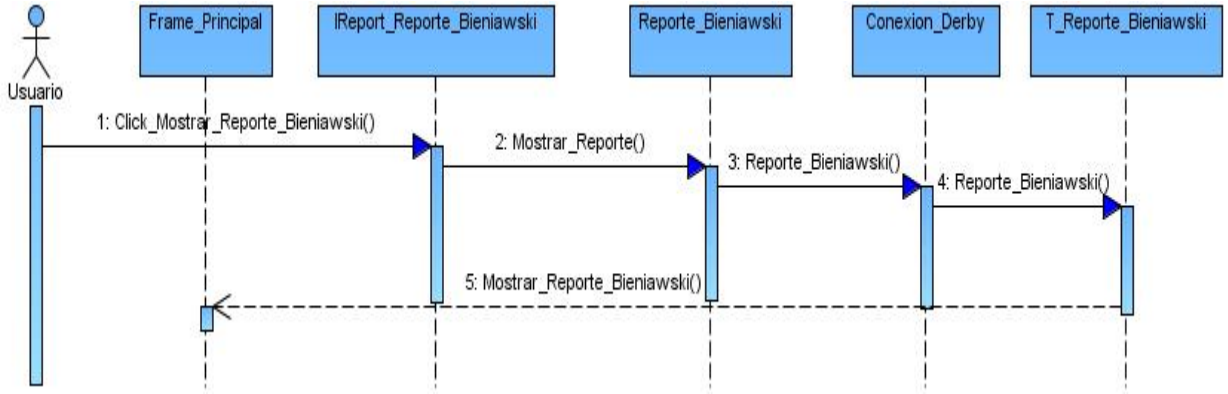

Figura 13. Diagrama de secuencia del CU "Mostrar_reporte_Bieniawski"

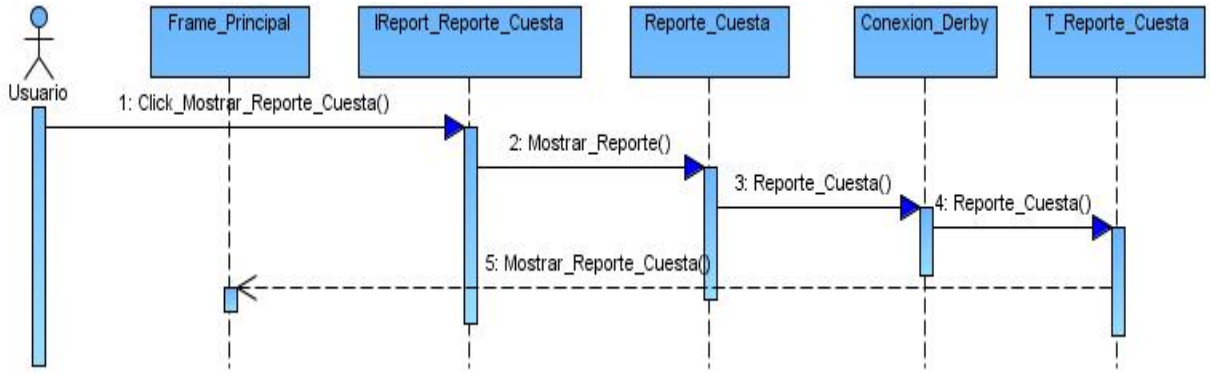

Figura 14. Diagrama de secuencia del CU "Mostrar_reporte_Cuesta"

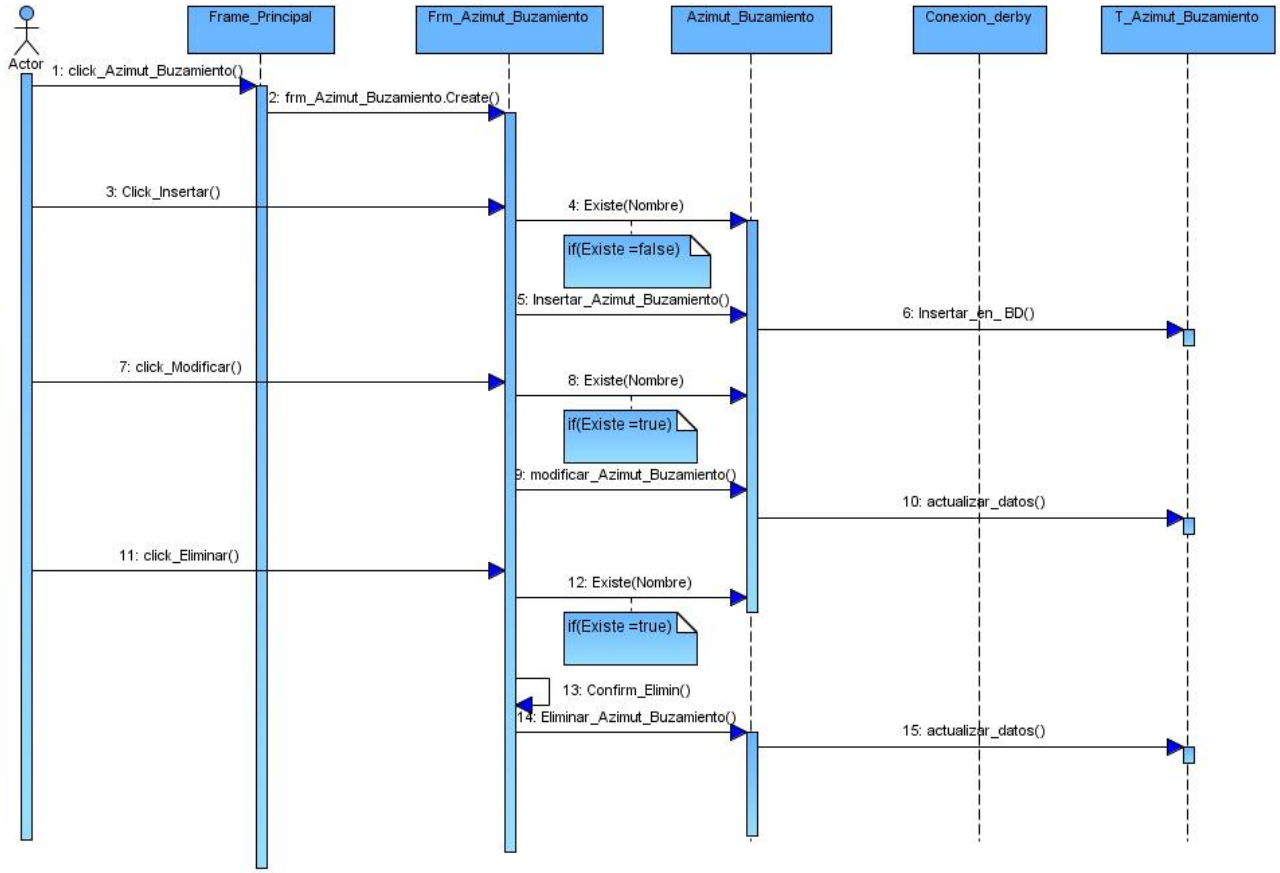

Figura 15. Diagrama de secuencia del CU "Gestionar_Azimut_Buzamiento"

Glosario de Términos

Aplicación: Es el programa que el usuario activa para trabajar en el ordenador. Existen muchos programas de ordenador que pueden clasificarse como aplicación. Generalmente se les conoce como Software.

API: Una API (del inglés Application Programming Interface - Interfaz de Programación de Aplicaciones) es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

JDBC: es el acrónimo de Java Database Connectivity, un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede utilizando el dialecto SQL del modelo de base de datos que se utilice.

RMI: (Java Remote Method Invocation) es un mecanismo ofrecido en Java para invocar un método remotamente. Al ser RMI parte estándar del entorno de ejecución Java, usarlo provee un mecanismo simple en una aplicación distribuida que solamente necesita comunicar servidores codificados para Java.

RUP: El Proceso Unificado Rational (RUP) es una metodología de desarrollo para la programación orientada a objetos. Según Rational (diseñadores de Rose Rational y el Idioma Modelado Unificado), RUP está como un mentor en línea que mantiene pautas, plantillas, y ejemplos de todos los aspectos y fases de desarrollo del programa. RUP es un software comprensivo que diseña herramientas que combinan los aspectos procesales de desarrollo (como las fases definidas, técnicas, y prácticas) con otros componentes de desarrollo (como los documentos, modelos, manuales, el código, y así sucesivamente) dentro de una armazón unificándose.

Metodologías de Desarrollo: Se define como un conjunto de filosofías, etapas, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de sistemas de información.

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.

Programación Extrema(XP): Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo.

Software Libre: Es el software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente, aunque conserve su carácter de libre, puede ser vendido comercialmente.