UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 09

SUBSISTEMA DE IMPORTACIÓN Y EXPORTACIÓN DE REGISTROS DE POZO

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN INFORMÁTICA

AUTOR: *Miriam Yilieny González Torres.*

TUTOR: *Ing. Luis Manuel Refeca González.*

HABANA, 29 DE JUNIO DE 2010

"AÑO 52 DE LA REVOLUCIÓN"

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 09 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los 29 días del mes de junio del año 2010.

_____ _____

Firma del Autor **Firma del Autor**

Miriam Yilieny González **Luis Manuel Refeca González**

Resumen

La investigación surge en el marco de trabajo del Proyecto: “Petrofísica”, convenio de colaboración entre el Centro de Investigaciones del Petróleo y el Polo PetroSoft de la Universidad de las Ciencias Informáticas (UCI). En ésta se realizará la implementación de un subsistema para la importación y exportación de registros de pozo. El subsistema visualiza los datos asociados a los archivos, además de guardar dichos archivos modificados. La aplicación será compatible con los formatos de archivos de pozo más comunes y completos para este fin, tales como: LAS (Log ASCII Standard), y ASCII.

Índice de Figuras

Índice de Tabla

Introducción

El petróleo es la fuente de energía más importante de la sociedad actual. Este hidrocarburo es un recurso natural no renovable que aporta el mayor porcentaje del total de la energía que se consume en el mundo, ya sea proporcionando fuerza, luz o calor y su importancia no ha dejado de crecer desde sus primeras aplicaciones industriales a mediados del siglo XIX.

La búsqueda de petróleo en el subsuelo cubano se inició hace más de 125 años, remontándose al año 1881, con el descubrimiento de un campo de nafta cerca de Motembo, en la parte central del país, en el municipio de Coralillo, provincia de Villa Clara. Sin embargo, fue después de 1960 que se inició un programa sistemático y detallado de exploración, con investigaciones y estudios geológicos y geofísicos, y la perforación de pozos profundos de carácter estratigráfico y de exploración. De esta forma, se han delimitado dos grandes cuencas sedimentarias: la Cuenca Norte y la Cuenca Sur.

En Cuba se han perforado más de 2500 pozos de petróleo. Durante la perforación se genera abundante información, almacenada en diferentes registros de pozo, uno de ellos es el registro eléctrico que ayuda a conocer los tipos de formación y las características físicas de las rocas, tales como densidad, porosidad, contenidos de agua, de petróleo y de gas natural.

La Schlumberger más conocida mundialmente como “la gigante furtiva del petróleo”, es una empresa capitalista y propietaria de algunos de los software utilizados por el Centro de Investigaciones del Petróleo (CEINPET) en las áreas de investigación.

CEINPET es una empresa cubana avalada por una alta calificación y que acumula una gran experiencia de trabajo en el sector del petróleo. CEINPET se encarga de dar respuesta de forma integral a toda la actividad petrolera, desde la Exploración hasta la Refinación. Una de las áreas de investigación de CEINPET es la petrofísica. La misma se encarga del estudio de las propiedades físicas de las rocas y, mediante el análisis petrofísico de los núcleos y registros de pozos, se obtiene una evaluación que describe a cada pozo analizado.

Para llevar a cabo trabajos en dicha área, la empresa cuenta con un conjunto de software propietario que se encargan de interpretar y graficar los registros de pozos. Estos software restringen su buen

funcionamiento debido al alto costo de las licencias para el centro, lo que provoca, unido a la falta de acceso a las actualizaciones, que el software pueda ser usado solo en una computadora a la vez. Esto implica que se acumule el trabajo tornándose engorroso, y que parte de sus investigadores se vean excluidos de las utilidades que brindan los mismos.

Por tales motivos se le ha dado la tarea a la Universidad de las Ciencias Informáticas (UCI) de desarrollar el software Sistema de Análisis de Registro de Pozos Petroleros (ANPER), el cual permitirá analizar y evaluar la información de los pozos en los procesos de perforación y producción, además de graficar todas las curvas presentes en los registros de pozos pero, para llevar a cabo la visualización de las curvas, es necesario realizar la carga y lectura de estos archivos además de almacenar cualquier cambio realizado.

Partiendo de la situación expuesta con anterioridad se ha definido como **problema a resolver** *¿Cómo obtener la información de los registros de pozo para el Sistema de Análisis de Registro de Pozos?*

La investigación estará centralizada fundamentalmente en *el proceso de análisis a los registros de pozo*, lo cual constituye el **objeto de estudio.**

Además, la investigación estará enmarcada en *la importación y exportación de registros de pozos*, representando el **campo de acción.**

En vista a resolver el problema descrito se hace necesario *desarrollar un subsistema de importación y exportación de registro de pozo para el Sistema de Análisis de Registros de Pozo*, lo cual representa el **objetivo general** de la investigación.

Para darle cumplimiento al objetivo general se han trazado las siguientes **tareas de investigación**:

- ✓ Caracterizar los registros de pozo.
- ✓ Caracterizar la estructura que poseen los archivos de almacenamiento de la información de los registros de pozo.
- ✓ Seleccionar la tecnología a utilizar.

- ✓ Analizar la información obtenida para determinar el método o algoritmo más factible de acuerdo con necesidades del sistema.
- ✓ Desarrollar la solución propuesta en la investigación.

El desarrollo exitoso de las tareas expuestas anteriormente contribuirá al cumplimiento de la **idea a defender** que propone que, *al implementar un subsistema de importación y exportación de registros de pozos, permitirá obtener información de estos para el Sistema de Análisis de Registros de Pozos.*

Para el desarrollo de la investigación se hizo necesario aplicar algunos métodos científicos dentro de los que se incluyen los **métodos teóricos y empíricos**, que tienen su sustento en la concepción materialista-dialéctica y facilitan la recopilación de la información necesaria para la realización de un subsistema de importación y exportación de registro de pozo.

De los teóricos se utilizaron el histórico-lógico, analítico-sintético e hipotético-deductivo y de los empíricos, la entrevista.

Histórico-Lógico: El método facilitó la realización de la primera parte de la investigación, permitiendo hacer un análisis a nivel nacional e internacional de las empresas que utilizan software para el análisis del proceso petrofísico en pozos petroleros y las características de los registros de pozo, lo cual permite conocer el estado actual y la evolución del fenómeno.

Analítico-Sintético: Se define con el objetivo de analizar las teorías, documentos, etc., permitiendo obtener, resumir y describir los elementos más importantes relacionados con los procesos de importación y exportación de registros de pozo.

Hipotético-Deductivo: Este método permitió, a partir del problema, plantear objetivos y fundamentar la idea a defender verificando elementos que se puedan inferir a través de teorías para establecer conclusiones previas.

Entrevista: Mediante este método, se le realizó la encuesta a la Dra. Olga Castro Castiñeira del Centro de Investigaciones del Petróleo de Cuba, con el fin de acumular datos, que dan una visión del estado actual del fenómeno para fundamentar la problemática.

Como **posibles resultados** de la investigación se espera obtener la implementación de un subsistema de importación y exportación de registros de pozo.

El presente trabajo se encuentra estructurado de la siguiente forma:

El Capítulo 1. Fundamentación teórica y estado del arte: En este capítulo se hará una descripción de los métodos utilizados, se abordarán conceptos referentes al dominio del problema y se realizará un estudio del estado del arte.

El Capítulo 2. Tendencias y tecnologías actuales a desarrollar: En este capítulo se realizará un estudio de los lenguajes de programación, metodologías y herramientas que darán soporte al desarrollo del subsistema.

El Capítulo 3. Descripción y análisis de la solución propuesta: En este capítulo se exponen los principales elementos que sustentan la construcción de la propuesta de solución.

El Capítulo 4. Validación de la solución propuesta: En este capítulo se lleva a cabo la validación de la solución propuesta, mediante las pruebas de caja negra.

Capítulo 1: "Fundamentación teórica y estado del arte"

1.1. Introducción

Este capítulo ofrece una panorámica y un bosquejo histórico sobre el objeto de estudio de la investigación y contempla los conceptos asociados a la problemática. Se conceptualizan términos como registro de pozo y otros tantos importantes para un mejor entendimiento del problema. Se describen herramientas que en la actualidad son utilizadas en la importación y exportación de registro de pozos.

1.2. Conceptos asociados al dominio del problema

1.2.1- Definición de subsistema

Un subsistema es un paquete de clases, asociaciones, operaciones, sucesos y restricciones interrelacionadas, y que tienen una interfaz razonablemente bien definida. Normalmente, un subsistema se identifica por los servicios que proporciona. Un servicio es un grupo de funciones relacionadas que comparten algún propósito común, tal como el procesamiento de entrada-salida, dibujar imágenes o efectuar cálculos aritméticos. Un subsistema define una forma coherente de examinar un aspecto del problema. [3]

Por tanto, los subsistemas son los principales componentes de un sistema.

1.2.2- Petrofísica

La petrofísica se ocupa del estudio de las propiedades físicas de las rocas como reservorios de hidrocarburos a través de análisis de laboratorio en núcleos y de registros de pozo, es decir, es el estudio de las propiedades físicas y químicas que describen la incidencia y el comportamiento de las rocas, los sólidos y los fluidos. [1]

Por tanto, la petrofísica es la encargada del estudio de todas las propiedades físicas y químicas de las rocas y describe el comportamiento de los fluidos presente en las mismas.

1.2.3- Log ASCII Standard (LAS)

Es un formato de archivo estándar común en el petróleo y el gas de la industria para almacenar la información de los registros de pozo.

1.3. Objeto de Estudio

El objeto de estudio es consecuencia del planteamiento del problema, delimita aquella parte de la realidad que interesa estudiar. La precisión del investigador, en este sentido, se demuestra en la redacción minuciosa y cuidada, con la cual formula el objeto de estudio.

1.3.1- Descripción General.

En nuestro país el Centro de Investigaciones del Petróleo (CEINPET) es el encargado de toda la actividad petrolera desde la Exploración hasta la Refinación. Una de las actividades fundamentales que allí se desarrolla es la petrofísica que no es más que el estudio de las propiedades físicas de las rocas, como los reservorios de hidrocarburos. Los reservorios son conocidos como roca almacén o roca colectora ya que en esta roca es donde está almacenado el petróleo. Todo el proceso de análisis petrofísico se realiza a través de estudios en laboratorios de los núcleos y registros de pozo.

La información del estudio de las condiciones de los pozos de petróleo a diferentes profundidades es guardada en los registros de pozo. Al inicio todo el trabajo fue rústico, se utilizaban cables y sondas. Más tarde con la ayuda de la electrónica, la automatización y las unidades computacionales, se logra una mejor toma de los parámetros. Con el pasar de los años, el aumento de las tecnologías, las necesidades técnicas y económicas; el estudio de los pozos adquirió un fuerte desarrollo. En la actualidad el trabajo de forma manual con los registros de pozo es casi imposible.

En términos generales los registros de pozo tienen los siguientes usos:

- Identificación de la formación y las características físicas de la roca, como son la porosidad, la permeabilidad, los fluidos, etc.
- Determinación del tipo y geometría del flujo de fluidos presentes.
- Evaluación de la perforación del pozo, etc.

Los registros de pozo más utilizados en la industria del petróleo actualmente son:

- **Registro del Potencial Espontáneo (SP)**: Este registro identifica las formaciones permeables y porosas; además de obtener indicaciones del volumen de arcilla en una zona dada.

- **Registro de Rayos Gamma (GR)**: Este registro mide la radioactividad, permitiendo a su vez determinar qué tipos de rocas están presentes en el pozo. Define capas arcillosas y formaciones de lutitas que son las más radioactivas.

- **Registro Sónico:** Se utiliza para obtener la porosidad de las rocas.

- **Registro de Densidad:** Determina la porosidad midiendo la densidad de las rocas.

- **Registro de Neutrón:** Determina la porosidad de la formación midiendo la cantidad de átomos de hidrógeno (neutrones) existentes en los poros.

- **Registro de Inducción o de Resistividad:** Determinan los tipos de fluidos presentes en las rocas además de medir la resistividad.

Los datos digitales representados en estos registros se representan en dos formatos principales: Código Estándar Americano para el Intercambio de Información (ASCII, por sus siglas en inglés) y Binario.

El formato de datos binarios Log Information Standard (LIS) fue producido a partir de los sistemas de adquisición de Schlumberger. Era el formato convencional de datos dentro de la industria del perfilaje hasta que fue superado por el Digital Log Interchange Standard (DLIS). El formato DLIS es un estándar sintáctico para sísmica, perforación y perfilaje de pozos. Asegura la rastreabilidad requerida por la industria de Exploración y Producción, al especificar el equipamiento, las herramientas, los procesos y los datos.

La investigación está centralizada en el formato ASCII, el archivo Log ASCII Standard (LAS), comprende archivo de datos individuales escritos en ASCII, que están presentes en el encabezado del registro de

pozo y las curvas ópticas en forma digital. Este formato es ampliamente aceptado por parte de las compañías de Exploración y Producción para el envió rápido de registros de pozos en el mundo entero. Debido a su estructura de archivo ASCII puede ser escrito y leído por cualquier sistema operativo de computadora. Además, puede ser abierto con cualquier editor de texto y extraer información sobre el pozo en forma visual, aunque no pueden ser leídos directamente por ningún programa convencional que realice procesos matemáticos o de graficación. Los archivos Log ASCII Standard (LAS) están estructurados por secciones, las que se explican a continuación:

~V - Version: Datos de la versión del archivo.

~W - Well: Datos de identificación del pozo.

~C - Curve: Identificación de los diferentes perfiles registrados.

~P - Parameter: Parámetros y Constantes.

~O - Other: Otra información de importancia.

~A - ASCII: Contiene los datos de las lecturas de los registros de cada intervalo.

```
~VERSION INFORMATION
VERS.            2.0    :CWLS Log ASCII Standard - VERSION 2.0
WRAP.             NO    :One Line per depth step
DLM.           SPACE    :Column Data Section Delimiter Character
PROD.   Schlumberger    :LAS Producer
PROG.   DLIS to ASCII 14C0-302                          :LAS Program name and version
CREA.     2007/10/15 12:06                              :LAS Creation date {YYYY/MM/DD hh:mm}
DLIS_CREA.  2007-Oct-15 11:29                           :DLIS Creation date and time {YYYY-MMM-DD hh:mm}
SOURCE.   Canasi-100_(OH)_FMI_NGS_HRLA_TLD_15October2007.DLIS   :DLIS File Name
FILE-ID.    FMI_NGS_HRLA_TLD_048PUP                     :File Identification Number
#---------- ------------------------------------------
~WELL INFORMATION
#MNEM.UNIT       DATA              DESCRIPTION
#---------- ------------------  ------------------------------
STRT .M        1613.5           :START DEPTH
STOP .M        3275.5           :STOP DEPTH
STEP .M         0.5             :STEP
NULL .          -999.25         :NULL VALUE
COMP .          SHERRITT INT. OIL & GAS                 :COMPANY
WELL .          POZO                                    :WELL
#---------- ------------------------------------------------------------------------
~PARAMETER INFORMATION
#MNEM.UNIT     VALUE                    DESCRIPTION
#---------- ------------------  ------------------------------
BSAL  .PPM      7900.00000              :Borehole Salinity
MRT   .DEGF      139.99992              :Maximum Recorded Temperature
RMB   .OHMM        0.10791              :Resistivity of Mud - BHT
#---------- ------------------------------------------------------------------------
~CURVE INFORMATION
#MNEM.UNIT    API CODE                                  DESCRIPTION
#---------- ---------                                  ------------------------------
DEPT .M                                                :DEPTH (BOREHOLE) {F10.1}
DEVI .DEG                                              :Hole Deviation {F13.4}
HAZI .DEG                                              :Hole Azimuth {F13.4}
CGR  .GAPI                                             :Computed Gamma Ray {F13.2}
POTA .                                                 :Potassium {F13.4}
SGR  .GAPI                                             :Gamma Ray {F13.2}
THOR .PPM                                              :Thorium {F13.4}
URAN .PPM                                              :Uranium {F13.4}
HTEM .DEGC                                             :HTC Temperature {F13.2}
GR   .GAPI                                             :Gamma-Ray {F13.2}
#---------- ------------------------------------------------------------------------
#
#
#      DEPT          DEVI          HAZI          CGR           POTA          SGR           THOR          URAN
#
~Ascii | Curve
     1613.5       63.6953       160.0896       83.65        0.0429        41.67        3.9336       -5.5657
     1614.0       63.6953       160.0896       83.65        0.0429        41.67        3.9336       -5.5657
     1614.5       63.6953       160.0896       83.65        0.0429        41.67        3.9336       -5.5657
     1615.0       63.6953       160.0896       83.65        0.0429        41.67        3.9336       -5.5657
     1615.5       63.6953       160.0896       83.65        0.0429        41.67        3.9336       -5.5657
```

Figura 1. Archivo Log ASCII Standard (LAS).[1]

[1] Versión 2.0 del archivo Log Acsii Standard (LAS), muestra claramente cada una de las secciones que lo conforman.

La información que contienen los registros de pozos, en cualquiera de los formatos anteriores, es interpretada mediante productos de software especializados que permiten la visualización de las curvas que representan las mediciones contenidas en los registros y el cálculo de las propiedades físicas. Esta interpretación se realiza con el fin de establecer la relación que existe entre las mediciones de los campos y las propiedades físicas de las rocas. Las mediciones resistividad, neutrones y densidad posibilitan cuantificar las propiedades físicas permeabilidad, saturaciones y porosidad efectiva, que permiten caracterizar un depósito de petróleo o gas. La interpretación se realiza a través de relaciones matemáticas que involucran las mediciones de los registros y los análisis en laboratorio de los núcleos.

1.3.2- Descripción actual del dominio del problema.

Actualmente en el Centro de Investigaciones del Petróleo se desarrollan los procesos de importación y exportación de archivos Log ASCII Standard y ASCII a través de tres software, de los que se tiene conocimientos básicos, medios y avanzados en dependencia de la experiencia de trabajo y tiempo de uso de sus ejecutivos. Como CEINPET no cuenta con un software para el análisis petrofísico que sea netamente cubano, compra a empresas reconocidas internacionalmente estos software en vista de darle solución a la actividad petrofísica que aquí se desarrolla y buscando alternativas factibles que propicien un trabajo más hacedero, ya que ninguno de estos sistemas informáticos están diseñados para satisfacer necesidades específicas de los trabajadores de la empresa.

1.3.3- Situación Problemática.

CEINPET está avalado por su alta calificación, es el encargado de dar respuesta a todas las actividades petroleras de Cuba, dentro de la misma se encuentra el área análisis petrofísico, que se encarga del estudio de las propiedades físicas de las rocas a través del análisis de los registros de pozos.

Este centro está obligado a utilizar software propietarios con los inconvenientes que los mismos poseen, la imposibilidad de copia y redistribución, el coste de las aplicaciones es mayor y el pago de las licencias se vuelve insostenible, el soporte de la aplicación es exclusivo del propietario, entre otros.

Por tal motivo es que el equipo de Proyecto Petrofísica del Polo Productivo Petrosoft de la facultad 9 de la UCI se encuentra desarrollando el sistema ANPER que aún no cuenta con el subsistema de importación y exportación de registros de pozo.

1.3.4- Análisis de soluciones de sistemas existentes.

PETROLUKE

PETROLUKE es capaz de cargar archivos en formatos LAS y ASCII de forma eficiente y correcta, interpretarlos, guardarlos en archivos MAT y luego abrirlo una vez más para continuar la interpretación, lo cual constituye una ventaja para el intérprete y facilita este proceso. Este software cuenta con diferentes módulos, como el Editor de Ecuaciones, que permite de forma versátil y amigable, la mayor parte de los cálculos necesarios para realizar una interpretación petrofísica avanzada. Por otra parte, pese a no ser aún capaz de calcular resistividades del agua de formación por medio del Hingle o el Pickett Plot, el módulo de Crossplots demostró ser útil para discriminar litología y la posible presencia o ausencia de Gas.

RESys

RESys es un sistema de software integrado para la evaluación de yacimientos petrolíferos, basado en una base de datos integrada y usando técnicas de diseño con objetos, RESys es un sistema modular y extensible que consiste de varios módulos para análisis y evaluación de registros de pozos, datos de núcleo, datos de producción y más. El diseño modular e integrado permite que el sistema pueda ser extendido, para agregar módulos adicionales sin la necesidad de reescribir o rediseñar del resto del sistema. Adicionalmente, debido a la modularidad y la base de datos integrada, pueden ser optimizados módulos individuales.

RESys es diseñado como un sistema autosuficiente, pero integrado. No hay necesidad de procedimientos especiales para su instalación y no usa el registro de Windows. Tampoco requiere ningún sistema especial para base de datos, porque todos los módulos contienen los drivers necesarios. Aunque hay un programa de instalación para facilidad, su instalación solamente requiere la creación de un directorio en el disco duro con una copia de los archivos en ese directorio. Se puede correr todos los módulos de RESys de una red, disco local o un “flash disk” o “pen drive.”

RELogAnalysis y RELogView representan el módulo más importante para evaluación y visualización de registros de pozos. Ambos programas comparten una base de código fuente e interfaces al usuario en común y este manual explica los dos. Esencialmente RELogView permite solamente la visualización de registros, mientras RELogAnalysis adicionalmente agrega la habilidad de editar curvas y hacer cálculos y evaluaciones petrofísicas. Si RECores está usado para descripciones sedimento lógicas, las descripciones están disponibles para visualización en RELogView y RELogAnalysis. Adicionalmente RELogAnalysis puede crear conjuntos de puntos para uso en el módulo de mapeo, REMaps. RELogAnalysis también incluye una opción para crear un proyecto geo-estadístico compatible con REGeostat.

Interactive Petrophysics

Interactive Petrophysics es una la aplicación de software para el análisis de la propiedad y depósito de petróleo. De los flujos de trabajo sencillos tales como la calidad del control de los datos de registro para realizar tareas complejas de varias zonas y varios análisis, Interactive Petrophysics es ideal tanto para petrofísicos y geólogos.

Interactive Petrophysics usa los modelos deterministas y probabilísticos para calcular la porosidad, la saturación de agua, los volúmenes de sales y otras propiedades en las zonas definidas por el usuario. Los usuarios pueden elegir los parámetros y criterios de valoración como crossplot directamente en las parcelas. Como parámetros se seleccionan de las parcelas de registro, crossplots interactivos e histogramas, los resultados de su análisis actualizado al instante. Interactive Petrophysics posibilita la interpretación en tiempo real, permitiendo la conectividad para leer datos de una red de Transferencia de Información del Pozo en el Lenguaje de Marcado Estándar (con sus siglas en ingles WITSML). Y con las pantallas 3D y utilidad de Generador de informes, la presentación profesional de sus resultados.

Luego de la realización del análisis de los anteriores software, se concluye que muchos podrían en cierta medida satisfacer y resolver la problemática actual, sin embargo son software de empresas privadas, y esto constituye una desventaja pues se dificulta e imposibilita el acceso a cada uno de ellos, a lo cual también se suma el elevado precio de los mismos y de sus actualizaciones. Razón por la cual se propone el desarrollo del Subsistema de Importación y Exportación de Registros de Pozos para el Sistema de Análisis de Registros de Pozo propiamente nacional, que permita la lectura y almacenamiento de la información de los registros de pozo con alta calidad en los procesos de perforación y producción y de

esta manera reducir las importaciones de software propietarios y contrarrestar las desventajas que implica la utilización de este tipo de software.

1.4. Conclusiones Parciales

En este capítulo se arriban a las siguientes conclusiones:

Con la revisión de la información se dan argumentos que facilitan la realización del inicio de la investigación, permitiendo hacer un estudio a nivel nacional e internacional de las empresas que utilizan software para el análisis del proceso petrofísico de pozos petroleros y las características de los registros de pozo, lo cual permite conocer el estado actual y la evolución de la problemática existente, demostrándose la necesidad de implementar un subsistema de importación y exportación de registros de pozos. A esto se suma que para lograr el óptimo desarrollo del subsistema es necesario hacer una selección adecuada de la metodología de desarrollo, el lenguaje de programación y las herramientas a utilizar.

Capítulo #2: "Tendencias y tecnologías actuales a desarrollar".

2.1. Introducción

Las tecnologías están presentes en todos los procesos de la sociedad y de la economía del ser humano. Su principal propósito es transformar el entorno, para adaptarlo mejor a las necesidades y deseos del hombre. Las tecnologías de la Información y Comunicación (TIC) son aquellas herramientas computacionales e informáticas que procesan, almacenan, sintetizan, recuperan y presentan información representada de la más variada forma.

En este capítulo se realiza una monografía de las principales herramientas, metodologías de desarrollo de software y lenguaje de programación, definido con anterioridad por el arquitecto del proyecto, donde se identifican y se describen las ventajas que ofrecen para incorporarlas a la solución del subsistema de importación y exportación de registros de pozo.

2.2. Paradigmas de la Programación.

Un paradigma de programación es un estilo fundamental de programación definido por la forma de dar soluciones a problemas. Proporciona y determina la visión que el programador tiene acerca de la ejecución de un programa. En conclusión, facilita expresar el concepto de computación de diversas maneras.

La programación orientada a objeto se ha convertido en un nuevo paradigma junto con la programación imperativa, la funcional, la lógica, entre otras. Cada uno de estos está definido por características específicas como son:

2.2.1. Paradigma Imperativo

Programa: serie de instrucciones (cálculos, entradas, salidas)

La orientación es hacia los estados.

Elementos de programación: abstracción procedimental, asignación, ciclos, condicionales.

Lenguajes: Cobol, Fortran, Pascal, C, C++

2.2.2. Paradigma Lógico

Programa: colección de declaraciones lógicas que especifican las características que debe tener la solución buscada.

Un programa es declarativo: no importa el cómo sino el qué.

La orientación es hacia las pruebas formales.

Elementos de programación: cláusulas de Horn, unificación, retroceso.

Lenguajes: Prolog, SOUL.

2.2.3. Paradigma Funcional

Programa: colección de funciones que se combinan mediante composición de forma compleja para construir nuevas funciones.

Un programa es declarativo: no importa el cómo sino el qué.

La orientación es hacia la evaluación.

Elementos de programación: composición, orden superior, currificación, recursivida.

Lenguajes:

- *Impuros y estrictos*: Lisp, Scheme, ML
- *Puros y perezosos:* Miranda1, Gofer, Haskell 98.

2.2.4. Paradigma Orientado a Objetos

Programa: colección de objetos que interactúan intercambiando mensajes que transforman estados.

La orientación es, obviamente, hacia los objetos.

Elementos de programación: modelado de objetos, clases, herencia, encapsulamiento.

Lenguajes: Smalltalk, C++, Java, C#, Eiffel, PHP.

La orientación a objeto (OO[2]) ha tomado por asalto y en forma legítima al mundo del software. Como medio para la generación de programas, tiene varias ventajas. Fomenta una metodología basada en componentes para el desarrollo de software, de manera que primero se genera un sistema mediante un conjunto de objetos, luego podrá ampliar el sistema agregándole funcionalidad a los componentes que ya había generado o agregándole nuevos componentes, y finalmente podrá volver a utilizar los objetos que generó para el sistema cuando cree uno nuevo, con lo cual reducirá sustancialmente el tiempo de desarrollo de un sistema. [43]

El paradigma orientado a objeto depende de ciertos principios fundamentales:

- ABSTRACCIÓN: Se centra en la vista externa de un objeto, de un modo que sirva para separar el comportamiento esencial de un objeto de su implementación. Definir una abstracción significa describir una entidad del mundo real, no importa lo compleja que pueda ser. [43]

- ENCAPSULACIÓN: Es la propiedad que permite asegurar que el contenido de la información de un objeto está oculta al mundo exterior, es el proceso de ocultar todos los objetos de un objeto que no contribuyen a sus características esenciales. La encapsulación permite la división de un programa en módulos, estos módulos se implementan mediante clases, de forma que una clase representa la encapsulación de una abstracción. [43]

- MODULARIDAD: Es la propiedad que permite subdividir una aplicación en partes más pequeñas (llamadas módulos), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes. La modularización, consiste en dividir un programa en módulos que se puedan compilar por separado, pero que tienen conexiones con otros módulos. [43]

[2] Orientación a objeto

- HERENCIA: Una de las propiedades más importantes de la programación orientada a objetos es la herencia. De hecho, algunos piensan que un programa que no emplea herencia no es un programa orientado a objetos. La herencia es aquella propiedad de la programación orientada a objetos que le permite a una clase, llamada clase derivada, compartir la estructura y el comportamiento de otra clase, llamada clase base. Otra razón para usar herencia es que permite construir una jerarquía entre clases. Las clases que incluyen aquellas cosas que se heredan más comúnmente se encuentran en la parte superior de la jerarquía, justo como sus antepasados están en la parte superior de la jerarquía de su familia genética. [43]

- JERARQUÍA: Es una propiedad que permite una ordenación de las abstracciones. Las dos jerarquías más importantes de un sistema complejo son: estructura de clases (jerarquía «es–un» (is–a): generalización/especialización) y una estructura de objetos (jerarquía «parte–de» (part– of): agregación). [43]

- POLIMORFISMO: Permite la posibilidad de construir varios métodos con el mismo nombre, pero con relación a la clase a la que pertenece cada uno, con comportamientos diferentes. Esta propiedad no suele ser considerada como fundamental en muchos de los diferentes modelos de los objetos existentes, pero, dada su importancia, no tiene sentido considerar un objeto modelo que no soporte esta propiedad. Polimorfismo es la propiedad que indica, literalmente, la posibilidad de que una entidad tome muchas formas. En términos prácticos, el polimorfismo permite a objetos de clases diferentes mediante el mismo elemento de programa y realizar la misma operación de diferentes formas, según sea el objeto que se referencia en ese momento. [43]

La programación orientada a objetos considera a un programa como una colección de agentes ampliamente autónomos, llamados objetos. Tiene características muy definidas como paradigma de programación:

- Cada objeto es responsable de tareas específicas. Es mediante la iteración de los objetos que avanza el cómputo. Por lo tanto, en cierto sentido la programación es, ni más ni menos, la simulación de un universo modelo.

- El comportamiento de cada objeto queda determinado por la clase que pertenece, cada objeto es un ejemplar de alguna clase, donde todos los objetos de la misma clase se comportarán de una forma similar antes de una solicitud similar.
- Un objeto exhibirá su comportamiento ante la invocación de un método como respuesta a un mensaje. La interpretación de un mensaje es decidido por el objeto y puede diferir de una clase de objetos a otra.
- Las clases pueden organizarse en un árbol de herencia jerárquico. Las clases que se encuentran en el nivel más bajo tienen acceso, pueden usar datos y comportamientos asociados con clases más altas del árbol.
- Al reducir la independencia entre los componentes de software, la programación orientada a objetos permite el desarrollo de sistemas de software reutilizables. Tales componentes pueden crearse y probarse como unidades independientes, aisladas de otras partes de una aplicación de software.

Siguiendo las características de estos paradigmas se plantea que la siguiente investigación está dirigida por la programación orientada a objetos. Siendo esta una de las más similares al pensamiento real, definiendo elementos como objetos, clases, mensajes y métodos; y situaciones como son: la herencia entre clases, la asignación de responsabilidades a los objetos y la ocultación de información con la utilización del polimorfismo.

2.3. Lenguaje de programación.

2.3.1. C Sharp.

C# es un nuevo lenguaje propuesto por Microsoft para satisfacer las necesidades actuales y de un futuro cercano. Es una herramienta, como todos los lenguajes de programación, pero adaptada al trabajo actual. [36]

El objetivo de Microsoft ha sido la creación del primer lenguaje orientado a componentes, al estilo de Visual Basic, pero con la flexibilidad y potencia de C++ y sin muchas de sus complejidades. Ha sido diseñado para una plataforma, la plataforma Microsoft .NET, en la que los servicios son ofrecidos en forma de componentes. Cuenta con construcciones sintácticas nativas para la definición, implementación y consumo de propiedades, métodos y eventos, lo cual le diferencia claramente de C++ o Java. [36]

Para construir un componente con no es necesario hacer nada especial aparte de crear una nueva clase. Dicho de otro modo, cualquier clase de objeto es un componente, sin necesidad de crear GUID (Globally Unique Identifier) para interfaces y clases de componentes. [36]

Es un lenguaje completamente orientado a objetos con una gran cantidad de características y mejoras sobre sus predecesores. Al igual que Java, trabaja en un entorno manipulado de memoria, aislando al programador de los engorrosos detalles del hardware. A pesar de esto, el programador tendrá la posibilidad de renunciar a esta característica y lidiar por sí mismo con la manipulación de la memoria [36].

Se dice que su competir más cercano es Java, lenguaje con el que guarda un enorme parecido. En este aspecto, es importante señalar que C# incorpora muchos elementos de los que Java carece como: el rendimiento, el cual es mejor, soporta más tipos primitivos, incluyendo tipos numéricos sin signo, compilación condicional, aplicaciones multi-hilo simplificadas; y otros [36].

2.3.2. Fundamentación de la selección del lenguaje de programación.

Se decide utilizar el lenguaje de programación C#, de la plataforma .NET, para generar los artefactos de tipo software, pues es simple, eficaz, orientado a objetos, permite desarrollar aplicaciones rápidamente y mantiene la expresividad y elegancia de los lenguajes de tipo C. Además presenta características esenciales como:

Sencillez de uso: Elimina muchos elementos añadidos por otros lenguajes y que facilitan su uso y compresión, como por ejemplo ficheros de cabecera, o ficheros fuentes IDL. [36]

Modernidad: Incorpora elementos que se ha demostrado a lo largo del tiempo que son muy útiles para el programador, como tipos decimales o booleanos, así como una instrucción que permita recorrer colecciones con facilidad (instrucción foreach). Estos elementos hay que simularlos en otros lenguajes como C++ o Java. [36]

Orientado a objetos: Como lenguaje de última generación es orientado a objetos. Además, soporta todas las características del paradigma de la programación orientada a objetos, como son la encapsulación, la herencia y el polimorfismo. [36]

Orientado a componentes: Su sintaxis incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular. [36]

Recolección de basura: Todo lenguaje incluido en la plataforma .NET tiene a su disposición el recolector de basura. [36]

Eficiente: Todo el código incluye numerosas restricciones para garantizar su seguridad, no permitiendo el uso de punteros. [36]

Compatible: Para facilitar la migración de programadores de C++ o Java a C#, no sólo se mantiene una sintaxis muy similar a la de los dos anteriores lenguajes, sino que también ofrece la posibilidad de acceder a código nativo escrito como funciones sueltas no orientadas a objetos, tales como las DLLs de la API de Win32.[36]

2.4. Metodología de desarrollo.

2.4.1. Rational Unified Process

RUP[3], una de las metodologías más usadas en la actualidad por las empresas de software, es validada continuamente para el perfeccionamiento de su uso. Está concebida para que desde el inicio del proceso se establezca una definición acertada del proyecto, haciendo innecesarias las reconstrucciones parciales posteriores. Además está dirigido a la programación orientada a objetos que permite obtener sistemas escalables en el tiempo que no necesitarán grandes inversiones de recursos en sus modificaciones posteriores. [36]

[3] Proceso Unificado de Desarrollo.

RUP es un proceso formal: Provee un acercamiento disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de software de alta calidad que satisfaga los requerimientos de los usuarios finales. Utiliza UML como lenguaje de notación, fue desarrollado por Rational Software, y está integrado con toda la suite Rational de herramientas. [36]

Los principales elementos de RUP son:

- Quién (Trabajadores)
- Cómo (Actividades)
- Qué (Artefactos)
- Cuando (Flujo de Actividades)

En RUP se han definido las actividades en grupos lógicos definiéndose nueve flujos de trabajo principales, los seis primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo. [36]

RUP presenta 4 fases por las cuales transita el software, estas son:

- Inicio o Conceptualización.
- Elaboración.
- Construcción.
- Transición.

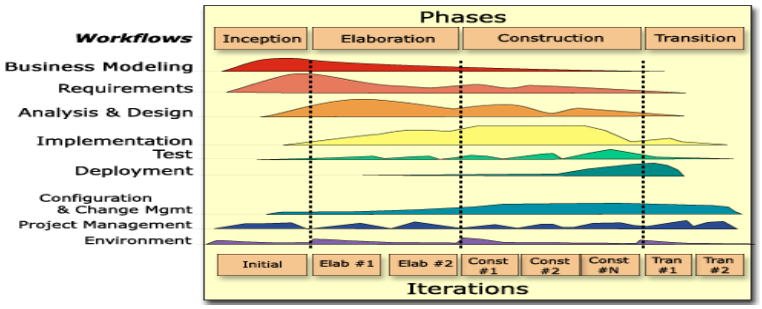


Figura 2. Fases de RUP.[4]

[4] RUP en dos dimensiones, las fases y los flujos de trabajos (Modelamiento del Negocio, Requerimiento, Análisis y Diseño, Implementación, Prueba e Instalación).

Es una metodología muy importante, presenta las siguientes características:

Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan, es una facilidad que el software debe proveer a sus usuarios. A partir de aquí guían el proceso de desarrollo incluyendo el diseño, la implementación y las pruebas del sistema. [36]

Centrado en la arquitectura: La arquitectura involucra los elementos más significativos del sistema. Surge de las necesidades de la empresa y se refleja en los casos de uso. También se ve influida por otros factores como las plataformas de software, los sistemas operativos, protocolos y requerimientos no funcionales. El modelo de arquitectura se representa a través de vistas en las que se incluyen los diagramas de Lenguaje Unificado de Modelado (Unified Modeling Language, UML). [36]

Iterativo e incremental: RUP divide el proceso en cuatro fases, las cuales se desarrollan en iteraciones de las actividades principales básicas de cualquier proceso de desarrollo, hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. Estas iteraciones están controladas y se ejecutan de una forma planificada. [36]

2.4.2. Fundamentación de la metodología de desarrollo a utilizar.

Se selecciona RUP como metodología de desarrollo porque provee un entorno de proceso de desarrollo configurable, basado en estándares, permite tener claro y accesible el proceso de desarrollo que se sigue, además de ser configurado según las necesidades de la organización y del proyecto. Utiliza mejores prácticas probadas en la industria y provee a cada participante con la parte del proceso que le compete directamente, filtrando el resto.

2.5. Lenguaje de Modelación

2.5.1. Lenguaje Unificado de Modelado (UML).

Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema

(modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. [36]

Ofrece diagramas en los cuales modelan el sistema. Los Diagramas de Estructura Estática que se enfatizan en los elementos que deben existir en el sistema modelado; es decir, describen las propiedades estructurales del sistema; Diagramas de Clases para modelar la estructura estática de las clases en el sistema, contiene un conjunto de clases, interfaces y colaboraciones, así como sus colaboraciones; Diagramas de Objetos para modelar la estructura estática de los objetos en el sistema que contienen un conjunto de objetos y sus relaciones; Diagramas de Casos de Uso para modelar los procesos del negocio, contiene un conjunto de casos de uso, actores y sus relaciones. [36]

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Un sistema se modela como una colección de objetos discretos que interactúan para realizar un trabajo que finalmente beneficia a un usuario externo. El lenguaje de modelado pretende unificar la experiencia basada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. Permite además modelar sistemas utilizando técnicas orientadas a objetos (OO) y especifica todas las decisiones de análisis, diseño e implementación, construyéndose así modelos precisos. [36]

UML tiene un vocabulario en el que se identifican:

- Elementos: Abstracciones que constituyen los bloques básicos de construcción.
- Relaciones: Ligan los elementos.
- Diagramas: Es la representación gráfica de un conjunto de elementos y sus relaciones.
- UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real. UML ofrece nueve diagramas en los cuales modelar sistemas.
- UML es una consolidación de muchas de las notaciones y conceptos más usados orientados a objetos.

2.5.2. Fundamentación del Lenguaje de Modelación a utilizar.

Se escogió como lenguaje de modelado UML ("Unified Modeling Language") porque es posible establecer una serie de requerimientos y estructuras necesarias para modelar un sistema de software previo al proceso intensivo de escribir código. UML es un lenguaje que posee más características visuales que programáticas, las que facilitan a integrantes de un equipo participar e intercomunicarse fácilmente, siendo estos integrantes los analistas, diseñadores, especialistas de área y desde luego los programadores. Una de las características más importantes que hacen que se escoja UML como lenguaje de modelado es que está diseñado para uso con software orientado a objetos, y tiene un uso limitado en otro tipo de cuestiones de programación.

2.6. Herramientas a utilizar.

Las herramientas de desarrollo son aquellas aplicaciones que tienen cierta importancia en el desarrollo de un programa. A continuación se describen las características principales de las herramientas Visual Paradigm y el Visual Studio 2008, definido con anterioridad por el arquitecto del proyecto.

2.6.1. Visual Paradigm.

Para el modelado con UML utilizaremos la herramienta CASE Visual Paradigm for UML ya que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. [36]

Entre sus características se encuentra:

- Soporte UML versión 2.1.
- Ingeniería de ida y vuelta.
- Ingeniería inversa, código a modelo y código a diagrama.

- Ingeniería inversa Java, C++, Esquemas XML, XML, .NET exe/dll, CORBA IDL.
- Generación de código modelo a código, diagrama a código.
- Editor de Detalles de Casos de Uso.
- Diagramas EJB: Visualización de sistemas EJB.
- Generación de código y despliegue de EJB´s: Generación de beans para el desarrollo, despliegue de aplicaciones y diagramas de flujo de datos.
- Soporte ORM:
    - Generación de objetos Java desde la base de datos.
    - Generación de bases de datos.
    - Transformación de diagramas de Entidad-Relación en tablas de base de datos Ingeniería inversa de bases de datos.
- Generador de informes para generación de documentación.
- Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML importación y exportación de ficheros XMI.
- Integración con Visio -Dibujo de diagramas UML con plantillas de MS Visio Editor de figuras.

2.6.1.1. Fundamentación de la herramienta CASE a utilizar.

Además de todas estas características también hay que decir que Visual Paradigm for UML es una herramienta fácil de utilizar, es intuitiva para el usuario y multiplataforma. [36]

2.6.2. Visual Studio 2008.

.NET es la nueva tecnología desarrollada por Microsoft con los objetivos principales de mejorar los sistemas operativos y de obtener un entorno diseñado para el desarrollo y ejecución del software en forma de servicios que puedan ser accedidos a través de Internet de forma independiente al lenguaje de programación, sistema operativo y hardware utilizados tanto para desarrollarlos como para publicarlos.

Visual Studio .NET es la herramienta de desarrollo multilenguaje más completa para construir e integrar rápidamente aplicaciones y servicios Web XML (Extensible Markup Language, Lenguaje de Marcas Ampliable). Aumenta de un modo extraordinario la productividad de los desarrolladores y crea nuevas oportunidades de negocio. En su diseño se han integrado a fondo los estándares y protocolos de Internet, como XML y SOAP (Simple Object Access Protocol), por lo que Visual Studio .NET simplifica considerablemente el ciclo de vida del desarrollo de aplicaciones [34].

Visual Studio 2008 tiene un elevado aumento de la productividad al escribir aplicaciones dirigidas a la nueva versión de .NET Framework, esto incluye la ampliación de tipos de proyectos, la reducción de tareas mundanas y los aspectos de equipo orientados a la ingeniería de software [35].

A las mejoras de desempeño, escalabilidad y seguridad con respecto a la versión anterior, se agregan entre otras, las siguientes novedades:

1. **Mejora en las capacidades de Pruebas Unitarias:** son ejecutadas más rápido independientemente de si lo hacen en el entorno IDE o desde la línea de comandos.

2. **Visual Studio Tools for Office (VSTO):** integrado con Visual Studio 2008 es posible desarrollar rápidamente aplicaciones de alta calidad basadas en la interfaz de usuario de Office que personalicen la experiencia del usuario y mejoren su productividad en el uso de Word, Excel, PowerPoint, Outlook, Visio, InfoPath y Project.

3. **LINQ (Language Integrated Query):** nuevo conjunto de herramientas diseñado para reducir la complejidad del acceso a Base de Datos.

4. **Soluciones multiplataforma:** Visual Studio 2008 ahora permite la creación de soluciones multiplataforma adaptadas para funcionar con las diferentes versiones de .Net Framework: 2.0. (Incluido con Visual Studio 2005), 3.0 (incluido en Windows Vista) y 3.5 (incluido con Visual Studio 2008).

2.6.2.1. Fundamentación del IDE a utilizar.

Por la utilización del framework 3.5 .Net y el lenguaje de programación C# el arquitecto del proyecto ha decidido utilizar como IDE[5] de desarrollo el Visual Studio 2008, ya que ofrece la visión de las aplicaciones clientes inteligentes, al permitir a los desarrolladores con avanzadas herramientas de desarrollo y otras características innovadoras, la creación de aplicaciones de manera rápida a través de diversas plataformas. El subsistema se desarrolla con el framework 3.5, ya que se considera estable y brinda la posibilidad al IDE de migrar a Visual Studio 2005.

2.7. Conclusiones parciales

Según la documentación presentada en el capítulo 2, se concluye que para la implementación del subsistema de importación y exportación de registros de pozo se utilice el paradigma orientado a objeto, siguiendo una metodología tradicional como es RUP, así como, las herramientas a utilizar que son el Visual Paradigm para lograr una óptima modelación de la problemática, y el IDE Visual Studio 2008 con el objetivo de realizar un excelente manejo del lenguaje de programación C Sharp. A partir de aquí se identificarán los requerimientos del subsistema, y la descripción de los patrones de diseño a utilizar, por lo tanto, en el próximo capítulo se realizara la descripción y análisis de la solución propuesta.

[5] Entorno de Desarrollo Integrado.

Capítulo #3: "Descripción y análisis de la solución propuesta".

3.1. Introducción

El presente capítulo expone los principales elementos que sustentan la construcción de la propuesta de solución. De esta manera, se definen los requerimientos funcionales y no funcionales del subsistema propuesto y se describe la arquitectura que satisface los mismos.

3.2. Requerimientos.

3.2.1. ¿Qué es un requerimiento?

1. La IEEE Standard Glossary of Software Engineering Terminology define un requerimiento como condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo. [39]

2. Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente. [39]

Por lo tanto, se define como requerimiento a todas las ideas que los clientes, usuarios y miembros del equipo de proyecto tengan acerca de lo que debe hacer el sistema, deben ser analizadas como candidatas a requisitos.

3.2.2. Requerimientos funcionales y no funcionales.

Para modelar el sistema, se identifican sus requisitos, tanto funcionales como no funcionales, y se modelan los funcionales partiendo de los casos de uso del sistema. [39]

Los Requerimientos Funcionales (RF) y los Requerimientos no Funcionales (RNF) explicados en los siguientes epígrafes son los correspondientes con la parte de la implementación del subsistema.

3.2.2.1. Requerimientos funcionales

Los RF son capacidades o condiciones que el sistema debe cumplir [39].

- **RF1:** El subsistema debe permitir que el petrofísico pueda cargar los datos de los ficheros de formato TXT y las versiones 1.2, 2.0 y la 3.0 de los archivos .LAS [38].
  - ✓ **RF1.1:** Visualizar cada una de las sesiones de los archivos [38].
    - **RF1.1.1:** Las sesiones de los archivo .LAS son:
      - Versión: Datos de la versión del archivo.
      - Well: Datos de identificación del pozo.
      - Curve: Identificación de los diferentes perfiles registrados.
      - Parameter: Parámetros y Constantes.
      - Other: Otra información de importancia.
      - ASCII: Contiene los datos de las lecturas de los registros de cada intervalo.
    - **RF1.1.2:** Las sesiones de los archivos .TXT son:
      - La sesión de la información del pozo.
      - La sesión que los datos de las lecturas de los registros de cada intervalo.

- **RF2:** El subsistema debe permitir que el petrofísico realice la conversión de cualquiera de las versiones 1.2 y 2.0 de los ficheros .LAS a la 3.0.
  - ✓ **RF2.1:** El subsistema debe permitir que el petrofísico exporte el archivo convertido. [38]

3.2.2.2. Requerimientos no funcionales.

La determinación de los requisitos no funcionales implica asociar a las facilidades, funcionalidades y las características generales de la aplicación que contribuyan al control necesario para garantizar la confiabilidad de la aplicación, seguridad propuesta, requisitos de la calidad, interfaces con otros sistemas de procesamiento manual o automatizado, ambiente de software y hardware. Para la definición de los requisitos no funcionales se utiliza la clasificación de la Norma ISO-9126 (2000), el modelo de calidad que clasifica los atributos de la calidad del software en seis características, que son además divididas en sub-características. El efecto combinado de las características de calidad de software para el usuario se define

como la calidad en el uso. Las características definidas son aplicables a todo tipo de software. Por lo que los RNF son propiedades o cualidades que el producto debe tener [39].

3.2.2.2.1. Restricciones en el diseño y la implementación.

- ✓ El lenguaje de programación a utilizar en la implementación C Sharp (C#).
- ✓ La principal librería usada del lenguaje es la System.IO
- ✓ El lenguaje a utilizar en la modelación es el UML.
- ✓ Diagramas principalmente usados en el lenguaje UML. El de colaboración (DC) y el de caso de uso del sistema (DCUS).
- ✓ IDE Visual Studio 2008 para C#.
- ✓ Herramienta CASE Visual Paradigm 6.4 para UML.

3.2.2.2.2. Requerimientos de Hadware.

- ✓ Periféricos Teclado y Mouse PS2 o USB.
- ✓ 1 MB de cache L2.
- ✓ 512 MB o superior de memoria RAM.
- ✓ 1 GB de espacio libre en disco.
- ✓ CPU Pentium IV a 1.00 GHz o superior.

3.3. Descripción del subsistema propuesto.

El subsistema propuesto debe cumplir con determinadas funcionalidades. Como son la importación de archivo TXT y las tres versiones existentes del archivo LAS, además de exportar la conversión de las versiones 1.2 y 2.0 a la 3.0 del archivo LAS. Todas las operaciones que brinda el subsistema son realizadas por los ingenieros petrofísicos de CEINPET.

3.3.1. Interfaz de la aplicación.

3.3.2. Los actores son el entorno del sistema.

No todos los actores representan a personas. Pueden ser actores otros sistemas o hardware externo que interactuará con el sistema. Cada actor asume un conjunto coherente de papeles cuando interactúa con el sistema. Un usuario físico puede actuar como uno o varios actores, desempeñando los papeles de esos actores en su interacción con el sistema. Varios usuarios concretos pueden actuar como diferentes ocurrencias del mismo actor.

Los actores se comunican con el sistema mediante el envío y recepción de mensajes hacia y desde el sistema según éste lleva a cabo los casos de uso a medida que se define lo que hacen los actores y lo que hacen los casos de uso, se realiza una clara separación entre las responsabilidades de los actores y las del sistema. Esta separación ayuda a delimitar el alcance del sistema. [41]

3.3.2.1. Descripción de los actores.

Actores	Descripción
Petrofísico	Son las personas encargadas de realizar las operaciones para los procesos de importación y exportación de archivos LAS y TXT.

Table 1. Descripción de los actores

3.3.3. Diagrama de Caso de Uso del Sistema.

Un diagrama de Caso de Uso del Sistema (DCUS) describe lo que hace un sistema desde el punto de vista de un observador externo, debido a esto, un diagrama de este tipo generalmente es de los más sencillos de interpretar en UML, ya que su razón de ser se concentra en un *Qué hace el sistema*. [40]

El diagrama de Caso de Uso del Sistema (DCUS) ayuda al cliente, los usuarios y los desarrolladores a llegar a un acuerdo sobre cómo utilizar el sistema. La mayoría de los sistemas tienen muchos tipos de usuarios. Cada tipo de usuario se representa mediante un actor. Los actores utilizan el sistema al interactuar con los casos de uso. [41]

3.3.3.1. Diagrama de Caso de Uso del Sistema.

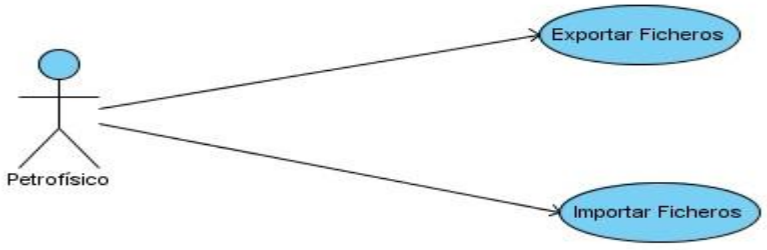


Figura 3. Diagrama de Caso de Uso del Sistema

3.3.4. Los casos de uso especifican el sistema.

Los Casos de Usos (CU) están diseñados para cumplir los deseos del usuario cuando utiliza el sistema. Un caso de uso se define como: *una secuencia de acciones, incluyendo variantes, que el sistema puede llevar a cabo, y que producen un resultado observable de valor para un actor concreto.* [41]

3.3.4.1. Descripción de los casos de uso.

Importar Fichero

Caso de Uso del sistema:	Importar ficheros
Actores:	Petrofísico.
Propósito:	El cliente tiene que ser capaz de importar los ficheros LAS y TXT.
Resumen:	El Caso de Uso se inicia cuando el petrofísico solicita cargar archivos de tipo .LAS o .TXT y finaliza cuando se logra cargar el archivo y mostrar la información.
Precondiciones:	Debe escoger el archivo que desea importar.
Referencias:	RF1,RF1.1,RF1.1.1, RF1.1.2
Prioridad:	Crítico.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El petrofísico escoge del menú Funcionalidades la opción Importar.	**2.** El sistema muestra un submenú que le muestra al petrofísico el tipo de fichero que desea importar: ➢ Si selecciona Archivo_LAS, ver sección “Importar Archivo_LAS” ➢ Si selecciona Archivo_ASCII, ver sección “Importar Archivo_ASCII”.

Sección: "Importar Fichero LAS"	
Acción del actor	**Respuesta del sistema**
1- El petrofísico escoge importar el Archivo_LAS.	**2-** El sistema abre una ventana para que el petrofísico escoja el fichero a importar.
3- El petrofísico señala donde está el fichero y elije el fichero a importar y da al botón "Abrir".	**4-** El sistema abre el fichero, lo carga y le muestra al petrofísico el contenido del mismo.
Prototipo de Interfaz de Usuario	
Ver figuras del anexo I.	
Sección: "Importar Fichero ASCII"	
Acción del actor	**Respuesta del sistema**
1- El petrofísico escoge importar el Archivo_ASCII.	**2-** El sistema abre una ventana para que el petrofísico escoja el fichero a importar.
3- El petrofísico señala donde está el fichero, elije el fichero a importar y selecciona el botón "Abrir".	**4-** El sistema abre el fichero, lo carga y le muestra al petrofísico el contenido del mismo.
Prototipo de Interfaz de Usuario	
Ver figuras del anexo II.	
Flujos alternos	
Sección: "Importar Fichero LAS"	
Acción del actor	**Respuesta del sistema**
	4- El sistema al abrir el fichero reconoce que el formato es incorrecto por lo que emitirá un mensaje "Archivo Log ASCII Standard Incorrecto"
3- El petrofísico selecciona el botón "Cancelar"	**4-** El sistema cierra la ventana, quedando la Forma Principal abierta dándole paso a otras operaciones.

Sección: "Importar Fichero LAS"	
Acción del actor	**Respuesta del sistema**
	4- El sistema al abrir el fichero reconoce que el formato es incorrecto por lo que emitirá un mensaje "Archivo ASCII Incorrecto"
3- El petrofísico selecciona el botón "Cancelar"	**4-** El sistema cierra la ventana, quedando la Forma Principal abierta dándole paso a otras operaciones.

Table 2. Descripción del caso de uso importar fichero

Exportar Fichero

Caso de Uso del sistema:	Exportar fichero.
Actores:	Petrofísico.
Propósito:	El cliente tiene que ser capaz de exportar la conversión de las versiones 1.2 y 2.0 a la 3.0 del fichero .LAS.
Resumen:	El caso de uso comienza cuando el petrofísico necesita exportar la conversión del archivo y finaliza cuando se logra exportar el archivo.
Precondiciones:	Debe escoger el archivo que desea exportar.
Referencias:	RF2.
Prioridad:	Crítico.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El petrofísico escoge del menú Funcionalidades la opción Exportar.	**2.** El sistema abre una ventana para que el petrofísico seleccione la dirección donde quiere guardar el fichero LAS.
3- El petrofísico señala el lugar que desea guardar el fichero LAS y elije el botón "Guardar".	**4-** El sistema guarda en la dirección que el petrofísico escogió para guardar el fichero LAS, emitirá un mensaje "Archivo Log ASCII Standard exportado correctamente" y termina el caso de uso.

Prototipo de Interfaz de Usuario	
Ver figuras del anexo III.	
Flujos alternos	
Sección: "Cancelar Fichero"	
Acción del actor	**Respuesta del sistema**
3- El petrofísico escoge el botón Cancelar.	**4-** El sistema cierra la ventana, quedando la Forma Principal abierta dándole paso a otras operaciones.

Table 3. Descripción del caso de uso exportar fichero

3.4. Diagrama de clase.

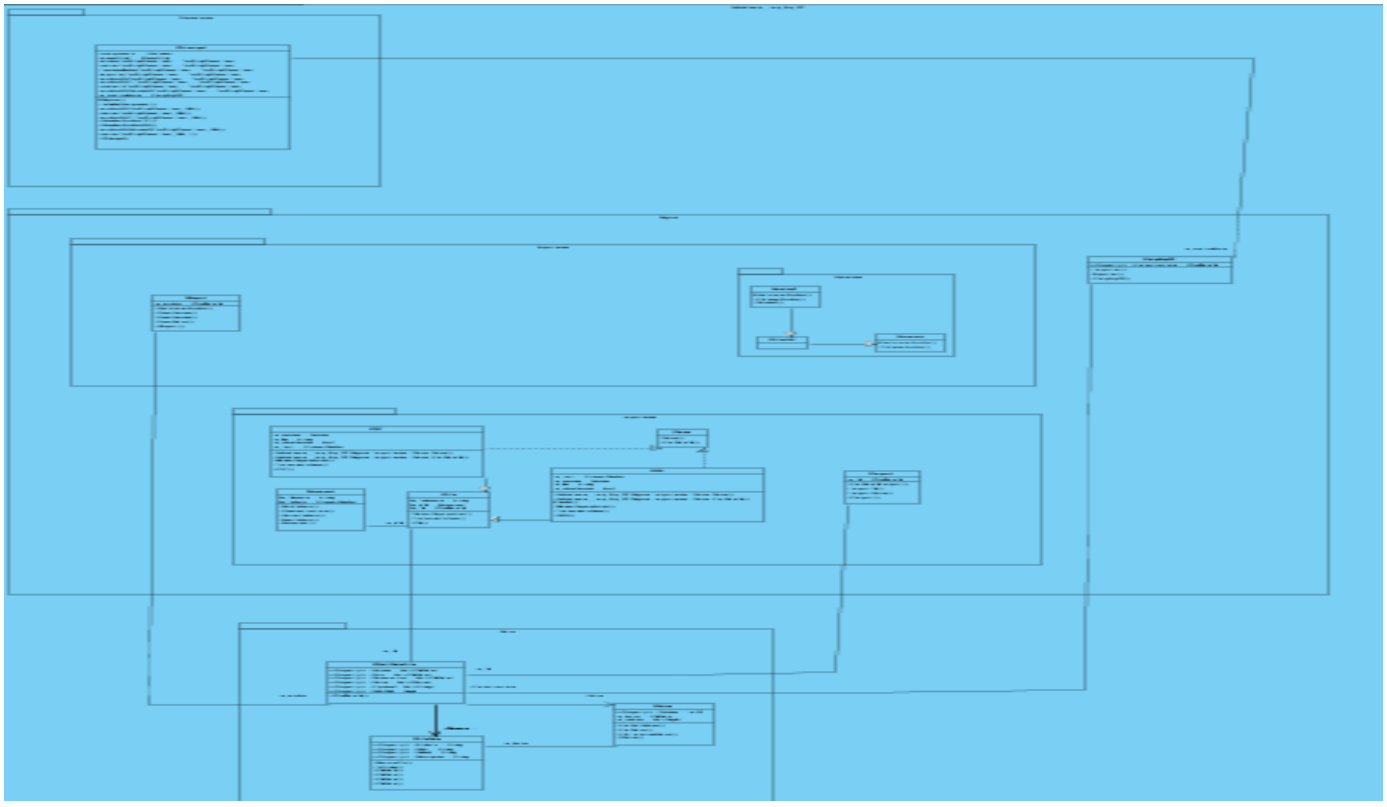


Figura 4. Diagrama de clase del subsistema

3.5. Manejo de archivos.

La lectura y escritura a un archivo en C# son hechas usando un concepto genérico llamado *Stream*. En el ambiente .NET se puede encontrar muchas clases que representan la transferencia de datos, como se muestra en la figura 5.

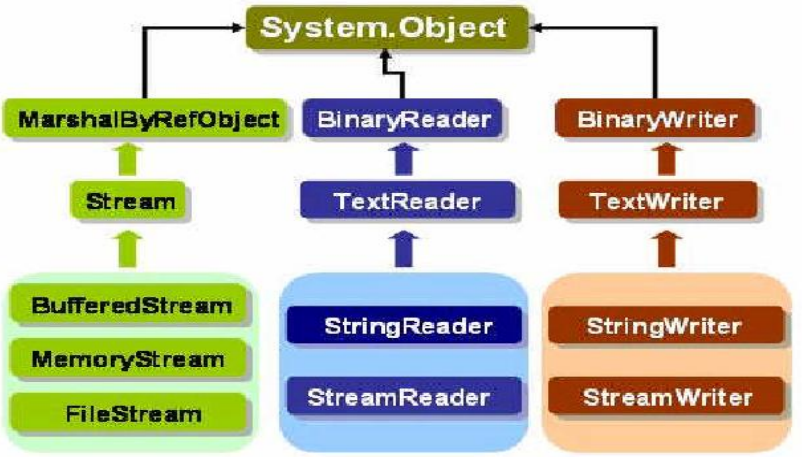


Figura 5. Clases del Framework .NET para el uso de Stream.[6]

Un stream es como se denomina a un objeto utilizado para transferir datos. Estos datos pueden ser transferidos en dos posibles direcciones:

- ✓ Si los datos son transferidos desde una fuente externa al programa, entonces se habla de "leer desde el stream".
- ✓ Si los datos son transferidos desde el programa a alguna fuente externa, entonces se habla de "escribir al stream".

El uso de la clase *stream* para la lectura y escritura de archivo es directa pero lenta. Por esta razón se crea la clase BufferedStream que es más eficiente y puede ser utilizado por cualquier clase de *stream*. Para operaciones de archivo es posible utilizar las clases *FileStream*, *StreamReader* y *StreamWriter*, donde el buffering está ya incluido.

[6] Jerarquía de clases de la librería System.IO (entrada y salida) de la plataforma .NET

- ✓ *FileStream*: Realiza operaciones de lectura y escritura, y crea cualquier tipo de archivos
- ✓ *StreamReader y StreamWriter*: Están diseñadas para lectura y escritura de archivos de texto. Son un nivel más alto que *FileStream.*

Mediante las clases *StreamWriter* y *FileStream* se realiza el proceso de exportación y con la *StreamReader* el proceso de importación de registros de pozo. Para el correcto uso de las mismas, es necesario referenciar el uso de la librería *System.IO,* debido que *System* no contiene los elementos para el manejo de archivos.

3.5.1. Clase StreamReader

La clase *StreamReader* realiza la lectura mediante tres métodos, el *ReadToEnd(), Read()* y *ReadLine().*

ReadLine(): Lee una línea completa de un archivo de texto hasta el cambio de línea más próximo. *StreamReader.ReadLine()* no incluye en el string el carácter de cambio de línea. Se utiliza en la aplicación, para leer línea a línea, dándole a su vez un tratamiento específico a cada línea del archivo, mediante el método *Parse(),* en las clases *CTXT* y *CLAS* respectivamente.

3.5.2. Clases StreamWriter y FileStream

La clase *FileStream* se utiliza para crea archivos de tipo LAS en el directorio que se le especifique, y de la clase *StreamWriter* se utiliza el método WriteLine(), para llenar con los datos necesarios el archivo creado con anterioridad.

3.6. Valoración crítica del diseño propuesto por el analista.

En la fase de análisis y diseño se realizó un excelente trabajo, esto ayudó a que la fase de implementación se realizara sin problemas, solo señalar que se decidió efectuar cambios en algunas clases y atributos asegurando que fuera más flexible, fácil y accesible la implementación utilizando las ventajas que brinda C Sharp.

3.7. Estilo de programación

En cualquier proyecto que intervenga más de una persona se hace necesario seguir unas guías comunes de desarrollo para asegurar la correcta compresión de todo el código, así como su mantenimiento posterior por personas que no han participado en el desarrollo. La disparidad de estilos, utilizada por programadores con poca experiencia, tiene un efecto mortalmente sedante en la salud del proyecto. Inicialmente aparentan ser productivas, ingeniosas y eficaces, pero a largo plazo, cuando llega el inevitable momento de corregir errores que ocurren en entornos o condiciones no previstas, cuando hay que hacer una ampliación del sistema, o cuando un programador nuevo necesita incorporarse al proyecto; únicamente entonces se pueden ver los verdaderos efectos negativos de esas prácticas.

A continuación se explicarán algunos de los estilos de programación más utilizados.

3.7.1. Estilo ALLMAN

El estilo Allman fue definido por Eric Allman[7]. Se trata de crear una nueva línea para las llaves, e identar el código debajo de ellas. La llave de cierre tiene el mismo identado que la de inicio.

Indicadores que plantea el estilo ALLMAN

- **Identación:** Poner las llaves de control en las líneas subsiguientes, así como definir los espacios con un tab para cada bloque**.**

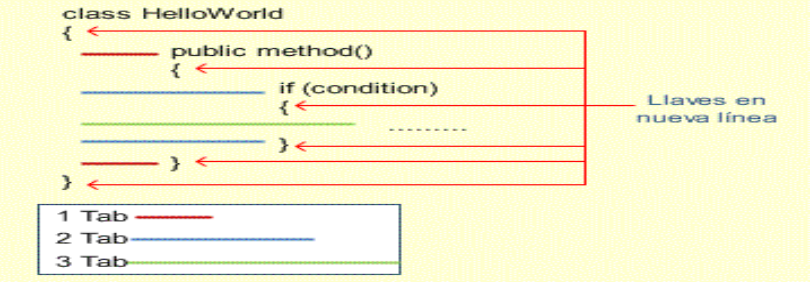


Figura 6. Identación del estilo ALLMAN[8]

[7] **Eric Allman Paul** (nacido el 02 de septiembre 1955) es un programador de computadoras estadounidense que desarrolló sendmail y su precursor delivermail a finales de 1970 y principios de 1980 en la Universidad de Berkeley.

[8] Explicación más detallada de cómo debe de quedar la identación del estilo ALLMAN en el código.

- **Saltos de Línea:** Añadir un salto de línea después del cierre de los paréntesis y cuando termina la sentencia después de un punto y coma.

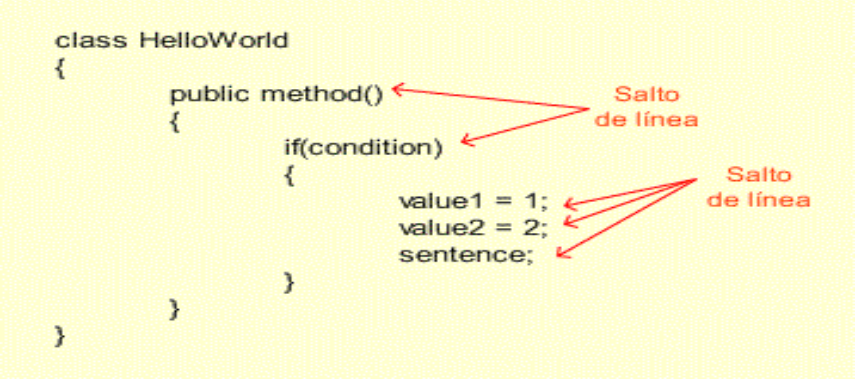


Figura 7. Salto de línea del estilo ALLMAN.[9]

- **Espacios y líneas en blanco:** Usar espacios en blanco para mejorar la legibilidad del código, en ambos lados del operador de símbolos, después de comas y después de las declaraciones. Usar líneas en blanco para separar trozos de código y antes de cada método dentro de la clase.

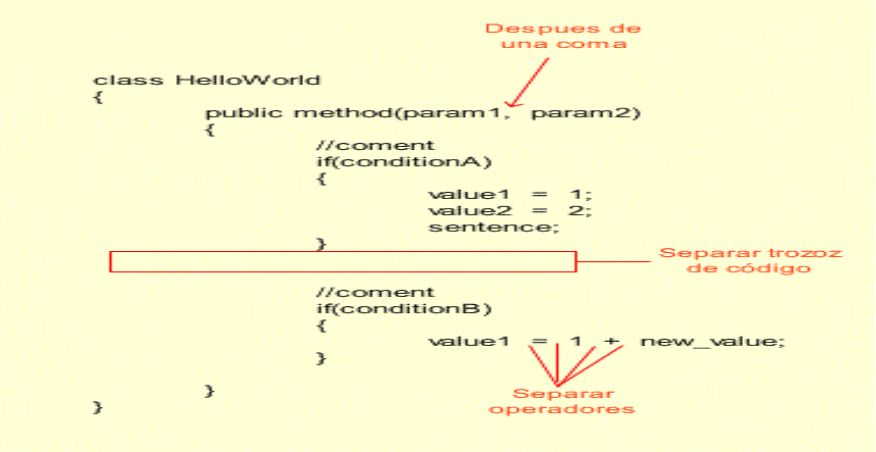


Figura 8. Espacios y líneas en blanco[10]

[9] Explicación más detallada de cómo debe de quedar el salto de línea del estilo ALLMAN en el código

[10] Explicación más detallada de cómo debe dejar un espacio delante y despúes de un operador, etc.

- **Longitud de la línea:** Evite las líneas de más de 80 caracteres, cuando supera se debe cortar bajo los siguientes principios:

  ✓ Salto de línea después de la coma.

  ✓ Salto de línea después de un operador.

  ✓ Alinear la nueva línea con el principio de la expresión en el mismo nivel de la línea anterior.

3.7.2. Estilo K&R

Se trata de abrir la llave en la misma línea de declaración de la orden, indentando los siguientes pasos al mismo nivel que la llave y cerrando la llave en el mismo nivel que la declaración.

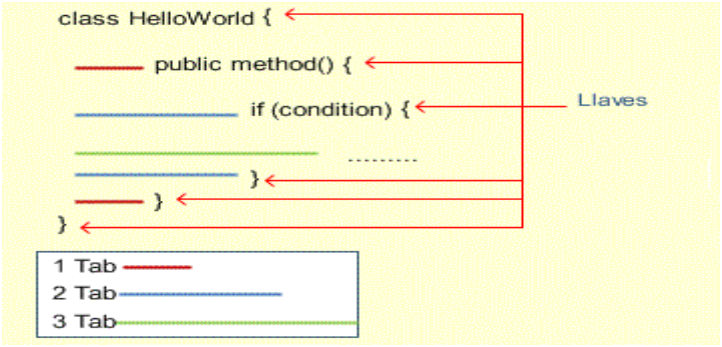


Figura 9. Identación del estilo K&R

- **Saltos de Línea:** Añadir salto de línea cuando termina la sentencia después de un punto y coma. Después de las llaves de cierre un salto de línea si necesitan una línea exclusiva para ellas.

- **Espacios y líneas en blanco:** Similar al estilo ALLMAN el K&R mantiene el uso de espacios en blanco para mejorar la legibilidad del código, en ambos lados del operador de símbolos, después de comas y después de las declaraciones.

3.7.3. Estilo Whitesmiths

El estilo Whitesmiths también llamado estilo Wishart. Coloca las llaves asociadas con las instrucciones de control indentada en la siguiente línea. Este estilo pone la llave que sigue a la

declaración de un bloque. Las instrucciones dentro del bloque son indentados en el mismo nivel que la llave.

Las ventajas obtenidas mediante la implementación de este estilo son las mismas del estilo Allman en que los bloques son claramente separados desde la instrucción de control. Sin embargo, en el estilo Whitesmiths, el bloque está conectado visualmente a su declaración de control. Otra ventaja es que la alineación de las llaves con el bloque entero es vista como un solo conjunto de instrucciones. Además, las llaves hacen hincapié en que el contenido del bloque esté subordinado a la declaración de control.

Una desventaja de este estilo es que las llaves ocupan una línea entera. Otro inconveniente podría ser que la llave de cierre no se alinea con la declaración a la que conceptualmente pertenecen, aunque otros sostienen que el cierre de llaves pertenece a la llave de apertura y no a la declaración.

3.8. Convención de nombres en identificadores

Los identificadores son símbolos léxicos (también llamados símbolos) que nombran entidades del lenguaje, tales como las constantes, los tipos de dato, las etiquetas, las subrutinas, los paquetes y las subrutinas (procedimientos y funciones). Existen algunas restricciones para la selección de los identificadores que dependen del lenguaje a utilizar:

- ✓ No se pueden utilizar identificadores que correspondan con palabras claves o reservadas del lenguaje y restricciones en qué caracteres pueden aparecer en un identificador (Ejem. no está permitido el uso de espacios en blanco, operadores del lenguaje, ni acentos, entre otros).
- ✓ Las selecciones apropiadas para los identificadores se consideran como la piedra angular para un buen estilo. Un nombre pobre para el identificador por ejemplo hace el código difícil de leer y entender.

Por ejemplo, considere el siguiente fragmento de código:

```
if (a < 24 && b < 60 && c < 60)
    return true;
else
    return false;
```

Debido a la elección de los nombres de las variables, la función del código es difícil de leer y entender. Sin embargo, si los nombres de variables se hacen más descriptivo:

```
if (horas < 24 && minutos < 60 && segundos < 60)
    return true;
else
    return false;
```

Así, dado el contexto (es decir, sabiendo de qué trata la clase), debería ser intuitivo, por el puro nombre, qué hace el método, o para qué se emplea la variable. Los nombres tienen que ser claros e intuitivos.

Beneficios:

- ✓ Para proporcionar información adicional (es decir, metadatos) sobre el uso de un identificador.
- ✓ Para ayudar a formalizar las expectativas y fomentar la coherencia dentro de un equipo de desarrollo.
- ✓ Para permitir el uso de reconstrucciones automatizadas o herramientas de búsqueda y reemplazo con un mínimo de posibilidad de error.
- ✓ Para aumentar la claridad en los casos de posible ambigüedad.
- ✓ Para mejorar la estética y apariencia profesional de los trabajos (por ejemplo, se desestiman los nombres excesivamente largos, nombres cómicos o abreviaturas).
- ✓ Para ayudar a evitar "colisiones de nombres" que podrían ocurrir cuando el resultado del trabajo de las diferentes organizaciones se combina (namespace).
- ✓ De proporcionar los datos que se utilizarán en la entrega de proyectos que requieren la presentación de código fuente de los programas y toda la documentación pertinente.

3.9. Identificadores con múltiples palabras

Al momento de nombrar identificadores es común que una sola palabra no sea suficiente para dar una idea al lector de su significado en el código. Por ello en muchos casos es necesario nombrar identificadores como la unión de varias palabras. Para esto, en muchos lenguajes de programación no es permitido escribir el nombre del identificador con espacios en blanco.

El enfoque más común consiste en definir delimitadores para realizar la unión, tales como el guión (-), y el subrayado (_):

(Ejem. dos-palabras, dos_palabras). En algunos lenguajes de programación el guión es reservado para identificar la operación de resta. Por lo cual han surgido distintas notaciones para permitir a los programadores la unión de palabras en la definición de identificadores, a continuación se explican algunas de las notaciones más utilizadas:

3.9.1. Notación Camel

Es la práctica de escribir frases o palabras compuestas en el que las palabras se unen sin espacios y se capitalizan. La notación consiste en escribir los identificadores con la primera letra de cada palabra en mayúsculas y el resto en minúscula:

DosPalabras. Se llama notación “Camel” porque los identificadores recuerdan las jorobas de un camello. Existen dos variantes:

● UpperCamelCase, CamelCase o PascalCase: en esta variante la primera letra también es mayúscula: DosPalabras.

● lowerCamelCase, camelCase o dromedaryCase: la primera letra es minúscula: dosPalabras.

En el lenguaje Java, se usa la notación CamelCase en identificadores para clases, y dromedaryCase para métodos y variables.

3.9.2. Notación C

Durante los años 1960, con la estandarización del código ASCII, los primeros programadores de C y UNIX utilizaron el carácter _ como separador: dos_palabras. Esta notación sigue siendo la más utilizada en C y entornos UNIX. Los defensores de esta notación argumentan que es más fácil de leer porque deja un

espacio entre palabras, al contrario que Camel. Además, en algunos teclados es más rápido de escribir el carácter _ que una mayúscula.

3.9.3. Notación Húngara

La notación Húngara se basa en Camel, añadiendo al principio del identificador una secuencia de letras en minúscula, que indica alguna característica del identificador, como su tipo en el caso de variables. Dentro de la notación Húngara, existen dos tipos; la de sistemas y la de aplicaciones. La primera, hace referencia a la tipo de variables utilizadas en el desarrollo de sistemas, tales como los tipos de datos: unsigned 32-bit interger, double. Mientras que la notación Húngara para aplicaciones, utiliza prefijos descriptivos, que indican al igual que en la de sistema el tipo de dato, pero estas últimas, son utilizadas para aplicaciones que se podrían denominar "independientes" del sistema.

Por ejemplo:

- ✓ fpPrecio: Precio es una variable en punto flotante.
- ✓ rgStudiantes: Estudiantes es una variable del tipo Arreglo.
- ✓ rgfpBalances: nBalances es un arreglo de punto flotante.
- ✓ OnMouseDown utiliza el prefijo on.

3.10. Estándar de codificación.

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. La legibilidad del código fuente repercute directamente en lo bien que un desarrollador comprende un sistema de software. El mejor método para asegurarse de que un equipo de programadores mantenga un código de calidad es establecer un estándar de codificación sobre el que se efectuarán luego revisiones del código de rutinas. Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento y mantenimiento del mismo.

Las convenciones de código son un conjunto de reglas que comprenden cómo se deben utilizar el espaciado y los saltos de línea, nombrar las variables, clases y ficheros, y cómo se deben escribir instrucciones específicas del lenguaje de programación.

Las convenciones ayudan a entender el código, por lo que su utilización es útil a los desarrolladores en las actividades de mantenimiento a programas escritos por otros o por ellos mismos en el pasado. Que un programa funcione o no es en buena medida independiente de que esté bien o mal escrito; sin embargo, uno mal escrito tiene una probabilidad mayor de no funcionar correctamente además de que es casi imposible de depurar o mantener.

Para la implementación del subsistema de importación y exportación de registros de pozos se define un estándar de codificación tomando como estilo de programación ALLMAN, que tiene como objetivos:

- Lograr una estructura uniforme para los bloques de código así como para los diferentes niveles de anidamiento.
- Establecer un modo común para comentar el código de forma tal que sea comprensible con solo leerlo una sola vez.
- Nombrar las clases e instancias de forma estándar para todas las aplicaciones.

Comentarios, separadores, líneas, espacios en blanco y márgenes.

Comentar al inicio de la clase o función especificando el objetivo de la misma así como los parámetros que usa (especificar tipos de datos, y objetivo del parámetro). Además de usar espacios en blanco entre estos operadores para lograr una mayor legibilidad en el código.

Ejemplo: producto = nomproducto;

Evitar comentar cada línea. Cuando el comentario se aplica a un grupo de instrucciones debe estar seguido de una línea en blanco. En caso de que se necesite comentar una sola instrucción se suprime la línea en blanco o se escribe a continuación de la instrucción.

Variables.

Todas las constantes deben ser declaradas con sus letras en mayúsculas y las variables la notación Came *dromedaryCase*. El nombre empleado debe permitir que con solo leerlo se conozca el propósito de la misma.

Clases y Objeto.

Los nombres de las clases declaradas por los programadores deben comenzar con una C indicando que es una clase, en caso de que sea un nombre compuesto se empleará la notación *PascalCase*. Dentro de las clases los atributos utilizan la unión de dos notaciones la notación *C* y la *Camel,* los mismos deben comenzar con la letra minúscula a_ (notación C) , seguido de la variante *dromedaryCase* de la notación Came, al igual que los parámetros con la diferencia que comienza con p_.

Ejemplo de clase: CMiClase

Ejemplo de atributo: a_nombreEstudiante.

Ejemplo de parámetro: p_nombreEstudiante.

Controles

El nombre de los controles deben comenzar con las primeras letras en minúscula, las cuales identificarán el tipo de datos al que se refiere, en caso de que sea un nombre compuesto se empleará la variante *dromedaryCase* de la notación Came.

Ejemplo: btnAceptar.

3.11. Patrones GRASP.

Los patrones son parejas de problema/solución con un nombre, que codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. En el diseño del sistema se pusieron en práctica el uso de algunos patrones de software para la asignación general de responsabilidades [41].

A continuación se detallan algunos de los patrones GRASP que se pusieron en práctica en el diseño de la aplicación.

3.11.1. Patrón experto.

Este patrón es el principio básico de asignación de responsabilidades. El mismo indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo se logra un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento). El mismo mantiene el encapsulamiento, los objetos utilizan su propia información para llevar a cabo sus tareas. Se distribuye el comportamiento entre las clases que contienen la información requerida. Son más fáciles de entender y mantener. [41]

Este patrón se pone de manifiesto en las clases CLAS y CTXT que son expertas ya que contiene toda la información necesaria para crear una estructura de tipo CWellDataFile.

En conclusión la responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Por lo que contiene toda la información necesaria para realizar la labor que tiene encomendada.

3.11.2. Patrón controlador.

El patrón *controlador* sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto es para aumentar la reutilización de código y a la vez tener un mayor control. Se recomienda dividir los eventos del sistema en el mayor número de controladores para poder aumentar la cohesión y disminuir el acoplamiento. [41]

La clase CImpExpRP asigna la responsabilidad de controlar el flujo de eventos del subsistema a las clases CImport y CExport, es decir que la clase CImpExpRP no realiza las actividades de importación y

exportación de archivos, sino que la delega a las clases CImport y CExport según corresponda, y mantiene un modelo de alta cohesión con las mismas.

En conclusión el patrón controlador asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión.

3.11.3. Patrón Creador.

El patrón *creador* asigna la responsabilidad de que una clase B cree un objeto de la clase A solamente cuando:

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

El patrón c*reador* se percibe claramente en las clases CLAS y CTXT ya que son las que poseen los datos necesarios para crear una estructura de tipo CWellDataFile.

En conclusión el creador posee como ventaja el bajo acoplamiento, lo cual admite facilidad de mantenimiento, mayor claridad, encapsulación y reutilización.

3.11.4. Patrón Polimórfico.

Cuando identificamos variaciones en un comportamiento, asignar la clase (interfaz) al comportamiento y utilizar polimorfismo para implementar los comportamientos alternativos, estamos en presencia del patrón polimórfico. [41]

En la aplicación se hizo uso del mismo en las clases CLAS y CTXT las que heredan de la interfaz IParse y que implementa los métodos de la interfaz, dándole un tratamiento específico según la clase.

En conclusión siempre que se tenga que llevar a cabo una responsabilidad que dependa del tipo, se tiene que hacer uso del polimorfismo, cuando las alternativas o comportamientos relacionados varían según el tipo (clase), se asigna responsabilidad al comportamiento, utilizando operaciones polimórficas, a los tipos que varían su comportamiento. Sería establecer el mismo nombre a servicios en diferentes objetos.

3.12. Conclusiones Parciales

En el capítulo se delimitó el estándar de codificación ganando la aplicación en organización y flexibilidad a nivel de código. Lográndose la implementación del subsistema de importación y exportación de registros de pozos, para almacenar la información en el Sistema de Análisis de Registro de Pozos. En el siguiente capítulo se procederá hacer la validación de dicho subsistema, con las pruebas de caja negra, por el método de partición equivalente.

Capítulo #4: "Validación de la solución propuesta".

4.1. Introducción.

Una de las últimas fases del ciclo de vida antes de entregar un programa para su explotación, es la fase de pruebas. Mediante estas se verifican el comportamiento externo del sistema y si satisfacen los requisitos establecidos por los clientes o futuros usuarios del mismo. En este capítulo se lleva a cabo la validación de la solución propuesta, mediante las pruebas de caja negra permitiendo verificar la implementación de los requisitos funcionales del subsistema.

4.2. Pruebas del subsistema.

Cualquier pieza de software completo, desarrollado o adquirido, puede verse como un sistema que debe probarse, ya sea para decidir acerca de su aceptación, para analizar defectos globales o para estudiar aspectos específicos de su comportamiento, tales como seguridad o rendimiento. A este tipo de pruebas donde se estudia el producto completo se les llama *Pruebas de Sistema.* Las pruebas de sistemas usualmente son de caja negra, especialmente si quien prueba no tiene acceso al código de fuente del producto a probar, que es lo más frecuente.

4.2.1. Pruebas de caja negra.

Se denomina **caja negra** a aquel elemento que es estudiado desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno. En otras palabras, de una *caja negra* nos interesará su forma de interactuar con el medio que le rodea (en ocasiones, otros elementos que también podrían ser *cajas negras*) entendiendo **qué es lo que hace**, pero sin dar importancia a **cómo lo hace**. Por tanto, de una *caja negra* deben estar muy bien definidas sus entradas y salidas, es decir, su interfaz; en cambio, no se precisa definir ni conocer los detalles internos de su funcionamiento.

4.2.1.1. Método a utilizar.

El método que se propone para el diseñar los casos de prueba es el de pruebas de caja negra basada en la descripción de los casos de uso, consta de tres pasos fundamentales:

1. **Para cada caso del uso, generar un sistema completo de escenarios:** Para cada caso de uso se crea una matriz denominada Matriz Parcial de Escenarios. Cada camino posible en un caso de uso (todas combinaciones entre el camino principal y los caminos alternativos) se denomina escenario. La matriz cuenta con tres columnas en las cuales se representan el nombre del escenario, el flujo donde comienza y si contiene algún flujo alternativo.

2. **Para cada escenario, identificar por lo menos un caso de prueba y las condiciones que hagan que se lleve a cabo:** Posteriormente se determinan las variables que determinan los escenarios de cada caso de uso y se representa en una matriz llamada Matriz de Casos de Prueba, la misma contiene además el comportamiento esperado del sistema.

3. **Para cada caso de prueba, identificar los valores de los datos con los cuales se hará la prueba:** Como el objetivo final es realizar las pruebas con datos reales, el último paso es identificar los valores para cada variable en los distintos escenarios. En este paso a la matriz de casos de pruebas se le agrega la columna donde se especifica el resultado de la prueba.

Teniendo en cuenta los pasos descritos anteriormente se presentan algunos casos de prueba diseñados para probar las funcionalidades del software.

4.2.1.1.1. Pruebas al caso de uso Importar Archivos

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Importar archivo LAS.	EC 1.1 Importar archivo LAS correcto.	El sistema carga archivos LAS versiones 1.2, 2.0 y 3.0. Visualiza la información.

	EC 1.2 Importar archivo LAS incorrecto.	El sistema carga archivos LAS. Muestra un mensaje de error.
	EC 1.3 Cancelar la Importación del archivo LAS.	El sistema cierra la ventana, quedando la Forma Principal abierta dándole paso a otras operaciones.
SC 2: Importar archivo ASCII.	EC 2.1 Importar archivo ASCII incorrecto.	El sistema carga archivos ASCII. Muestra un mensaje de error.
	EC 2.2 Importar archivo ASCII .	Muestra la información del archivo.
	EC 2.3 Cancelar la Importación del archivo ASCII.	El sistema cierra la ventana, quedando la Forma Principal abierta dándole paso a otras operaciones.

Table 4. Descripción de las secciones y escenarios del Caso de Uso Importar Archivos

ID Sección	Escenario	Respuesta del Sistema	Resultado de la Prueba
SC 1:	Importar archivo LAS correcto.	Muestra la información del archivo.	Prueba satisfactoria.
	Importar archivo LAS incorrecto.	Muestra un mensaje “Archivo Log ASCII Standard Incorrecto”	Prueba satisfactoria.
	Cancelar la Importación del archivo LAS.	El sistema cierra la ventana, quedando la Forma Principal abierta dándole paso a otras operaciones.	Prueba satisfactoria.
SC 2:	Importar archivo ASCII correcto.	Muestra la información del archivo.	Prueba satisfactoria.
	Importar archivo ASCII incorrecto.	Muestra un mensaje “Archivo ASCII Incorrecto”	Prueba satisfactoria.
	Cancelar la Importación del archivo ASCII.	El sistema cierra la ventana, quedando la Forma Principal abierta dándole paso a otras operaciones.	Prueba satisfactoria.

Table 5. Caso de prueba del caso de uso importar archivo

Con los siguientes casos de prueba se procede a la realización de las pruebas, algunas de las cuales se muestran en los anexos IV, con valores reales según las especificaciones de casos de prueba. El objetivo principal de esta etapa es la detección de errores, los cuales serán registrados para su posterior corrección.

4.3. Conclusiones Parciales.

En el capítulo se resume los principales elementos para la realización del sistema propuesto. Todo lo referente a la concepción del sistema y los principales elementos de programación junto con las pruebas realizadas constituyen elementos de gran importancia para futuras versiones del subsistema.

Conclusiones

El desarrollo de este subsistema brindó la posibilidad de documentar parte de las tareas propuestas para el desarrollo de la aplicación y la implementación de la misma. Se afirma que con el desarrollo de esta investigación:

- ✓ Se logró un incremento en los conocimientos acerca de algunas de las técnicas y tendencias de la programación actuales.
- ✓ Se caracterizó el empleo de estándar de codificación.
- ✓ Se modelaron y ejecutaron algunos casos de pruebas que certificaron el funcionamiento de la aplicación desarrollada.
- ✓ Se implemento el subsistema de Importación y Exportación de Registros de Pozo.

De forma general se concluye que se cumplieron satisfactoriamente el objetivo enunciado desde el inicio de la investigación, por lo que se le dio solución al problemática planteada.

Recomendaciones

- ✓ Ampliar las funcionalidades del subsistema, basándose en la importación de otros tipos de registros de pozos y la exportación de los mismos.

- ✓ Permitir en nuevas versiones exportar los archivos Log ASCII Standard a cualquiera de sus versiones.

- ✓ Realizar pruebas de caja blanca para probar más exhaustivamente el producto obtenido.

- ✓ Utilizar la solución en cualquier otro software probándose así su reusabilidad.

Bibliografía y Referencias Bibliográficas

[1] Nathaly Famiglietti. Licenciada en Ingeniería Geofísica de la Universidad Simón Bolívar, Sartenejas Campus Venezuela, 1999. Visitado en el Sitio SEED. [Disponible en: http://www.seed.slb.com/qa2/FAQView.cfm?ID=914&Language=ES]. [Consultado el día 14 de octubre del 2009].

[2] 2008. Visitado en el Sitio El Boomeran(g) (blog literario en español.). [Disponible en: http://www.elboomeran.com/blog-post/18/3074/andres-ortega/schlumberger/] [Consultado el día 18 de agosto del 2009].

[3] Visitado en el Sitio Neuralog. [Disponible en: http://www.neuralog.com/espanol/index.htm]. [Consultado el día 18 de octubre del 2009].

[4] 31 de marzo del 2008. Visitado en el Sitio patagónico.net. Disponible en: http://www.elpatagonico.net/index.php?item=energia%7Cview&ref=suplementos&id=21429 [Consultado el día 18 de octubre del 2009].

[5] Visitado en el Sitio Energía. [Disponible en: http://www.energia.inf.cu/instituciones/ceinpet.html]. [Consultado el día 18 de octubre del 2009].

[6] 2007. Visitado en el Sitio RCLAB SRL (Laboratorio de Estudios Petrofísicos). [Disponible en: http://www.rclabsrl.com.ar/Pagina1empresa.htm]. [Consultado el día 18 de octubre del 2009]

[7] MARCELO ANDRÉS SARAVIA GALLARDO, Ph. D. METODOLOGÍA DE INVESTIGACIÓN CIENTÍFICA “Orientación metodológica para la elaboración de proyectos e informes de investigación”. Master en Investigación y Doctor en Investigación y Evaluación de la Calidad Educativa. UNIVERSIDAD DE BARCELONA. 2004. Visitado en el Sitio Viceministerio de Ciencia y Tecnología (Ministerio de Planificación del Desarrollo). [Disponible en:

http://www.conacyt.gov.bo/convocatorias/publicaciones/Metodologia.pdf. [Consultado el día 10 de octubre del 2009].

[8] 2007. Visitado en el Sitio Mis respuestas [Disponible en: http://www.misrespuestas.com/que-es-el-petroleo.html]. [Consultado el día 23 de noviembre del 2009]

[9] Lisbeth Camacho. Visitado en el Sitio Monografía [Disponible en: http://www.monografias.com/trabajos7/arch/arch.shtm]. [Consultado el día 13 de diciembre del 2009].

[10] Sugar Land, Estados Unidos, 25 agosto 2008 Visitado en el Sitio PETRÓLEO (Información técnica y de negocios para la industria de los hidrocarburos en América Latina.). [Disponible en: http://www.petroleo.com/pi/secciones/PI/ES/MAIN/N/NOTICIAS1/doc_65217_HTML.html?idDocumento=65217]. [Consultado el día 20 de noviembre del 2009].

[11] 2008 Visitado en el Sitio ESSS (Engineering Simulation and Scientific Software.). [Disponible en: http://www.esss.com.br/index.php?pg=viewprojeto&idProject=9]. [Consultado el día 24 de enero del 2009].

[12] 2005. Visitado en el Sitio HDS (Housing and Development Software). [Disponible en: http://www.hdsoftware.com/about_us.htm][Consultado el día 24 de enero del 2009].

[13] 2002. Visitado en el Sitio The Scotia Group Inc. (Servicios Consultivos Nacionales e Internacionales para Petróleo y Gas). [Disponible en: http://www.scotia-group.com/Spanish/services/reservoir/petrophysics.asp] [Consultado el día 25 de enero del 2009].

[14] 1 marzo 2009. Visitado en el Sitio Gaceta IMP. [Disponible en: http://akbal.imp.mx/gaceta_e/nota_p.asp?nt=intjunE]. [Consultado el día 25 de enero del 2009].

[15] 1 marzo 2009. Visitado en el Sitio Gaceta IMP. [Disponible en: http://akbal.imp.mx/gaceta_e/nota_p.asp?nt=intjunE]. [Consultado el día 25 de enero del 2009].

[16] Martín Jiménez Guerrero. “Caracterización Integral de Yacimientos: Integración de Datos de Registros de Pozo con Atributos Sísmicos Usando Geoestadística”. Universidad Nacional Autónoma de México. México. 2005. 109. [Disponible en: http://132.248.182.189/mdiaz/Tesis/2005/Martin_J/]. [Consultado el día 17 de enero del 2009].

[17] Olga Castro Castiñería, Juan Rodríguez Loeches Diez Argüelles y Dania Brey del Rey. “Modelo petrofísico determina porosidad”. 25 de febrero del 2009. Visitado en el Sitio Modelaje de Yacimientos. [Disponible en: http://modelaje-de-yacimientos.blogspot.com/2008/02/modelo-petrofsico-determina-porosidad.html]. [Consultado el día 19 de enero del 2009].

[18] Visitado en el Sitio Glosario del petróleo y el gas. [Disponible en: http://www.elchenque.com.ar/eco/petro/glosdefi.htm]. [Consultado el día 18 de diciembre del 2009].

[19] Radeck, K. C# and Java: Comparing Programming Languages. Microsoft Corporation, Febrero de 2003. [Citado el: 31 de Octubre de 2009.]

[20] Oualline, S. Practical C++ Programming. O'Reilly and Associates, Inc. Pag: 3-7. 1995. [Citado el: 31 de Octubre de 2009.]

[21] Charte, F. Visual C#.Net. Anaya Multimedia, SA. 2002. [Citado el: 31 de Octubre de 2009.]

[22] Smith, E. A., Whisler, V., et al. Visual Basic 6 Bible. IDG Books WorldWide, Inc. USA, 1998. [Citado el: 31 de Octubre de 2009.]

[23] Deitel, H. M. y Deitel, P. J. Java How to Program. Prentice Hall. 2002. [Citado el: 5 de Noviembre de 2009.]

[24] Robinson, S., Cornes, O., et al. Professional C#. Wrox Press Ltd. 2001. [Citado el: 5 de Noviembre de 2009.]

[25] URRIELLUnet. [En línea] [Citado el: 5 de Noviembre de 2009.] Disponible en: http://urriellu.net/es/articles-software/csharp-advantages.html

[26] Metodologías de Desarrollo Software [En línea] Febrero de 2009 [Citado el: 25 de Noviembre de 2008] Disponible en: http://www.scribd.com/doc/2050925/metodologias-de-desarrollo-software

[27] eduangi telecom – Proyectos de Open Source y Telecomunicaciones. [En línea] 4 de Agosto de 2003[Citado el: 25 de Noviembre de 2009.] Disponible en: http://web.madritel.es/personales3/edcollado/ingsw/tema2/2-4.htm.

[28] Beck, K. Extreme Programming Explained. Addison-Wesley Professional, 1999.

[29] Proceso de Desarrollo OpenUP – Córdova Software Factory [En línea] 2 de Septiembre de 2008 [Citado el: 25 de Noviembre de 2009.] Disponible en: http://cbasqa.wordpress.com/2008/09/02/proceso-de-desarrollo-openup/

[30] IBM Rational Unified Process – Grupo Soluciones Innova [En línea] 2007 [Citado el: 25 de Noviembre de 2009] Disponible en: http://www.rational.com.ar/herramientas/rup.html

[31] Jacobson, Ivar; Booch, Grady; Rumbaugh, James. El Proceso Unificado de Desarrollo de Software. [Citado el: 10 de Diciembre de 2009.]

[32] Visitado en el Sitio GSInnova (Grupo Soluciones). [Disponible en: http://www.rational.com.ar/herramientas/roseenterprise.html]. [Consultado el 25 de febrero del 2009].

[33] Falla Villega, Elias Jhon. Interpretación de registro de pozos de petróleo. [Disponible en: http://sisbib.unmsm.edu.pe/bibvirtualdata/Tesis/Basic/falla_ve/cap2.intro.pdf]. [Consultado el 25 de febrero del 2009].

[34] Vitter, D; Templeman J. Visual Studio .NET, 2002. [Citado el: 10 de Enero de 2009]

[35] Powers, R; Snell, M. Microsoft Visual Studio 2008 UNLEASHED, 2008. [Citado el: 12 de Enero de 2009]

[35] “Material_para_la_Conf_1.doc”. Visitado en el Sitio Teleformación.uci.cu. [Disponible en: http://eva.uci.cu/mod/resource/view.php?id=11402]. [Consultado el 26 de enero del 2009].

[36] Francisco Vega Leyte Vidal. Diseño de la arquitectura de software para el proyecto petrofísica. [Citado el: 10 de marzo de 2010.]

[37] Ingeniería de Software 1. Conferencia #4. Fase de Inicio. Flujo de trabajo de requerimientos. Modelo de Diseño. Curso 2008-2009. 27. [Citado el: 8 de abril del 2010].

[38] Yusleidy Neira Rodríguez. Análisis y Diseño del Sistema para el análisis petrofísico. [Citado el: 8 de abril del 2010].

[39] Suzanne Robertson, James Robertson, Mastering the Requirements Process Second Edition, Addison Wesley Professional, 2006. [Citado el: 12 de abril del 2010].

[40] IngenieroSoftware. Análisis y Diseño. UML: Casos de Uso. Por Joaquin Gracia. [Disponible en: http://www.ingenierosoftware.com/analisisydiseno/casosdeuso.php]. [Citado el: 12 de abril del 2010]

[41] JACOBSON, J. R. G. B. I. El Proceso unificado de desarrollo de software Madrid, Pearson Educación, 2000.

[42] LARMAN, Craig “UML y patrones” 1999, Prentice Hall Iberoamericana. Capítulos 13, 16, 17, 18, 19, 21, 34 y 35.

[43] Martínez Santos, Eusebio Antonio, 2008. ESTADO DEL ARTE DE LA PROGRAMACIÓN ORIENTADA A OBJETO (POO). Marzo 10, 2010.

Glosario de Términos

Porosidad: Es una medida de la capacidad de almacenamiento de fluidos que posee una roca y se define como la fracción del volumen total de la roca que corresponde a espacios que pueden almacenar fluidos. [1]

Permeabilidad: Se define como la capacidad que tiene una roca de permitir el paso de fluidos a través de sus poros interconectados. La idea de permeabilidad fue desarrollada a partir de los experimentos de Henry Darcy. [1]

Saturación: Es el porcentaje del espacio poroso de una roca, ocupada por un fluido. [1]

Presión Capilar: Es la diferencia de presión a través de la interface que separa dos fluidos inmiscibles, cuando se ponen en contacto en un medio poroso. [1]

Tortuosidad: Se define debido a que los poros si existen y las presencia de las interface originan presiones capilares que afectan los procesos de desplazamiento de las sustancias ya que los poros interconectados que en la roca representan los canales de flujo de los fluidos en el yacimiento (gas, petróleo, gas) no son tubos capilares rectos ni tampoco tienen pared lisa. [1]

Yacimiento petrolífero: Un yacimiento petrolífero o campo petrolífero es una zona con abundancia de pozos de los que se extrae petróleo del subsuelo.[1]

Anexos

Anexos I: Importar Archivo LAS.

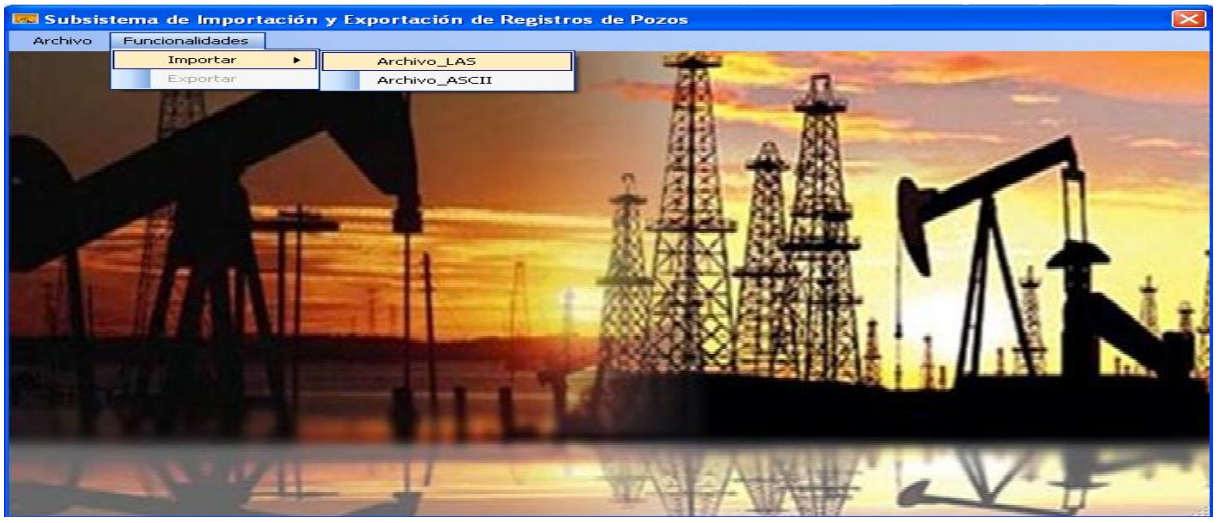


Figura 10. Selección de la funcionalidad importar archivo Log ASCII Standard

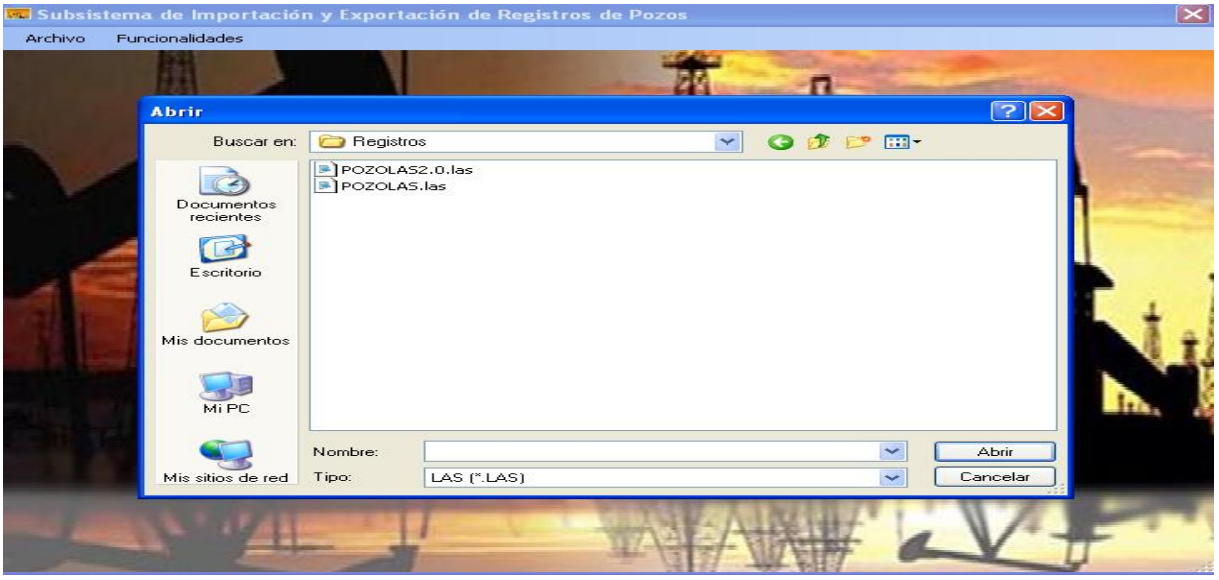
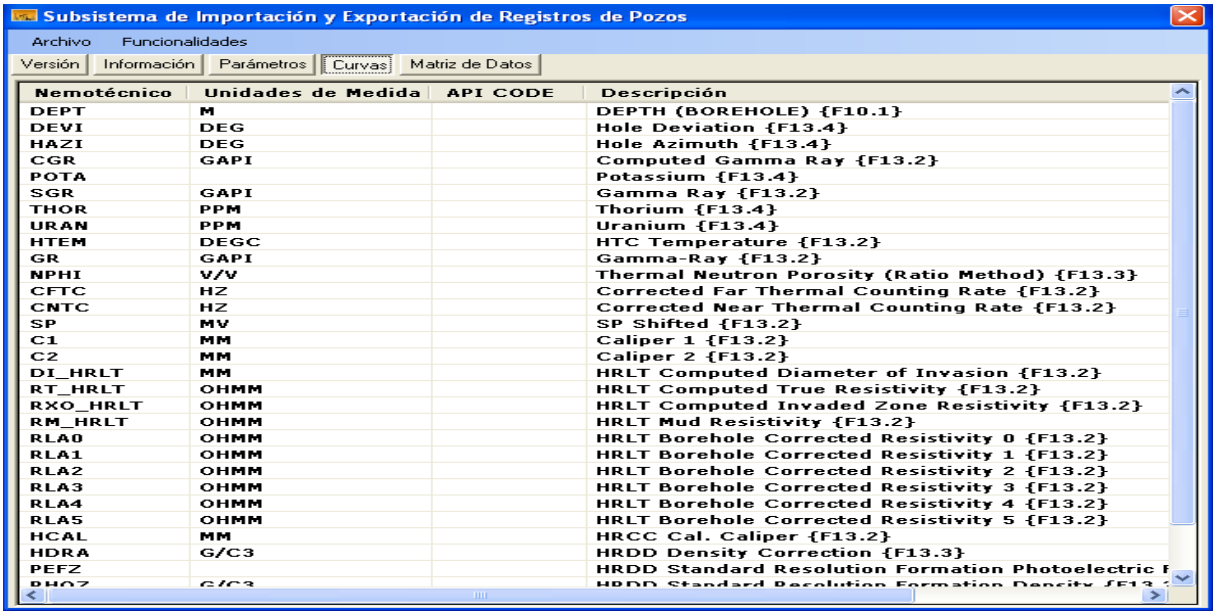


Figura 11. Selección del archivo Log ASCII Standard a importar.



Nemotécnico	Unidades de Medida	API CODE	Descripción
DEPT	M		DEPTH (BOREHOLE) {F10.1}
DEVI	DEG		Hole Deviation {F13.4}
HAZI	DEG		Hole Azimuth {F13.4}
CGR	GAPI		Computed Gamma Ray {F13.2}
POTA			Potassium {F13.4}
SGR	GAPI		Gamma Ray {F13.2}
THOR	PPM		Thorium {F13.4}
URAN	PPM		Uranium {F13.4}
HTEM	DEGC		HTC Temperature {F13.2}
GR	GAPI		Gamma-Ray {F13.2}
NPHI	V/V		Thermal Neutron Porosity (Ratio Method) {F13.3}
CFTC	HZ		Corrected Far Thermal Counting Rate {F13.2}
CNTC	HZ		Corrected Near Thermal Counting Rate {F13.2}
SP	MV		SP Shifted {F13.2}
C1	MM		Caliper 1 {F13.2}
C2	MM		Caliper 2 {F13.2}
DI_HRLT	MM		HRLT Computed Diameter of Invasion {F13.2}
RT_HRLT	OHMM		HRLT Computed True Resistivity {F13.2}
RXO_HRLT	OHMM		HRLT Computed Invaded Zone Resistivity {F13.2}
RM_HRLT	OHMM		HRLT Mud Resistivity {F13.2}
RLA0	OHMM		HRLT Borehole Corrected Resistivity 0 {F13.2}
RLA1	OHMM		HRLT Borehole Corrected Resistivity 1 {F13.2}
RLA2	OHMM		HRLT Borehole Corrected Resistivity 2 {F13.2}
RLA3	OHMM		HRLT Borehole Corrected Resistivity 3 {F13.2}
RLA4	OHMM		HRLT Borehole Corrected Resistivity 4 {F13.2}
RLA5	OHMM		HRLT Borehole Corrected Resistivity 5 {F13.2}
HCAL	MM		HRCC Cal. Caliper {F13.2}
HDRA	G/C3		HRDD Density Correction {F13.3}
PEFZ			HRDD Standard Resolution Formation Photoelectric F
RHOZ	G/C3		HRDD Standard Resolution Formation Density {F13.2

Figura 12. Visualización de los datos del archivo Log ASCII Standard.

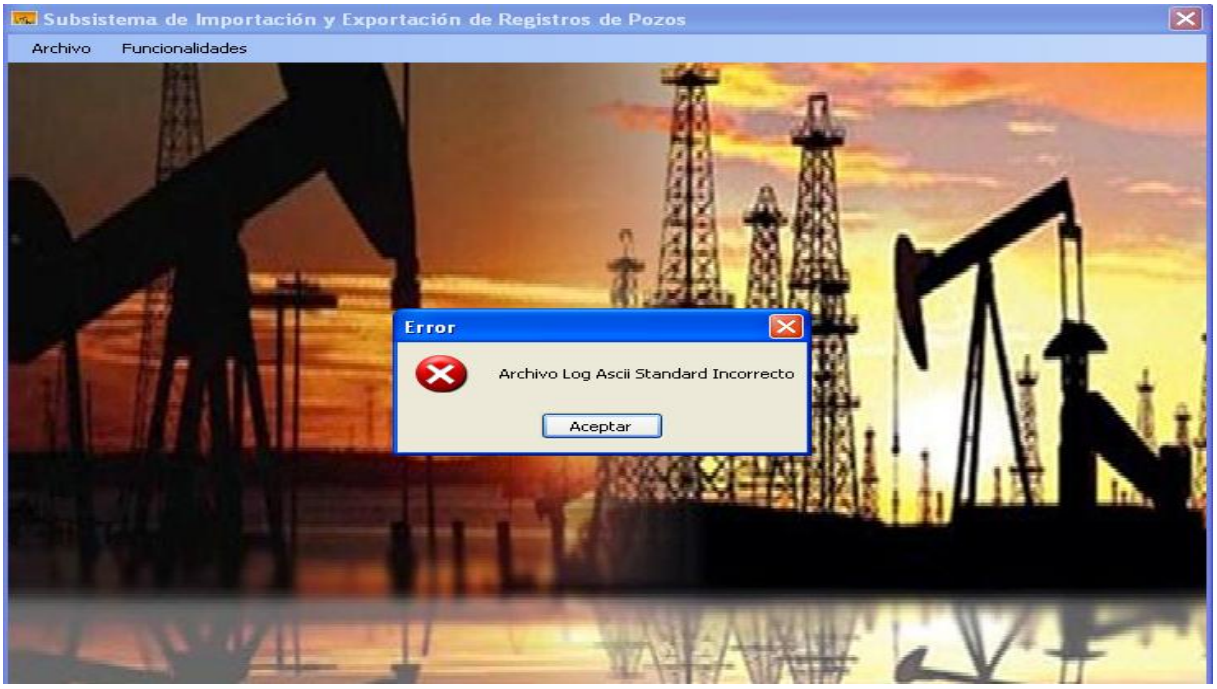


Figura 13. Formato del archivo Log ASCII Standard incorrecto

Anexos II: Importar Archivo ASCII.

Figura 14. Selección de la funcionalidad importar archivo ASCII

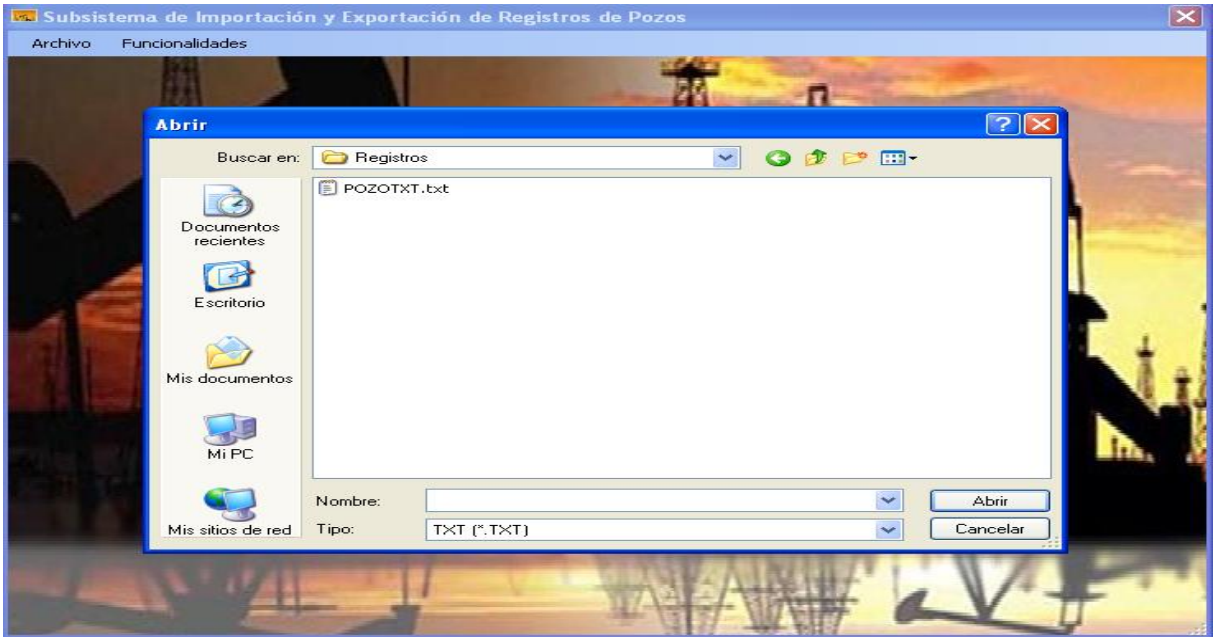


Figura 15. Selección del archivo ASCII a importar.

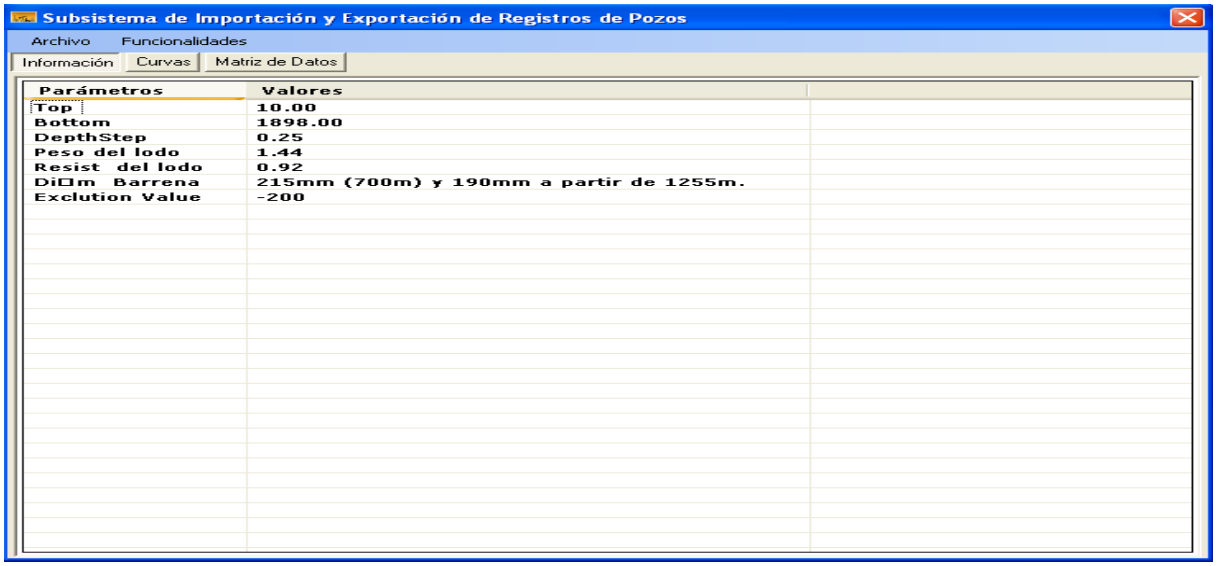


Figura 16. Visualización de los datos del archivo ASCII.

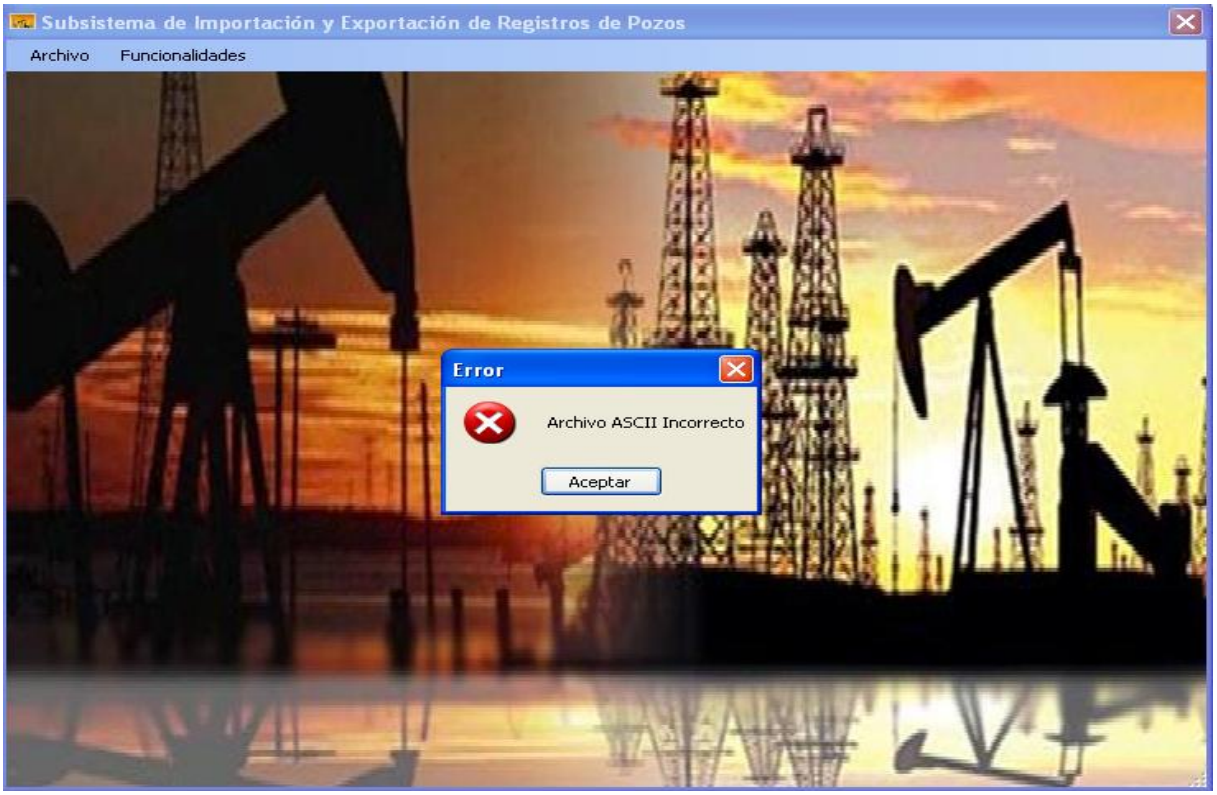


Figura 17. Formato del archivo ASCII incorrecto

Anexo III: Exportar Archivo LAS.

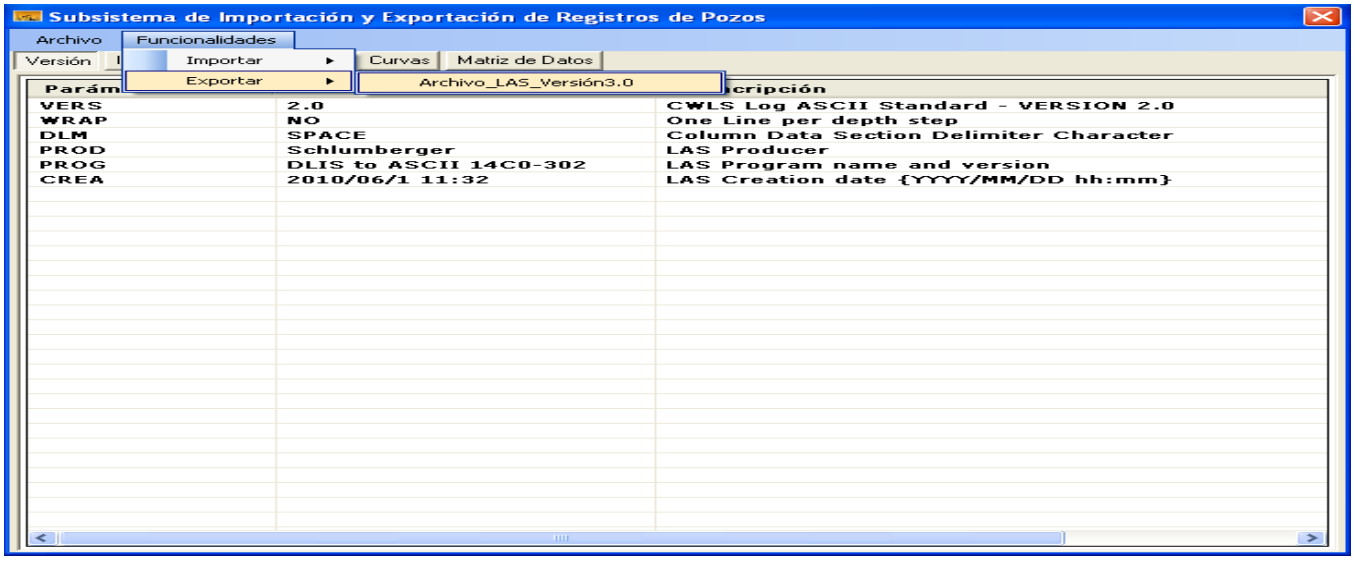


Figura 18. Selección de la funcionalidad exportar archivo Log ASCII Standard a la versión 3.0

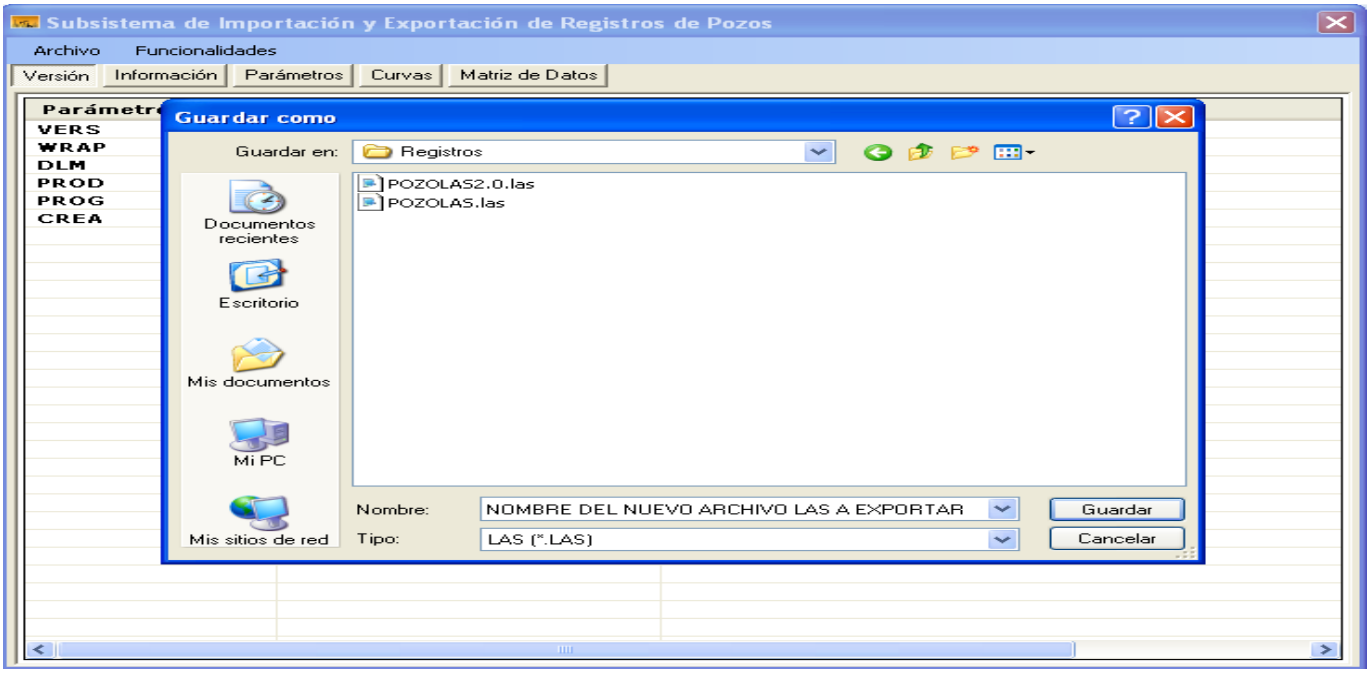


Figura 19. Selección de la dirección a guardar del archivo que se exportara.

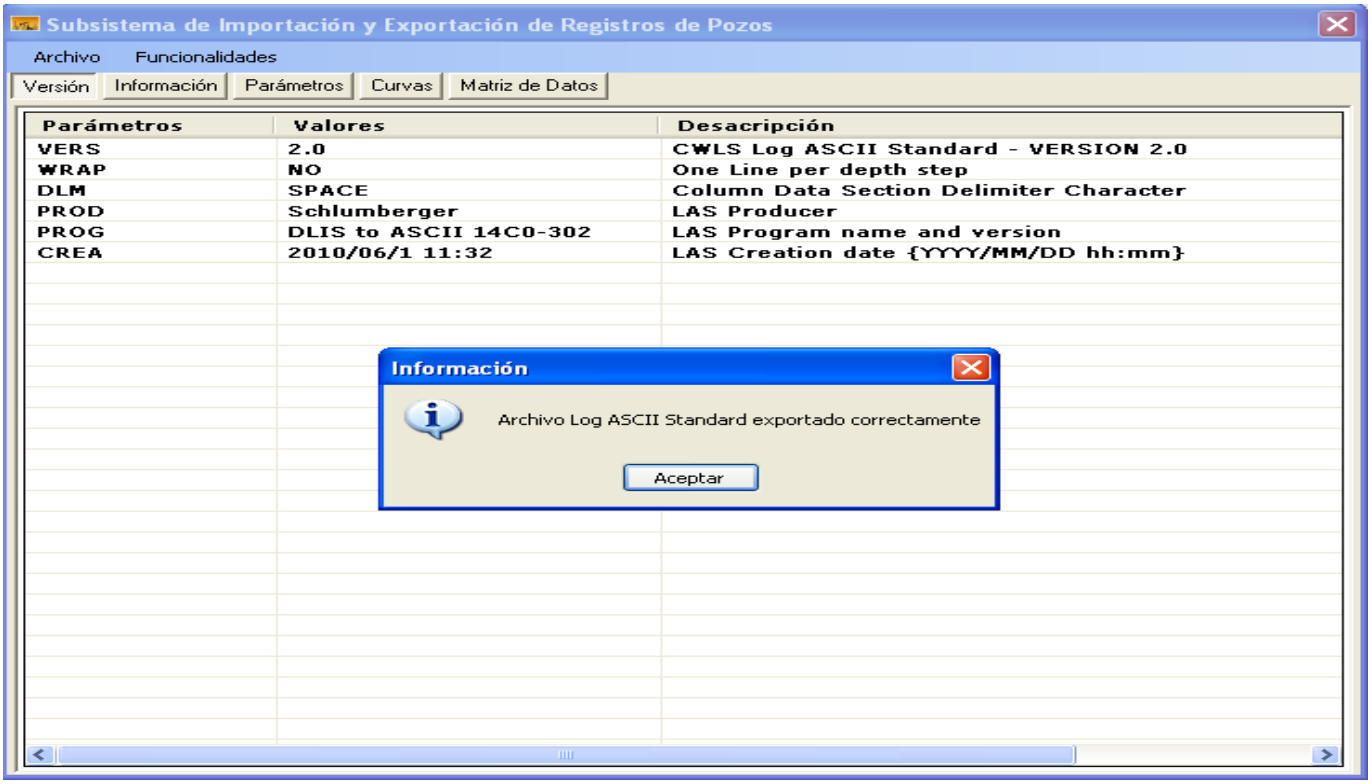


Figura 20. Archivo exportado.

Anexo IV: Casos de pruebas para el caso de uso Exportar Fichero

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Exportar Archivo.	EC 1.1 Exportar archivo LAS.	Exporta el archivo a la versión 3.0.
	EC 1.2 Cancelar la exportación del archivo LAS.	El sistema cierra la ventana, quedando la Forma Principal abierta dándole paso a otras operaciones.

Table 6. Descripción de las secciones y escenarios del Caso de Uso Exportar Archivos

ID Sección	Escenario	Respuesta del Sistema	Resultado de la Prueba
SC 1:	Exportar archivo LAS.	El sistema guarda en la dirección que el petrofísico escogió para guardar el fichero LAS, emitirá un mensaje "Archivo Log ASCII Standard exportado correctamente" y termina el caso de uso.	Prueba satisfactoria.
	Cancelar la exportación del archivo LAS.	El sistema cierra la ventana, quedando la Forma Principal abierta dándole paso a otras operaciones.	Prueba satisfactoria.

Table 7. Caso de prueba del caso de uso exportar archivo

Anexo V: Encuesta realizada a los trabajadores del CEINPET

¿Cómo se realiza el trabajo de Visualizar Registros?

¿Cuál es el orden de las acciones para la Visualización de Registros?

¿Para realizar el proceso de Visualización de Registros, necesita cargar solamente un archivo .LAS? En caso de necesitar algún otro diga cuál es?