Universidad de las Ciencias Informáticas

Facultad 5

Módulo para el cálculo de parámetros petrofísicos en el Sistema de Análisis e Interpretación de Registros de Pozos AnPer

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor:
Yoandri Matos De La Cruz

Tutor:
Ing. Yunier Alexander Pimienta Fernández

Cotutores:
Ing. Alexey Díaz Domínguez
Ing. Danay Leyva Abrahantes

La Habana, Cuba

2013

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor del presente trabajo y autorizo a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

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

Yoandri Matos De La Cruz

_____

Firma del Autor

Ing. Yunier Alexander Pimienta Fernández

_____

Firma del Tutor

Dedicatoria

A Mercedes y Yanet, porque forman el regalo más increíble que me ha dado la vida…

A mi familia en general por su apoyo en todos los sentidos…

A mis amigos de ahora y de siempre…

Resumen

El Centro de Investigaciones del Petróleo (CEINPET) es el encargado de dar respuesta de forma integral a los procesos de exploración y producción de petróleo en Cuba. Dentro de las actividades que realiza se encuentra el análisis petrofísico de registros de pozos. Adquirir programas para el desarrollo de esta tarea resulta muy costoso. Estas soluciones pertenecen a empresas que comercializan software propietario. El CEINPET y la Universidad de Ciencias Informáticas (UCI) han iniciado el desarrollo del Sistema de Análisis e Interpretación de Registros de Pozos AnPer, software que pretende resolver las necesidades propias de la entidad cubana en el análisis de registros para la evaluación de yacimientos.

El presente trabajo muestra el desarrollo de un módulo para el cálculo de parámetros petrofísicos integrado en AnPer, sustentado por la necesidad de los especialistas de obtener parámetros que no se pueden medir directamente en los registros de pozos, sino que se infieren a partir de la información que estos contienen. Se brinda una herramienta que permite calcular parámetros mediante la interpretación de modelos petrofísicos, los cuales pueden estar definidos o ser codificados según la naturaleza del problema que resuelvan. Esto permitirá realizar una evaluación de los yacimientos obteniendo información relevante sobre el potencial económico de los mismos.

Palabras clave:

Curva, intérprete, lenguaje, modelo petrofísico, parámetro, registro de pozo.

Índice

Introducción

Por la magnitud y complejidad de los procesos de exploración-producción en la industria petrolera, la variedad de sistemas informáticos que se utilizan en esta rama se encuentra en ascenso. Los programas desarrollados para el área de análisis petrofísico, subproceso en la exploración, ocupan un lugar clave para la localización y evaluación de zonas de alto valor productivo. Su complejidad e importancia los ubican como soluciones que garantizan mantener claridad, confidencialidad y rapidez en los análisis estadísticos e indicadores, generación de reportes e integridad de la información.

En Cuba, el Centro de Investigaciones del Petróleo (CEINPET), es la empresa encargada de dar respuesta de forma integral a la actividad petrolera en el territorio nacional, dígase exploración, perforación, producción y refinación. Actualmente el centro necesita el uso de este tipo de sistema informático; sin embargo, las alternativas existentes no son las más factibles para Cuba debido a las condiciones políticas y económicas que se mantienen con Estados Unidos. El costo en el mercado asciende a los 150 mil dólares e impone a su vez un pago de licencia anual por cuestiones de uso, soporte, capacitación y actualización. En este sentido, el Centro de Informática Industrial (CEDIN) de la Universidad de las Ciencias Informáticas (UCI), en colaboración con el CEINPET, se ha propuesto desarrollar el Sistema de Análisis e Interpretación de Registros de Pozos AnPer.

El objetivo fundamental de este software es interpretar y analizar la información de registros de pozos de petróleo, proceso clave para conocer la cantidad de crudo contenido en los yacimientos y cuánto puede ser extraído, incluso para la toma de decisiones que conduzcan a un mejor desempeño en las tareas de perforación. Desde este punto de vista, se hace necesario obtener algunos parámetros petrofísicos para determinar el volumen de hidrocarburos existente en un yacimiento. AnPer permite analizar la información de los registros mediante la visualización de curvas[1] en diferentes tipos de gráficas, pero carece de algún mecanismo que posibilite calcular dichos parámetros para una evaluación petrofísica del pozo.

Los parámetros petrofísicos necesarios para la evaluación de las formaciones resultan difíciles de obtener directamente con las herramientas de registros geofísicos. Estas herramientas permiten hacer mediciones de otros parámetros de donde se puedan inferir las características petrofísicas de las rocas

[1] Son las que conforman los registros de pozo, representan las variaciones de las propiedades de los mismos.

del yacimiento. La interpretación de registros permite traducir estos parámetros medibles en los parámetros petrofísicos deseados, proporcionando las ecuaciones y técnicas para que dichos cálculos puedan llevarse a cabo.

En la industria petrolera existen programas que calculan parámetros petrofísicos, pero las soluciones más completas pertenecen a las grandes compañías, las cuales además de costosas, no muestran la forma utilizada para definir los cálculos. Esto impide realizar modificaciones que pueden ser requeridas para adecuar los resultados a la litología del yacimiento o para realizar la estimación de otros parámetros. Estos paquetes de software contienen varios modelos petrofísicos[2] para determinar ciertos parámetros, pero cuando se requiere calcular otros no definidos, o hacer variaciones en los cálculos y formulaciones, se necesita que nuevos modelos sean desarrollados. La construcción de nuevos modelos es normalmente impulsada por los datos disponibles y la naturaleza del problema a resolver. Algunos programas permiten al usuario especificar sus cálculos usando algún lenguaje de programación. La complejidad del uso de estos lenguajes hace necesario que los petrofísicos posean ciertos conocimientos de programación o dependan de algún informático.

Teniendo en cuenta la carencia en AnPer de algún mecanismo para obtener parámetros petrofísicos y lo que ello implica, se define como **problema a resolver**: ¿Cómo calcular parámetros petrofísicos inferidos a partir de la información de los registros de pozos en el sistema AnPer?

El problema anterior tiene como **objeto de estudio** el proceso de cálculo de parámetros petrofísicos.

Para dar solución al mismo se establece como **objetivo general**: Desarrollar un módulo en el sistema AnPer que permita interpretar modelos petrofísicos para el cálculo de parámetros.

El trabajo se enmarca en la interpretación de modelos petrofísicos para el cálculo de parámetros que representa su **campo de acción**.

Para el cumplimiento del objetivo se conciben las siguientes tareas investigativas:

- Análisis del proceso de caracterización petrofísica y cálculo de parámetros para definir funcionalmente el módulo a desarrollar.
- Análisis del proceso de compilación e interpretación de lenguajes para la construcción del intérprete.

[2] Es un proceso o procedimiento utilizado para interpretar datos petrofísicos. Por lo general representan un conjunto de ecuaciones, algoritmos u otros procesos matemáticos.

- Especificación de un lenguaje para desarrollar modelos petrofísicos.
- Desarrollo de un intérprete para ejecutar modelos petrofísicos.
- Desarrollo de un editor de código para la codificación de modelos petrofísicos.
- Diseño de interfaces para el cálculo de parámetros petrofísicos básicos.
- Integración del módulo al sistema AnPer.
- Ejecución de pruebas al módulo para comprobar su funcionamiento.

En el desarrollo de la investigación se aplicaron métodos científicos que facilitaron la recopilación y análisis de la información necesaria para la confección de la solución propuesta.

Métodos teóricos:

- Método Histórico-Lógico: Se aplica en el análisis de los procedimientos que se han desarrollado para determinar parámetros y las características de algunos sistemas informáticos usados en procesos petrofísicos, así como investigaciones realizadas anteriormente sobre el tema.
- Método Analítico-Sintético: Se utiliza con el objetivo de modelar el funcionamiento de los procesos de compilación e interpretación de lenguajes, y describir los elementos más importantes relacionados con estos.
- Método Inductivo-Deductivo: Se aplica en la revisión y justificación del modelo y la tecnología para diseñar intérpretes.
- Modelación: Se utiliza para la creación de modelos, que no son más que una reproducción simplificada de la realidad y que proporcionan una mejor comprensión de los temas abordados.

Métodos empíricos:

- Observación: Se aplica con el objetivo de obtener un registro visual de la información referente a los procesos de caracterización petrofísica y el funcionamiento del CEINPET en una situación real.

El presente documento se encuentra estructurado en Introducción, Desarrollo (dividido en tres Capítulos), Conclusiones y Recomendaciones.

Capítulo 1. Fundamentación teórica: Se analizan elementos teóricos de la investigación relacionados con el cálculo de parámetros petrofísicos y el proceso de compilación e interpretación de lenguajes. Además se definen el ambiente y las tecnologías de desarrollo.

Capítulo 2. Presentación de la solución propuesta: Se describe el modelo de dominio y se especifican

detalladamente los requisitos funcionales y no funcionales, los actores y los casos de uso del sistema.

Capítulo 3. Construcción de la solución propuesta: Se exponen algunos de los principales artefactos generados en el análisis, diseño y construcción de la solución como diagramas de clases, diagramas de interacción, diagramas de componentes y características generales de la implementación. Además se realiza la comprobación funcional de la solución mediante la ejecución de pruebas.

Capítulo 1: Fundamentación teórica

Introducción

En este capítulo se abordan elementos asociados al objeto de estudio, identificando y definiendo los conceptos, características y principios fundamentales. Se analizan aspectos del proceso de interpretación petrofísica, cálculo de parámetros y soluciones existentes. Se realiza un análisis del proceso de compilación describiendo las fases y técnicas para el desarrollo de compiladores e intérpretes. Además se define el ambiente de desarrollo y las tecnologías que son usadas en la construcción de la solución.

1.1. Caracterización petrofísica de un yacimiento

La caracterización de un yacimiento es un proceso de amplia base científica en el cual son aplicados diversos conocimientos sobre ingeniería, para así interpretar lógicamente los datos y características del yacimiento mediante herramientas y técnicas modernas. Es el conjunto de productos orientados a la definición y al estudio de las características geológicas, petrofísicas y dinámicas que controlan la capacidad de almacenamiento y de producción de los yacimientos petroleros, así como la cuantificación del volumen de hidrocarburos. También se incluye la definición de las estrategias y alternativas de explotación de los yacimientos, con el propósito de apoyar los planes de operación para optimizar la explotación del área de estudio, incrementando las reservas o la producción de los mismos. (1)

La evaluación petrofísica es relevante para la cuantificación del volumen de los fluidos presentes en cualquier yacimiento. Tiene como objetivo principal la determinación de parámetros básicos del yacimiento, inferidos a partir de registros, para generar modelos a través de los cuales estimar el volumen de arcilla, la porosidad y la saturación de agua. (2) El análisis de los pozos que penetran el yacimiento genera una serie de parámetros importantes para el estudio y comparación de las diferentes propiedades físicas y texturales de la roca, que son imprescindibles en cualquier prospección geológica detallada.

1.2. Registros geofísicos

Un registro o perfil de pozo es una grabación contra profundidad de las características de las formaciones rocosas atravesadas, hechas por aparatos de medición en el hoyo del pozo. Se componen mediante curvas que representan las mediciones digitales. Los registros geofísicos de pozos aportan información de los yacimientos. El aspecto relevante de esta información es la posibilidad de realizar una estimación de las propiedades del yacimiento. (3)

1.3. Parámetros petrofísicos

El análisis de los parámetros petrofísicos aporta resultados necesarios para definir el potencial de un yacimiento. Estos parámetros no se obtienen de manera directa sino que se deducen a partir de las características de la formación medidas con las herramientas de registros geofísicos.

1.3.1. Porosidad

La porosidad (PHI) es el volumen de poros por cada unidad volumétrica de formación. Puede ser primaria o secundaria. En una arena limpia, la matriz de la roca se compone de granos individuales de arena apiñados de manera que los poros se hallan entre los granos. A esta porosidad se le llama porosidad primaria. La porosidad secundaria se debe a la acción de aguas de formación o fuerzas tectónicas en la matriz de la roca después del depósito. (3)

Porosidad total (PHIT): Total de huecos de las rocas, o sea, la suma de las porosidades primaria y secundaria. (4) La porosidad total se calcula como:

$$PHIT = \frac{\mathrm{NPHI} + \mathrm{DPHI}}{2}$$

En caso de estar en presencia de rocas gasíferas:

$$PHIT = \sqrt[2]{\frac{NPHI^2 + DPHI^2}{2}}$$

Siendo:

NPHI: Porosidad neutrónica.

DPHI: Porosidad por densidad.

Porosidad efectiva (PHIE): Es la suma de las porosidades conectadas, tanto primaria como secundaria. Es la que realmente se utiliza en los cálculos de saturación de agua, cálculos de reservas y simulaciones numéricas de yacimientos, ya que refleja el volumen de los poros interconectados. (3) La porosidad efectiva se calcula como:

$$PHIE = PHIT - (VSh * PHISh)$$

Siendo:

VSh: Volumen de arcilla.

PHIT: Porosidad total.

PHISh: Porosidad de la arcilla.

Para que un yacimiento sea comercialmente productivo es obvio que debe tener una porosidad

suficiente para almacenar un volumen apreciable de hidrocarburos. Por tanto, la porosidad es un parámetro petrofísico muy importante de las rocas productivas.

1.3.2.Saturación

La saturación de una formación es el porcentaje del volumen poroso ocupado por el fluido en consideración. La saturación de agua (Sw) es la fracción o porcentaje del volumen poroso que contiene agua de formación. (3)

Todas las determinaciones de saturación de agua a partir de registros de resistividad en formaciones limpias (sin arcilla) se basan en la ecuación de Archie[3]:

$$\mathrm{Sw} = \sqrt[n]{\frac{a * Rw}{PHIE^m * Rt}}$$

Siendo:

Rw: Resistividad del agua de formación.

PHIE: Porosidad efectiva.

Rt: Resistividad de la capa.

a: Tortuosidad.

m: Exponente de cementación.

n: Exponente de saturación.

Las arenas arcillosas son mucho más complejas de analizar, para esto existen varios modelos: Simandoux, Saraband e Indonesian. (Anexo 1. Modelos para el cálculo de saturación de agua (Sw) en arenas arcillosas)

Los valores de saturación de agua proporcionan información acerca del tipo de fluidos que están presentes en la formación de interés.

La determinación de la saturación de la zona lavada (Sxo), presenta gran importancia en el cálculo de hidrocarburos móviles de la formación, ya que estos son los que realmente pueden ser extraídos por métodos convencionales. Al igual que en el caso de la saturación de agua, es necesario tener en cuenta si el colector es limpio o arcilloso para su correcta determinación. En el caso de rocas limpias, en la

[3] Ecuación que relaciona la resistividad de la formación con la porosidad y la saturación de agua. Demostrada en las investigaciones de Gustavo E. Archie. Es la base o punto de partida para otros modelos desarrollados posteriormente para la evaluación de formaciones arcillosas.

ecuación de Archie se sustituye Rt por Rxo, y Rw por Rmf.

$$Sxo = \sqrt[n]{\frac{a * Rmf}{PHIE^m * Rxo}}$$

En el caso de arenas arcillosas:

$$Sxo = \sqrt[n]{\frac{1}{Rxo} * \left(\frac{VSh * \frac{1-VSh}{2}}{\sqrt{RSh}} + \sqrt{\frac{PHIE^m}{a * Rmf}} \right)^2}$$

Siendo:

Rxo: Resistividad de la zona lavada.

Rmf: Resistividad del filtrado de lodo corregida a la temperatura de capa.

El cálculo de la saturación de hidrocarburos se define como:

- Saturación de hidrocarburos residuales (SHR):

$$SHR = 1 - Sxo$$

- Saturación de hidrocarburos totales (SHT):

$$SHT = 1 - Sw$$

- Saturación de hidrocarburos móviles (SHM):

$$SHM = SHT - SHR$$

1.3.3.Volumen de arcilla

Las arcillas y lutitas[4] tienen valores de porosidad muy altos, pero debido al pequeño tamaño de sus granos, tienen muy baja permeabilidad, por lo cual funcionan como un sello del reservorio. En los colectores que presentan un cierto volumen de arcilla (VSh), la porosidad total está seriamente influida por la arcilla, presentando valores altos que no responden realmente a las potencialidades del colector, por eso se hace imprescindible calcular el volumen de arcilla con la mayor precisión posible para poder determinar la porosidad efectiva, que sí da una medida real del volumen de poros interconectados. (3)

A partir del registro Gamma Natural el volumen de arcilla se obtiene como:

$$VSh = \frac{CGRr - CGRl}{CGRa - CGRl}$$

Siendo:

CGRr: Valor de GR en el registro.

[4] Es la roca de minerales de arcilla y algunas variedades de minerales con grano muy fino.

CGRl: Valor de GR en la capa limpia (sin arcilla).

CGRa: Valor del GR en la capa arcillosa.

A partir del registro de resistividad (Rt):

$$VSh = \sqrt[2]{\frac{RSh}{Rt} * \frac{Rl - Rt}{Rl - RSh}}$$

Siendo:

RSh: Resistividad de la arcilla.

Rt: Resistividad verdadera.

Rl: Resistividad de la roca limpia.

A partir de la curva de potencial espontáneo (SP):

$$VSh = 1 - \frac{PSP}{SSP}$$

Siendo:

SSP: Valor de SP en una capa saturada al 100% de agua.

PSP: Valor de SP frente a la capa.

1.4. Soluciones existentes para el cálculo de parámetros

Muchos parámetros petrofísicos se obtienen utilizando modernos programas. En la actualidad el CEINPET utiliza soluciones de empresas reconocidas internacionalmente cuyo costo se eleva por el pago de licencias y actualización. Estos sistemas están diseñados para cualquier tipo de empresa que trabaje en la rama, alejándose de las necesidades específicas que presentan los trabajadores del CEINPET, siendo necesario preparar al personal mediante capacitaciones y cursos. Dentro de los sistemas más usados se encuentran el *Elemental Analysis* (ELAN) y el *Interactive Petrophysics* (IP), ambos de la compañía Schlumberger[5]. Además del costo monetario que implica la adquisición de estos programas, los sistemas comerciales no permiten que se les realice ningún tipo de modificación, como por ejemplo el uso de otro tipo de formulación, lo que es de suma importancia para los especialistas petrofísicos.

IP permite realizar análisis de registros de pozos. Con IP se tiene experiencia media, éstas se están fortaleciendo actualmente a través de cursos, debido a las numerosas ventajas que tiene respecto a los

[5] Compañía estadounidense que provee servicios de yacimiento petrolífero que aportan una variedad de sub-servicios y soluciones a la industria de petróleo internacional.

demás sistemas informáticos. Uno de sus módulos permite al usuario crear sus propias rutinas de análisis para determinar parámetros y manejar la información de los registros. Las rutinas son escritas en FORTRAN, PASCAL o C++. (5) El uso de estos lenguajes implica que los especialistas tengan que poseer ciertos conocimientos de programación o trabajen directamente con algún informático, ya que la sintaxis puede resultar compleja por el uso de bibliotecas y funciones para poder acceder a los valores de las curvas y parámetros que están contenidos en los registros de pozos.

ELAN es un programa de análisis de registros capaz de calcular la formación más probable de minerales y volúmenes de fluidos de poros. Es en particular valioso en áreas de litología variada, minerales especiales y sistemas de porosidad duales. (6) Su uso se hace difícil porque se necesita mucha experiencia y práctica, contradiciendo el propósito de tener un software amigable con el usuario.

También existen pequeñas soluciones de software petrolero para evaluar modelos así como propiedades petrofísicas de un yacimiento. Concretamente son en su mayoría hojas de cálculos sencillas pero versátiles, las cuales pueden ser modificadas de acuerdo a los parámetros petrofísicos que se tengan y resultados que se deseen obtener. Constituyen en general soluciones aisladas, donde cada una responde a una determinada necesidad petrofísica. Entre estas están:

- DST: Software para calcular la presión.
- FRF: Hoja de cálculo para analizar las propiedades eléctricas en registros.
- TVD: Programa para encontrar la profundidad vertical verdadera (TVD, *True Vertical Depth*) mediante siete métodos.
- TOC: Pequeño software que calcula el contenido total orgánico de registros sónicos.

Por lo complejo que resulta la adquisición de estos programas en cuanto a costo y usabilidad es que el CEDIN y el CEINPET se involucran en el desarrollo del sistema AnPer. Este software trata de suplir las necesidades propias de los especialistas en el análisis de yacimientos. De ahí la importancia de integrarle un módulo para el cálculo de parámetros, buscando una alternativa donde el usuario pueda determinar parámetros petrofísicos básicos, pero además, tenga un mecanismo con el cual desarrollar sus propios modelos para realizar el cálculo de otros parámetros y adecuarlos a la litología del yacimiento.

1.5. Lenguaje de programación

Los lenguajes de programación son los que permiten establecer una comunicación hombre-máquina. Intentan conservar una similitud con el lenguaje humano, con la finalidad de que sean más naturales a

quienes los usan. Establecen un conjunto de reglas sintácticas y semánticas, las cuales rigen la estructura del programa de computación que se escribe o edita. (7) De esta forma, permiten a los programadores o desarrolladores, poder especificar de forma precisa los datos sobre los que se va a actuar, su almacenamiento, transmisión y demás acciones a realizar bajo las distintas circunstancias consideradas.

1.5.1.Lenguaje de Dominio Específico

En los últimos años se ha popularizado el término de "Lenguaje de Dominio Específico" (DSL, *Domain Specific Language*) en la industria del software para indicar un lenguaje de programación o especificación dedicado a un dominio particular o una técnica particular de solución de un problema. De forma concreta se puede definir como: "Un lenguaje de programación o lenguaje de especificación ejecutable que ofrece potencia expresiva enfocada y restringida a un dominio de problema concreto". (8)

Entre las principales características de los DSL se pueden encontrar:

- Generalmente estos lenguajes son pequeños y ofrecen un restringido conjunto de notaciones y abstracciones.
- Suelen ser lenguajes declarativos, pudiendo considerarse lenguajes de especificación, además de programación.
- Un objetivo común de muchos de estos lenguajes es la programación realizada por los usuarios, los que se encargan de desarrollar sus propios programas. Estos usuarios no suelen tener grandes conocimientos informáticos pero pueden realizar tareas de programación en dominios concretos con un vocabulario cercano a su especialización.

1.5.2.Especificación de un lenguaje

La sintaxis de un lenguaje de programación es el conjunto de reglas formales que especifican la estructura de los programas pertenecientes a dicho lenguaje, mientras que la semántica es el conjunto de reglas que especifican el significado de cualquier sentencia sintácticamente válida. La especificación formal de un lenguaje desde el punto de vista sintáctico se suele realizar mediante la descripción estándar de su gramática en notación BNF (*Backus-Naur-Form*)[6]. En el caso de la especificación semántica no existe ningún método estándar globalmente extendido y cada uno la describe desde diferentes puntos de vista. La razón es que la descripción del comportamiento de los programas tiene una mayor complejidad que la de su estructura.

[6] Se utiliza para definir formalmente las reglas gramaticales (sintaxis) que rigen la construcción de los símbolos y de las secuencias de símbolos escritos en un programa.

1.6. Proceso de compilación

Un traductor es cualquier programa que toma como entrada un texto escrito en un lenguaje, llamado fuente y da como salida otro texto en un lenguaje, denominado objeto. (8)

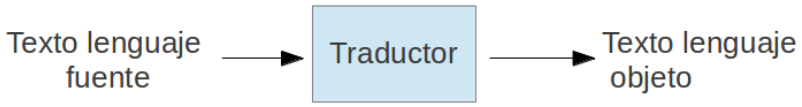

Figura 1. Representación de un traductor

Un traductor engloba tanto a compiladores como intérpretes, donde el "programa de salida" en los primeros suele ser código de máquina, mientras que en los segundos está constituido por una serie de acciones atómicas que serán ejecutadas posteriormente.

Un intérprete es un programa que analiza y ejecuta simultáneamente un programa escrito en un lenguaje fuente. (9)

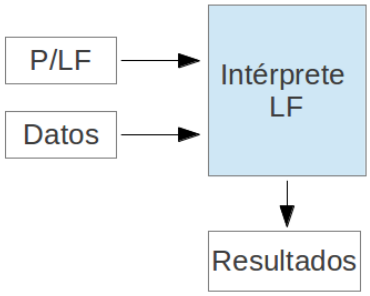

Figura 2. Esquema general de un intérprete

Los compiladores, a diferencia de los intérpretes, transforman el programa a un programa equivalente en un código objeto (fase de compilación), y en un segundo paso generan los resultados a partir de los datos de entrada (fase de ejecución). (9)

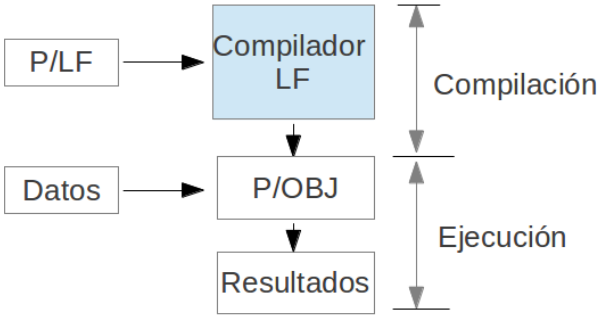

Figura 3. Esquema general de un compilador

1.6.1.Fases del proceso de compilación

Conceptualmente un compilador opera en fases. Cada una de las cuales transforma el programa fuente de una representación en otra. En la práctica se pueden agrupar fases y las representaciones intermedias entres las fases agrupadas no necesitan ser construidas de forma explícita.

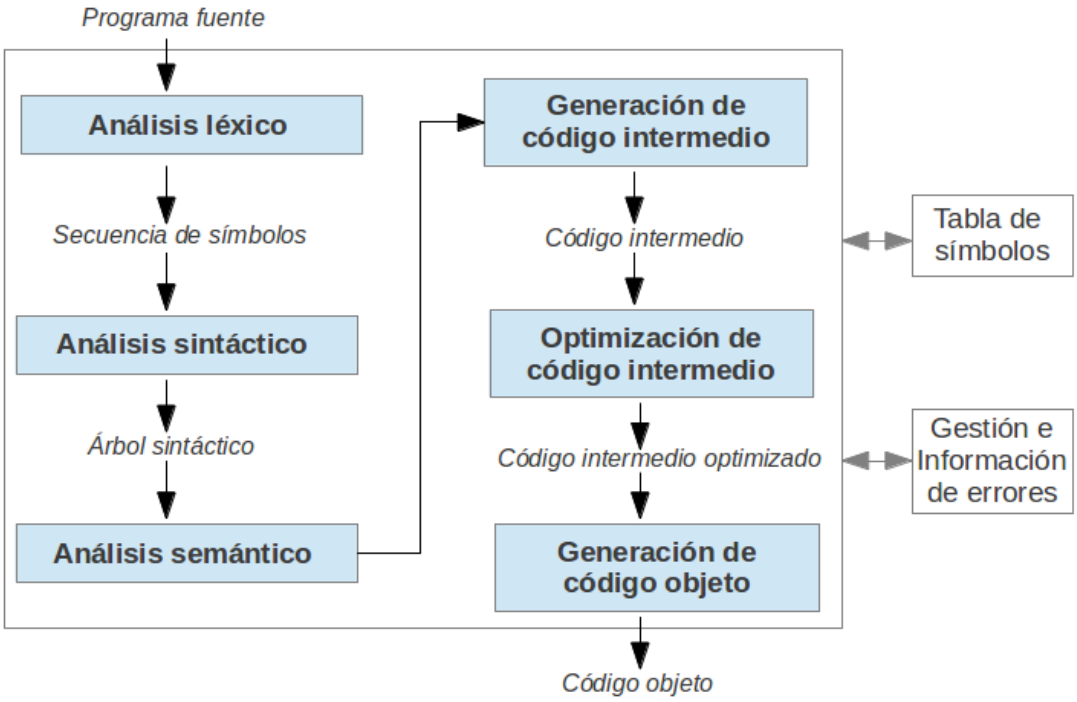

Figura 4. Fases del proceso de compilación

Fase 1. Análisis léxico: El analizador léxico o *scanner* se encarga de tomar la secuencia de símbolos pertenecientes al alfabeto de un determinado lenguaje y agruparlos en entidades sintácticas simples o elementales denominadas *tokens* o lexemas. (10)

Fase 2. Análisis sintáctico: El analizador sintáctico o *parser* examina la secuencia de *tokens* para determinar si el orden de la secuencia es correcto de acuerdo a ciertas convenciones estructurales (reglas) de la definición sintáctica del lenguaje. (10)

Fase 3. Análisis semántico: En la fase de análisis semántico el compilador adiciona información al árbol de sintaxis abstracta[7] generado en la fase de análisis sintáctico. (8)

Fase 4. Generación de código intermedio: Después del análisis sintáctico y semántico muchos

[7] Es una representación de árbol de la estructura sintáctica abstracta (simplificada) del código fuente escrito en cierto lenguaje de programación.

compiladores generan una representación explícita intermedia del código fuente. Dicha representación puede verse como la representación de un programa para una máquina abstracta.

Fase 5. Optimización del código intermedio: La fase de optimización se efectúa con el objetivo de mejorar la eficiencia del código intermedio de modo que en la siguiente fase resulte un código de máquina más rápido de ejecutar.

Fase 6. Generación del código objeto: La fase final de un compilador es la de generación del código objeto, es decir, generar el código máquina o código ensamblador.

Operaciones sobre la tabla de símbolos: Una tarea fundamental en un compilador es la de almacenar los identificadores utilizados en un programa y sus atributos principales, de manera que en cualquier momento pueda conocerse la información que se necesita sobre las variables del programa. Esta información se almacena generalmente en una estructura conocida como tabla de símbolos, la cual tiene una entrada para cada identificador y sus atributos.

Gestión e información de errores: Los errores en un programa pueden clasificarse en cuatro grandes grupos: lexicológicos, sintácticos, semánticos y lógicos o de programación. El analizador léxico detecta errores cuando los caracteres que restan de la entrada no forman ningún *token* válido en el lenguaje. Los errores referentes a que los *tokens* no cumplan con las reglas estructurales (sintaxis) del lenguaje se detectan en la fase de análisis sintáctico. Durante el análisis semántico el compilador trata de detectar estructuras que tengan una sintaxis correcta pero incorrecta semánticamente de acuerdo a las operaciones involucradas.

1.7. Estructura de un intérprete

A la hora de construir un intérprete es conveniente utilizar una representación interna (RI) del lenguaje fuente (LF) a analizar. De esta forma, la organización interna de la mayoría de los intérpretes se descompone en los módulos:

- Traductor a representación interna: Toma como entrada el código del programa P en lenguaje fuente, lo analiza y lo transforma a la representación interna correspondiente a dicho programa P.
- Representación interna (P/RI): La representación interna debe ser consistente con el programa original. Entre los tipos de representación interna, los árboles sintácticos son los más utilizados y, si las características del lenguaje lo permiten, pueden utilizarse estructuras de pila para una mayor eficiencia.

- Tabla de símbolos: Durante el proceso de traducción, es conveniente ir creando una tabla con información relativa a los símbolos que aparecen. La información a almacenar en dicha tabla de símbolos depende de la complejidad del lenguaje fuente.
- Evaluador de representación interna: A partir de la representación interna anterior y de los datos de entrada, se llevan a cabo las acciones indicadas para obtener los resultados.
- Tratamiento de errores: Durante el proceso de evaluación pueden aparecer diversos errores como desbordamiento de la pila, divisiones por cero, etc. que el intérprete debe contemplar.

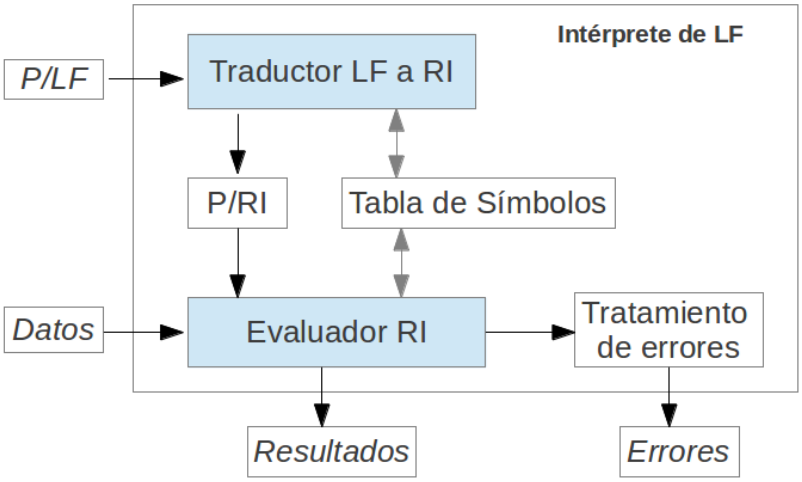

Figura 5. Organización interna de un intérprete

Dependiendo de la complejidad del código a analizar, el intérprete puede contener módulos similares a los de un compilador tradicional: análisis léxico, sintáctico y semántico.

1.8. Herramientas para generar analizadores del proceso de compilación

Actualmente existen herramientas que facilitan la tarea de escribir compiladores o intérpretes informáticos. Estas herramientas permiten generar el esqueleto del analizador sintáctico a partir de una definición formal del lenguaje de partida, especificada normalmente mediante una gramática formal, dejando únicamente al programador del compilador la tarea de programar las acciones semánticas asociadas.

Lex/Yacc: Lex es un programa para generar analizadores léxicos, se utiliza comúnmente con el programa Yacc que genera análisis sintáctico. (11)

Flex: Herramienta que implementa un analizador léxico. Flex lee los ficheros de entrada dados con la descripción de un escáner a generar. (12)

Bison: Herramienta que implementa un analizador sintáctico. Es un generador de analizadores sintácticos perteneciente al proyecto GNU disponible para prácticamente todos los sistemas operativos. Bison convierte la descripción formal de un lenguaje en un programa en C, C++, o Java que realiza análisis sintáctico. (12)

El uso de estas herramientas tiene como desventaja que las acciones semánticas asociadas con las producciones de los no terminales de las gramáticas son difíciles de depurar. También mezclan las especificaciones sintácticas con las semánticas. Es complejo poner en práctica mensajes de error significativos en el analizador. Si un error ocurre, puede ser difícil para determinar si está en la gramática o en el código que se analiza. Además, para especificar la gramática en los ficheros de estos programas habría que tener en cuenta la sintaxis para definir las curvas y parámetros, y estos nomencladores van a depender de la información que contenga el registro de pozo, un proceso que debe tratarse de manera diferenciada ya que cada registro presenta esta información con nomencladores diferentes.

1.9. Herramientas y tecnologías

Debido a que el módulo a desarrollar formará parte del sistema AnPer, las metodologías, herramientas y lenguajes a utilizar durante el desarrollo de la solución deben corresponder e integrarse correctamente con las definidas por el equipo de desarrollo del proyecto. Las mismas se adaptan perfectamente a las necesidades del módulo para el cálculo de parámetros petrofísicos.

1.9.1. Metodología de desarrollo, lenguaje de modelado y herramienta CASE

Durante el desarrollo de software muchas tareas y actividades se tornan engorrosas, trayendo consigo que el proceso de desarrollo de software se torne riesgoso y difícil de controlar, obteniendo clientes insatisfechos con el resultado final. La forma para solucionar o tratar de llevar a cabo un proyecto eficiente es aplicando una metodología de desarrollo de software.

El Proceso Unificado de Desarrollo (RUP, *Rational Unified Process*) es una plataforma de procesos de desarrollo de software que brinda guías consistentes y personalizadas para todo el equipo de proyecto. RUP describe cómo utilizar de forma efectiva reglas de negocio y procedimientos comerciales probados en el desarrollo de software, conocidos como “mejores prácticas”. Se caracteriza por ser dirigido por casos de uso, centrado en la arquitectura y ser iterativo e incremental, donde cada fase se desarrolla en iteraciones, de forma tal que se pueda dividir en pequeños proyectos mejorando su comprensión y desarrollo. (13)

Es importante señalar que la metodología RUP ha sido seleccionada teniendo en cuenta las

especificaciones de la propuesta de solución que hace que las partes a entregar deban quedar bien documentadas internamente, así como lo distante que se encuentran desarrolladores y clientes.

En el proceso de desarrollo de software el modelado ejerce un papel fundamental. RUP utiliza como lenguaje de modelado el Lenguaje Unificado de Modelado (UML) que con la utilización de diagramas y gráficos brinda una mejor perspectiva de lo que se quiere. El mismo se utiliza para definir un sistema de software, detallar los artefactos que se generan durante su desarrollo, documentarlo y construirlo. De esta forma, ofrece una amplia vista del sistema, desde la cual se logra un entendimiento entre todos los vinculados al desarrollo de software. UML ofrece un estándar para describir un plano del sistema, incluyendo aspectos conceptuales, tales como procesos de negocios, funciones del sistema y aspectos concretos, como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. Da apoyo a la mayoría de los procesos de desarrollo orientados a objetos. (14) UML es el enlace entre quién tiene la idea y el desarrollador, siendo la comunicación su principal objetivo.

Las herramientas para la Ingeniería de Software Asistida por Computadoras (CASE, *Computer Aided Software Engineering*) de modelado con UML permiten representar el software mediante diagramas que se generan durante las diferentes etapas del proyecto. Permiten además hacer análisis y diseño orientado a objetos y facilitan el desarrollo del proceso. La herramienta definida es *Visual Paradigm for UML Enterprise Edition*. Esta soporta el ciclo de vida del desarrollo de software: Análisis y Diseño, Construcción, Pruebas y Despliegue. Su diseño está centrado en casos de uso y enfocado al negocio generando un software de mayor calidad. Presenta capacidades de ingeniería directa e inversa y disponibilidad en múltiples plataformas. Permite el uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación. Es muy fácil de usar en la creación de todo tipo de diagramas UML, para los que dispone de un número considerable de estereotipos que permiten un mayor entendimiento de los mismos. (15)

1.9.2.Lenguaje de programación, framework y Entorno de Desarrollo Integrado

En el desarrollo de software cuando se desea seleccionar un lenguaje de programación se deben tener en cuenta elementos claves como: el entorno de ejecución, el rendimiento, la escalabilidad, la portabilidad y la seguridad. El lenguaje de programación C++ es un lenguaje versátil y potente. Entre los programadores ha llegado a ocupar el primer puesto como herramienta en el desarrollo de aplicaciones. C++ es un lenguaje orientado a objetos. Una de las ventajas que presenta es que los programas que genera son considerados entre los más compactos y rápidos. Permite la creación de punteros a objetos,

esto posibilita que aquellos programas que sean creados con este lenguaje se ejecuten más rápido, además de presentar mayor eficiencia en el consumo de recursos. Permite el control de memoria y una capacidad de programación de alto nivel. (16)

El lenguaje de programación por sí solo no es suficiente para el desarrollo de la aplicación, por lo que es necesario contar con un Entorno de Desarrollo Integrado (IDE, *Integrated Development Environment*). 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 de usuario (GUI, *Graphical User Interface*).

Qt Creator es un IDE multiplataforma, distribuido bajo los términos de *GNU Lesser General Public License*. Combina edición, depuración, gestión de proyectos, localización y herramientas de compilación. Está diseñado para hacer que el desarrollo en C++ de la aplicación Qt sea más rápido y fácil; posee un avanzado editor de código C++. El depurador visual para C++ es consciente de la estructura de muchas clases de Qt, lo que aumenta la capacidad de mostrar los datos con claridad. Constituye un entorno integrado para la creación y diseño de GUI para proyectos C++. Los formularios son totalmente funcionales y pueden ser previamente visualizados para asegurarse de que se verá y sentirá exactamente como lo pensó el usuario. (17)

El uso de un *framework* (ambiente de trabajo) para el desarrollo de la aplicación ayuda a construir una arquitectura sólida, robusta, brindando consistencia al código y facilitando la integración de nuevas funcionalidades. Básicamente un *framework* es una estructura de soporte mediante la cual puede ser desarrollado un proyecto de software. Generalmente incluye programas, bibliotecas y un lenguaje interpretado que optimizan y aceleran el proceso de desarrollo del software. Son desarrollados para brindar a los programadores y diseñadores una mejor organización y estructura a sus proyectos. Los objetivos principales que persigue un *framework* son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo tales como el uso de patrones.

Qt es un *framework* distribuido bajo los términos de *GNU Lesser General Public License* utilizado para el desarrollo de aplicaciones multiplataforma. La función más conocida de Qt es la de la creación de interfaces de usuario, sin embargo, no se limita a esto, ya que también provee varias clases para facilitar ciertas tareas de programación como el manejo de *sockets*, soporte para programación multihilo, comunicación con bases de datos, manejo de cadenas de caracteres, entre otras. (18) Qt es un *framework* muy poderoso, comparable con Swing de Java o .NET de Microsoft, además ofrece una suite

de aplicaciones para facilitar y agilizar las tareas de desarrollo, las aplicaciones que componen esta suite son:

- Qt Assistant: Herramienta para visualizar la documentación oficial de Qt.
- Qt Designer: Herramienta para crear interfaces de usuario.
- Qt Linguist: Herramienta para la traducción de aplicaciones.
- Qt Creator: IDE para el lenguaje C++, pero especialmente diseñado para Qt, integra las primeras dos herramientas mencionadas.

Conclusiones parciales

El análisis del proceso de caracterización petrofísica y cálculo de parámetros permitió establecer las pautas para el funcionamiento del módulo a desarrollar, enfatizando en la necesidad de permitir variabilidad en la forma de establecer los cálculos y formulaciones de acuerdo a las necesidades del usuario, además de definir modelos para determinar parámetros petrofísicos básicos. El estudio del proceso de compilación posibilitó diseñar y establecer los componentes para desarrollar un intérprete, definir un lenguaje y así crear un mecanismo para que el usuario pueda codificar y ejecutar modelos petrofísicos y obtener los parámetros deseados.

Los principales componentes tecnológicos para el desarrollo de la solución propuesta: metodología de desarrollo de software, lenguaje de programación, entorno de desarrollo, facilitan en conjunto alcanzar el objetivo general de la presente investigación. Se ha asumido como premisa lograr la libertad tecnológica del producto, teniendo en cuenta el soporte sobre diferentes sistemas operativos y el cumplimiento con los principios de la comunidad de software libre.

Capítulo 2: Presentación de la solución propuesta

Introducción

En el presente capítulo se describen los procesos del negocio que intervienen en la realización de las actividades vinculadas al objeto de estudio. Se representan los conceptos fundamentales del entorno del problema en un modelo de dominio. Se definen los requisitos funcionales y no funcionales que se tienen en cuenta para el diseño e implementación del módulo y se describen los actores y casos de usos del sistema.

2.1. Descripción de los procesos de negocio

Para la descripción de los procesos de negocio que se llevan a cabo dentro del campo de acción de este trabajo, se toma como premisa la ejecución de modelos para calcular parámetros petrofísicos. En la evaluación petrofísica de yacimientos se llevan a cabo diferentes cálculos de parámetros definidos por ecuaciones matemáticas, a partir de la información contenida en los registros de pozos. Como se ha visto anteriormente algunos programas de la rama petrofísica incorporan mecanismos para ejecutar esta actividad. Se determinan dichos parámetros a través de modelos predefinidos en estos sistemas.

Las soluciones más avanzadas incorporan una herramienta de compilación para la ejecución de otros modelos, utilizando nuevas definiciones matemáticas y manipulando la información de los registros. Es una forma de darle libertad al usuario de que establezca sus propios modelos y así resuelva necesidades específicas en el proceso de evaluación petrofísica. Esta tarea se podría llevar a cabo utilizando hojas de cálculo y otras herramientas, pero traería como consecuencias tener que insertar toda la información en estas hojas para posteriormente analizar los resultados, debido a la no interoperabilidad entre la hoja de cálculo y los sistemas.

2.2. Modelo de dominio

El modelo de dominio o modelo conceptual, permite de manera visual mostrar al usuario los principales conceptos que se manejan en el dominio del problema. Representa las clases conceptuales del mundo real, no de componentes de software. Puede considerarse como un diccionario visual de las abstracciones relevantes, vocabulario e información del dominio. Aprovechando las oportunidades de los diagramas UML para representar conceptos, el modelo de dominio se presenta en forma de diagrama de clases donde figuran los principales conceptos y roles del sistema en cuestión. (19)

El modelo de dominio se describe en la siguiente figura, utilizando el lenguaje UML. Cada clase representa un concepto significativo para el dominio del problema.

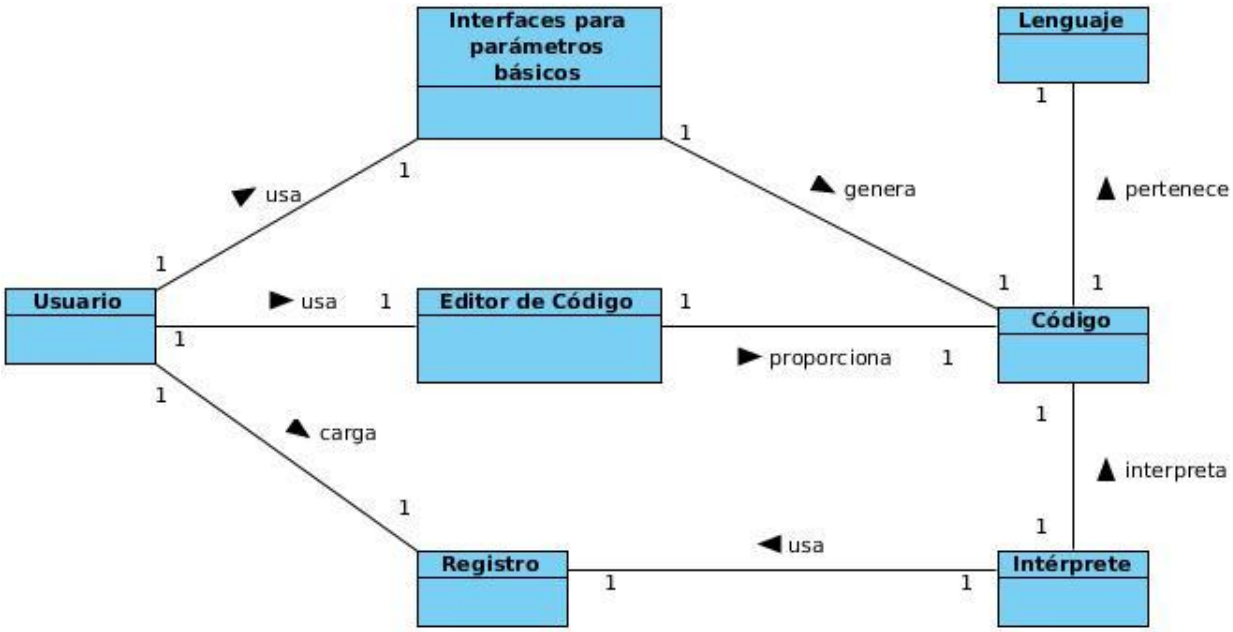

Figura 6. Modelo de dominio

Usuario: Representa al especialista petrofísico que hace uso del módulo en el sistema.

Interfaces para parámetros básicos: Representan las interfaces definidas para calcular parámetros petrofísicos básicos, cuyas formulaciones y procedimientos (modelos) ya están establecidas en el sistema. El usuario solamente debe introducir los valores necesarios para su ejecución.

Editor de código: Representa la herramienta que usa el usuario para codificar y ejecutar modelos petrofísicos que no estén definidos en el sistema para el cálculo de otros parámetros usando nuevas formulaciones.

Código: Representa la cadena de código que define un modelo petrofísico, ya sea generada a través de las interfaces definidas o proporcionada por el usuario a través del editor. Es la codificación que debe ser evaluada y ejecutada para obtener los resultados de las instrucciones que define.

Intérprete: Representa el mecanismo para evaluar y ejecutar el código o la cadena de instrucciones y obtener los resultados, haciendo uso del registro de pozo que proporciona la información, y al cual se le integrarán los resultados.

Registro: Representa la información de un registro de pozo, conteniendo las curvas y parámetros que lo conforman.

Lenguaje: Representa el conjunto de símbolos y reglas sintácticas y semánticas que debe cumplir el código a ejecutar.

2.3. Requerimientos del sistema

Los requisitos o requerimientos definen el comportamiento del software. Se pueden encontrar dos tipos de requisitos: funcionales y no funcionales. Los requisitos funcionales no son más que las condiciones o capacidades que deben estar presentes en un sistema o componentes del sistema. Los requisitos no funcionales son aquellos que describen las cualidades, características y propiedades del producto. (20)

Todas las ideas que tengan los clientes, usuarios y miembros del equipo de proyecto acerca de lo que debe hacer el sistema, deben ser analizadas como candidatas a requisitos. La validación de requisitos es una actividad muy importante, pues un levantamiento de requisitos con errores que no se detecten a tiempo, además de conducir a resultados inesperados, provoca costos excesivos y gran pérdida de tiempo.

A continuación se enuncian los requerimientos funcionales y no funcionales que el módulo desarrollado debe cumplir.

2.3.1.Requerimientos funcionales

El sistema propuesto debe permitir:

- **RF1**: Calcular la porosidad total a través de un modelo predefinido en el sistema.
- **RF2**: Calcular la porosidad efectiva a través de un modelo predefinido en el sistema.
- **RF3**: Calcular la saturación de agua a través de un modelo predefinido en el sistema.
- **RF4**: Calcular el volumen de arcilla a través de un modelo predefinido en el sistema.
- **RF5**: Calcular la saturación de la zona lavada a través de un modelo predefinido en el sistema.
- **RF6**: Calcular la saturación de hidrocarburos (móviles, residuales y totales) a través de un modelo predefinido en el sistema.
- **RF7**: Codificar modelos para calcular parámetros petrofísicos.
- **RF8**: Guardar modelos desarrollados por el usuario.
- **RF9**: Cargar modelos desarrollados por el usuario.
- **RF10**: Ejecutar modelos petrofísicos para obtener los parámetros que definen.
- **RF11**: Informar errores que puedan generarse al ejecutar modelos petrofísicos.
- **RF12**: Manipular la información de curvas y parámetros del registro de pozo.
- **RF13**: Incorporar al registro los parámetros obtenidos al ejecutar un modelo petrofísico.

2.3.2.Requerimientos no funcionales

Usabilidad

- El sistema debe tener un diseño de interfaz que facilite la realización de las funciones con que fue concebido.
- El sistema debe estar disponible para usuarios con experiencia básica, media o avanzada en los procesos de análisis petrofísicos.

Rendimiento

- El sistema debe garantizar un tiempo de respuesta de 3 a 5 segundos en dependencia de la complejidad de los modelos a evaluar.

Portabilidad

- El sistema debe ser compatible con los sistemas operativos: Windows XP o versiones superiores y GNU/Linux.

Soporte

- El sistema debe ser capaz de dar las mismas salidas para diferentes ambientes de trabajo.

Hardware

- El sistema debe trabajar satisfactoriamente en máquinas con memoria RAM de 512 Mb o superiores.

2.4. Descripción del sistema propuesto

Para dar respuesta al problema identificado al inicio de este trabajo, y teniendo en cuenta los requerimientos planteados, la solución que se propone es el desarrollo de un módulo para calcular parámetros petrofísicos mediante la interpretación de modelos que definan dichos cálculos, integrado al sistema AnPer. Dentro de sus funcionalidades principales tiene calcular parámetros básicos cuyos modelos ya están definidos y establecidos en el sistema. Se incorpora al módulo la posibilidad de codificar y ejecutar nuevos modelos según la naturaleza del problema que se necesite resolver. Se considera la existencia de un solo rol, el especialista petrofísico, que es el usuario del sistema que va a interactuar con el módulo.

Utilizando las facilidades que brinda UML, se representarán los requisitos funcionales del sistema mediante un diagrama de casos de uso. Para ello se definen cuáles serían los actores que van a interactuar con el sistema, y los casos de uso que van a representar las funcionalidades.

2.4.1.Descripción de los actores del sistema

Un actor no es parte del sistema, sino un rol que se juega dentro del sistema, que puede intercambiar información o puede ser un recipiente pasivo de información. Representa a un ser humano, a un software o a una máquina que interactúa con el sistema. (21)

El sistema cuenta con el siguiente actor:

Tabla 1. Descripción de actores

Actor	Descripción
	Representa la persona que interactúa con el módulo para calcular parámetros petrofísicos. Carga el registro, establece valores de entrada para determinar parámetros básicos y crea y ejecuta modelos petrofísicos.

2.4.2.Modelo de casos de uso del sistema

El modelo de casos de uso del sistema es una representación de las funciones deseadas para el sistema y su entorno, y sirve como contrato entre el cliente y los desarrolladores. Se utiliza como entrada esencial para las actividades de análisis y diseño. (13)

El siguiente diagrama muestra el modelo de casos de uso del sistema para el módulo propuesto:

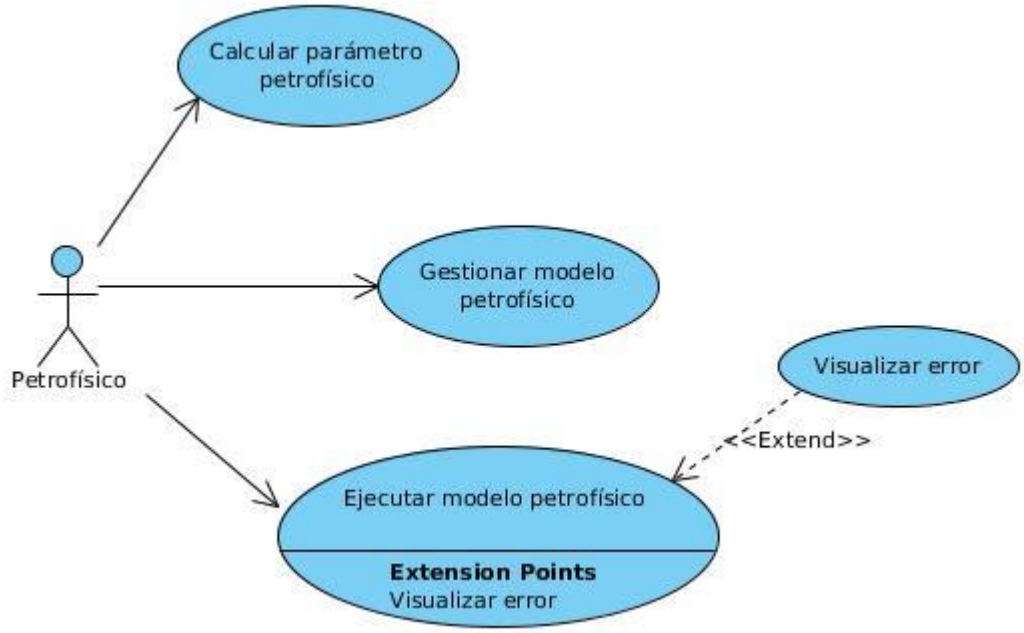

Figura 7. Modelo de CUS

2.4.3.Descripción textual de los casos de uso del sistema

Los casos de uso del sistema son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Por lo tanto, establece un acuerdo entre clientes y desarrolladores sobre las condiciones y posibilidades (requisitos) que debe cumplir el sistema. (13)

A continuación se exponen las descripciones textuales de cada caso de uso del sistema.

Tabla 2. Descripción textual CU1

Caso de Uso	CU1. Calcular parámetro petrofísico
Objetivo	Calcular parámetro petrofísico mediante la ejecución de modelos definidos en el sistema.
Actores	Petrofísico
Resumen	Este CU permite calcular parámetros petrofísicos a través de modelos definidos en el sistema. Los parámetros que se pueden calcular son: Porosidad Total, Porosidad Efectiva, Saturación de Agua, Volumen de Arcilla, Saturación de la zona lavada y Saturación de Hidrocarburos.
Referencias	RF1, RF2, RF3, RF4, RF5, RF6, RF14
Prioridad	Crítico
Precondiciones	El petrofísico debe estar autenticado en el sistema y tener activo un registro de pozo.
Poscondiciones	Integración del parámetro calculado al registro de pozo.
Flujo de eventos	
Flujo Normal de Eventos	
Actor	**Sistema**
1. El caso de uso inicia cuando el petrofísico da clic en la opción “Parámetros Petrofísicos Básicos” en la pestaña “Cálculo”.	2. Verifica que esté activo un registro de pozo. (alterno 1) 3. Muestra una ventana con los siguientes parámetros: • PHIT Porosidad Total (Ir a Sección 1) • PHIE Porosidad Efectiva (Ir a Sección 2) • Sw Saturación de Agua (Ir a Sección 3) • VSh Volumen de Arcilla (Ir a Sección 4)

	• Sxo Saturación de la zona lavada (Ir a Sección 5) • SH Saturación de Hidrocarburos (Ir a Sección 6)
	4. Termina el caso de uso.
Flujos alternos	
Evento 1	
Actor	**Sistema**
	1. Si no existe ningún registro de pozo activo muestra el mensaje “Debe cargar un fichero LAS para calcular parámetros petrofísicos”. 2. Ir al paso 4 del flujo normal de eventos.
Sección 1 “PHIT Porosidad Total”	
Flujo de eventos	
Actor	**Sistema**
1. Selecciona el parámetro “PHIT Porosidad Total”.	2. Muestra una ventana solicitando la siguiente información: • Porosidad Neutrónica NPHI • Porosidad por Densidad DPHI • Si es o no Rocas Gasíferas
3. Introduce los datos de entrada para NPHI, DPHI y si es o no Rocas Gasíferas. 4. Da clic en el botón Aceptar. (alterno 1)	5. Genera la codificación para el cálculo de PHIT. 6. Analiza y ejecuta el código. 7. Informa el resultado de la ejecución.
8. Confirma el resultado de la ejecución.	9. Ir al paso 4 del flujo normal de eventos.
Flujos alternos	
Evento 1	
Actor	**Sistema**
1. Da clic en el botón Cancelar.	2. Ir al paso 4 del flujo normal de eventos.

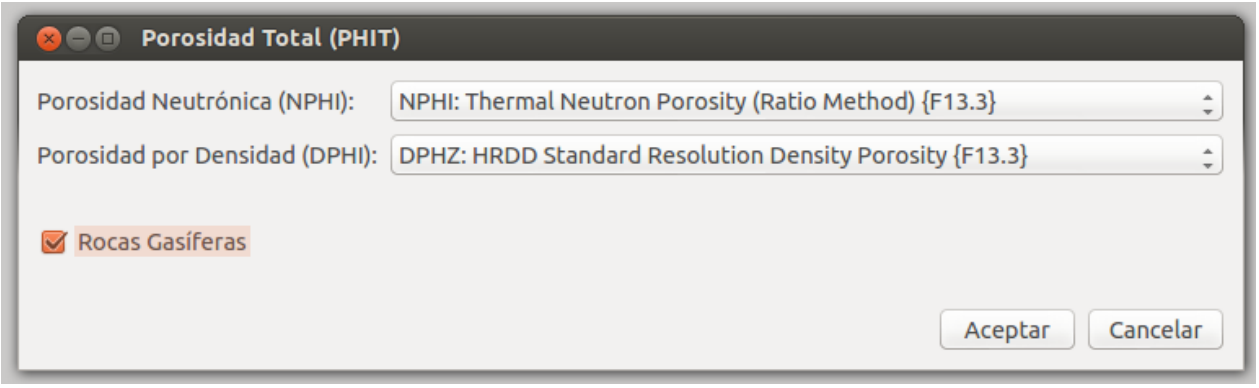

Figura 8. Prototipo de interfaz PHIT

Sección 2 “PHIE Porosidad Efectiva”	
Flujo de eventos	
Actor	**Sistema**
1. Selecciona el parámetro “PHIE Porosidad Efectiva”.	2. Muestra una ventana solicitando la siguiente información: • Volumen de Arcilla VSh • Porosidad Total PHIT • Porosidad de la Arcilla PHISh
3. Introduce los datos de entrada para VSh, PHIT y PHISh. 4. Da clic en el botón Aceptar. (alterno 1)	5. Genera la codificación para el cálculo de PHIE. 6. Analiza y ejecuta el código. 7. Informa el resultado de la ejecución.
8. Confirma el resultado de la ejecución.	9. Ir al paso 4 del flujo normal de eventos.
Flujos alternos	
Evento 1	
Actor	**Sistema**
1. Da clic en el botón Cancelar.	2. Ir al paso 4 del flujo normal de eventos.

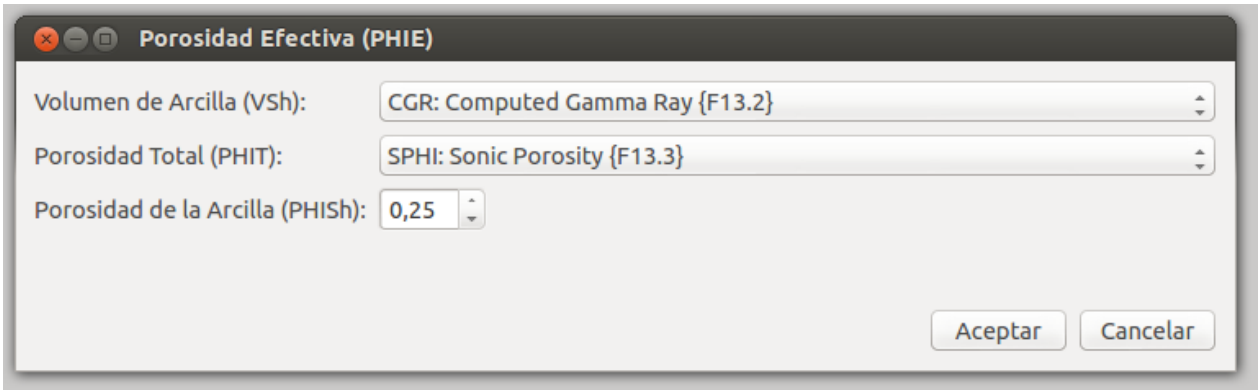

Figura 9. Prototipo de interfaz PHIE

Sección 3 "Sw Saturación de Agua"	
Flujo de eventos	
Actor	**Sistema**
1. Selecciona el parámetro "Sw Saturación de Agua".	2. Muestra una ventana solicitando la siguiente información: • Resistividad del agua de formación Rw • Resistividad de la capa Rt • Porosidad Efectiva PHIE • Tortuosidad • Exponente de Cementación • Exponente de Saturación • Si es o no Arenas Arcillosas • Volumen de Arcilla VSh • Resistividad de la roca en formación RSh • Modelo (Indonesian, Simandoux, Saraband)
3. Introduce los datos de entrada para Rw, Rt, PHIE, Tortuosidad, Exponente de Cementación, Exponente de Saturación, si es o no Arenas Arcillosas, VSh, RSh y el Modelo.	5. Genera la codificación para el cálculo de Sw. 6. Analiza y ejecuta el código. 7. Informa el resultado de la ejecución.

4. Da clic en el botón Aceptar. (alterno 1)	
8. Confirma el resultado de la ejecución.	9. Ir al paso 4 del flujo normal de eventos.
Flujos alternos	
Evento 1	
Actor	**Sistema**
1. Da clic en el botón Cancelar.	2. Ir al paso 4 del flujo normal de eventos.

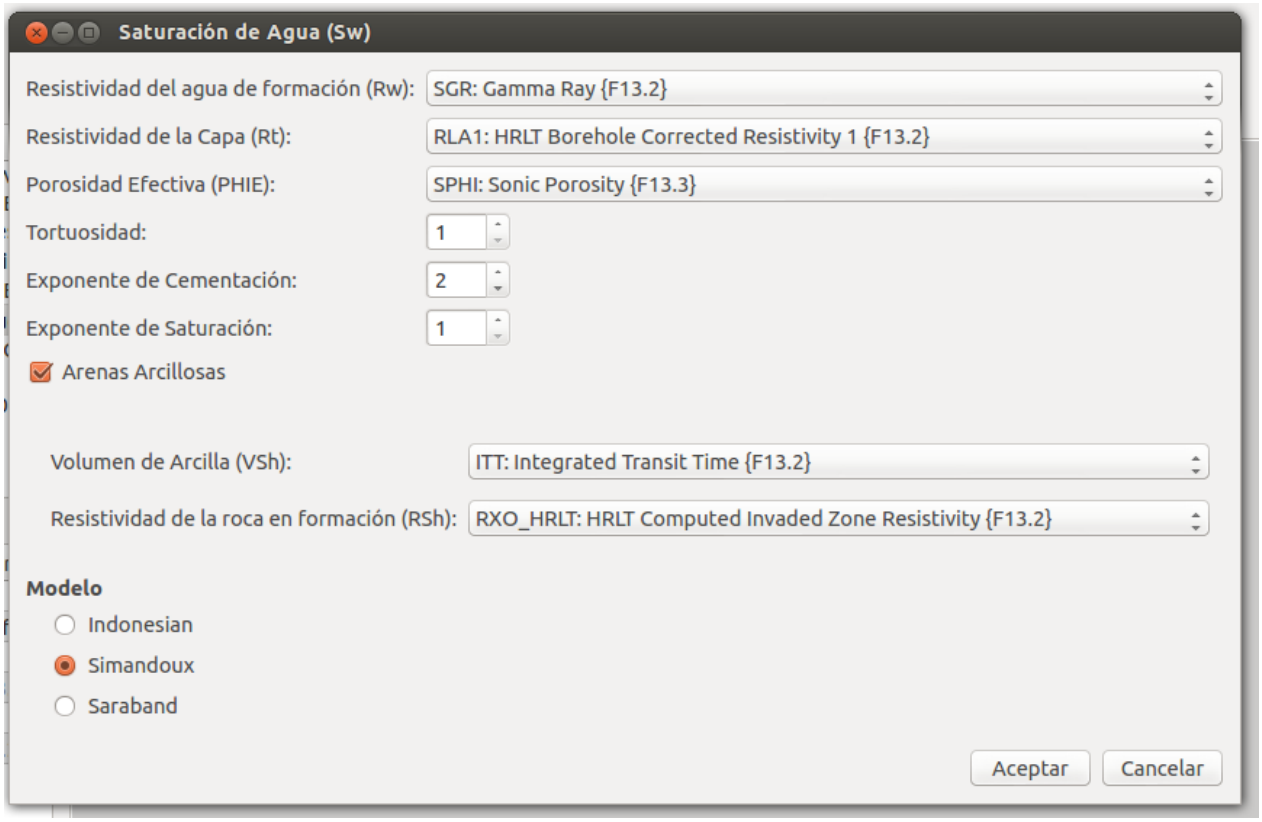

Figura 10. Prototipo de interfaz Sw

Sección 4 “VSh Volumen de Arcilla”	
Flujo de eventos	
Actor	**Sistema**
1. Selecciona el parámetro “VSh Volumen de Arcilla”.	2. Muestra una ventana solicitando la siguiente información: • Registro (Rayos Gamma CGR,

	Resistividad Rt, Potencial Espontáneo SP) • GR en el registro CGRr • GR en la capa limpia CGRl • GR en la capa arcillosa CGRa
3. Introduce los datos de entrada para el Registro, CGRr, CGRl y CGRa. 4. Da clic en el botón Aceptar. (alterno 1)	5. Genera la codificación para el cálculo de VSh. 6. Analiza y ejecuta el código. 7. Informa el resultado de la ejecución.
8. Confirma el resultado de la ejecución.	9. Ir al paso 4 del flujo normal de eventos.
Flujos alternos	
Evento 1	
Actor	**Sistema**
1. Da clic en el botón Cancelar.	2. Ir al paso 4 del flujo normal de eventos.

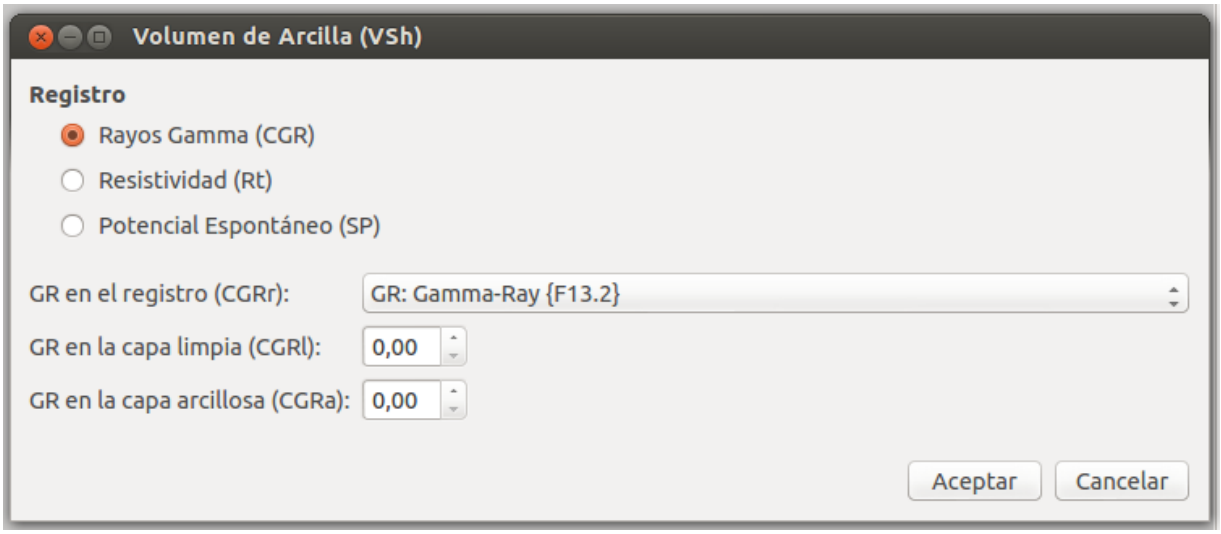

Figura 11. Prototipo de interfaz VSh

Sección 5 "Sxo Saturación de la zona lavada"	
Flujo de eventos	
Actor	**Sistema**
1. Selecciona el parámetro "Sxo	2. Muestra una ventana solicitando la siguiente

Saturación de la zona lavada”.	información: • Resistividad de la zona lavada Rxo • Resistividad del filtrado de lodo Rmf • Porosidad Efectiva PHIE • Tortuosidad • Exponente de Cementación • Exponente de Saturación • Si es o no Arenas Arcillosas • Volumen de Arcilla VSh • Resistividad de la roca en formación RSh
3. Introduce los datos de entrada para Rxo, Rmf, PHIE, Tortuosidad, Exponente de Cementación, Exponente de Saturación, si es o no Arenas Arcillosas, VSh y RSh. 4. Da clic en el botón Aceptar. (alterno 1)	5. Genera la codificación para el cálculo de Sxo. 6. Analiza y ejecuta el código. 7. Informa el resultado de la ejecución.
8. Confirma el resultado de la ejecución.	9. Ir al paso 4 del flujo normal de eventos.
Flujos alternos	
Evento 1	
Actor	**Sistema**
1. Da clic en el botón Cancelar.	2. Ir al paso 4 del flujo normal de eventos.

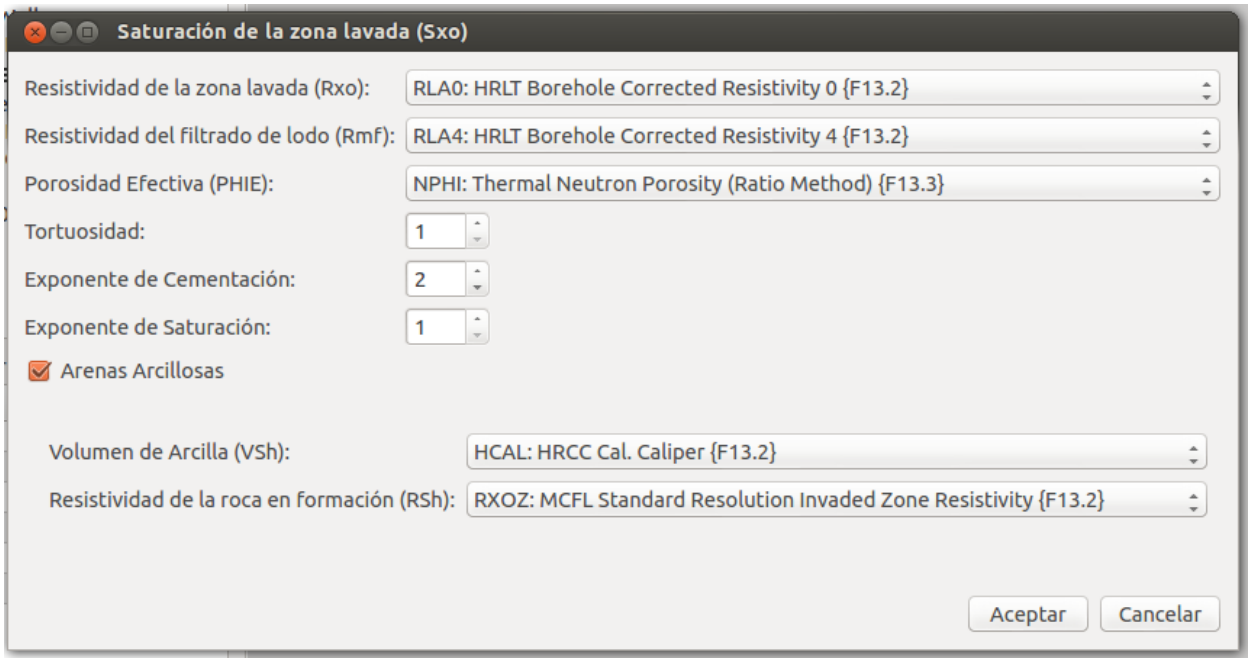

Figura 12. Prototipo de interfaz Sxo

Sección 6 "SH Saturación de Hidrocarburos"	
Flujo de eventos	
Actor	**Sistema**
1. Selecciona el parámetro "SH Saturación de Hidrocarburos".	2. Muestra una ventana solicitando la siguiente información: • Saturación de la zona lavada Sxo • Saturación de agua Sw • Resultados (Saturación de Hidrocarburos Residuales SHR, Saturación de Hidrocarburos Totales SHT, Saturación de Hidrocarburos Móviles SHM)
3. Introduce los datos de entrada para Sxo, Sw y Resultados. 4. Da clic en el botón Aceptar. (alterno 1)	5. Genera la codificación para el cálculo de SH. 6. Analiza y ejecuta el código. 7. Informa el resultado de la ejecución.
8. Confirma el resultado de la ejecución.	9. Ir al paso 4 del flujo normal de eventos.

Flujos alternos	
Evento 1	
Actor	**Sistema**
1. Da clic en el botón Cancelar.	2. Ir al paso 4 del flujo normal de eventos.

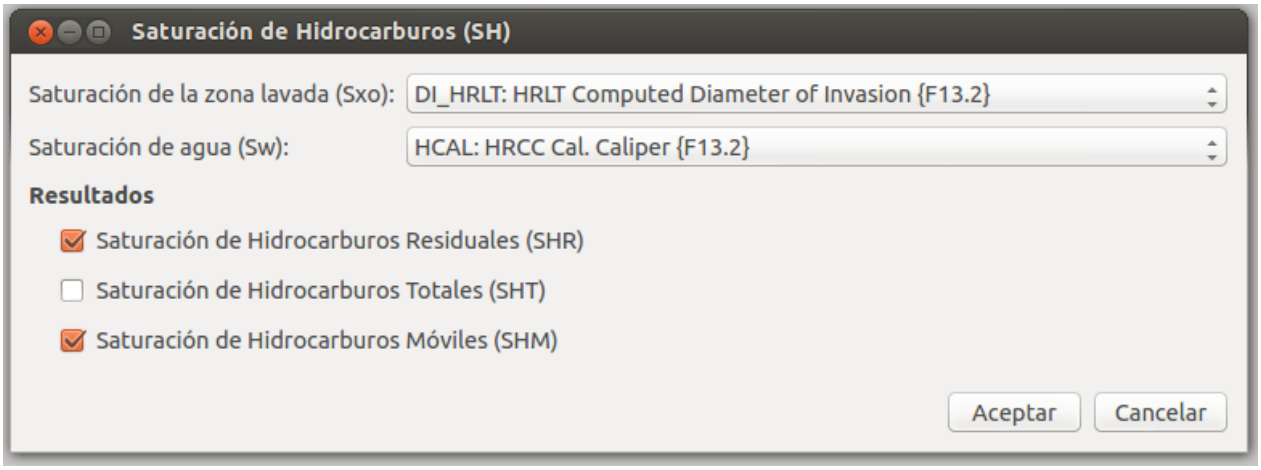

Figura 13. Prototipo de interfaz SH

Relaciones	CU Incluidos	No procede
	CU Extendidos	No procede
Requisitos no funcionales	No procede	
Asuntos pendientes	No procede	

Tabla 3. Descripción textual CU2

Caso de Uso	CU2. Gestionar modelo petrofísico
Objetivo	Codificar, guardar y cargar modelos petrofísicos.
Actores	Petrofísico
Resumen	Permite realizar la codificación de un modelo petrofísico usando el lenguaje definido, guardar la codificación de los modelos desarrollados en un fichero de texto y cargar modelos de usuarios.
Referencias	RF7, RF8, RF9
Prioridad	Crítico
Precondiciones	El petrofísico debe estar autenticado en el sistema y tener activo un

	registro de pozo.
Poscondiciones	Codificación, salva y carga de un modelo petrofísico.
Flujo de eventos	
Flujo Normal de Eventos	
Actor	**Sistema**
1. El caso de uso inicia cuando el petrofísico da clic en la opción “Editor” en la pestaña “Cálculo”.	2. Verifica que esté activo un registro de pozo. (alterno 1) 3. Muestra una ventana para gestionar modelos petrofísicos con las opciones de: • Codificar modelo (Ir a Sección 1) • Guardar modelo (Ir a Sección 2) • Cargar modelo (Ir a Sección 3)
	4. Termina el caso de uso.
Flujos alternos	
Evento 1	
Actor	**Sistema**
	1. Si no existe un registro de pozo activo muestra el mensaje “Debe cargar un fichero LAS para crear un programa de usuario”. 2. Ir al paso 4 del flujo normal de eventos.
Sección 1 “Codificar modelo”	
Flujo de eventos	
Actor	**Sistema**
1. Da clic en la opción Nuevo modelo.	2. Incorpora al completamiento de código las funciones del lenguaje y los nomencladores de las curvas y parámetros que conforman el registro activo. 3. Establece el formato de codificación.
4. Escribe la codificación del modelo.	5. Activa el formato y el completamiento de código. 6. Ir al paso 4 del flujo normal de eventos.
Sección 2 “Guardar modelo”	

Flujo de eventos	
Actor	**Sistema**
1. Da clic en la opción Guardar modelo.	2. Muestra una ventana para especificar la ruta y el nombre del fichero en el que se va a guardar la codificación del modelo.
3. Establece la ruta y el nombre del fichero. (alterno 1)	4. Salva la codificación del modelo en el fichero indicado. (alterno 2) 5. Ir al paso 4 del flujo normal de eventos.
Flujos alternos	
Evento 1	
Actor	**Sistema**
1. Establece un nombre de fichero vacío.	2. Muestra una etiqueta con el mensaje “Escriba un nombre de archivo”. 3. Ir al paso 3 del flujo de eventos de la sección 2.
Evento 2	
Actor	**Sistema**
	1. Si el fichero establecido no es válido muestra un mensaje “No se pudo escribir en el fichero”. 2. Ir al paso 4 del flujo normal de eventos.
Sección 3 “Cargar modelo”	
Flujo de eventos	
Actor	**Sistema**
1. Da clic en la opción Abrir modelo.	2. Muestra una ventana para especificar el fichero que contiene la codificación del modelo.
3. Establece el fichero que contiene la codificación del modelo.	4. Obtiene la codificación del modelo desde el fichero indicado. (alterno 1) 5. Muestra la codificación en el editor de código. 6. Ir al paso 4 del flujo normal de eventos.
Flujos alternos	
Evento 1	

Actor	Sistema
	1. Si el fichero establecido no es válido muestra un mensaje “No se pudo leer el fichero”. 2. Ir al paso 4 del flujo normal de eventos.

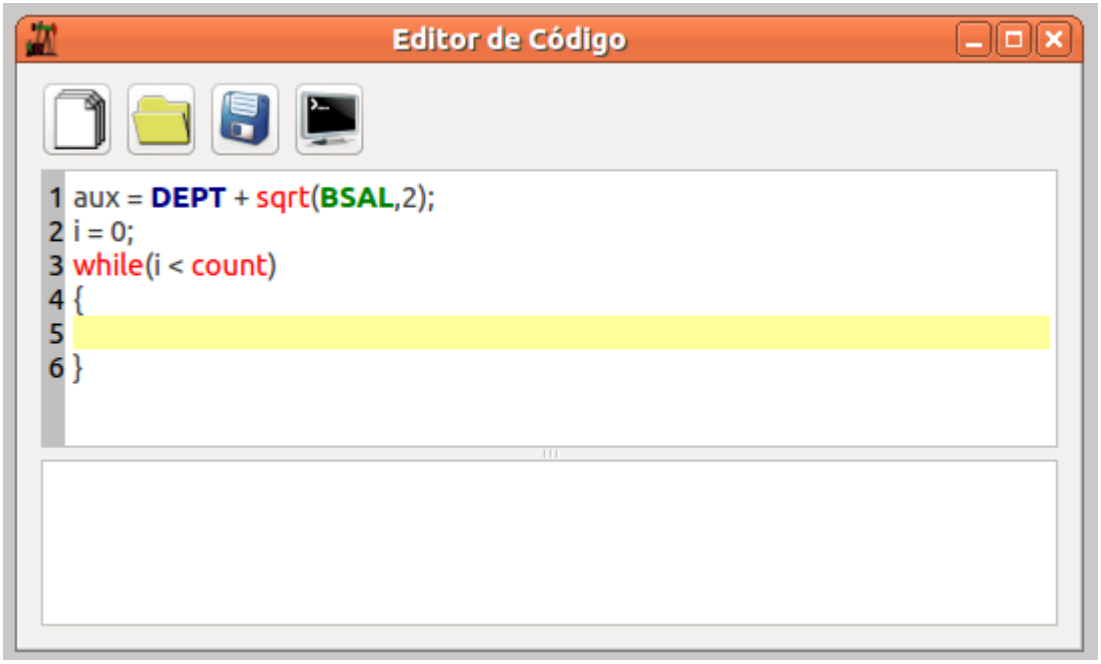

Figura 14. Prototipo de interfaz Gestionar modelo petrofísico

Relaciones	CU Incluidos	No procede
	CU Extendidos	No procede
Requisitos no funcionales	No procede	
Asuntos pendientes	No procede	

Tabla 4. Descripción textual CU3

Caso de Uso	CU3. Ejecutar modelo petrofísico
Objetivo	Ejecutar la codificación de un modelo petrofísico.
Actores	Petrofísico
Resumen	Permite ejecutar la codificación de un modelo petrofísico desarrollado por el usuario e integrar los resultados al registro de pozo. Informa los errores que ocurren durante el análisis y ejecución del modelo.

Referencias	RF10, RF11, RF12, RF13, CU4
Prioridad	Crítico
Precondiciones	El petrofísico debe estar autenticado en el sistema y tener activo un registro de pozo. El petrofísico debe tener codificado un modelo en el editor de código.
Poscondiciones	Integración de los resultados de la ejecución de un modelo petrofísico al registro de pozo.

Flujo de eventos	
Flujo Normal de Eventos	
Actor	**Sistema**
1. El caso de uso inicia cuando el petrofísico da clic en la opción Ejecutar modelo.	2. Valida la codificación del modelo petrofísico (alterno 1). 3. Ejecuta la codificación del modelo petrofísico (alterno 2). 4. Integra los resultados al registro de pozo activo. 5. Informa la culminación de la ejecución del modelo mostrando el mensaje “Código ejecutado correctamente”.
	6. Termina el caso de uso.
Flujos alternos	
Evento 1	
Actor	**Sistema**
	1. Si la codificación del modelo está vacía muestra el mensaje “Código vacío”. 2. Ir al paso 6 del flujo normal de eventos.
Evento 2	
Actor	**Sistema**
	1. Si en la ejecución del modelo se generan errores informa al usuario sobre la ocurrencia de estos. 2. Ir al paso 6 del flujo normal de eventos.

Relaciones	CU Incluidos	No procede
	CU Extendidos	CU4
Requisitos no funcionales	No procede	
Asuntos pendientes	No procede	

Tabla 5. Descripción textual CU4

Caso de Uso	CU4. Visualizar error	
Objetivo	Visualizar e indicar la aparición de errores en la codificación de un modelo petrofísico después de la ejecución del mismo.	
Actores	Petrofísico	
Resumen	Permite visualizar los errores ocurridos durante el análisis y ejecución de un modelo petrofísico en el editor de código.	
Referencias	RF11	
Prioridad	Secundario	
Precondiciones	El petrofísico debe estar autenticado en el sistema y tener activo un registro de pozo. El petrofísico debe haber ejecutado un modelo que contenga errores.	
Poscondiciones	Visualización del error en el editor de código.	
Flujo de eventos		
Flujo Normal de Eventos		
Actor		**Sistema**
1. El caso de uso inicia cuando el petrofísico da doble clic en un error de la lista de errores resultante de la ejecución de un modelo.		2. Obtiene el número de la línea donde ocurrió el error. 3. Oscurece el texto de la codificación en la línea indicada.
		4. Termina el caso de uso.
Relaciones	CU Incluidos	No procede
	CU Extendidos	No procede
Requisitos no funcionales	No procede	
Asuntos pendientes	No procede	

En la descripción textual de los casos de uso se han detallado las actividades que se realizan dentro de los procesos, dando así un mayor entendimiento y seguimiento de los mismos.

Conclusiones parciales

A partir del análisis del proceso de negocio se obtuvo el modelo de domino, permitiendo conocer con mayor profundidad los conceptos presentes en el entorno donde coexiste el problema. Se definieron los requisitos funcionales y no funcionales que el sistema debe cumplir, contribuyendo a la calidad del producto final y al grado de satisfacción del cliente. La descripción y representación gráfica de los casos de uso del sistema constituyen la base para enmarcar el análisis y diseño del módulo, haciendo énfasis en las principales funcionalidades que debe tener, y estableciendo una comprensión entre desarrolladores y clientes para una mayor satisfacción y cumplimiento de sus necesidades.

Capítulo 3: Construcción de la solución propuesta

Introducción

En este capítulo se hace una descripción del modelo de análisis y los artefactos que se generan en el mismo, como los diagramas de clases del análisis que describen gráficamente las especificaciones de las clases de software y de las interfaces de la aplicación. Se muestran los diagramas de interacción que reflejan el comportamiento y la trazabilidad de los mensajes entre los objetos. Se describen los estilos arquitectónicos adoptados y los patrones de diseño utilizados. El modelo de análisis dará paso a realizar el modelo de diseño que tendrá como artefactos principales los diagramas de clases del diseño. Se describe el modelo de implementación mediante diagramas de componentes, se define el lenguaje que permitirá codificar modelos petrofísicos y se muestran los casos de pruebas realizados a la solución.

3.1. Modelo de análisis

El modelo de análisis ayuda a refinar los requisitos y permite razonar sobre los aspectos internos del sistema. Proporciona una estructura centrada en el mantenimiento, en aspectos tales como la flexibilidad ante los cambios y la reutilización. Esta estructura no solo es útil para el mantenimiento de los requisitos como tal, también se utiliza como entrada en las actividades de diseño e implementación. Es descrito con el lenguaje del desarrollador, utilizado fundamentalmente para comprender como debería darse forma al sistema, es decir, como debería ser diseñado e implementado. (22)

3.1.1. Diagramas de clases del análisis

Un diagrama de clases del análisis (DCA) es un artefacto en el que se representan los conceptos de un dominio del problema. Representa el funcionamiento del mundo real, no de la implementación automatizada del mismo. (22)

Las clases se clasifican en:

Entidad: Modelan información que posee larga vida y que es a menudo persistente.

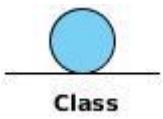

Figura 15. Clase entidad

Interfaz: Modelan la interacción entre el sistema y sus actores.

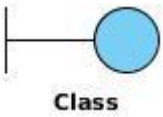

Figura 16. Clase interfaz

Control: Coordinan la realización de uno o unos pocos casos de uso coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso.

Figura 17. Clase control

A continuación se muestran los diagramas de clases del análisis por cada caso de uso del sistema.

3.1.1.1. DCA CU Calcular parámetro petrofísico

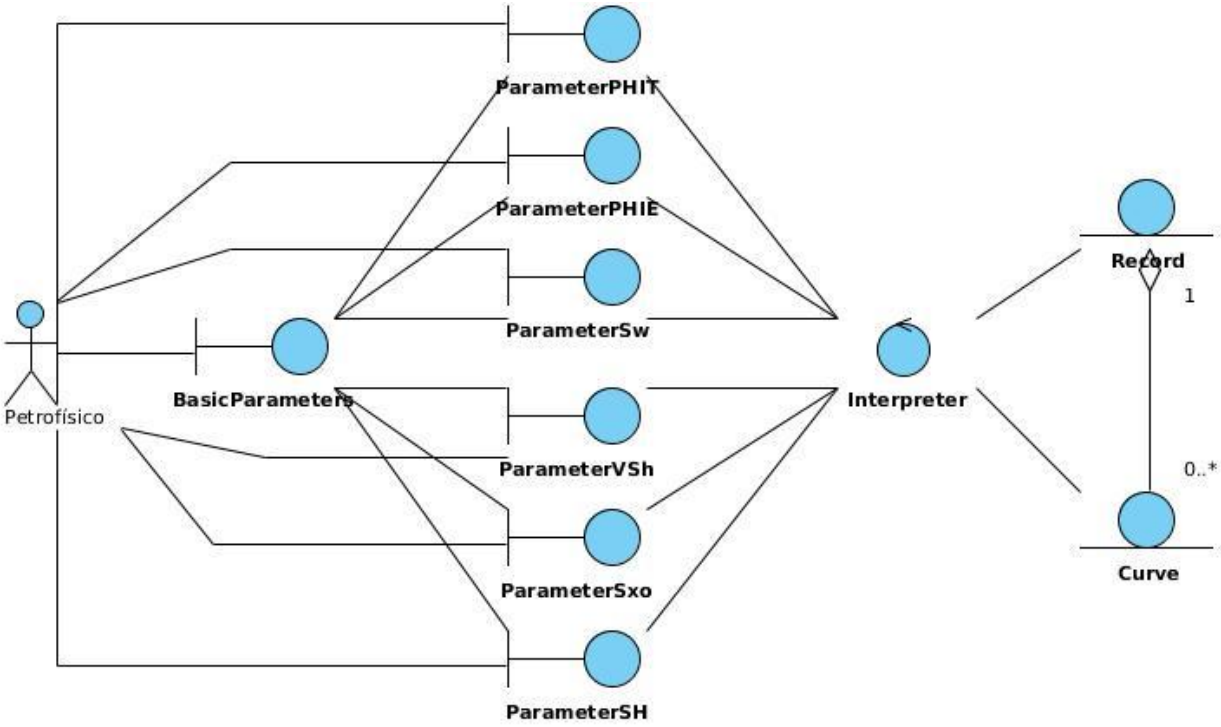

Figura 18. DCA CU1

3.1.1.2. DCA CU Gestionar modelo petrofísico

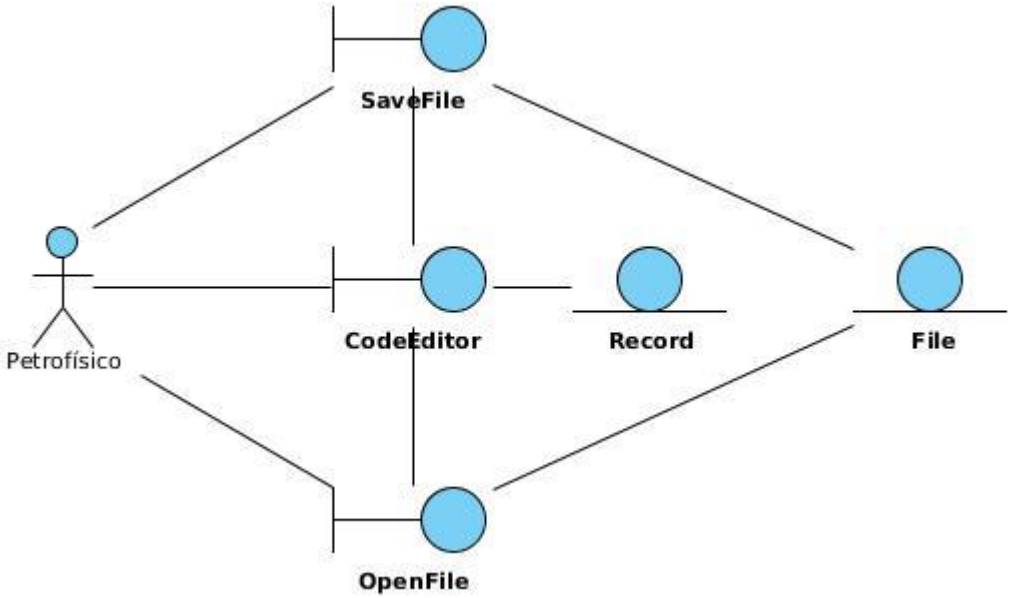

Figura 19. DCA CU2

3.1.1.3. DCA CU Ejecutar modelo petrofísico

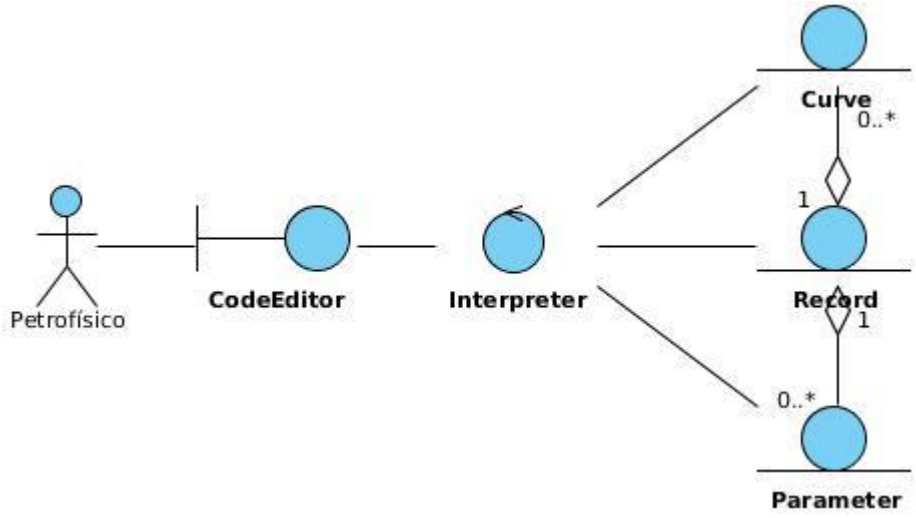

Figura 20. DCA CU3

3.1.1.4. DCA CU Visualizar error

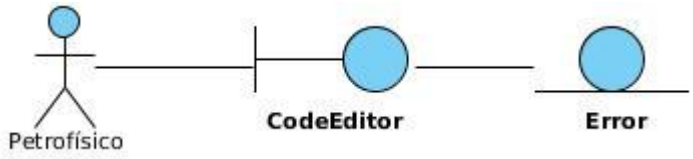

Figura 21. DCA CU4

3.1.2.Diagramas de interacción del análisis

Los diagramas de interacción incluyen la interacción de los mensajes entre los objetos que se definen en el modelo conceptual y otras clases de objetos. Existen dos tipos de diagramas de interacción: diagramas de secuencia y diagramas de colaboración.

Un diagrama de secuencia muestra una interacción que está organizada como una secuencia temporal. En particular, muestra los objetos que participan en la interacción mediante sus líneas de vida y mediante los mensajes que intercambian, organizados en forma de una secuencia temporal. (22)

Una colaboración modela los objetos y los enlaces significativos dentro de una interacción. Los objetos y los enlaces son significativos solamente en el contexto proporcionado por la interacción. Un rol describe un objeto, y un rol en la asociación describe un enlace dentro de una colaboración. Un diagrama de colaboración muestra los roles en la interacción en una disposición geométrica. Los mensajes se muestran como flechas, ligadas a las líneas de la relación, que conectan a los roles. La secuencia de mensajes, se indica con los números secuenciales que preceden a las descripciones del mensaje. (22)

Se muestran a continuación los diagramas de secuencia del análisis (DSA) por cada caso de uso.

3.1.2.1. DSA CU Calcular parámetro petrofísico (Sección PHIT)

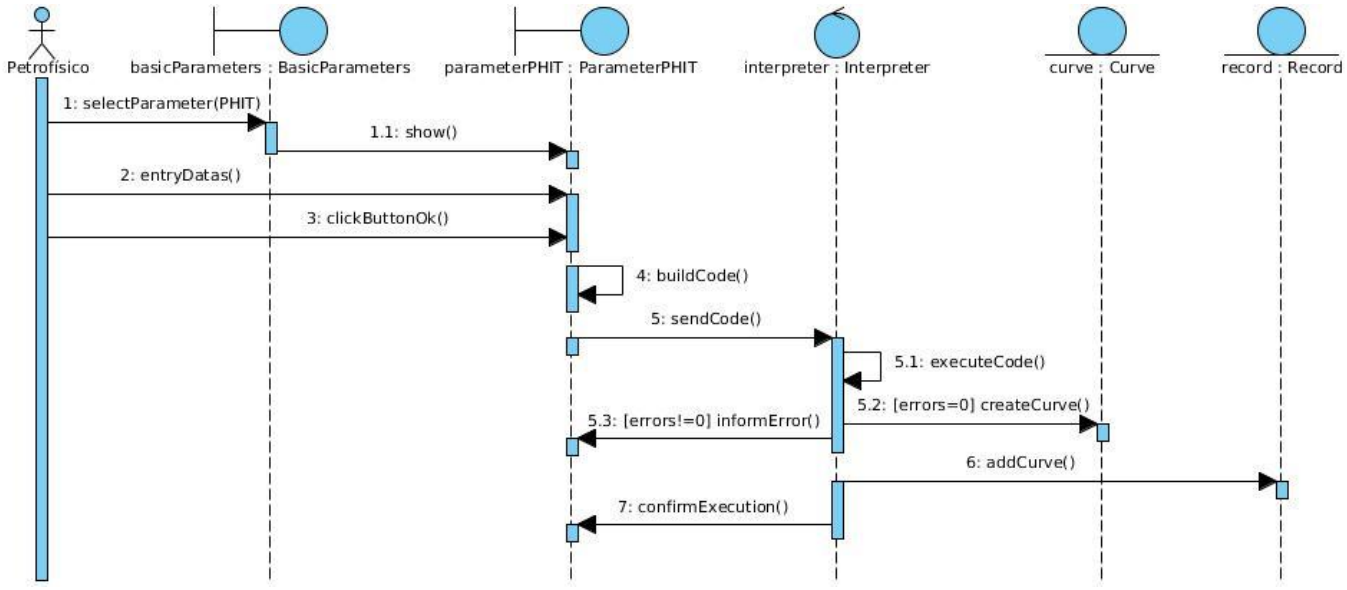

Figura 22. DSA CU1 (PHIT)

El flujo de mensajes para las secciones restantes del CU Calcular parámetro petrofísico se comporta de manera similar. Solo se establece la interfaz correspondiente con cada parámetro.

3.1.2.2. DSA CU Gestionar modelo petrofísico (Sección Codificar)

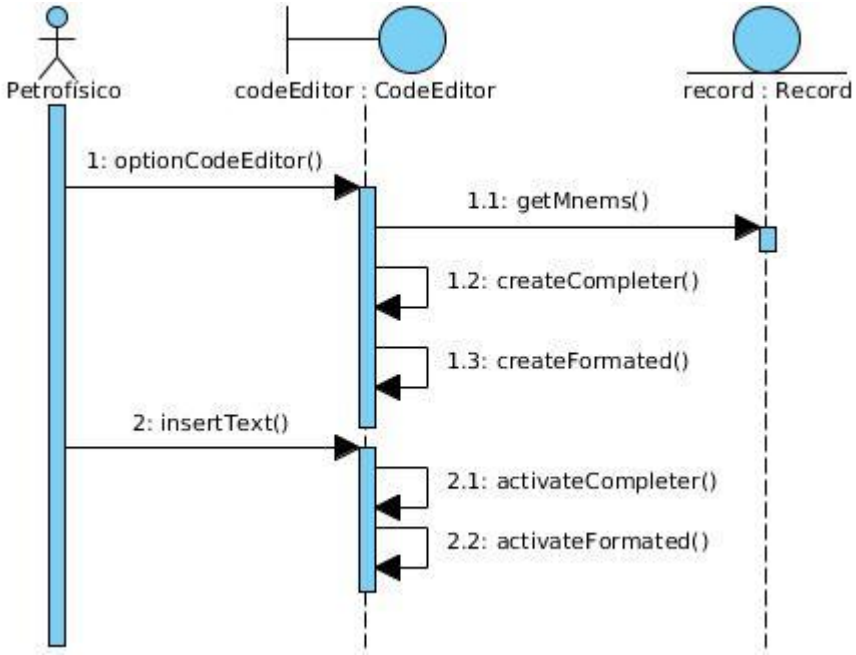

Figura 23. DSA CU2 (Codificar)

3.1.2.3. DSA CU Gestionar modelo petrofísico (Sección Guardar)

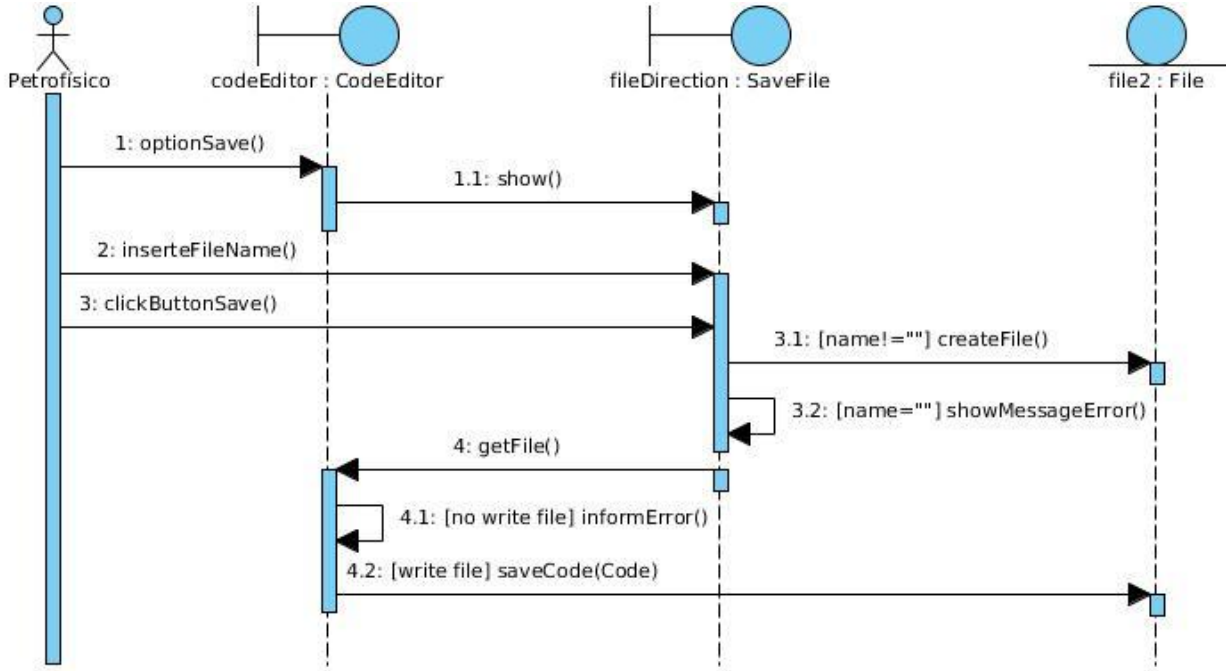

Figura 24. DSA CU2 (Guardar)

3.1.2.4. DSA CU Gestionar modelo petrofísico (Sección Cargar)

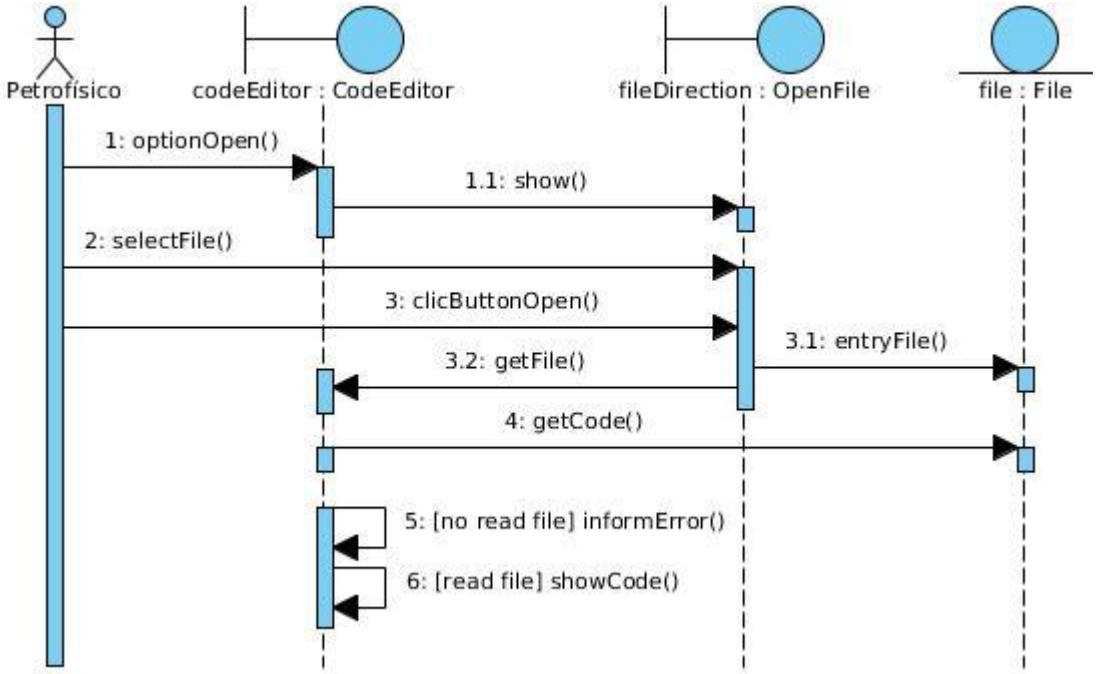

Figura 25. DSA CU2 (Cargar)

3.1.2.5. DSA CU Ejecutar modelo petrofísico

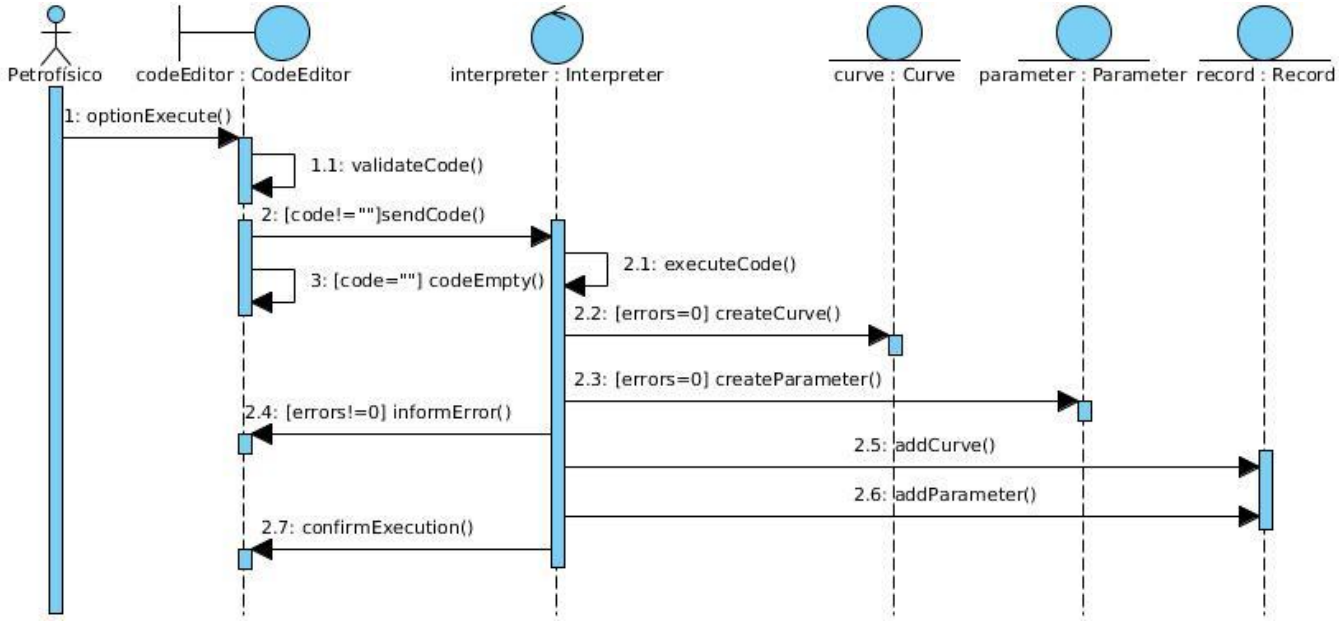

Figura 26. DSA CU3

3.1.2.6. DSA CU Visualizar error

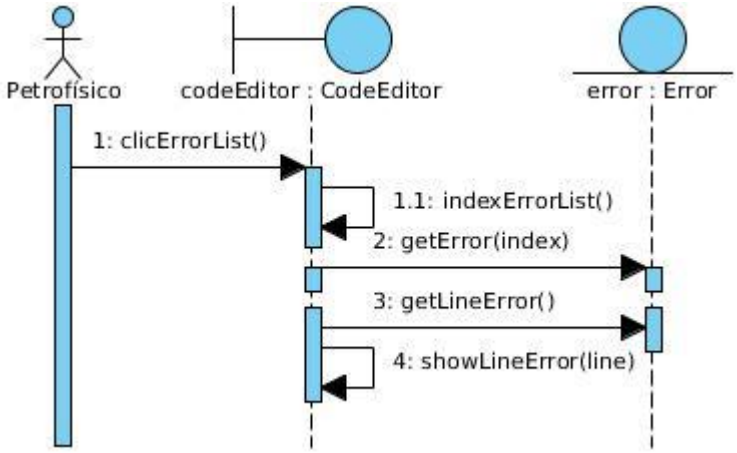

Figura 27. DSA CU4

Después de haberse expuesto los principales eventos y operaciones que se desarrollan dentro de cada caso de uso en el análisis, se tiene una primera aproximación de lo que será el sistema.

3.2. Estilo arquitectónico

La arquitectura del software es el diseño de más alto nivel de la estructura de un sistema. La importancia de esta radica en que sus diferentes elementos garanticen el cumplimiento de las características tanto en la funcionalidad que el sistema debe ofrecer como en los requerimientos de calidad que debe satisfacer.

Cada estilo arquitectónico describe una categoría del sistema que contiene: un conjunto de componentes, que realiza una función requerida por el sistema; un conjunto de conectores que posibilitan la comunicación, la coordinación y la cooperación entre los componentes; restricciones que definen como se puede integrar los componentes que forman el sistema; y modelos semánticos que permiten al diseñador entender las propiedades globales de un sistema para analizar las propiedades conocidas de sus partes constituyentes. (23)

La arquitectura definida en el módulo para el cálculo de parámetros petrofísicos está sustentada por el estilo arquitectónico en capas. Este tipo de arquitectura permite probar los componentes por separado, que se lleve a cabo un desarrollo paralelo (en cada capa), mantenimiento y soporte más sencillo, mayor flexibilidad y también provee una alta escalabilidad. Las capas definidas son: capa de presentación, capa de negocio y capa de datos.

Figura 28. Arquitectura basada en capas

Capa de presentación: Esta capa es la que ve el usuario, presenta el sistema al usuario, le comunica la información y captura la información en un mínimo de proceso. Esta capa se comunica con la capa de negocio. Contiene las vistas que permiten la entrada de datos para calcular parámetros petrofísicos básicos y desarrollar la codificación de modelos petrofísicos.

Capa de negocio: Aquí es donde se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para almacenar o recuperar información. Contiene las clases que permiten la interpretación y ejecución de modelos petrofísicos.

Capa de datos: Es donde residen los datos y es la encargada de acceder a los mismos. La información está representada por las clases que definen el registro de pozo activo con el cual trabaja el usuario.

Algunas vistas de la capa de presentación acceden directamente a la capa de datos en determinadas funcionalidades de actualización de campos que dependen de la información del registro, usando la variación del estilo de arquitectura basado en capas: saltos de capas (*layer skipping*). Este permite que las capas puedan invocar otras capas más profundas que las que están directamente debajo de ellas, incrementando el rendimiento.

Dentro de la capa de negocio, específicamente en la implementación de las fases definidas para la construcción del intérprete, se usó el estilo de flujo de datos: tuberías y filtros. El sistema tubería-filtros se percibe como una serie de transformaciones sobre sucesivas piezas de los datos de entrada. Los datos entran al sistema y fluyen a través de los componentes.

3.3. Patrones de diseño

Los patrones de diseño (*design patterns*) son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. A continuación se hace referencia a algunos patrones usados en la solución propuesta.

3.3.1.Patrones GRASP

Los patrones GRASP (*General Responsibility Assignment Software Patterns*) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones.

A continuación se describen un conjunto de este tipo de patrones tenidos en cuenta en el diseño e implementación del módulo presentado.

Experto: Asignar una responsabilidad al experto en información; la clase que tiene la información necesaria para llevar a cabo la responsabilidad.

Creador: Crear una nueva instancia por la clase que tiene la información necesaria para realizar la creación del objeto, usa directamente las instancias creadas del objeto, almacena o maneja varias instancias de la clase, o contiene o agrega la clase.

Bajo acoplamiento: Diseñar con el objetivo de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

Controlador: Asignar la responsabilidad de gestionar un mensaje de un evento del sistema a una clase que represente una de estas dos opciones: representa el sistema global, dispositivo o subsistema (controlador de fachada), o representa un escenario de caso de uso en el que tiene lugar el evento del sistema (controlador de caso de uso o de sesión).

3.3.2.Patrones GoF

Los patrones de diseño GoF (*Gang of Four*), término que hace referencia a los cuatro autores del libro "*Design Patterns: Elements of Reusable Object-Oriented Software*": Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides, constituyen una serie de posibles soluciones a problemas que suelen ser comunes en el desarrollo de software.

Clasificación según su propósito:

- Creacionales: Definen la mejor manera en que un objeto es instanciado. El objetivo de estos patrones es abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.

- De Comportamiento: Permiten definir la comunicación entre los objetos del sistema y el flujo de la información entre los mismos.
- Estructurales: Permiten crear grupos de objetos para ayudar a realizar tareas complejas.

Facade: Patrón estructural cuyo propósito es proporcionar una interfaz de alto nivel, unificada a un conjunto de interfaces en un subsistema, haciendo más fácil su uso, minimizando las comunicaciones y dependencias entre el cliente y los subsistemas. (24) Para estructurar un sistema en capas, la fachada define el punto de entrada de cada nivel, se pueden simplificar las dependencias obligando a los subsistemas a comunicarse únicamente a través de sus fachadas. Es implementado para acceder a la capa de negocio a través de la clase *Interpreter*.

Visitor: Patrón de comportamiento que representa una operación sobre los elementos de una estructura de objetos. Permite definir una nueva operación sin cambiar las clases de los elementos sobre los que opera. Proporciona una forma fácil y sostenible de ejecutar acciones en una familia de clases. (24) Es implementado por la clase *Encoder* para recorrer el árbol de sintaxis abstracta (AST) y construir el código intermedio visitando a cada uno de sus nodos.

3.4. Modelo de diseño

El diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción. Esto contribuye a una arquitectura estable y sólida. En el diseño se modela el sistema y se le da forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. Una entrada esencial en el diseño es el resultado del análisis, o sea el modelo de análisis, que proporciona una comprensión detallada de los requisitos.

3.4.1. Diagrama de clases del diseño

El diagrama de clases del diseño (DCD) describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. A diferencia de las clases conceptuales del modelo del dominio, las clases de diseño muestran las definiciones de las clases de software en lugar de los conceptos del mundo real. (13)

Para una mejor organización el modelo de diseño se ha estructurado de acuerdo a las capas definidas en la arquitectura. A continuación se muestran los diagramas de clases del diseño obtenidos.

3.4.1.1. DCD Capa de presentación

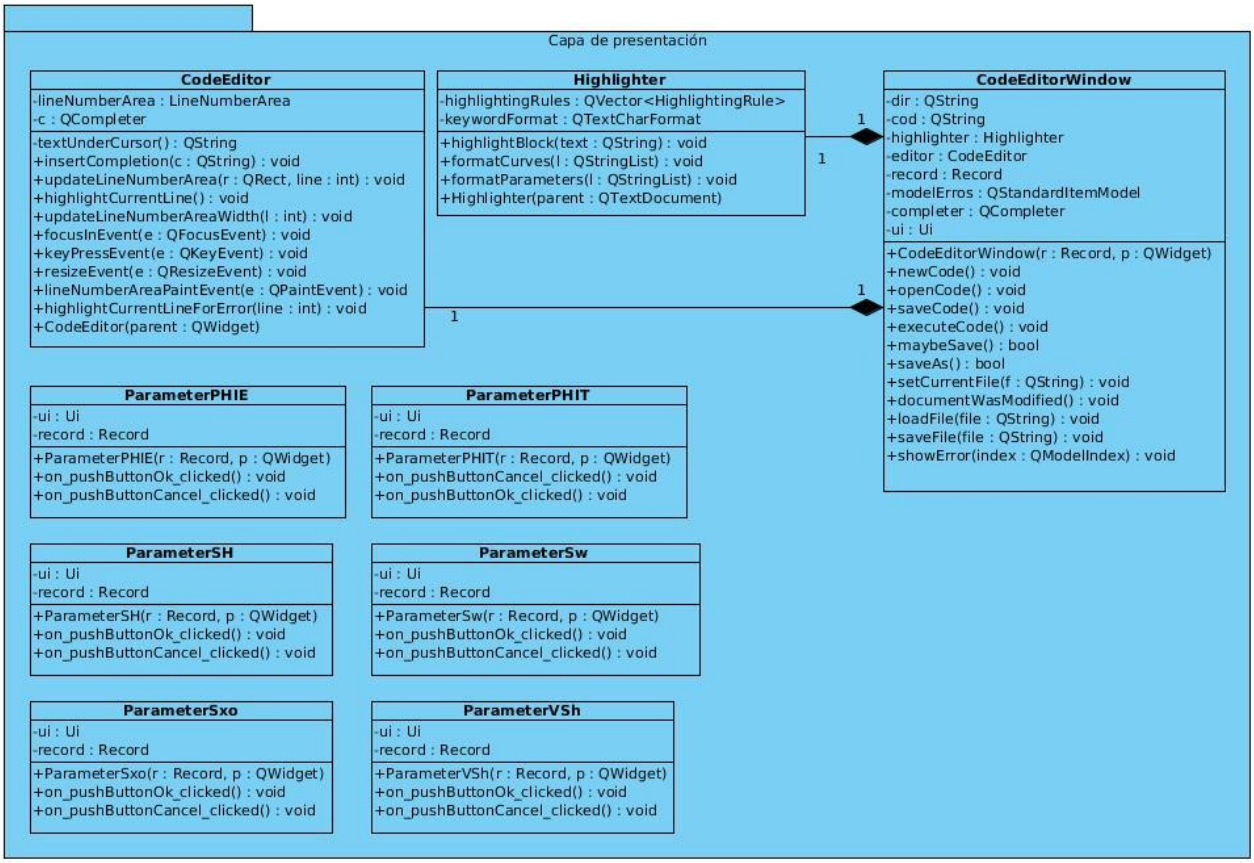

Figura 29. DCD Capa de presentación

La capa de presentación contiene las clases e interfaces con las que el usuario interactúa para la realización de las funcionalidades definidas en el módulo.

CodeEditorWindow: Clase que define la interfaz para la creación, ejecución, salva y carga de la codificación de modelos petrofísicos.

CodeEditor: Clase que se integra al *CodeEditorWindow* representando el editor de código.

HighLighter: Clase que se integra al *CodeEditorWindow* y posibilita el diseño y aplicación de un formato a la codificación.

ParameterPHIT: Clase que define la interfaz con la que interactúa el usuario para el cálculo de Porosidad Total.

ParameterPHIE: Clase que define la interfaz con la que interactúa el usuario para el cálculo de Porosidad Efectiva.

ParameterSw: Clase que define la interfaz con la que interactúa el usuario para el cálculo de Saturación de Agua.

ParameterVSh: Clase que define la interfaz con la que interactúa el usuario para el cálculo del Volumen de Arcilla.

ParameterSxo: Clase que define la interfaz con la que interactúa el usuario para el cálculo de Saturación de la zona lavada.

ParameterSH: Clase que define la interfaz con la que interactúa el usuario para el cálculo de Saturación de Hidrocarburos.

3.4.1.2. DCD Capa de negocio

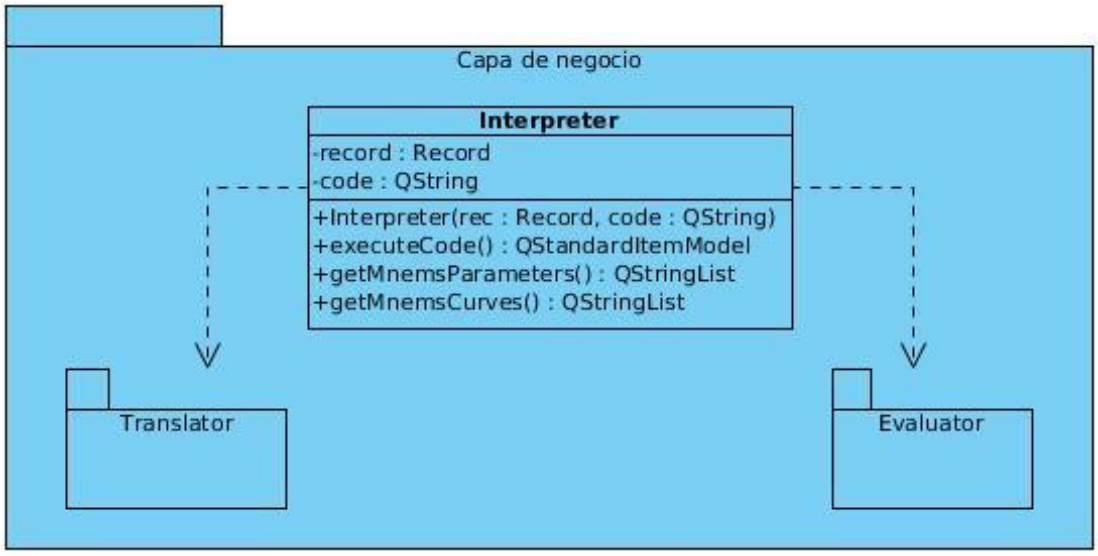

Figura 30. DCD Capa de negocio

La capa de negocio contiene las clases encargadas de analizar y ejecutar la codificación de los modelos petrofísicos. Las clases que analizan el código se han agrupado en el paquete *Translator* y las que ejecutan las instrucciones del código en el paquete *Evaluator*.

Interpreter: Clase que representa la fachada con la que se comunican las vistas de la capa de presentación para la ejecución de la codificación de un modelo petrofísico.

3.4.1.2.1. DCD Translator

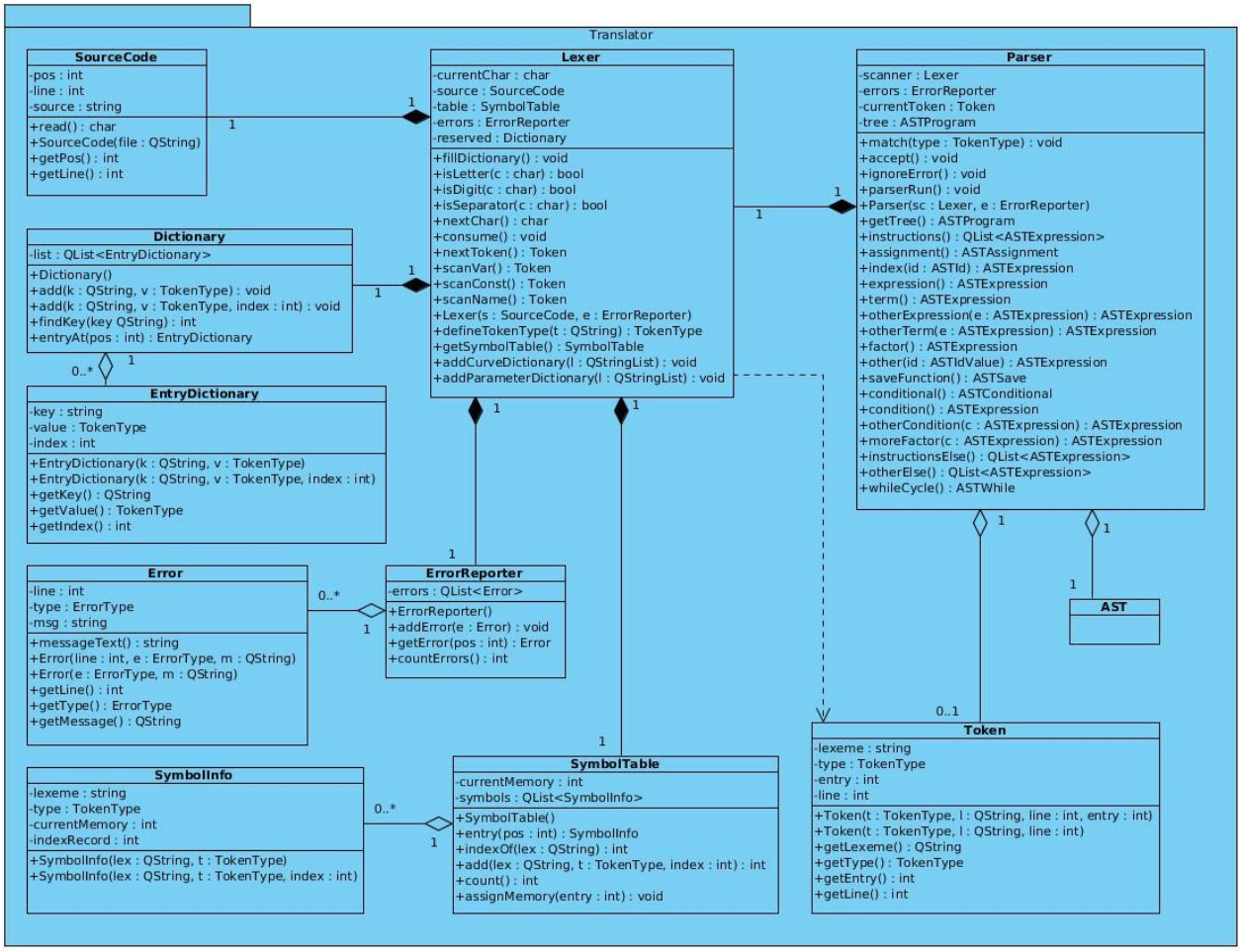

Figura 31. DCD Translator

El paquete *Translator* contiene las clases que realizan el análisis léxico, sintáctico y semántico del código fuente, generando una representación interna del mismo.

SourceCode: Clase que representa el código fuente, o sea, la codificación de un modelo petrofísico.

EntryDictionary: Estructura para almacenar cadenas definas en el lenguaje como funciones, propiedades, constantes y nomencladores de curvas y parámetros del registro activo.

Dictionary: Clase que gestiona los objetos *EntryDictionary*.

Error: Estructura para representar los diferentes tipos de errores que puedan surgir durante el análisis y ejecución del código.

ErrorReporter: Clase para gestionar los objetos *Error*.

SymbolInfo: Estructura para representar las unidades básicas del código conocidas como lexemas y algunas características que se van a ir actualizando en las diferentes fases para el análisis del código.

SymbolTable: Clase para gestionar los objetos *SymbolInfo*, representa la tabla de símbolos que se usa en los procesos de compilación.

Token: Estructura básica del código que va siendo generada por el analizador léxico (*Lexer*).

Lexer: Clase que implementa el análisis léxico del código. Recibe como entrada el código fuente y va generando como salida los objetos *Token*.

Parser: Clase que implementa el análisis sintáctico del código. Recibe como entrada los objetos *Token* generados por el *Lexer* y va analizando la estructura del código según las reglas sintácticas del lenguaje definido. Produce como representación interna del código el árbol de sintaxis abstracta (AST).

AST: Clase que representa el árbol de sintaxis abstracta generado por el *Parser*. (Anexo 2. Diagrama de clases del diseño AST)

No se diseñó ninguna estructura que implementara la fase de análisis semántico del código ya que las restricciones semánticas que impone el lenguaje se basan en el tratamiento de los tipos de datos para las operaciones y en el origen de las variables, es decir, si constituyen una curva, un parámetro o una variable creada por el usuario. Las variables que almacenen valores de curvas tienen un comportamiento diferente a las que almacenan valores de parámetros o valores numéricos. El diseño de las reglas sintácticas impide la ocurrencia de estas restricciones. Tener identificados los nomencladores de curvas y parámetros previamente posibilita identificar el tipo de variable en cuestión. Las variables creadas por el usuario pueden cambiar su tipo de acuerdo al valor que van almacenando o al comportamiento que le da el usuario en la codificación. En este caso, durante la ejecución se puede fácilmente tener en cuenta estas restricciones según la función que definan los operadores.

3.4.1.2.2. DCD Evaluator

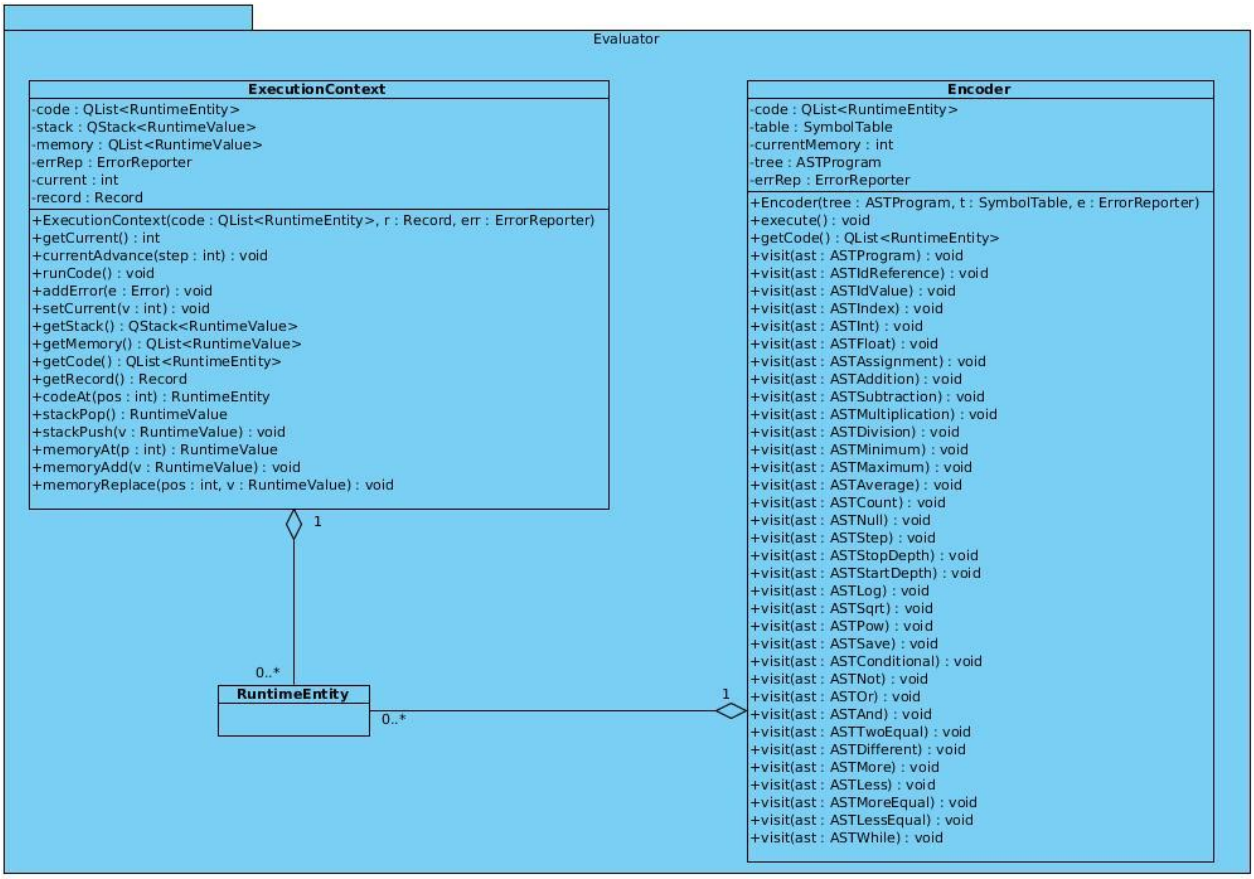

Figura 32. DCD Evaluator

El paquete *Evaluator* contiene las clases que toman la representación interna generada por el analizador sintáctico (*Parser*) y permiten la ejecución de las instrucciones para obtener los resultados. En general representan una máquina abstracta para ejecutar el conjunto de instrucciones del lenguaje definido.

Encoder: Clase que permite conformar un código intermedio ya que las instrucciones están dispersas en cada nodo del árbol sintáctico. Recibe como entrada el AST generado por el *Parser* y mediante la implementación del patrón *Visitor* visita cada uno de los tipos de nodos del árbol. Lo único que realiza dicha operación es identificarse a sí mismo invocando el método correspondiente de la clase visitante.

ExecutionContext: Clase que representa el contexto de ejecución agrupando las estructuras necesarias para simular el funcionamiento de la máquina abstracta en tiempo de ejecución. Simula la memoria de la máquina y contiene la pila de ejecución donde se almacenan los valores intermedios.

RuntimeEntity: Clase que permite representar el código intermedio con una subclase por cada tipo de instrucción. Implementará la forma en que se ejecuta cada instrucción actuando sobre el contexto de ejecución. (Anexo 3. Diagrama de clases del diseño RuntimeEntity)

3.4.1.3. DCD Capa de datos

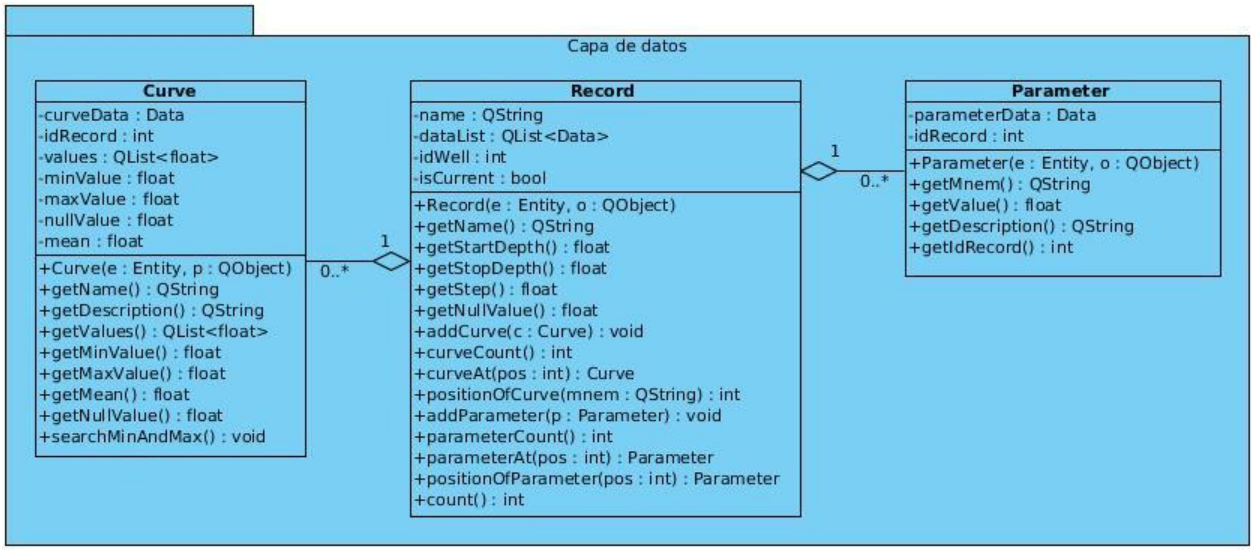

Figura 33. DCD Capa de datos

La capa de datos contiene las clases de acceso y representación de la información que maneja el módulo, es decir, los datos contenidos en el registro de pozo activo.

Record: Estructura que representa un registro de pozo. Contiene la información de las curvas y parámetros que lo conforman.

Curve: Estructura que representa la información de una curva de registro de pozo.

Parameter: Estructura que representa la información de un parámetro de registro de pozo.

3.4.2.Diagramas de secuencia del diseño

En esta etapa, las clases tienen ya definidas las operaciones que en la parte de análisis constituían fundamentalmente abstracciones del dominio del problema. A continuación se presentan los diagramas de secuencia del diseño (DSD) en donde se aplican las modificaciones realizadas.

3.4.2.1. DSD CU Calcular parámetro petrofísico (Sección PHIT)

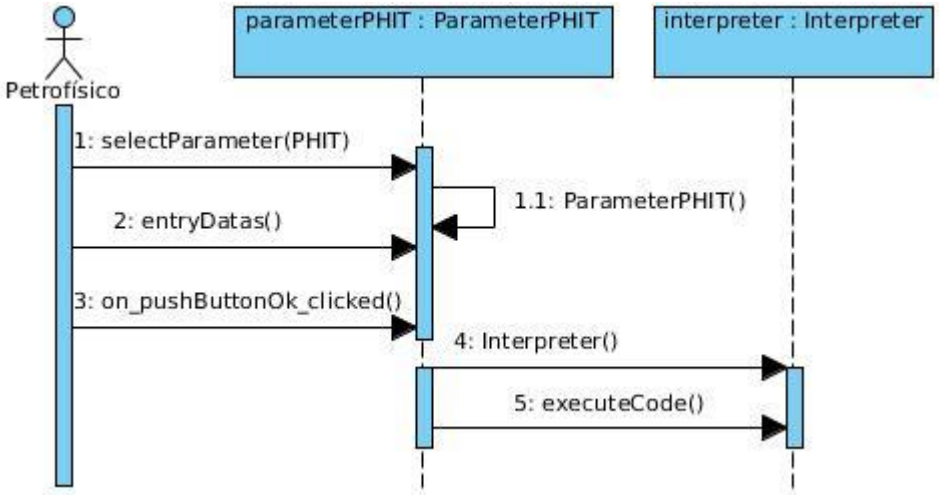

Figura 34. DSD CU1 (PHIT)

Los diagramas de secuencia para las otras secciones del CU Calcular parámetro petrofísico se comportan con el mismo flujo de mensajes y eventos entre objetos, solo que cada caso define su interfaz correspondiente.

3.4.2.2. DSD CU Gestionar modelo petrofísico (Sección Codificar)

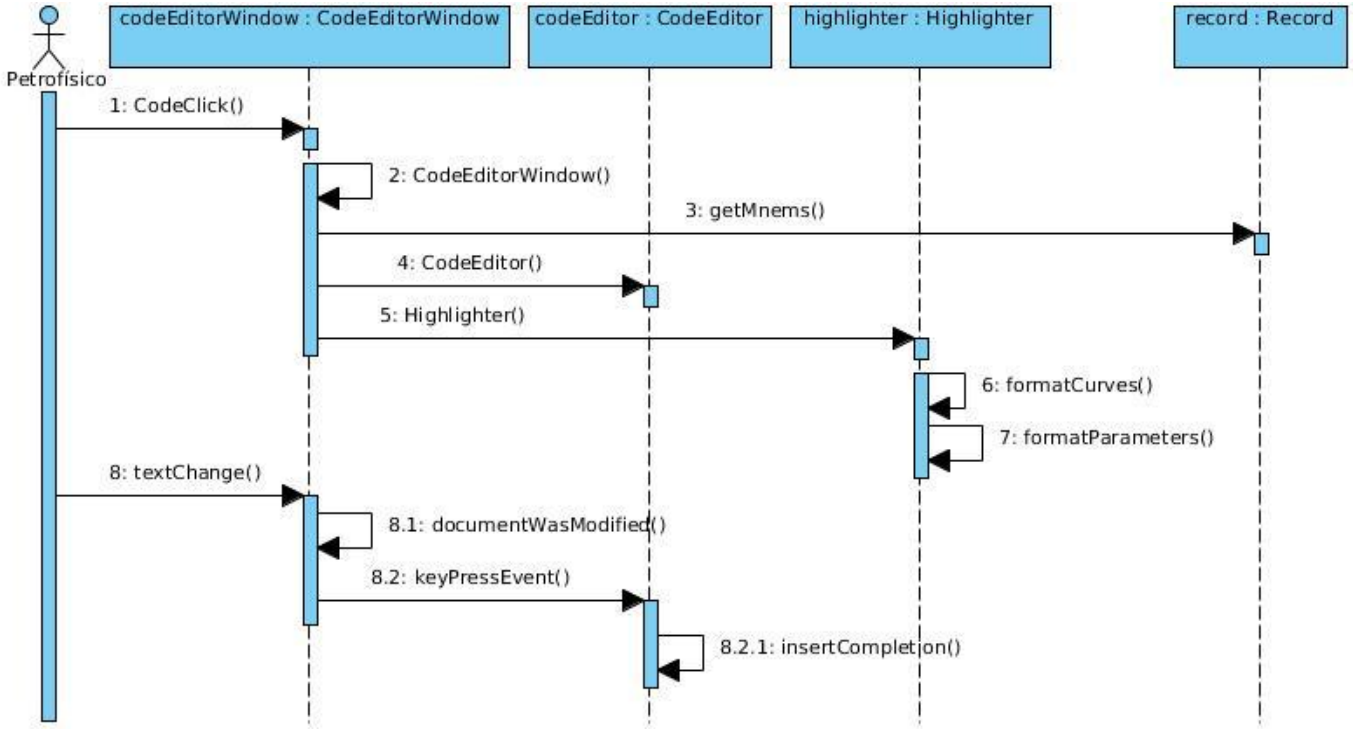

Figura 35. DSD CU2 (Codificar)

3.4.2.3. DSD CU Gestionar modelo petrofísico (Sección Guardar)

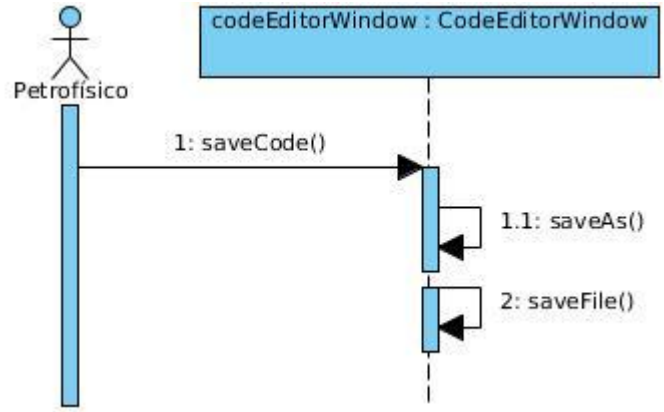

Figura 36. DSD CU2 (Guardar)

3.4.2.4. DSD CU Gestionar modelo petrofísico (Sección Cargar)

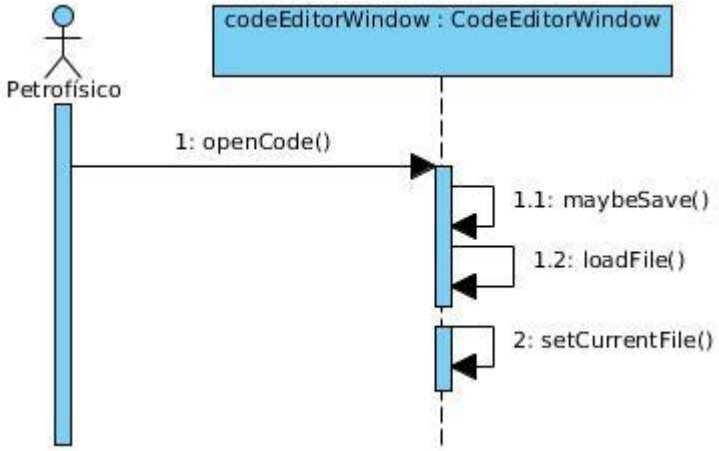

Figura 37. DSD CU2 (Cargar)

3.4.2.5. DSD CU Ejecutar modelo petrofísico

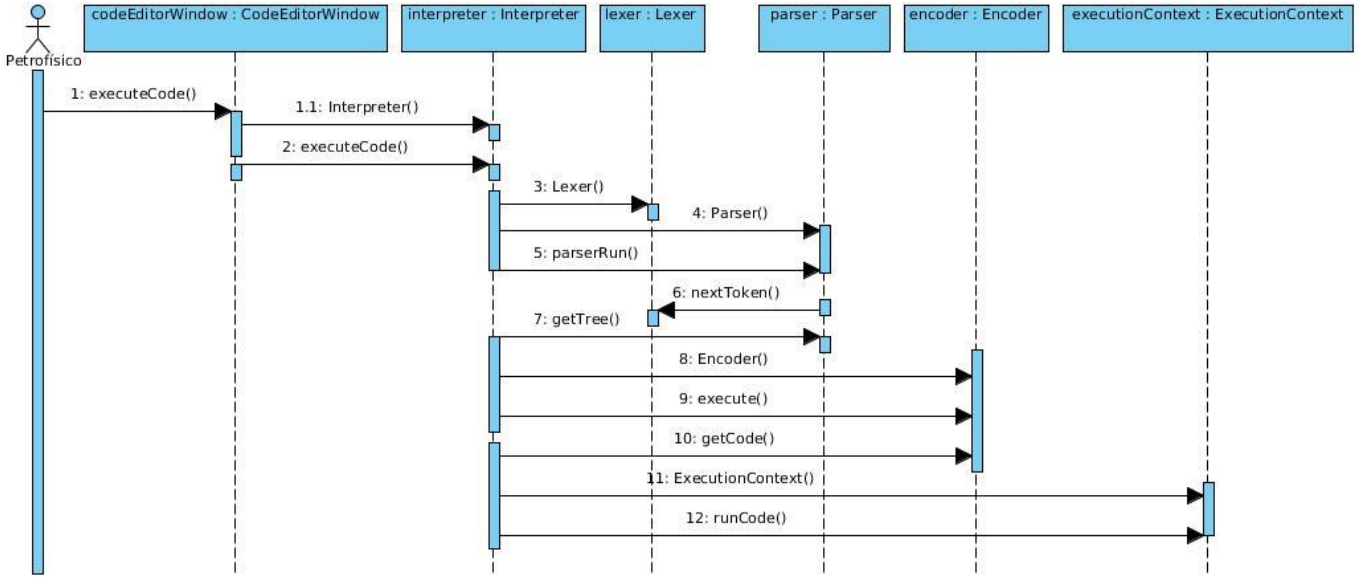

Figura 38. DSD CU3

3.4.2.6. DSD CU Visualizar error

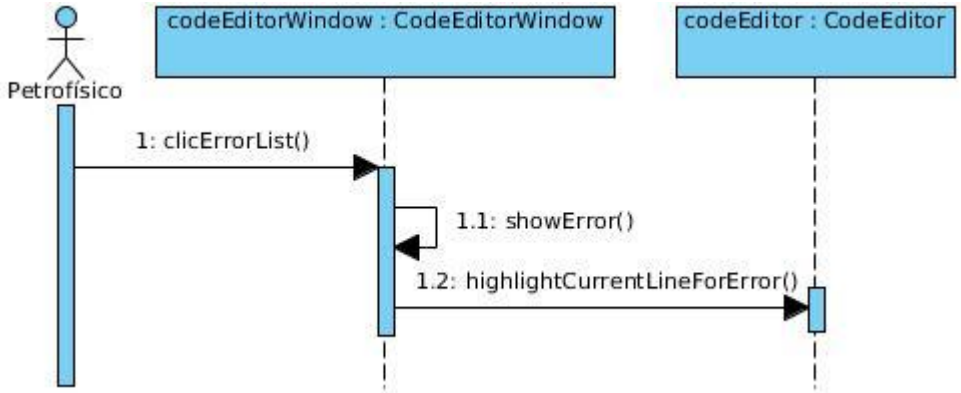

Figura 39. DSD CU4

Al terminar la etapa de diseño se ha refinado suficientemente el diagrama de clases y las relaciones entre estas. También se han definido los mensajes que intercambian los objetos para la realización de los casos de uso.

3.5. Modelo de implementación

El modelo de implementación está compuesto por un conjunto de subsistemas y componentes que constituyen la composición física de la implementación del sistema. Para lograr una mejor estructura y organización en la implementación del módulo propuesto se organizaron los componentes de acuerdo a su ubicación en las capas definidas en la arquitectura.

3.5.1.Diagrama de componentes

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre sus elementos. Los componentes representados pueden ser datos, archivos, ejecutables, código fuente y directorios. (22)

A continuación se muestran los diagramas de componentes obtenidos en la implementación de la solución propuesta.

3.5.1.1. Diagrama de componentes Capa de presentación

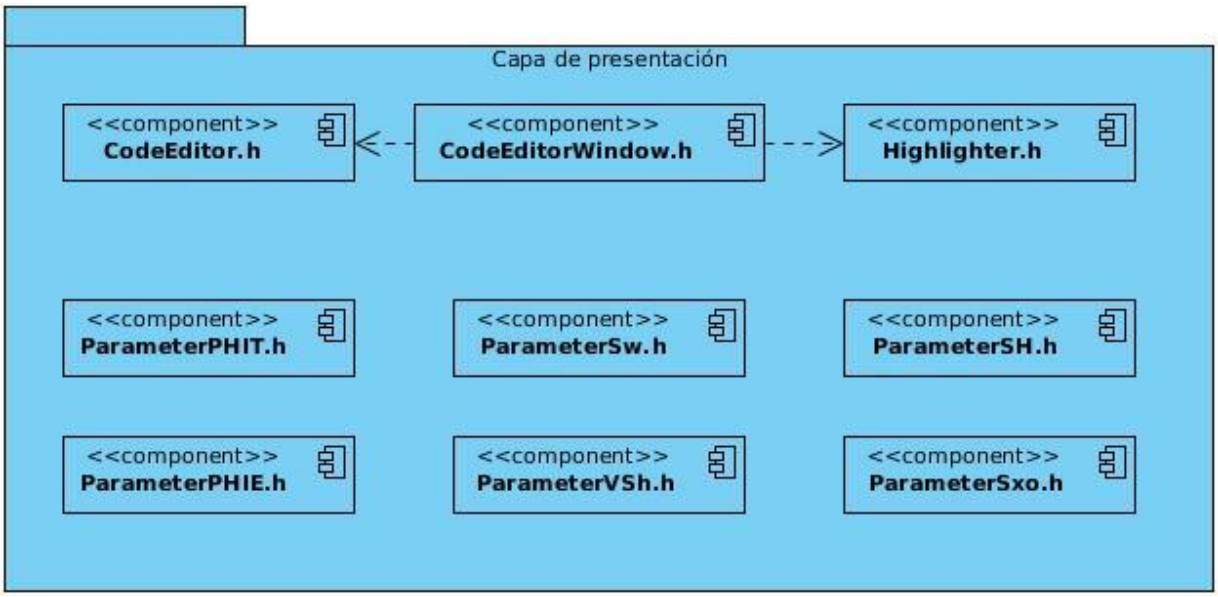

Figura 40. Diagrama de componentes Capa de presentación

3.5.1.2. Diagrama de componentes Capa de negocio

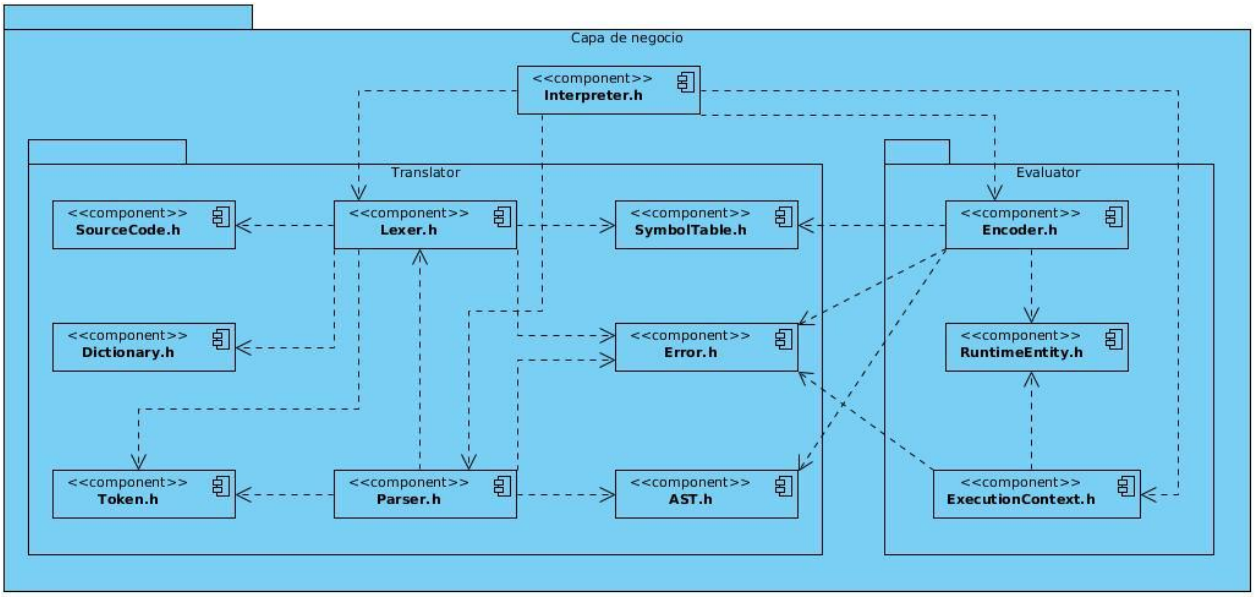

Figura 41. Diagrama de componentes Capa de negocio

3.5.1.3. Diagrama de componentes Capa de datos

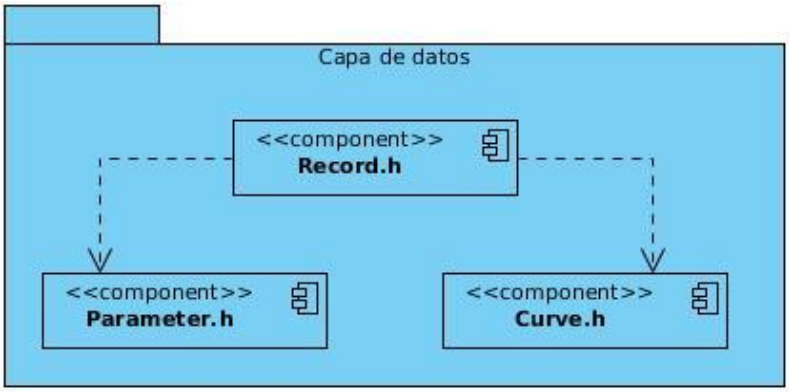

Figura 42. Diagrama de componentes Capa de datos

3.6. Definición del lenguaje

Un lenguaje de programación permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis. Estos programas se ejecutarán por un computador que realizará las tareas descritas. El programa debe ser comprendido tanto por personas como por computadores. La utilización de un lenguaje de programación requiere, por tanto, una comprensión mutua por parte de personas y máquinas.

Para la codificación de modelos petrofísicos se ha diseñado un lenguaje que permita a los especialistas desarrollar programas para el cálculo de parámetros manipulando la información de los registros de

pozos. Constituye una solución de compromiso entre las necesidades de los especialistas petrofísicos que serán los que codifiquen sus programas, y la máquina que ejecutará dichos programas. Las declaraciones, instrucciones y funciones se han concebido para entender mejor lo que se ha escrito en la codificación. Por otro lado, la utilización de un vocabulario limitado y de unas reglas estrictas son concesiones para facilitar el proceso de traducción.

Se utilizó la notación BNF para especificar la sintaxis del lenguaje. (Anexo 4. Gramática del lenguaje)

3.6.1.Variables, comportamiento y tipos de datos

Las variables se especifican con caracteres alfanuméricos y el signo guión bajo. Todas deben iniciar con una letra alfabética. Ejemplo: Depth, aux, RXO_1, VSh_0.

El comportamiento de las variables se diferencia en cuanto al origen de las mismas. Las variables pueden representar una curva o un parámetro del registro de pozo a través de los nomencladores con que están definidos en el mismo, o una variable creada por el usuario.

Los tipos de datos que se manejan son números enteros, números reales, valores de curva y valor puntual de parámetro. Estos no se especifican en la declaración de las variables. Las variables creadas por el usuario son las únicas que adquieren un determinado tipo de dato en alguna instrucción de asignación. Las variables que representan curvas contienen los valores de la curva (lista de números reales) y las variables que representan parámetros contienen el valor del parámetro (número real).

3.6.2.Funciones y constantes

Para las variables que representan curvas o que su tipo de dato se maneja como valores de curva, se pueden utilizar las siguientes funciones:

- min: devuelve el valor mínimo de los valores de curva
- max: devuelve el valor máximo de los valores de curva
- average: devuelve el promedio de los valores de curva

Su acceso es a través del signo de punto después del nombre de la variable, ejemplo: DPTH.min.

Se manejan constantes numéricas enteras y reales, ejemplo: 12, 0.25.

Se puede acceder a los valores constantes del registro a través de:

- count: devuelve la profundidad del registro (cantidad de valores de curva)

- nullValue: devuelve el valor nulo que maneja el registro
- step: devuelve el paso de profundidad del registro
- startDepth: devuelve la profundidad inicial del registro
- stopDepth: devuelve la profundidad final del registro

Para salvar los resultados de alguna operación en el registro se usa la función save(A,B), donde A es una variable creada por el usuario y B la descripción con que se guardará la misma. Ejemplo: save(Sw_1,"Saturacion de agua").

3.6.3.Operadores y funciones matemáticas

Operadores: + (adición), - (substracción), * (multiplicación), / (división).

Funciones:

- pow(A,B): exponencial de A de orden B
- sqrt(A,B): raíz de A de orden B
- log(A,B): logaritmo de A en base B

Tanto los operadores y funciones matemáticas van a manejar internamente su comportamiento de acuerdo al tipo de dato de sus operandos. Por ejemplo: Si A representa una curva y B un valor puntual, cuando se ejecuta A+B el resultado representaría los valores de A, cada uno sumado con el valor de B. Si A y B fueran valores puntuales, pues el resultado sería la suma de los dos valores, representando un valor puntual.

3.6.4.Operadores lógicos y de comparación

Operadores lógicos:

- A or B: verdadero si A o B es verdadero, falso en caso contrario
- A and B: verdadero si A y B son verdaderos, falso en caso contrario
- not(A): verdadero si A es falso, falso en caso contrario

Operadores de comparación:

- A == B: verdadero si A igual B, falso en caso contrario
- A != B: verdadero si A diferente de B, falso en caso contrario
- A > B: verdadero si A es mayor que B, falso en caso contrario
- A < B: verdadero si A es menor que B, falso en caso contrario

- A >= B: verdadero si A es mayor o igual que B, falso en caso contrario
- A <= B: verdadero si A es menor o igual que B, falso en caso contrario

3.6.5.Ciclos y condicionales

Existe la posibilidad de que el usuario necesite manipular los valores puntuales de las curvas o establecer un comportamiento diferente según el problema. Para dar respuesta a este tipo de situación se crearon estructuras para poder especificar ciclos y condicionales.

Para los ciclos se definió la estructura while:

```
while(<condición>){
        <instrucciones>
}
```

Para las condicionales se definió la estructura if-else:

```
if(<condición>){
        <instrucciones>
}
else{
        <instrucciones>
}
```

3.6.6.Otros signos y operadores

Para la instrucción de asignación se utilizó el signo de igualdad. Ejemplo: A=B, donde A tiene que ser una variable creada por el usuario y B alguna expresión aritmética.

El signo punto y coma ‘;’ se emplea para separar una instrucción de otra.

Los signos de corchetes ‘[ ]’ se usan para acceder a los valores de las curvas. Por ejemplo: DPTH[0] indica el valor de la curva DPTH en la posición 0. Solo es aplicable a las variables que contengan como tipo de dato valores de curva, o las que representan curvas del registro.

3.7. Pruebas de la solución

El instrumento adecuado para determinar el status de la calidad de un producto de software es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que cumple con los

requerimientos. Las pruebas son técnicas de validación predominantes o procesos de ejecución de un programa con la intención de descubrir errores de una aplicación que antes no se habían descubierto.

Al módulo desarrollado se le aplicaron varias pruebas para validar el cumplimiento de los requisitos enunciados, dar una indicación de calidad y comprobar su funcionamiento. Las pruebas que se muestran son de tipo caja negra (*Black-Box Testing*). Estas pruebas están centradas en los requerimientos funcionales del software. Permiten derivar conjuntos de condiciones de entrada que ejerciten completamente los requerimientos funcionales de un programa.

3.7.1.Casos de prueba

Un caso de prueba permite detallar la forma en que se va a probar el sistema, incluyendo los datos de entrada con los que se realizará la prueba correspondiente, las condiciones de ejecución y resultados obtenidos. Deben verificar si el producto satisface los requerimientos del usuario, tal y como se describe en las especificación de los requerimientos y si el producto se comporta como se desea, tal y como se describe en las especificaciones funcionales del diseño. (13)

A continuación se muestran los casos de prueba realizados a la aplicación, teniendo en cuenta la ejecución de cada caso de uso del sistema.

3.7.1.1. Caso de prueba CU Calcular parámetro petrofísico

Tabla 6. Caso de prueba CU1

Sección “PHIT Porosidad Total”		
Entrada	**Resultados**	**Condiciones**
El usuario especifica los datos para el cálculo de PHIT: Porosidad Neutrónica: NPHI Porosidad por Densidad: DPHZ Rocas Gasíferas: Si	El sistema ejecuta el modelo con los datos de entrada para el cálculo de PHIT. El sistema informa el resultado: PHIT_0	El usuario está registrado en el sistema y tiene un registro activo. NPHI y DPHZ son curvas del registro.

Para las diferentes secciones del CU Calcular parámetros petrofísicos se ejecutaron pruebas. Cada una de ellas demostró que el diseño de las interfaces definidas para la entrada y especificación de los datos, no permite que el usuario inserte o seleccione valores inválidos, ya que los campos que representan curvas tienen por defecto las curvas del registro, y los campos numéricos se capturan con los componentes visuales spinBox y doubleSpinBox. Estos manejan las restricciones en su programación. En cuanto a los resultados, los parámetros que adquieran un valor no válido en alguna operación, toman

el valor nulo que establece el registro activo. Esta tarea se realiza en el momento de ejecución de los modelos petrofísicos.

3.7.1.2. Caso de prueba CU Gestionar modelo petrofísico

Tabla 7. Caso de prueba CU2

Sección "Codificar modelo"		
Entrada	**Resultados**	**Condiciones**

El usuario escribe la siguiente codificación:
aux=DEPT+BSAL*pow(step,2);
save(aux,"Prueba");
	El sistema activa el completamiento de código para DEPT, BSAL, pow, step y save.
	El sistema establece el formato al código escrito.
		El usuario está registrado en el sistema y tiene un registro activo.
		DEPT es una curva del registro.
		BSAL es un parámetro del registro.

Sección "Guardar modelo"		
Entrada	**Resultados**	**Condiciones**
El usuario escribe un nombre de archivo vacío.	El sistema muestra la etiqueta "Escriba un nombre de archivo".	El usuario está registrado en el sistema y tiene un registro activo. El usuario ha codificado un modelo petrofísico.
El usuario escribe como nombre del archivo "micode".	El sistema guarda la codificación del modelo en el archivo "micode.txt".	El usuario está registrado en el sistema y tiene un registro activo. El usuario ha codificado un modelo petrofísico.
Sección "Cargar modelo"		
Entrada	**Resultados**	**Condiciones**
El usuario indica el archivo origen "code.txt".	El sistema muestra el código que contiene el archivo "code.txt" en el editor.	El usuario está registrado en el sistema y tiene un registro activo.

3.7.1.3. Caso de prueba CU Ejecutar modelo petrofísico

Tabla 8. Caso de prueba CU3

Entrada	Resultados	Condiciones
El usuario indica ejecutar un código vacío.	El sistema muestra el mensaje “Código vacío”.	El usuario está registrado en el sistema y tiene un registro activo.
El usuario indica ejecutar el código: aux = DEPT; DTP1 = aux + gl;	El sistema muestra errores en la lista de errores. Error semántico por el uso de la variable gl en la línea 2.	El usuario está registrado en el sistema y tiene un registro activo en el sistema. DEPT es una curva del registro.

El usuario indica ejecutar el código:
aux = DEPT;
i = 0;
while(i < count){
if(aux[i] > 0){
aux[i] = DEPT[i] + DFV[i];
}
i = i +1;
}
	El sistema muestra errores en la lista de errores.
	Uso incorrecto del carácter [en la línea 5.
		El usuario está registrado en el sistema y tiene un registro activo en el sistema.
		DEPT es una curva del registro.
		DFV es un parámetro del registro.

El usuario indica ejecutar el código:
aux = DEPT;
i = 0;
while(i < count){
if(aux[i] > 0){
aux[i] = DEPT[i] + DFV;
}
i = i +1;
}
MPar = aux*pow(GR,2);
	El sistema ejecuta el código y muestra el mensaje “Código ejecutado correctamente”.
	Adiciona al registro activo el resultado MPar.
		El usuario está registrado en el sistema y tiene un registro activo en el sistema.
		DEPT y GR son curvas del registro.
		DFV es un parámetro del registro.

save(MPar,"Param M");		

3.7.1.4. Caso de prueba CU Visualizar error

Tabla 9. Caso de prueba CU4

Entrada	Resultados	Condiciones

El usuario selecciona el error de la lista de errores que indica su ocurrencia en la línea 3.	El sistema señala la línea 3 del editor de código con un color oscuro.	El usuario está registrado en el sistema y tiene un registro activo.
		DEPT es una curva del registro.
		Se ha ejecutado el código:
		a = 0;
		VV = DEPT + a;
		save();

Para el desarrollo de las pruebas se cargaron diversos registros de pozos. Las pruebas funcionales se realizaron para verificar que el sistema implementa adecuadamente cada uno de los requisitos acordados para el software. De esta manera se comprobó que el módulo responde a las necesidades que dieron origen a su desarrollo. Se revisó también que las interfaces de la aplicación utilizarán correctamente el idioma definido, que mantuvieran concordancia entre sus textos y buena ortografía, y que los mensajes mostrados fueran claros y concisos. De forma general, las pruebas se ejecutaron exitosamente partiendo de que el éxito de las pruebas se encuentra en la detección de errores.

Conclusiones parciales

Durante este capítulo se han expuesto los principales artefactos de los flujos de trabajo: Análisis y Diseño, Implementación, Prueba y Despliegue. Quedaron evidenciadas las características principales del módulo desarrollado como solución al problema planteado, a través de las vistas de los diagramas de análisis y diseño. Los patrones de diseño utilizados ayudaron eficientemente y con resultados positivos a lograr un adecuado rendimiento del sistema. La realización del modelo de implementación permitió detallar los componentes creados en el desarrollo de la aplicación y la relación entre ellos. La definición de un lenguaje para la manipulación de la información de los registros de pozos y el cálculo de parámetros es la base para poder llevar a cabo la codificación y ejecución de modelos petrofísicos. Mediante las pruebas ejecutadas a la solución, se comprobaron las funcionalidades identificadas durante el proceso de desarrollo del software.

Conclusiones

AnPer es un sistema desarrollado con la finalidad de analizar e interpretar la información de los registros de pozos de petróleo. Las aplicaciones de los resultados de este proceso en el campo de la petrofísica son múltiples y por tanto puede llegar a ser de gran utilidad para la industria petrolera cubana. Después del desarrollo e integración del módulo presentado, AnPer proporciona la realización de una evaluación petrofísica más completa, brindando una herramienta capaz de calcular parámetros petrofísicos que no se obtienen directamente en los registros de pozos, dándole respuesta al problema y cumplimiento al objetivo planteado. Esto facilita la toma de decisiones que conduzcan a un mejor desempeño en las tareas de exploración y perforación.

En general, la solución presentada arroja las siguientes conclusiones:

- Se definieron modelos petrofísicos para el cálculo de parámetros básicos.
- Se especificó un lenguaje para la codificación de modelos petrofísicos.
- Se obtuvo un intérprete que permite ejecutar modelos, obtener los parámetros deseados e integrar los resultados al registro de pozo.
- Se obtuvo un editor de código que permite codificar, salvar, cargar y ejecutar modelos petrofísicos para el cálculo de parámetros.

Recomendaciones

Se ha desarrollado un módulo integrado al sistema AnPer que permite calcular parámetros petrofísicos, necesarios en el proceso de evaluación de los yacimientos. Sin embargo, es importante destacar que durante el desarrollo del mismo se han identificado ciertas mejoras que podrían implementarse en un futuro, en aras de darle una mayor efectividad y utilidad al producto obtenido. Es por ello que se recomienda para versiones posteriores, tener en cuenta los siguientes puntos:

- Permitir en la codificación y ejecución de modelos petrofísicos acceder simultáneamente a la información de varios registros de pozos.
- Incluir nuevos modelos petrofísicos para el cálculo de parámetros básicos.
- Integrar modelos petrofísicos creados por el usuario con componentes visuales para la entrada de datos.

Referencias bibliográficas

1. Vera Callao, Raysha. Interpretación básica de los registros geofísicos. Ingeniería Petrolera. [En línea] 2012. http://ingeniera-petrolera.blogspot.com/2012/04/interpretacion-basica-de-los-registros.html.

2. Estimación de Parámetros Petrofísicos y Máquinas de Vectores de Soporte. Del Mar, Angel. Colombia : UNAB - UIS, 2009.

3. Bisbé York, Esther María. Elementos básicos de geofísica de pozos. s.l. : CUPET, 2011.

4. Del Mar Ortega, Angel Ciro. Estimación de Porosidad y Volumen de Arcilla en formaciones rocosas. Venezuela : Universidad de Los Andes, 2008.

5. Interactive Petrophysics - User Manual. s.l. : Schlumberger, 2007.

6. Osorio, Rafael. Software petrolero petrofísico. Petroblogger. [En línea] 2010. http://www.ingenieriadepetroleo.com/2010/06/software-petrolero-petrofisico.html.

7. Labra Gayo, Jose Emilio, Cueva Lovelle, Juan Manuel y Izquierdo Castanedo, Raúl. Intérpretes y Diseño de Lenguajes de Programación. 2004.

8. Ruiz Catalán, Jacinto. Compiladores: teoría e implementación. s.l. : RC Libros, 2010.

9. Ceballos Carmona, Miguel Ángel. Del Funcionamiento y Comportamiento de los Compiladores. 2002.

10. Pérez Delgadillo, Lizbeth. Fases del proceso de compilación. Compiladores. [En línea] 2010. http://compiladorsistemas.blogspot.com/2010/11/fases-del-proceso-de-compilacion.html.

11. Rojas, Sergio y Mora Mata, Miguel. Traductores y Compiladores con LEX/YACC , JFLEX /CUP y JAVACC. 2005.

12. Herramientas para generar compiladores. Compiladores. [En línea] 2010. http://compiladorsistemas.blogspot.com/2010/11/herramientas-para-generar-compiladores.html.

13. Pressman, Roger. Ingeniería de Software un Enfoque Práctico. Quinta Edición.

14. Hernández Orallo, Enrique. El Lenguaje Unificado de Modelado UML. [En línea] http://www.disca.upv.es/enheror/pdf/ActaUML.PDF.

15. Sierra, Daniel. Visual Paradigm For Uml. 2007.

16. Gómez Ruiz, José Antonio. Introducción al lenguaje C/C++. Universidad de Málaga. [En línea] http://www.lcc.uma.es/~janto/ftp/fundinf/trans_t3.pdf.

17. Qt Creator. Qt Project. [En línea] 2012. http://qt-project.org/wiki/Category:Tools::QtCreator_Spanish.

18. Portilla Pauca, Christian. Qt Programando en C++. [En línea] 2012. http://xhrist14n.wordpress.com/2012/04/04/qt-programando-en-c/.

19. Presman, Roger. Ingeniería de Software un Enfoque Práctico. Sexta Edición.

20. Chaves, Michael. La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de Software. 2005.
21. González Cornejo, José Enrique. El Lenguaje de Modelado Unificado (UML). DocIRS. [En línea] http://www.docirs.cl/uml.htm.
22. Sommerville, Ian. Ingeniería de Software. 2005.
23. Land, Rikard. A brief survey of software architecture. 2002.
24. Gamma, Erich, y otros. Design Patterns: Elements of Reusable Object Oriented Software. 1994.
25. Integración de la información petrofísica y geológica: una tarea para los petrofísicos. Jerry, Lucia. s.l. : Oilfield Review, 2001.

Anexos

Anexo 1. Modelos para el cálculo de saturación de agua (Sw) en arenas arcillosas

Modelo Simandoux

$$Sw = \sqrt{\frac{a * Rw}{PHIE^2 * Rt} + \left(\frac{a * Rw * VSh}{2 * PHIE^2 * RSh}\right)^2} - \frac{a * Rw * VSh}{2 * PHIE^2 * RSh}$$

Modelo Saraband

$$Sw = \sqrt{\frac{a * Rw * (1 - VSh)}{PHIE^2 * Rt} + \left(\frac{a * Rw * VSh * (1 - VSh)}{2 * PHIE^2 * RSh}\right)^2}$$

Modelo Indonesian

$$Sw = \sqrt[n]{\frac{1}{Rt} * \left(\frac{VSh * \frac{1-VSh}{2}}{\sqrt{RSh}} + \sqrt{\frac{PHIE^m}{a * Rw}}\right)^2}$$

Siendo:

Rw: Resistividad del agua de formación.

PHIE: Porosidad efectiva.

Rt: Resistividad de la capa.

a: Tortuosidad.

m: Exponente de cementación.

n: Exponente de saturación.

VSh: Volumen de arcilla.

PHIE: Porosidad efectiva.

RSh: Resistividad de la roca arcillosa.

Anexo 2. Diagrama de clases del diseño AST

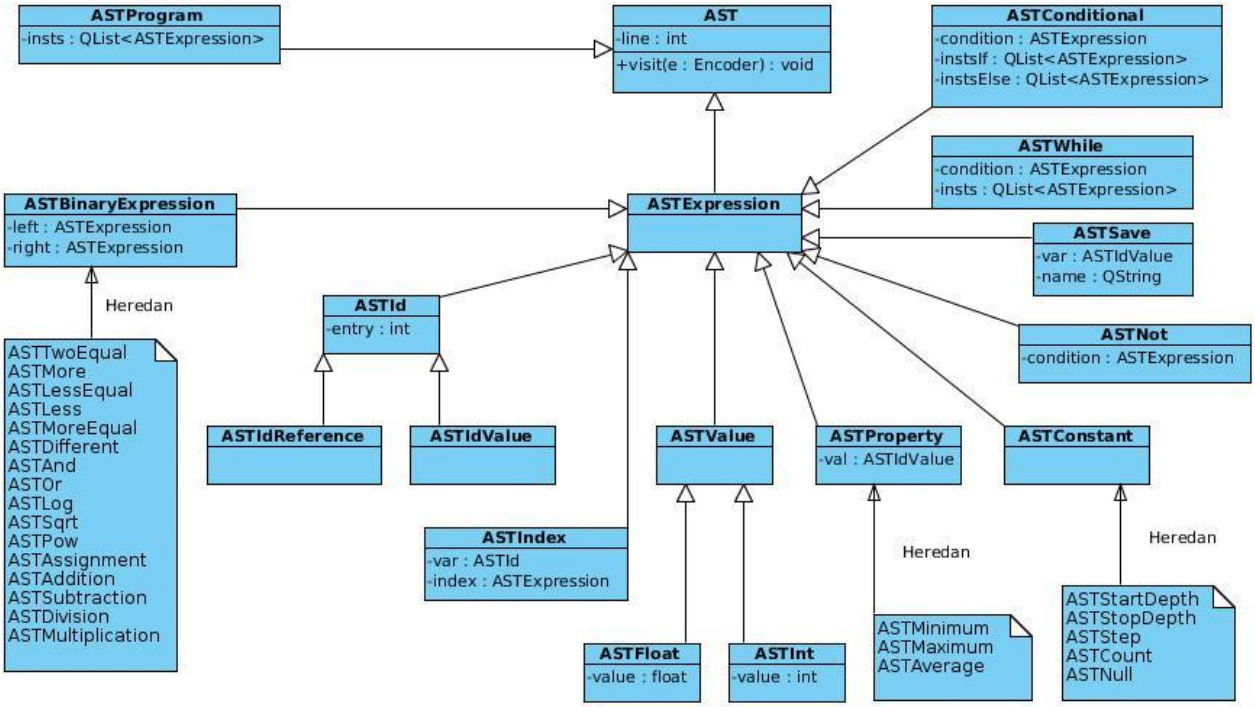

Figura 43. DCD AST

Anexo 3. Diagrama de clases del diseño RuntimeEntity

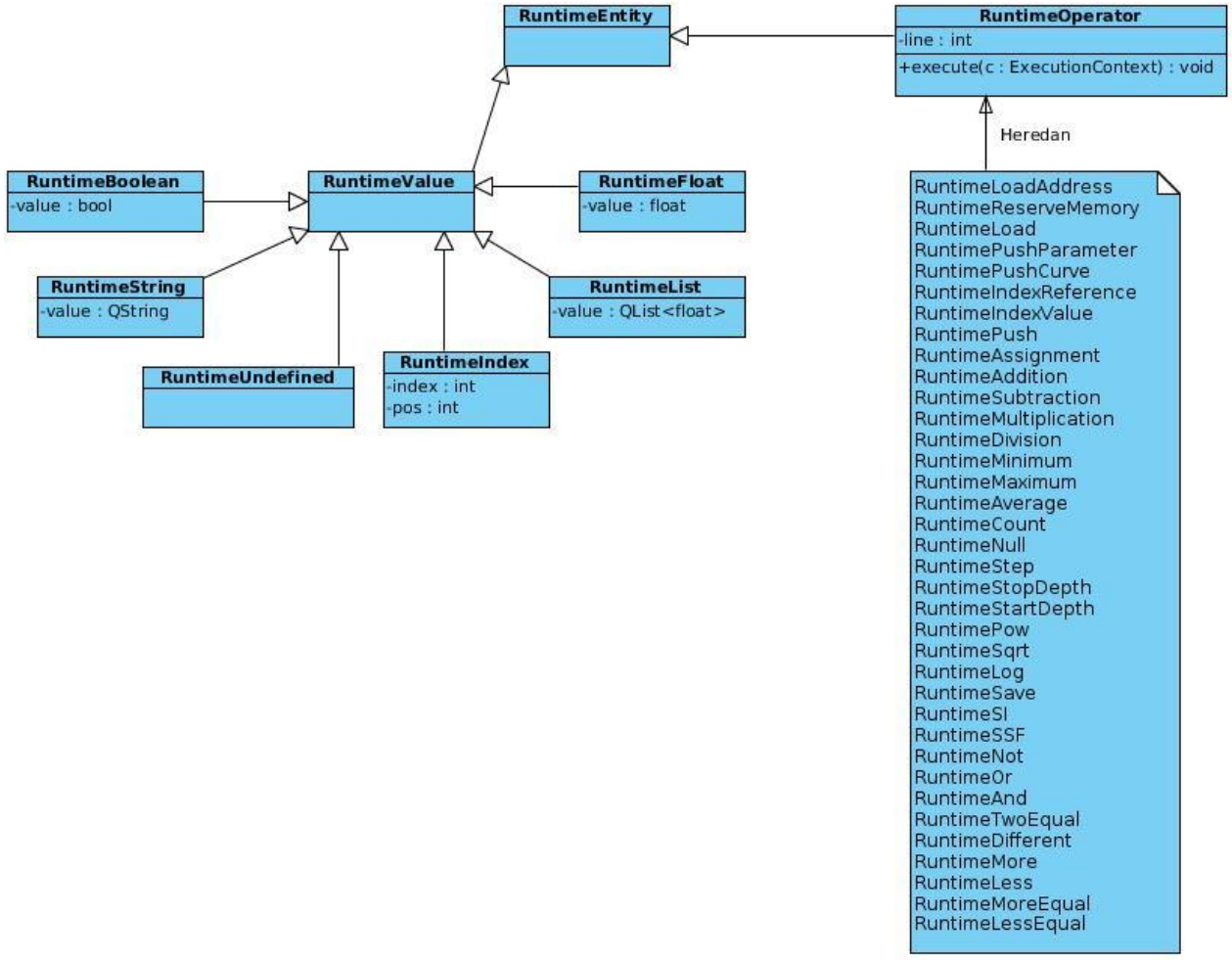

Figura 44. DCD RuntimeEntity

Anexo 4. Gramática del lenguaje

Terminales:

<instructions> <assignment> <index> <expression> <otherExpression> <term> <otherTerm> <factor> <other> <saveFunction> <conditional> <condition> <otherCondition> <moreFactor> <instructionsElse> <otherElse> <WhileCycle>

No Terminales:

var curv parameter float int log sqrt pow count nullValue step stopDepth startDepth min max average save name not or and if else while + - * / == < > <= >= . , ; ( ) [ ] { } = e

Símbolo Inicial: <instructions>

Producciones:

<instructions> --> <assignment> <instructions> | <conditional> <instructions> |

<saveFunction> <instructions> | <whileCycle> <instructions> | e

<assignment> --> var <index> = <expression> ;

<index> --> [<expression>] | e

<expression> --> <term> <otherExpression>

<otherExpression> --> + <term> <otherExpression> | - <term> <otherExpression> | e

<term> --> <factor> <otherTerm>

<otherTerm> --> * <factor> <otherTerm> | / <factor> <otherTerm> | e

<factor> --> (<expression>) | curv <other> | var <other> | parameter | float | int |
log (<expression> , <expression>) | pow (<expression> , <expression>) |
sqrt (<expression> , <expression>) |
count | nullValue | step | stopDepth | startDepth

<other> --> . <property> | <index>

<saveFunction> --> save (var , name) ;

<conditional> --> if (<condition>) { <instructions> } <instructionsElse>

<condition> --> not (<condition>) <otherCondition> |
<expression> <moreFactor> <otherCondition>

<otherCondition> --> or <condition> | and <condition> | e

<moreFactor> --> == <expression> | != <expression> |
> <expression> | < <expression> |
>= <expression> | <= <expression>

<instructionsElse> --> else <otherElse> | e

<otherElse> --> { <instructions> } | <conditional>

<WhileCycle> --> while (<condition>) { <instructions> }

Glosario de términos

Geofísica: Ciencia que estudia los fenómenos físicos que se producen en el planeta, destacando entre estos, el electromagnetismo, la propagación de ondas mecánicas en la corteza terrestre y la gravedad. Esta ciencia puede definirse como la aplicación de la física y la geología al estudio de los materiales que componen la corteza terrestre y de los campos de fuerza que surgen de ella y ejercen su influencia hacia el exterior. (3)

Petrofísica: Es la especialidad de caracterizar las propiedades físicas de las rocas mediante la integración del entorno geológico, perfiles de pozo, análisis de muestra de roca, sus fluidos e historias de producción. (3)

Litología: Es la parte de la geología que estudia las rocas, especialmente su tamaño de grano, tamaño de las partículas y sus características físicas y químicas. Incluye también su composición, su textura, tipo de transporte así como su composición mineralógica, distribución espacial y material cementante. (25)

Yacimiento petrolífero: Un yacimiento, depósito o reservorio petrolífero, es una acumulación natural de hidrocarburos en el subsuelo contenidos en rocas porosas o fracturadas (roca almacén). (3)

Árboles de sintaxis abstracta (AST): Es una representación de árbol de la estructura sintáctica abstracta (simplificada) del código fuente escrito en cierto lenguaje de programación. Cada nodo del árbol denota una construcción que ocurre en el código fuente. La sintaxis es abstracta en el sentido que no representa cada detalle que aparezca en la sintaxis verdadera. Los AST sirven para manejar la información semántica de un código. La forma más eficiente de manejar la información proveniente de un lenguaje de programación es la forma arbórea; por eso la estructura de datos elegida es un árbol. Además, construyendo AST a partir de un texto podemos obviar mucha información irrelevante; si un AST se construye bien, no habrá que tratar con símbolos de puntuación. Al contrario de los flujos, una estructura en árbol puede especificar la relación jerárquica entre los símbolos de una gramática.