DESARROLLO DE UN COMPONENTE SOFTWARE PARA INTEGRAR LAS LIBRERÍAS DEL SOFTWARE DE MODELADO GEOMECÁNICO (G.M.S) EN LA PLATAFORMA PETREL

PEDRO ANDRÉS RODRÍGUEZ SÁNCHEZ

UNIVERSIDAD INDUSTRIAL DE SANTANADER

FACULTAD DE INGENIERÍAS FISICOMECÁNICAS

ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

BUCARAMANGA – COLOMBIA

2013

DESARROLLO DE UN COMPONENTE SOFTWARE PARA INTEGRAR LAS LIBRERÍAS DEL SOFTWARE DE MODELADO GEOMECÁNICO (G.M.S) EN LA PLATAFORMA PETREL

PEDRO ANDRÉS RODRÍGUEZ SÁNCHEZ

Trabajo de grado para optar al título de ingeniero de sistemas

DIRECTOR:

MSc. FERNANDO ANTONIO ROJAS MORALES,

Docente planta, Universidad Industrial de Santander

UNIVERSIDAD INDUSTRIAL DE SANTANADER

FACULTAD DE INGENIERÍAS FISICOMECÁNICAS

ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

BUCARAMANGA – COLOMBIA

2013

AGRADECIMIENTOS

El autor de este proyecto expresa su agradecimiento:

Al ingeniero Cesar augusto Ochoa, por su confianza, su gran colaboración, continuo apoyo y oportunos concejos que permitieron realizar este trabajo de grado.

Al grupo de investigación Estabilidad de pozo por la oportunidad que nos brindó de crecer en un ambiente investigativo, por la experiencia laboral que nos han permitido fortalecernos como personas.

A todos mis compañeros quienes de alguna forma se involucraron para hacer realidad este trabajo.

A la escuela de ingeniería de sistemas e informática y por ende a su cuerpo de docentes, por las enseñanzas recibidas durante el transcurrir de mi vida universitaria.

A la empresa Schlumberger que me brindó la oportunidad de trabajar con su SDK y puso a mi disposición todo el material necesario para el buen desarrollo de este proyecto.

DEDICATORIA

A mi madre Ana Inés Sánchez y a mi hermana Sandra Patricia Rodríguez por el ejemplo de vida que son, por todo su apoyo, comprensión, amor y sacrificio a través de los años.

Al ingeniero Cesar Augusto Ochoa por su gran apoyo, especialmente en los momentos de mayor dificultad.

A mi familia por ser parte incondicional de mi desarrollo personal.

A mis compañeros que de una u otra forma contribuyeron para el logro de esta meta.

PEDRO ANDRES RODRIGUEZ SANCHEZ

Tabla Contenido

ÍNDICE DE FIGURAS

LISTA DE ANEXOS

RESUMEN

TITULO: DESARROLLO DE UN COMPONENTE SOFTWARE PARA INTEGRAR LAS LIBRERÍAS DEL SOFTWARE DE MODELADO GEOMECÁNICO (G.M.S) EN LA PLATAFORMA PETREL*

AUTOR

PEDRO ANDRÉS RODRÍGUEZ SÁNCHEZ **

PALABRAS CLAVE

Petrel, Ocena, GSM, plataforma software, geomecánica, estabilidad de pozo, GIEP, Schlumberger.

DESCRIPCIÓN

El grupo de investigación estabilidad de pozo (GIEP) ha basado sus estudios en el análisis e investigación de los fenómenos geomecánicos que afectan a la industria del petróleo. Para corroborar el análisis y las interpretaciones de las pruebas que se realizan, constantemente se elaboran herramientas software que permiten apreciar el modelado de los diferentes fenómenos y permite a los investigadores tomar las mejores decisiones dependiendo de los resultados apreciados y obtenidos en la ejecución de estas herramientas software especializadas, gracias a la plataforma software de modelado geomecánico (G.M.S.). Desarrollada en el 2010 el grupo de investigación cuenta con la herramienta que permite unir estas investigaciones en un centro de control común, es así como el (GIEP) puede ahora ofrecer sus servicios por medio de su plataforma e ir incrementando estos servicios al incluir nuevos plug-ins elaborados bajo la guía de su estándar de desarrollo software (E.D.S.), sin embargo para que esta plataforma pueda ser tomada en cuenta por la industria de análisis geomecánicos debe entrar a un duro mercado donde plataformas software mundialmente conocidas como Petrel tiene el dominio de este mercado.

En este trabajo se diseñó, estructuro e implemento un complemento software utilizando el SDK Ocean para la plataforma software Petrel de la multinacional Schlumberger, que permite la integración de las librerías de evaluación geomecánica (G.M.S.) en esta plataforma, gracias a la integración dinámica de los servicios del (G.M.S.) se garantiza un acople completo de los servicios actuales y futuros que se desarrollen bajo el (E.D.S.), este componente le otorgará al (G.M.S.) el nivel de competencia necesario para incurrir en el mercado del software de modelado geomecánico, al ofrecer un servicio de integración con una de las plataformas software más usadas en esta área.

---

* Proyecto de Grado en la modalidad de investigación.

** Facultad de ingenierías físico-mecánicas. Ingeniería de sistemas e informática ROJAS MORELES, Fernando Antonio.

ABSTRACT

TITLE: DEVELOPMENT A SOFTWARE COMPONENT TO INTEGRATE GEOMECHANICAL LIBRARIES MODELING SOFTWARE (GMS) ON THE PLATFORM PETREL.

AUTHOR

PEDRO ANDRÉS RODRÍGUEZ SÁNCHEZ **

KEYWORDS

Petrel, Ocean, GSM, software platform, geomechanics, wellbore stability, GIEP, Schlumberger.

DESCRIPTION

The research group wellbore stability (GIEP) has based its studies and research in the analysis of geomechanical phenomena affecting the oil industry. To corroborate the analysis and interpretation of the tests performed, constantly developed software tools that allow us to appreciate the modeling of different phenomena and allows researchers to make the best decisions based on the results obtained in appreciated and implementation of these tools specialized software thanks to the geomechanical modeling software platform (GMS). Developed in 2010, the research group has the tool to unite these investigations in a common control center, so as (GIEP) can now offer its services through its platform and go increasing these services to include new plug -ins developed under the guidance of their standard software development (ESD), however for this platform can be accounted for by industry geomechanical analysis must enter a hard market where software platforms Petrel is known worldwide as the domain of this market.

This study was designed, structured and implemented using a plug-in SDK Ocean for Petrel software platform for the multinational Schlumberger, which allows the integration of geomechanical assessment libraries (GMS) in this platform, by integrating dynamic services (GMS) are guaranteed a full coupling of current and future services to be developed under the (EDS), this component will give the (GMS) the level of skill required to incur the software market geomechanical modeling, the offer an integration with a software platform used in this area.

---

* Proyecto de Grado en la modalidad de investigación.

** Facultad de ingenierías físico-mecánicas. Ingeniería de sistemas e informática ROJAS MORELES, Fernando Antonio.

1 INTRODUCCIÓN

Actualmente el grupo estabilidad de pozo (GIEP), cuenta con un estándar de desarrollo software que guía la elaboración de estas herramientas, esto ha permitido que se desarrollen herramienta software basados en el estándar y que trabajen y se vinculen por medio de dlls como plugins anexos a la plataforma G.M.S, esto ha generado software especializado en análisis matemáticos, pruebas de laboratorio y visualización de registros, entre otros, que satisfacen la demanda. Sin embargo al ser una plataforma tan joven aún le falta mejorar su capacidad de prestar un servicio especializado. Plataformas como PETEREL son usadas a diario en la industria de los hidrocarburos alrededor del mundo.

Es así como surge una necesidad de implementar un componente software que conecte estas dos plataformas de manera ágil y trasparente para el usuario, de esta forma evitar tiempos no productivos en la ejecución de las pruebas realizadas en laboratorio, esto debido al hecho de tener que migrar los datos de una plataforma a la otra para realizar las pruebas que se necesitan.

En este documento se describirá de manera general el procedimiento que se siguió para realizar este procedimiento y una visualización de los resultados de dicho componente software.

2 ESPECIFICACIONES DEL PROYECTO

En este capítulo se realiza la presentación del proyecto, mencionando la oportunidad observada en la plataforma software (G.M.S) y su estándar de desarrollo software (E.D.S) del grupo de investigación estabilidad de pozo (GIEP) y la posibilidad de integrar este estándar a PETREL usando el SDK OCEAN, igualmente durante este capítulo se describen los objetivos planteados y el alcance estimado del proyecto.

2.1 PLANTEAMIENTO DEL PROBLEMA

Los modelos geomecánicos son importantes en la industria de los hidrocarburos ya que su papel es proporcionar información de la estabilidad del campo a perforar o perforado. Para ello, los análisis geomecánicos que se realizan en laboratorios y campos necesitan herramientas software rápidas y efectivas para evitar, mitigar o corregir las anomalías presentadas en dicho proceso. Teniendo en cuenta esto, multinacionales como Schlumberger han optado por el desarrollo de herramientas software capaz de suplir muchas de las necesidades de esta industria. Con el objetivo de maximizar la explotación de los yacimientos fue desarrollado PETREL, una plataforma software que permite agregar datos de yacimiento provenientes de múltiples fuentes y convertirlos en información útil para la interpretación de datos sísmicos, construcción de modelos de yacimiento adecuados para la simulación, entre otras aplicaciones.

En el contexto local en el 2010 el Grupo de Investigación de Estabilidad de Pozo (GIEP) de la Universidad Industrial de Santander (UIS) dio a conocer una plataforma software de modelado geomecánico (G.M.S.) que permite, por medio de un estándar de desarrollo, integrar actuales y futuras herramientas software creadas en este grupo con el fin de suplir las necesidades crecientes de esta industria en el área de la geomecánica, la estabilidad de pozos y otros desafíos de esta industria, además de otorgándole a los semilleros herramientas adecuadas para su desarrollo intelectual.

El presente proyecto busca desarrollar un componente software para la plataforma PETREL usando el SDK OCEAN que permita cargar y ejecutar los plug-ins o librerías creados en el estándar de desarrollo implementado en la herramienta G.M.S. del grupo de Investigación Estabilidad de Pozo. Lo anterior permitirá a PETREL ejecutar los procesos almacenados en estas librerías, procesos tales como el cálculo de fallas geomecánicas en la estabilidad del pozo, donde estos

cálculos dan como resultado información importante para simulaciones futuras y que al ejecutarse dentro de la misma plataforma se puede almacenar directamente en la base de datos del proyecto que se está trabajando en PETREL.

Con ello se posibilita la apertura de un área de estudio relacionada con el SDK OCEAN para PETREL que permitirá que futuros desarrollos de la Escuela de Ingeniería de Sistemas e Informática de la Universidad Industrial de Santander se direccionen a una nueva área de desempeño, ofreciendo de esta manera un nuevo nivel de competencia a los futuros ingenieros de sistemas. Además el Grupo de Investigación de Estabilidad de Pozo (GIEP) se beneficiará de este proyecto al adquirir la capacidad que sus productos tecnológicos actuales y futuros basados en su estándar de desarrollo para la plataforma G.M.S. se ejecuten en la plataforma PETREL y así poder extender las propiedades del G.M.S. al aprovechar las propiedades de PETREL.

2.2 OBJETIVOS DEL PROYECTO

2.2.1 Objetivo general

Desarrollar un componente software que permita utilizar las librerías de evaluación geomecánica del software de modelado geomecánico (G.M.S.) del Grupo de Investigación en estabilidad de Pozo (GIEP) en la plataforma software PETREL utilizando el SDK OCEAN de Schlumberger.

2.2.2 Objetivos específicos

- Identificar las variables que intervienen en el estándar de desarrollo (EDS) del software de modelado geomecánico (G.M.S.)
- Diseñar un componente software con el SDK OCEAN que permita interpretar las librerías G.M.S. creadas con el EDS
- Desarrollar una interfaz gráfica de usuario que incluya un entorno de visualización, con el fin de presentar gráficamente las librerías G.M.S. disponibles en PETREL
- Diseñar un módulo en el componente creado para PETREL que incluya la información de los resultados de los procesos ejecutados en las librerías G.M.S.
- Generar un manual de referencia del SDK OCEAN para futuros desarrollos basado en el desarrollo de este componente.

2.3 ALCANCE DEL PROYECTO

Diseñar e implementar un componente software para para la plataforma software PETREL, que permita incluir las librerías desarrolladas para el G.M.S del grupo de investigación de estabilidad de pozo (GIEP) para dicha plataforma, aprovechando la capacidad de PETREL y el desarrollo de nuevas herramientas software especializadas del GIEP basados en es EDS del G.M.S

Con el desarrollo de este componente se pretende marcar una pauta que permita expandir los alcances de las investigaciones que se realizan en el grupo de investigación de estabilidad de pozo (GIEP), proporcionándoles un nuevo atributo a su plataforma G.M.S, el cual sería el enlace con la plataforma software mundial PETREL desarrollada por Schlumberger.

Las tecnologías utilizadas en este proyecto están en constante evolución y las estructuras planteadas en su implementación deberán cumplir con las reglas y estándares planteados en el proyecto G.M.S del grupo de investigación de estabilidad de pozo (GIEP). Gracias a esto, se permitirá un crecimiento continuo, lo que otorgará una vida útil más extensa del componente software y una mayor adaptabilidad a futuros cambios del G.M.S.

Se espera que el desarrollo de este proyecto abra las puertas a futuros desarrollos con el SDK OCEAN, no solo en la universidad industrial de Santander (UIS) y el grupo de investigación de estabilidad de pozo (GIEP) y otras universidades interesadas sino también en el instituto colombiano del petróleo (ICP) y en la industria del petróleo en general.

2.4 METODOLOGÍA DE DESARROLLO

El trabajo de investigación y desarrollo realizado para el grupo de investigación de estabilidad de pozo (GIEP), se encuentra enmarcado dentro de un modelo de maduración de proyectos de investigación (MMPI), el cual fue elaborado dentro del convenio UIS-ICP y que en la actualidad es implementado en el grupo como medida de mejoramiento y seguimiento de los trabajos de investigación que se realizan los semilleros y tesistas activos del grupo.

El MMPI consiste en cuatro fases que permiten ir construyendo de manera clara y precisa el trabajo de investigación en desarrollo, brindándole a quien lo implemente (semillero o tesista) la posibilidad de estructurar la investigación, garantizando no solo buenos resultados sino resultados coherentes a lo planteado desde un inicio en el trabajo de grado.

La estandarización para el desarrollo de procesos es un tema a nivel mundial. Para este trabajo, es necesario aplicar los mismos estándares usados para el desarrollo del G.M.S, por lo que se estudiaron dos estándares reconocidos como lo son: (1) el RUP (Proceso Unificado de Rational), uno de los estándares más utilizados en análisis, diseño, implementación y documentación de sistemas orientados a objetos para producir software de calidad, esta metodología es adaptable al contexto y necesidades de cada organización, cualidad imprescindible para el buen desarrollo de este trabajo; y (2) el formato de Especificación de Requisitos Software (ERS) presentado en el estándar de la IEEE 830, que al igual que los anteriores es muy utilizado en la ingeniería de software debido a que proporciona a quien lo utiliza bases sólidas para la elaboración de herramientas software.

La metodología implementada está constituida de una serie de etapas, organizadas en un ciclo constructivo que permite la corrección a tiempo de cualquier irregularidad presentada al momento de elaborar el proyecto de investigación y desarrollo.

A continuación se muestra un diagrama correspondiente al ciclo de desarrollo de la metodología implementada.

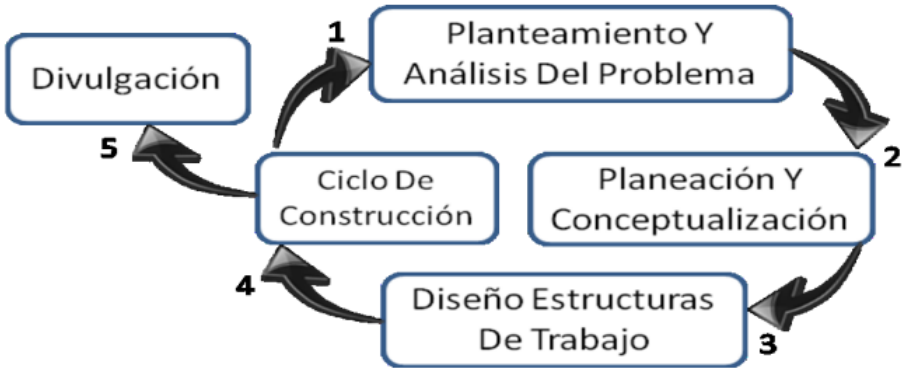


Figura 1 Ciclo de desarrollo

Fuente: ROJAS, 2010

2.4.1 Planteamiento y análisis del problema

La metodología a seguir comienza con la etapa donde se abarca la definición del problema y realiza la investigación de los antecedentes mediante la revisión bibliográfica. En esta etapa se reúne el análisis de requerimientos junto con las partes implicadas en el trabajo a desarrollar, permitiendo tener claro los parámetros y funciones que deberá cumplir el proyecto.

Se incluye además la búsqueda de documentación referente a esta investigación, usando los siguientes medios:

- Revisión bibliográfica de antecedentes del tema, estándares, componentes software e intérpretes software.
- Artículos a nivel mundial referentes al tema a componentes software.
- Recopilación de trabajos pasados realizados en el grupo de investigación estabilidad de pozo (GIEP).

Como producto final de esta fase, se redactan los alcances que se pretenden lograr, mediante la formulación del objetivo general y se formaliza la justificación al tema de investigación basándose en la revisión bibliográfica encontrada.

2.4.2 Planeación y conceptualización

Una vez ya planteados los objetivos y definido el camino a seguir, se inicia con la segunda etapa. En esta fase se contempla a fondo los antecedentes del trabajo a realizar, ya no de manera general, sino de forma específica, estudiando así los componentes software para PETREL utilizando el SDK OCEAN y la plataforma G.M.S.

Las bases teóricas necesarias para el desarrollo de este proyecto se recopilaron en esta etapa, sirviendo como soporte conceptual para el proyecto de investigación y desarrollo. Por medio esto, se logró de la delimitación de los alcances del proyecto estableciendo objetivos más claros y específicos.

Para finalizar esta fase, partiendo del estudio de alternativas y posibles soluciones al tema planteado, se realiza el planteamiento concreto de la hipótesis, en la que se describe el flujo de trabajo guía para la ejecución del proyecto. Esta hipótesis se ha planteado gracias a las metodologías, modelos, estándares y artículos que han servido para dar soporte teórico a este proyecto.

2.4.3 Diseño estructuras de trabajo

Esta fase se enfoca en la elaboración de los esquemas y diagramas correspondientes al funcionamiento del trabajo final. Estos diagramas se usarán en la etapa de construcción, lo cual permitirá realizar un trabajo más eficaz y ordenado, utilizando no sólo estos diagramas, sino los requerimientos extraídos con la participación de los actores implicados en el proyecto.

Las labores que se usaron para la construcción de esta etapa son:

- Interpretación y asimilación de los estándares que servirán como guía en la construcción del intérprete de las librerías G.M.S para la plataforma PEPTREL
- Extracción de las librerías actuales presentes en G.M.S que comprenden las metodologías implementadas bajo su estándar E.D.S para realizar las pruebas en la etapa final del proyecto de investigación y desarrollo.
- Delimitación del alcance a partir de estas librerías, con lo cual se estructura el cronograma a seguir en la fase de construcción.
- Modelado estructural de la metodología elegida para la construcción del intérprete de las librerías de la herramienta software de modelado geomecánico (G.M.S) y la validación de este interprete.
- Realización de diagramas y esquemas de las funciones que deben cumplir el intérprete.
- Diseño concreto del interprete software con las especificaciones correspondientes a cada etapa que compone dicho estándar
- Planteamiento de la estructura operativa que se usara en la fase de construcción.

2.4.4 Ciclo de construcción

Esta etapa contempla la ejecución de todos los planes anteriormente escritos, así como la construcción del intérprete, corrección de errores y ajustes de rendimiento del sistema. La realización de las labores se basa en un ciclo que conforma las

cuatro primeras fases. Si la labor se ejecuta sin cambios estructurales ni complicaciones, se pasa de inmediato a la etapa de difusión de resultados. En caso de que se encuentre algún tipo de error, se requieran cambios en la estructura del sistema o existan improvistos en el desarrollo, se requerirá una nueva evaluación de las fases anteriores con el fin de corregir o evaluar las nuevas estructuras del sistema.

Con las estructuras funcionales que se elaboraron con anterioridad, se busca crear el sistema central para el intérprete de las librerías G.M.S en la plataforma PETREL, validando dicha herramienta mediante la ejecución de las diferentes librerías G.M.S disponibles para el Grupo de Investigación Estabilidad de Pozo (GIEP).

En este punto se sigue un ciclo creado por los autores del G.M.S para la implementación del estándar en el intérprete software para PETREL, este ciclo está compuesto por 3 etapas descritas a continuación:

Figura 2 Ciclo de desarrollo

Fuente: ROJAS, 2010

- **Análisis e interpretación:** se hace un análisis de la metodología que se quiere incluir en el sistema central de la herramienta software, por medio de análisis de requerimientos y otras fuentes.
- **Corrección y acoplamiento:** realización de pruebas y ajustes necesarios a la metodología para permitir la interpretación de las librerías G.M.S en PETREL
- **Implantación:** inclusión formar de la metodología a PETREL por medio de un componente software que se anexara a la plataforma PETREL, en esta etapa del ciclo se debe reiniciar desde la etapa análisis e interpretación para realizar las debidas correcciones detección de errores y su corrección adecuada.

Este ciclo de desarrollo permitirá detectar y corregir los errores de manera más ágil, controlada y precisa. Al finalizar esta fase se entrega una documentación

completa de las pruebas realizadas los errores encontrados, las correcciones y modificaciones realizadas, y toda aquella información relevante, no sólo para la comprensión del interprete software en PETREL, sino que además las experiencias obtenidas en el desarrollo de dicho componente usando el SDK OCEAN.

2.4.5 Divulgación

Una vez finalizado el trabajo de investigación, se realizará la divulgación de los resultados obtenidos por medio de una presentación al grupo de investigación de estabilidad de pozo (GIEP) como al encargado del SDK OCEAN en Colombia, también se plantea en el MMPI que este tipo de divulgación se realice en revistas de carácter científico, de manera que sirva de punto de referencia en trabajos posteriores. La difusión hace parte importante de las etapas del ciclo de desarrollo, debido a que es importante dar a conocer los resultados que se obtienen y sentar precedente de las dificultades en el desarrollo de este proyecto.

3 MARCO TEORICO

Este capítulo se centra en las bases teóricas que soportan la construcción del intérprete software para la plataforma PETREL, para el grupo de investigación estabilidad de pozo e igualmente se describe de manera general las teorías bases en la que se enfoca dicho grupo de investigación.

3.1 GENERALIDADES DE LA GEOMECÁNICA

La geomecánica es la encargada de estudiar la respuesta del material geológico a los cambios del entorno físico tales como cambios de temperatura, presión, esfuerzos y químicos. El estudio de estos cambios es importante en el área de la explotación y producción de crudos, gas y otros materiales fósiles que requieren la manipulación del suelo generando, hoyos de grandes profundidades que alteran la estabilidad del terreno por el proceso de perforación y la interacción química de la cara del pozo con los fluidos de perforación o lodo de perforación y la extracción del material de interés. Esto ocasiona una disminución en la saturación de la roca que puede llevar al hundimiento del terreno y otros accidentes medio-ambientales es así como la geomecánica es la encargada de realizar estudios al terreno para garantizar la óptima explotación, esta área basa su conocimiento en los conceptos y teorías de la mecánica de rocas y la mecánica de suelos.

La geomecánica de yacimientos es la encargada del estudio especializado de la interacción de las rocas del yacimiento y el flujo de fluidos entre ellas. Esta área es la encargada de predecir el comportamiento mecánico de la formación a evaluar, de manera que se puedan predecir las posibles causas que llevarían a una inestabilidad del pozo y con estas predicciones poder mitigar, evitar, corregir los problemas ocasionados por la perforación y extracción del material proporcionando así un pozo estable.

Estas inestabilidades se producen por efectos mecánicos, químicos o una combinación de ellas, se presentan a nivel mundial en operaciones como perforación, completamiento, evaluación de formaciones, cementación, registros y producción.

En Colombia el estudio de la estabilidad de pozo es de gran importancia para la perforación de pozos, debido a la alta complejidad estructural de la zona por ende este estudio se tiene presente en la planeación, seguimiento y cierre de un pozo.

Son muchos y frecuentes los problemas de inestabilidad de pozo durante la perforación, que representan consecuencias por mencionar algunas:

- Ensanchamiento del pozo.

- Perdida de circulación.
- Daño de pozos inducidos por los esfuerzos.
- Fallas de pozo inducidas por la perforación.
- Deterioro de casing (recubrimiento).
- Producción de arena.
- Dificultad en toma de registros.

Estas consecuencias repercuten en los altos costos de perforación.

Los análisis de estabilidad involucra aspectos como:

- Toma de núcleos geológicos.
- Ensayos geomecánicos en un laboratorio con muestras del núcleo.
- Elaboración de modelos sobre comportamiento esfuerzo-deformación.
- Pruebas de campo.
- Toma de registros.
- Elaboración correlaciones núcleo-perfil.
- Usar métodos analíticos o numéricos para predecir las condiciones mecánicas y químicas de estabilidad en el pozo.

Los análisis anteriores y simulaciones generadas por software especializado como Petrel ofrecen herramientas muy valiosas para los analistas, una de ellas es el modelo estructural de un campo.

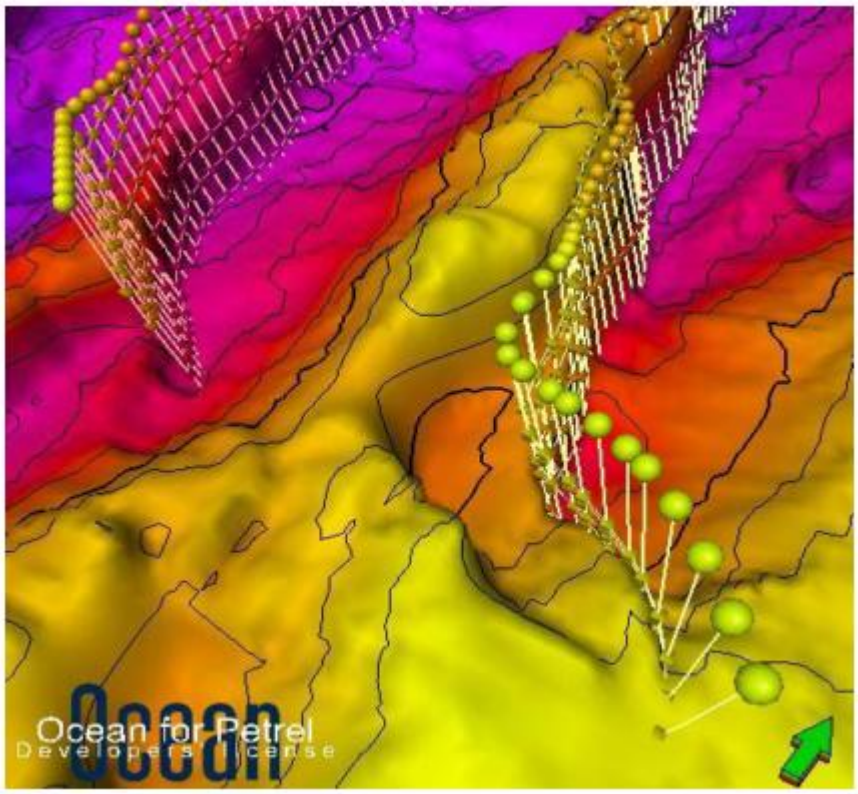


Figura 3 Modelo estructural

Fuente: SCHLUMBERGER, 2010

3.2 BASES PARA EL DESARROLLO DEL PROYECTO

La estandarización de procesos y desarrollos ha permitido a muchas empresas a nivel mundial ampliar su cobertura tecnológica y humana obteniendo mejores resultados con el aumento en la implementación de proyectos de alta calidad.

Tanto Schlumberger con su plataforma software PETREL y el grupo de investigación estabilidad de pozo (GIEP) con su plataforma software G.M.S han desarrollado estándares para sus procesos, permitiendo que futuros desarrollos software puedan ser agregados debido a que estas dos plataformas has sido construidas por teorías modulares en las cuales la plataforma software final ejecuta complementos software o plug-ins utilizando un estándar propio creado por la empresa responsable de cada plataforma. Por su parte Schlumberger tiene su SDK OCEAN, y por otra, el grupo de investigación de estabilidad de pozo (GIEP) tiene el E.D.S o Estándar de Desarrollo Software.

3.2.1 Proceso Unificado de Rational (RUP)

El proceso unificado de rational es uno de los procesos de desarrollo más general que existen actualmente. También se ha identificado como uno de los mejores procesos de desarrollo software que captura las mejores prácticas del conocimiento en ingeniería del software por lo que proporciona a los equipos de desarrollo no solo guías, estándares sino recomendaciones para la construcción de software de calidad.

Al ser una infraestructura flexible de desarrollo software que proporciona practicas recomendadas y probadas además de una arquitectura configurable estableciendo un proceso práctico y fácil de seguir.

Debido a las enormes diferencias entre los proyectos de desarrollo software, cada uno con diferentes prioridades, requerimientos, y tecnologías muy diferentes. Estos proyectos nunca son iguales, sin embargo, en todos los proyectos, se debe minimizar el riesgo, garantizar la predictibilidad de los resultados y entregar software no solo de calidad sino a tiempo.

Las diferentes características que posee el RUP lo hace una herramienta necesaria para el desarrollo de este proyecto.

Principales características

- Forma disciplinaria de asignar tareas y responsabilidades (quién hace qué, cómo y cuándo)
- Pretende implementar las mejores prácticas en Ingeniería de Software
- Desarrollo iterativo
- Administración de requisitos
- Uso de arquitectura basada en componentes
- Control de cambios
- Modelado visual del software
- Verificación de la calidad del software

Ciclo de vida del Proceso Unificado de Rational (RUP)

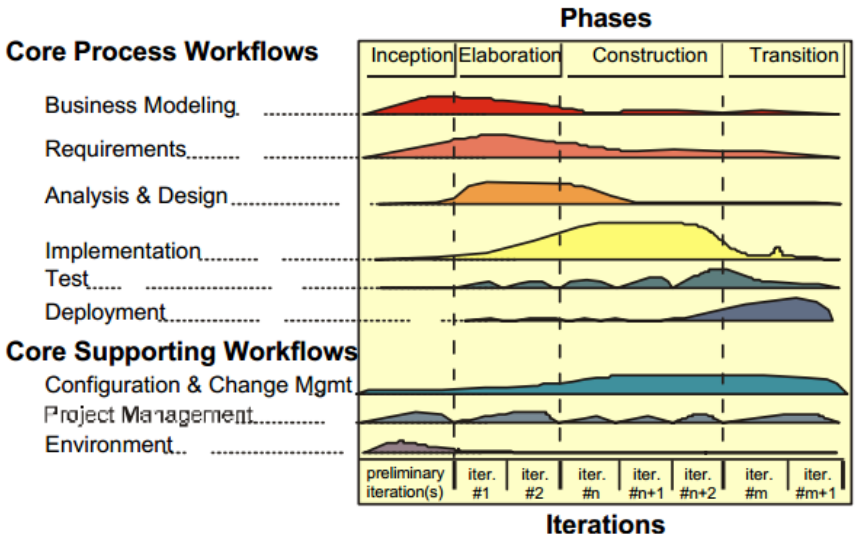


Figura 4 Ciclo de vida RUP

Fuente: RATIONAL SOFTWARE CORPORATION, 1998

El ciclo de vida RUP en una implementación del desarrollo en espiral, creado ensamblando los elementos en secuencias semi-ordenadas. Se ordenan las tareas por medio de fases e iteraciones, este proceso se ve dividido en cuatro fases y un determinado número de iteraciones por fases, las cuales se describen a continuación.

- Inicio: esta fase se enfoca en la comprensión del problema definiendo el alcance del proyecto, la delimitación del ámbito del proyecto, durante esta fase las iteraciones hacen más énfasis en actividades de modelado y de requisitos
- Elaboración: en esta fase se enfoca en la eliminación de los riesgos críticos y el establecimiento de una línea base para el desarrollo de la arquitectura final del software, las iteraciones en esta fase se orientan al desarrollo de la línea base, análisis, requerimientos, arquitectura del producto final
- Construcción: en esta fase se construye el sistema planteado en la línea base y en las dos anteriores fases del ciclo, las iteraciones se centran en la construcción del producto por medio de la selección de algunos casos de uso y se procede a su implementación y pruebas.
- Transición: etapa final del proyecto donde se garantiza que se tiene un producto preparado para su entrega a la comunidad de usuarios.

La fragmentación en pequeños proyectos por medio de las iteraciones hace de este proceso, un proceso muy utilizado en grupos de trabajo que les permiten desarrollar sistemas de calidad en fechas razonables de acuerdo a la dificultad de las tareas o iteraciones.

3.2.2 Estándar IEEE 830

Para el correcto desarrollo de una herramienta software que pueda ser considerada como un producto de calidad, es necesario elaborar una buena estructura que permita construir un producto final de calidad, es necesario elaborar una buena estructura que permita construir el producto final sin retrasos, con errores mínimos o en lo posible sin ellos y con un funcionamiento completo.

Una de las actividades que generalmente un desarrollador de software debe tener presente es la búsqueda para determinar los soportes para la elaboración del proyecto, teniendo presente los aspectos generales que las personas interesadas quiere que haga el sistema a desarrollar.

Una de las formas más prácticas y completas de satisfacer los intereses del cliente es por medio del uso de la especificación de requisitos software (ERS), que a grandes rasgos es una descripción completa del comportamiento del sistema a desarrollar. Involucrando las interacciones que los usuarios tendrán con el software.

Usando este estándar el desarrollador de software describe las estructuras, el contenido deseable y las cualidades del sistema.

La descripción de los requisitos según el estándar IEE 830 – 1998 se dividen en tres:

- Funcionales: son los que el usuario necesita que efectué el software
- No funcionales: estos requisitos son los recursos para que trabaje el sistema de información.
- Empresariales u organizacionales: son el marco contextual en el cual se implantara el sistema para conseguir un objetivó macro.

3.2.3 Patrones de Diseño

Los patrones de diseño son definidos como el esqueleto de soluciones reutilizables simples y elegantes a problemas comunes que aparecen en el desarrollo software.

Estas soluciones han sido basadas en experiencias probadas y documentadas para resolver diversos problemas, donde demostraron un óptimo funcionamiento.

Surgen los patrones de diseño por la necesidad de transmitir conocimientos y experiencias, evitando con ello que los nuevos programadores en accenso sufran los mismos inconvenientes y errores ya vividos por antecesores, estos permiten

que la evolución de los sistemas de programación siga su curso, impidiendo el desgaste de tiempo y recursos en problemas ya resueltos.

Los patrones de diseño han sido descritos con el fin de alcanzar objetivos que pretenden:

- Proporcionar elementos reutilizables en el diseño de herramientas software.
- Evitar el desgaste en buscar soluciones a problemas ya resueltos con anterioridad.
- Estandarizar entre los diseñadores el modo de realizar su labor.
- Facilita el aprendizaje de las nuevas generaciones, permitiendo que estas logren alcanzar metas de mayor dificultad.

Existen numerosos patrones de diseño, por lo cual fue requerido clasificarlos por categorías, lo que facilita su aprendizaje y permite referenciar a patrones similares mediante la localización de la familia a la cual pertenece.

La utilización de los patrones de diseño es de libre decisión por parte del desarrollador de software, pero se recomienda no abusar de estos, debido a que puede generar redundancias, conflictos entre los objetos definidos para cada patrón y quitarles claridad al software, haciéndolo difícil de entender.

Cada patrón es adecuado para ser adaptado a un tipo de problema y en base a esto se clasifican según el tipo de propósito para el que han sido definidos.

- **Patrones creacionales:** para solucionar problemas de creación de instancias y configuración de objetos. Algunos ejemplos son los siguientes:

    - **Builder:** separa la construcción de un objeto complejo de su representación, de forma que el mismo proceso de construcción pueda crear diferentes representaciones.
    - **Factory Method:** define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar. Permite que una clase delegue en sus subclases la creación de objetos.
    - **Prototype:** especifica los tipos de objetos a crear por medio de una instancia prototípica, y crear nuevos objetos copiando este prototipo.
    - **Singlenton:** garantiza que una clase solo tenga una instancia, y proporciona un punto de acceso global a ella.
    - **MVC:** plantea la separación del problema en tres capas: el modelo, la vista y el controlador
- **Patrones estructurales:** separan la interfaz de la implementación. Se ocupan de solucionar problemas de como las clases y objetos se agrupan, para formar estructuras más grandes. Algunos ejemplos son:

    - **Adapter:** convierte la interfaz de una clase en otra distinta que es la que esperan los clientes. Permiten que cooperen clases que de otra manera no podrían por tener interfaces incompatibles.
    - **Bridge:** desvincula una abstracción de su implementación, de manera que ambas puedan variar de forma independiente.
    - **Composite:** combina objetos en estructuras de árbol para representar jerarquías de parte-todo. Permite que los clientes traten de manera uniforme a los objetos individuales y a los compuestos.
    - **Decorator:** añade dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender la funcionalidad.
- **Patrones de comportamiento:** soluciones respecto a la interacción, comunicación y responsabilidades entre las clases y objetos, así como los algoritmos que encapsulan. Algunos ejemplos serian :
    - **Command:** encapsula una petición en un objeto, permitiendo así parametizar a los clientes con distintas peticiones, encolar o llevar un registro de las peticiones y poder deshacer la operaciones.
    - **Interpreter:** dado un lenguaje, define una representación de su gramática junto con un intérprete que usa dicha representación para interpretar las sentencias del lenguaje.
    - **Iterator:** proporciona un modo de acceder secuencialmente a los elementos de un objeto agregado sin exponer su representación interna.
    - **Mediator:** define un objeto que encapsula el como interactúan un conjunto de objetos. Promueve un bajo acoplamiento al evitar que los objetos se refieran unos a otros explícitamente, y permite variar la interacción entre ellos de forma independiente.
    - **Observer:** define una dependencia de uno a muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos.
    - **Strategy:** Define una familia de algoritmos, encapsula uno de ellos y los hace intercambiables. Permite que un algoritmo varié independientemente de los clientes que los usan.
    - **Visitor:** representa una operación sobre los elementos de una estructura de objetos. Permite definir una nueva operación sin cambiar las clases de los elementos sobre los que opera.

Todos los patrones de diseño poseen en su organización estructural cuatro elementos esenciales

- Nombre como referencia significativa del patrón
- Una descripción del área del problema que explica en que casis puede aplicarse el patrón correspondiente.
- Una descripción de la solución del diseño, sus relaciones y las responsabilidades. Por lo general se expresa gráficamente mostrando la relación entre los objetos y las clases de los objetos en la solución

- La declaración de las consecuencias de aplicar el patrón, lo cual permite al diseñador comprender si el patrón en cuestión se aplica de forma efectivamente en la situación particular a tratar.

3.2.3.1 Patrón de diseño Singleton

Garantiza que una clase solo tenga una instancia, y proporciona un punto de acceso global a ella. El patrón singleton se implementa creando en nuestra clase un método que crea una instancia del objeto sólo si todavía no existe alguna. Para asegurar que la clase no puede ser instanciada nuevamente se regula el alcance del constructor (con atributos como protegido o privado).

La instrumentación del patrón puede ser delicada en programas con múltiples hilos de ejecución. Si dos hilos de ejecución intentan crear la instancia al mismo tiempo y esta no existe todavía, sólo uno de ellos debe lograr crear el objeto. La solución clásica para este problema es utilizar exclusión mutua en el método de creación de la clase que implementa el patrón.

Las situaciones más habituales de aplicación de este patrón son aquellas en las que dicha clase controla el acceso a un recurso físico único (como puede ser el ratón o un archivo abierto en modo exclusivo) o cuando cierto tipo de datos debe estar disponible para todos los demás objetos de la aplicación.

El patrón singleton provee una única instancia global gracias a que:

- La propia clase es responsable de crear la única instancia.
- Permite el acceso global a dicha instancia mediante un método de clase.
- Declara el constructor de clase como privado para que no sea instanciable directamente.

Estructura

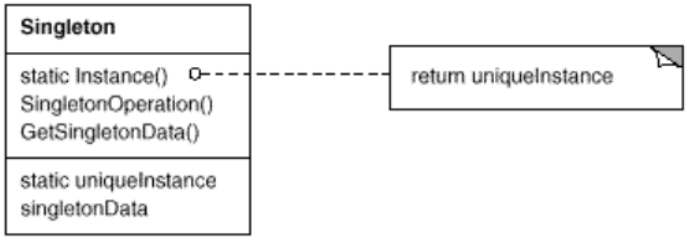


Figura 5 Estructura patrón Singleton

Fuente: GAMMA, 2004

Consecuencias de implementar el patrón de diseño Singleton

- Controla el acceso a su única instancia, debido a que la clase singleton encapsula su instancia e implementa un riguroso control sobre como otras clases acceden a ella.
- Reducción del namespace. El patrón Singleton es una mejora sobre las variables globales. Evita contaminar el namespace de variables globales que almacenar instancias individuales.
- Permite el perfeccionamiento de las operaciones. La clase Singleton puede tener subclases, y es fácil de configurar una aplicación con una instancia de esta clase extendida. Se puede configurar la aplicación, con una instancia de la clase que necesita en tiempo de ejecución.
- Permite un número variable de instancias. El patrón hace que sea fácil de cambiar su comportamiento para permitir más de una instancia de la clase Singleton. Por otra parte, se puede utilizar el mismo método para controlar el número de instancias que utiliza la aplicación.

Implementación en C#

```
namespace EjemploPrueba

{

        public class CentalDatos

        {

                private static CentalDatos datos;

                private CentalDatos()

                {


                }

                public static CentalDatos Datos

                {

                        get
```

```
                {
                        if (datos == null)
                        {
                                datos = new CentalDatos();
                        }
                        return datos;
                }
        }
    }
}
```

3.2.3.2 Patrón de diseño Adapter

Este patrón convierte la interfaz de una clase en otra distinta que es la que esperan los clientes y permite que cooperen clases que de otra manera no podrían por tener interfaces incompatibles.

Se encarga de proporcionar a una clase cliente la interfaz que necesita para trabajar utilizando en su implementación una clase o clases que no cumplen con dicha interfaz.

Aplicabilidad

- desea utilizar una clase existente, y su interfaz no coincide con la que usted necesita.
- Se desea crear una clase reutilizable que coopere con clases no relacionada, es decir, clases que no necesariamente tienen interfaces compatibles.
- (adaptador de objeto único) es necesario utilizar varias subclases ya existentes, pero es poco práctico para adaptar su interfaz por medio de subclases para cada una. Un objeto adaptador puede adaptar la interfaz de su clase padre.

Estructura

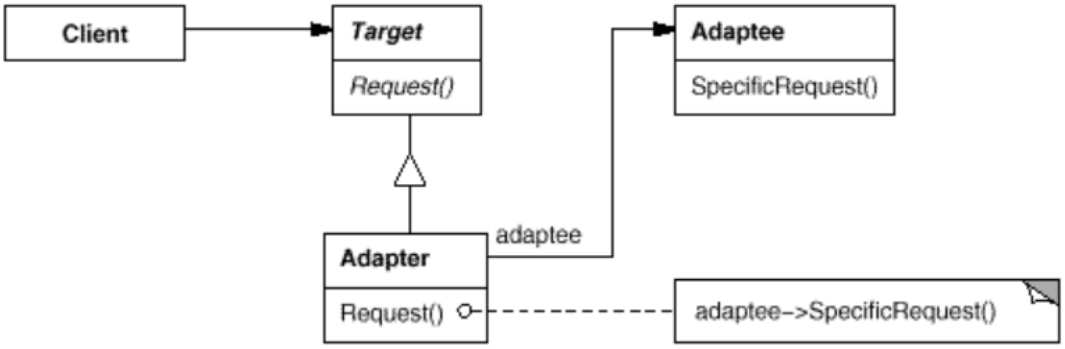


Figura 6 Estructura patrón Adapter

Fuente: GAMMA, 2004

3.2.3.3 Patrón de diseño Bridge

El patrón Bridge, también conocido como Handle/Body, es una técnica usada en programación para desacoplar una abstracción de su implementación, de manera que ambas puedan ser modificadas independientemente sin necesidad de alterar por ello la otra.

Esto es, se desacopla una abstracción de su implementación para que puedan variar independientemente.

Aplicabilidad

- Se desea evitar un enlace permanente entre la abstracción y su implementación. Esto puede ser debido a que la implementación debe ser seleccionada o cambiada en tiempo de ejecución.
- Tanto las abstracciones como sus implementaciones deben ser extensibles por medio de subclases. En este caso, el patrón Bridge permite combinar abstracciones e implementaciones diferentes y extenderlas independientemente.
- Cambios en la implementación de una abstracción no deben impactar en los clientes, es decir, su código no debe tener que ser recompilado.
- Se desea compartir una implementación entre múltiples objetos (quizá usando contadores), y este hecho debe ser escondido a los clientes.

Estructura

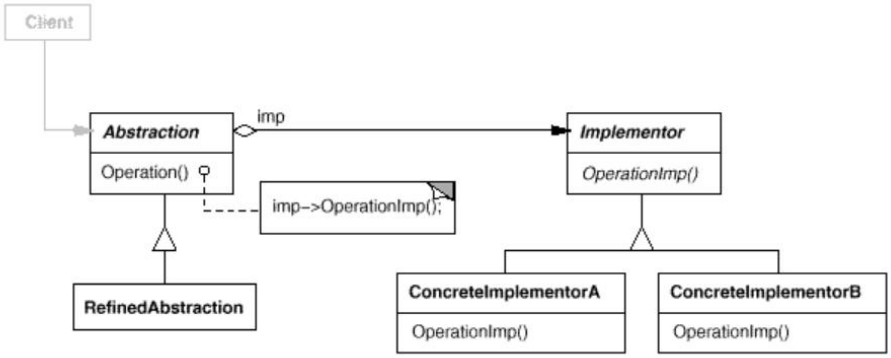


Figura 7 Estructura Patrón Bridge

Fuente: GAMMA, 2004

Consecuencias de implementar el patrón Bridge

- Desacopla interface e implementación: una implementación no es limitada permanentemente a una interface. La implementación de una abstracción puede ser configurada en tiempo de ejecución. Además le es posible a un objeto cambiar su implementación en tiempo de ejecución. Desacoplando Abstraction e Implementor también elimina las dependencias sobre la implementación en tiempo de compilación. Cambiar una clase de implementación no requiere recompilar la clase Abstraction ni sus clientes.
- Mejora la extensibilidad: se puede extender las jerarquías de Abstraction e Implementor independientemente.
- Esconde los detalles de la implementación a los clientes.

3.2.4 Modelo De Maduración De Proyectos De Investigación (MMPI)

Este modelo recopila diferentes elementos para el desarrollo de proyectos de investigación, partiendo desde la concepción de la idea hasta la divulgación de los resultados obtenidos.

La realización del MMPI se da en el grupo de investigación estabilidad de pozo del convenio UIS-ICP y tiene como objetivo optimizar el proceso de investigación direccionando de forma correcta la planeación y ejecución del trabajo a elaborar.

Figura 8 Modelo de Maduración de proyectos de investigación

Fuente: Lilian Mantilla, MANTILLA, 2009

El MMPI establecido en un manual de desarrollo, brinda las bases conceptuales para elaborar un proyecto de investigación, mediante la realización de actividades contempladas en fases progresivas, que fueron diseñadas de tal forma que permiten a los semilleros de los diferentes grupos de investigación que implementen el modelo, ir construyendo de forma adecuada el trabajo.

Las faces que se observan en la anterior imagen se describen a continuación:

Fase preliminar: Se realiza la sesión de lluvia de ideas, sesión de priorización y el árbol causa efecto. Durante esta fase son asignadas las personas responsables al trabajos a elaborar y se determina el tiempo a emplear en encontrar las posibles soluciones al tema correspondiente.

Fase 1: Búsqueda y revisión bibliográfica, definición título del problema, formulación de objetivos descripción justificación del trabajo y delimitación del problema.

Fase 2: Examinar antecedentes del problema, establecer bases teóricas, definición de términos básicos, planteamiento de las posibles hipótesis, determinación de las variables influyentes en el tema.

Fase 3: Elaborar las estructuras de trabajo, Establecer técnicas e instrumentos de recolección de datos y técnicas de procesamiento para análisis de datos, definir cronograma de actividades y calcular el presupuesto a emplear.

Fase 4: Desarrollo de la metodología de la investigación propuesta, de acuerdo al calendario establecido.

Fase final: Publicación de los resultados obtenidos.

4 TECNOLOGÍAS APLICADAS AL PROYECTO

En la elaboración del proyecto de investigación y desarrollo, fue necesario utilizar diferentes tecnologías, las cuales permitieron desarrollar los objetivos que se habían planteado al inicio de este proyecto. La elección de estas tecnologías, se hiso bajo diversos factores, como lo era compatibilidad, documentación, seguridad y capacidad de orientación a objetos y componentes. En el transcurso de este capítulo se describirá las tecnologías que fueron utilizadas para la construcción del BRIDGE-GMS.

4.1 VISUAL ESTUDIO PROFESIONAL 2008

El componente software BRIDGE-GMS, se ha elaborado bajo el entorno de desarrollo de visual studio versión 2008.

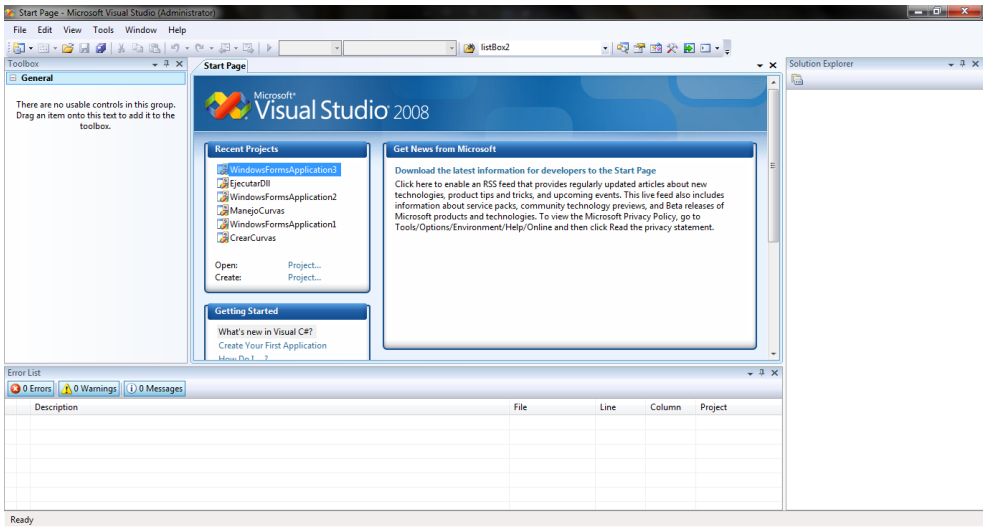


Figura 9 Entorno de desarrollo Visual Studio 2008

Fuente: Autor de este proyecto

Esta edición de Visual Studio, aunque no es gratuita cuenta con más herramientas y lenguajes de programación que la versión Express, tiene todas las fortalezas de los productos Visual Studio y actualmente es uno de los productos comerciales más utilizados alrededor del mundo.

El Framework (.NET 3.5) fue diseñado para aprovechar las ventajas del sistema operativo Windows Vista a través de sus subsistemas Windows Communication Foundation (WCF) y Windows Presentation Foundation (WPF). El primero tiene como objetivo la construcción de aplicaciones orientadas a servicios, mientras que

el último apunta a la creación de interfaces de usuario más dinámicas que las conocidas hasta el momento.

Entre las novedades respecto a las anteriores versiones de Visual Studio se encuentra la escalabilidad y seguridad.

- Mejora de la capacidad de pruebas unitarias, permitiendo ejecutarlas más rápido independientemente de si lo hacen en el entorno IDE o desde la línea de comando y con la herramienta de pruebas de Visual Studio reformada, ofrece un soporte para el diagnóstico y optimización de código.
- Con Visual Studio Tools for Office (VSTO) integrado con Visual Studio 2008 permite desarrollar aplicaciones de alta calidad basadas en la interfaz de usuario (UI) de Office, permitiendo personalizar el escritorio de trabajo y así mejorar la experiencia y productividad de uso de herramientas como lo son Word, Exel, PowerPoint, entre otras.
- Al incorporar el Windows Presentation Foundation (WPF) ahora es posible incorporar estas características tanto en formularios de windows excistentes como en nuevos, permitiendo actualizar el estilo visual de las aplicaciones debido a las mejoras en Microsoft Foundation Class Library (MFC) y Visual C++
- LINQ (Language Integrated Query) es un nuevo conjunto de herramientas diseñado para reducir la complejidad del acceso a bases de datos a través de extensiones para C++ y Visual Basic, así como para Microsoft .NET Framework. Permite filtrar, enumerar, y crear proyecciones de muchos tipos y colecciones de datos utilizando todos la misma sintaxis, prescindiendo del uso de lenguajes especializados.
- 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).

El entorno de desarrollo integrado (IDE) delk Visual Studio versión 2008, presenta un ambiente amigable para el usuario, facilitándole el entendimiento de las diferentes funciones, características y detalles visuales, este entorno está compuesto en su parte más externa por un compilador, un depurador, un constructor de interfaz gráfica (UI) y un editor de código.

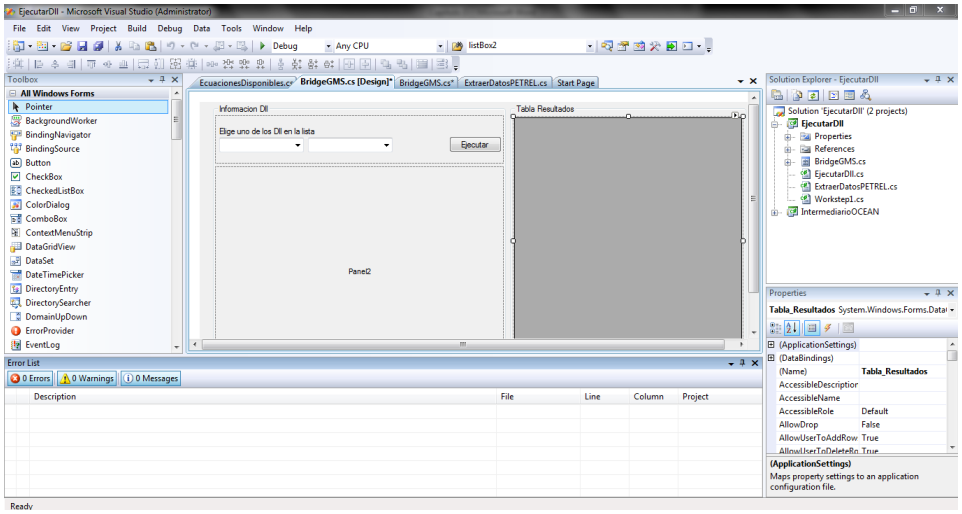


Figura 10 interfaz gráfica UI Visual Studio 2008

Fuente: Autor de este proyecto

4.2 LENGUAJE DE PROGRAMACIÓN: VISUAL C#

La creación de visual C# o Visual C Sharp es un lenguaje creado por Microsoft para su plataforma .NET, con el fin de aprovechar al máximo sus características.

La sintaxis y estructura que se usa en este lenguaje es similar a la de otros lenguajes de programación orientado a objetos como lo es Java y C++, gracias a estas similitudes la migración de código escrito en estos tipo de lenguajes es posible además de facilitar el aprendizaje para los desarrolladores nuevos en este lenguaje, algunas de las características comunes entre los lenguajes de la familia C son:

- Todas las declaraciones finales con un punto y coma
- Los bloques de código se escriben entre llaves
- El lenguaje es sensible a mayúsculas

Se listaran a continuación solo algunas de las principales características del lenguaje de programación C# y aunque no todas son claramente propias del lenguaje como tal sino de la plataforma .NET tienen aplicación directa con este lenguaje.

- Sencillez: el código escrito en C# es auto contenido, no necesita de ficheros adicionales a los de su propia fuente. Sus tipos de datos básicos es fijo e

independiente del compilador, sistema operativo o maquina en que se compile, estas características facilitan la portabilidad de código.

- Modernidad: Con su vieja trayectoria la familia C ha ido desarrollando elementos a través de los años de experiencia por lo tanto C# incorpora elementos como:
  - Un tipo básico decimal que permita realizar operaciones de alta precisión
  - La inclusión de la instrucción foreach que permita recorrer colecciones de objetos permitiendo aplicarlos a tipos definidos por el usuario
  - La inclusión del tipo básico string para representar cadenas
  - La distinción de un tipo bool específico para representar valores lógicos
- Orientación a objetos: C# mantiene el enfoque orientado a objetos y se caracteriza por no admitir funciones ni variables globales sino que todo el código y datos han de definirse por medio de tipos de datos, lo que reduce problemas por conflictos de nombres y facilita la legibilidad del código
- Orientación a componentes: la sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones complejas
- Gestión automática de memoria el lenguaje .NET tiene a su disposición el recolector de basura CLR. Esto tiene el como efecto el no incluir instrucciones que destruyan los objetos creados por la ejecución del código
- Seguridad de tipos: C# asegura que los accesos a tipos de datos, se realicen correctamente, lo que evita que se produzcan errores difíciles de detectar por acceso a memoria, como lo es el acceso a un objeto no existente.
- Eficiente: en c todo código incluye numerosas restricciones para asegurar su seguridad y no permite el uso de punteros. Sin embargo, y a diferencia de otros lenguajes de programación, es posible saltarse dichas restricciones manipulando objetos a través de punteros. Para ello basta marcar regiones de código como inseguras y podrán usarse en ellas punteros de forma similar a como se hace en C++, lo que puede resultar vital para situaciones donde se necesite una eficiencia y velocidad procesamientos muy grandes.
- Compatible: la migración de programadores de sus lenguajes actuales a C# es una de las prioridades de Microsoft, por este motivo C# no solo mantiene una sintaxis muy similar a C, C++ o JAVA que permite incluir directamente el código escrito en estos lenguajes en C#, el CLR también ofrece, a través de los llamados platform invocetion services (PInvoke), la posibilidad de acceder a código nativo escrito como funciones sueltas no orientadas a objetos tales como DLLs de las API Win32.

4.3 PLATAFORMA PETREL

Petrel es una plataforma software propiedad de Schlumberger, esta plataforma es destinada a ordenadores con sistema operativo Windows y es destinada al análisis de la totalidad de los datos del yacimiento de campos de explotación petrolera.

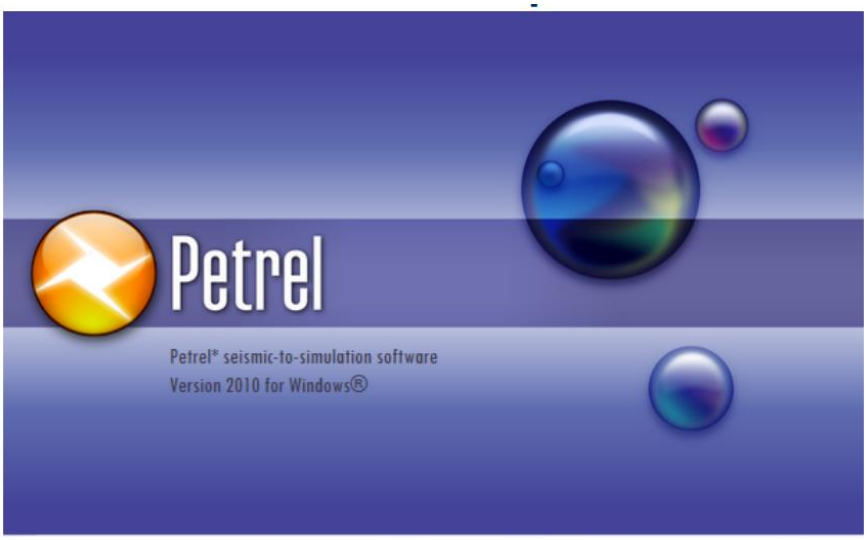


Figura 11 Logo Petrel

Fuente: SCHLUMBERGER, 2010

Esta plataforma permite al usuario interpretar los datos sísmicos por medio de correlaciones, construir modelos de yacimientos adecuados para la simulación, presentar y visualizar los resultados de la simulación, cálculo de volúmenes, elaborar mapas y diseñar estrategias de desarrollo para maximizar la explotación de yacimientos. En base a estos objetivos fue que Schlumberger abordo la necesidad de una única aplicación capaz de soportar la "sísmica y simulación", reduciendo la necesidad de una multitud de herramientas altamente especializadas.

Versión Petrel 2007.1

Esta versión ofrece una amplia gama de aplicaciones para la sísmica y la simulación otorgando una mayor capacidad en los flujos de trabajo en el campo de la explotación de hidrocarburos. Petrel 2007.1 ahora maneja a gran escala los estudios sísmicos 2D y líneas regionales de escala. Fractura de modelado y capacidades de doble porosidad además de apoyar los flujos de trabajo no convencionales de gas. Petrel 2007.1 establece la conexión de plataforma software al implementar el modelado por componentes gracias a esto se establece el SDK OCEAN, que permite a terceras partes, como las universidades, las

empresas petroleras ajenas a Schlumberger generar productos tecnológicos que se puedan usar directamente en PETREL.

Versión Petrel 2008.1

Lanzado en marzo de 2008. Las principales mejoras incluyen soporte para las fracturas hidráulicas, modelado de sector, procesos multi hilos, y mejoras en los flujos de trabajo Auto seguimiento sísmicos 3D. La representación en volumen y un módulo de extracción ahora permite a los usuarios mezclar interactivamente múltiples volúmenes sísmicos, aislar las áreas de interés y luego extraer instantáneamente lo que se ve en un objeto 3D.

Versión Petrel 2009.1

Lanzado en febrero de 2009, esta es la primera versión de Petrel en ser completamente de 64 bits y para ejecutarse en Microsoft Windows Vista de 64 bits. Esto trae grandes beneficios de rendimiento para los usuarios especialmente los que trabajan en la exploración o con grandes volúmenes sísmicos y de modelos geológicos. Otras mejoras incluyen un nuevo tipo de Inversión sísmica llamada Inversión genética basada en una relación no lineal. Geoestadística multipunto entre muchas otras mejoras.

Versión Petrel 2010.1

Lanzado en mayo de 2010. Las principales mejoras incluyen un nuevo flujo de trabajo de modelado estructural que permite al usuario construir modelos estructurales de agua. Otra de las mejoras incluye el modelado de fractura, geoestadística multipunto, y los flujos de trabajo de interpretación de volumen. Esta versión también integra PetroMod y sistemas de modelado avanzado de RDR módulo de análisis estructural y de fallos que permite un enfoque integrado para el análisis de exploración. Este lanzamiento coincidió con el lanzamiento de la tienda de Ocean en línea donde los usuarios pueden descargar plug-ins para Petrel.

4.3.1 SDK OCEAN

Ocean es un marco de desarrollo de aplicaciones con capacidad de trabajar en todos los dominios de datos. Proporciona servicios, componentes, y una interfaz gráfica de usuario común que permite eficiente integración entre aplicaciones. Esto permite que los desarrolladores de aplicaciones interactúen directamente con el modelo central de datos de petrel.

La arquitectura de Ocean

La arquitectura Ocean consta de tres niveles: el núcleo, el de Servicios, y la familia de productos. Para las aplicaciones centradas en el modelo, la familia de

productos seria Petrel. Los módulos Ocean son gestionados por La capa de núcleo. Ellos interactúan con todos los niveles de la estructura

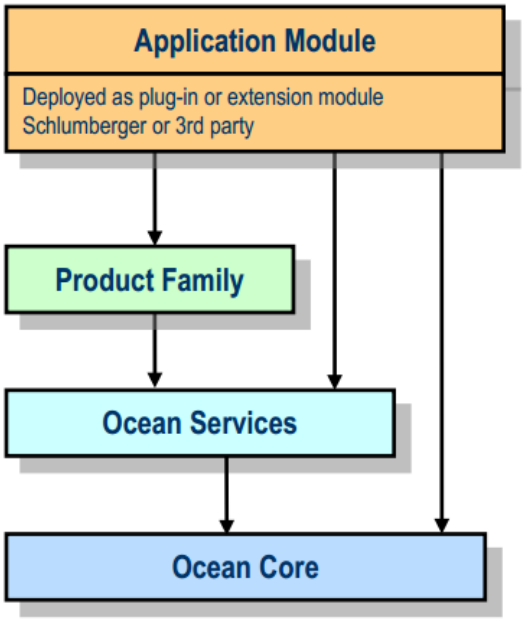


Figura 12 Arquitectura Ocean

Fuente: SCHLUMBERGER 2010

El núcleo Ocean tiene el papel de interpretar la infraestructura básica, gestiona los módulos y los servicios Ocean, tanto los servicios pre-cargados por la familia de productos, como los servicios que se van a añadir de forma dinámica a través de la interfaz de programación de aplicaciones (API). El núcleo Ocean gestiona las fuentes de datos proporcionados por el producto de la familia o cualquier otra fuente de datos externa que podría definirse por cualquier módulo. También lleva a cabo la gestión de eventos y mensajes de registro básicos. Los servicios ofrecidos por Ocean son un conjunto de utilidades de aplicaciones independientes. Son módulos que se benefician de ser estandarizados en familias de productos. Un ejemplo es el Servicio de coordenadas - una utilidad para convertir entre coordenadas geodésicas y proyección de sistemas.

La familia de productos es el anfitrión para aplicaciones Ocean y es el entorno en el que el módulo de Ocean necesita para funcionar. La familia de productos ofrece:

- Los objetos de dominio y su fuente de datos
- El entorno gráfico en el que las aplicaciones visualizan sus datos
- Un aspecto común para todas las interfaces de usuario de aplicaciones

4.4 PLATAFORMA G.M.S.

La plataforma de modelado geomecánico (G.M.S.) fue desarrollada con el objetivo de disponer de una herramienta software propia dentro del grupo de investigación estabilidad de pozo (GIEP), que permitiera acoplar los diferentes programas y aplicaciones que han sido elaboradas dentro de este grupo de investigación, para la validación de las metodologías de estudio, surgió la necesidad de crear un sistema que tuviera la facilidad de ir aumentando sus funcionalidades cada vez que se quisiera implementar una nueva metodología.

Figura 13 Logo G.M.S.

Fuente: Rojas, 2010

La plataforma de modelado geomecánico (G.M.S.) fue desarrollada bajo el entorno de programación de Visual Studio Express Edition 2010 y el lenguaje de programación C#, la cual fue diseñada con una arquitectura centrada en componentes, que permite la utilización de sus funciones en otras aplicaciones más pequeñas, tal que estas pueden ser implementadas y utilizadas de forma eficiente dentro del entorno de la herramienta software.

Esta plataforma permite el acople de subprogramas que le añaden nuevas funcionalidades, haciendo que el software aumente su utilidad. Debido a este tipo de arquitectura aquellos programas que quieran ser anexados deben cumplir con ciertas características establecidas en el estándar de desarrollo software (E.D.S).

5 DESARROLLO DEL COMPLEMENTO BRIDGE-GMS

Esta sección está dedicada a el complemento software Bridge-GMS, partiendo de una descripción general del software y continuando con una especificación más detallada de su constitución, donde se mencionan cada uno de los subsistemas que hacen posible que Bridge-GMS cumpla a cabalidad los objetivos para los que fue desarrollada.

5.1 DESCRIPCIÓN DEL COMPONENTE

Con el objetivo de conocer y mejorar las necesidades presentes en la industria de los hidrocarburos referente a los análisis de los datos de pozo, e incluir el desarrollo de algunas de las metodologías ya implementadas en la plataforma software G.M.S. al escritorio de trabajo de la plataforma PETREL, se desarrolla un componente software ajustado a estas necesidades, que integrara las bibliotecas de enlace dinámico (.dll) pertenecientes al G.M.S. en una interfaz gráfica de usuario (UI) desarrollado en el lenguaje de programación C# y apoyado por el SDK Ocean para Petrel.

Figura 14 Logo BridgeGMS

Fuente: Autor de este proyecto

5.2 ESTRUCTURA DEL COMPONENTE

Para cumplir con las necesidades de este proyecto es preciso analizar las dos plataformas software, de esta forma se obtendrán las bases necesarias para poder

integrar un módulo en Petrel que se comporte como puente hacia las bibliotecas de enlace dinámico del G.M.S.

PETREL y G.M.S cuentan con estándares que permiten agregar nuevos componentes software, siempre y cuando cumplan con los requerimientos especificados y documentados en el SDK Ocean y el E.D.S. respectivamente. Aunque estas plataformas cumplen a cabalidad con el funcionamiento por el cual fueron creadas, el uso continuo y simultaneo de estas disminuye en una gran medida las capacidades individuales de cada una de ellas, teniendo que compartir recursos hardware de los ordenadores en la que se encuentran en funcionamiento, y produciendo retardos en la entrega de resultados además de errores al tener que pasar a mano los datos que se encuentran en una plataforma a la otra.

Es por eso que surge la necesidad de integrar estas dos plataformas, no sólo para conocer la capacidad que ofrece el SDK Ocean, sino que también para evitar todos los problemas que surgen a la hora de utilizar las bibliotecas de enlace dinámico (.dll) en la plataforma G.M.S mientras se hace una evaluación de campo en Petrel. Teniendo en cuenta estas necesidades, es necesario determinar las capacidades de comunicación y bajo qué metodología se va a comunicar estas dos plataformas.

5.3 ESTRUCTURA DE COMUNICACIÓN CON PETREL

Petrel, al ser una plataforma software, su programación está basada en componentes por lo tanto posee un estándar que permite, una vez se implementa que la plataforma acepte ese componente como suyo y lo utilice directamente sobre su escritorio de trabajo para así aprovecha los datos almacenados en la base de datos del proyecto Petrel, en este orden de ideas el SDK Ocean sería el estándar requerido para que petrel acepte un componente como suyo, los métodos y bibliotecas de enlace dinámico (.dll) necesarias para que un componente pueda ser usado en Petrel, están disponibles una vez se instala el SDK Ocean, a continuación se hará una breve descripción de estos métodos.

Petrel por medio del SDK Ocean ofrece acceso a los siguientes datos de dominio:

- Pozos (Well): para aplicaciones petrofísicas y geológicas.
- Sismica (Seismic): para aplicaciones geofísicas.
- Shapes: para modelos estructurales.
- pilarGrid: para geo-modelado
- simulación (Simulation): para reservoir evaluations.

Además del acceso a estos datos de dominio Ocean ofrece la capacidad de extender la interfaz de usuario Petrel por medio de su API estas extensiones son las siguientes:

- Ventanas: Puede adherirse cualquier número de ventanas tanto nuevas como por defecto
- Renderers: adhiere renderers del dominio de objetos, nativos o propios
- Interacciones: adhiere configuraciones propias que integran diferentes tipos de ventanas.
- Menús y barras de herramientas: adhiere nuevos menús o barras de herramientas o extiende los ya existentes
- Exploradores de proyecto petrel: adhiere o modifica los arboles de exploración de datos de un proyecto petrel

Específicamente esta Proyecto trabaja en base a los datos de dominio albergados en los pozos de un proyecto Petrel en la sección de datos de entrada o Input:

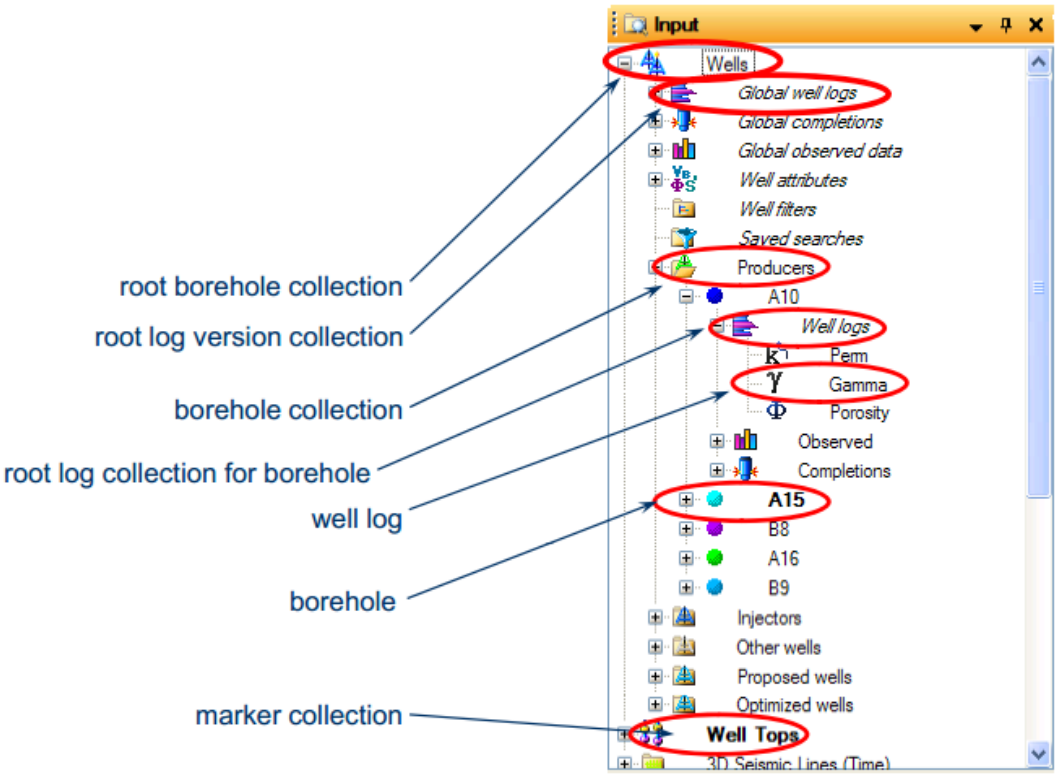


Figura 15 Árbol de trabajo Petrel

Fuente: SCHLUMBERGER, 2010

- **Root borehole collection:** es la raíz del proyecto en ella se encuentra de manera general toda la información necesaria que se va a utilizar para ejecutar las bibliotecas de enlace dinámico G.M.S.
- **Borehole Collection:** en esta agrupación de datos se encuentran los datos generales de los pozos del campo específico.
- **Root log collection for borehole:** en esta agrupación de datos se encuentran todos los registros actuales que posee un pozo en específico
- **Well log:** son los datos específicos de un registro en un pozo.
- **Borehole:** cada uno de los pozos individuales de un campo.

5.3.1 Estructura Base de un módulo Ocean

Un módulo Ocean es una extensión de una familia de productos, los programadores de aplicaciones crean módulos para satisfacer una necesidad, la estructura básica de un módulo Ocean para petrel es la siguiente:

La clase modulo que se desea implementar deberá implementar la interfaz IModule perteneciente a la biblioteca de enlace dinámico Slb.Ocean.Core, la estructura definida de la interfaz IModule y su ciclo de vida se describirá a continuación:

La interfaz IModule define cinco métodos que son las fases del ciclo de vida de un módulo en Petrel

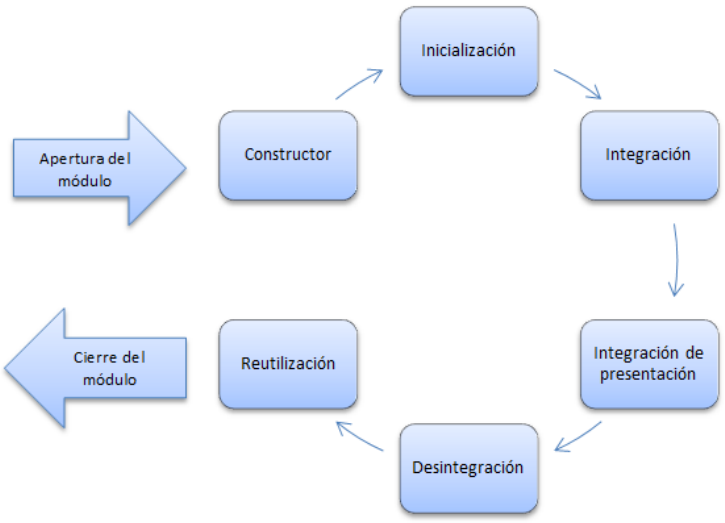


Figura 16 Ciclo de vida Modulo Ocean

Fuente: RAMIREZ, 2012

- **Default Constructor:** durante la apertura del módulo, el núcleo (Core) carga los módulos pertenecientes a la familia de productos que han sido definidos por el archivo de configuración, Ocean llama al constructor por defecto para instanciar cada módulo. Este constructor puede usarse para verificar la existencia o ausencia de licencias campos protegidos a adquirir cualquier recurso que el necesite para su correcto funcionamiento.
- **inicialization:** la segunda fase del ciclo de vida de un módulo Ocean es la ejecución del método initialize, este método tiene como fin o propósito el registro de los servicios ofrecidos por el modulo usando la clase SeviceLocator, cuando este método finaliza quedaran registrados los servicios disponibles para usar por todos los módulos Ocean.
- **Integrate:** El método de Integración es el primer punto en el que un módulo puede consumir servicios registrados en Ocean. Esto es porque todos los módulos han pasado por la fase de inicialización, donde sus servicios se ha registrado en Ocean.
- **IntegratePresentation:** El método de IntegratePresentation es la fase del ciclo de vida de IModule en el que los componentes de interfaz de usuario (menús, barras de herramientas, menús contextuales, ventanas, etc.) se añaden a la familia de productos.
- **Disintegrate:** Cuando la familia de productos empieza su proceso de cierre, el método disintegrate de IModule es llamado disintegrate tiene la responsabilidad de la limpieza de todos los elementos de presentación instaladas por el módulo.
- **Dispose:** proviene del patrón IDisposable y debe ser implementado como parte de la aplicación IModule. El propósito de Dispose es liberar los recursos no administrados y licencias adquiridas por el módulo.

5.4 PRINCIPALES REQUERIMIENTOS DEL SISTEMA

5.4.1 Análisis de requisitos (Adquisición y tratamiento de datos PETREL).

Introducción:

Propósito: Crear un módulo que permita adquirir, tratar y guardar información de los registros (Logs) perteneciente a un pozo especifico seleccionado de un campo actualmente activo en el escritorio de trabajo PETREL, una vez se obtienen esa información se le realizara una conversión al tipo de dato que acepta el G.M.S., además cuando G.M.S. suministre los resultados estos datos serán pasados de nuevo a información que se almacenara en la base de datos del proyecto actualmente activo en PETREL.

Ámbito del sistema: Este módulo se llamara ExtraerDatosPetrel, y será el encargado de comunicarse con petrel extraer la información de los pozos y

registros seleccionados por el usuario y enviarle esta información a la central de datos, además una vez otros se obtienen los resultados será el encargado de convertir los datos en información disponible en PETREL, el objetivo principal del módulo será de solo extracción y almacenamiento de información en PETREL, esto beneficiara en gran medida a la organización y mantenimiento de código además permitirá aislar errores más rápido al ser un módulo especializado en una sola tarea.

Descripción general:

Perspectiva del producto: Este producto será una parte importante del producto final debido a que se encargara de uno de los dos principales objetivos de componente software el cual es comunicarse con PETREL y así poder acceder a su información. Como modulo del sistema completo BridgeGMS, será el encargado de suministrarle a la central de datos la información necesaria para que las bibliotecas de enlace dinámico del G.M.S se puedan ejecutar.

Funciones del producto: como funciones principales serán suministrarle a la central de datos la información del proyecto actualmente activo en petrel, cuantos pozos tiene dicho proyecto, y una vez el actor selecciones un pozo los registros disponibles de ese pozo, finalmente agregara un nuevo registro a un pozo con los datos de salida que le entregara la central de datos producto de la ejecución de la biblioteca de enlace dinámico del G.M.S.

Características de los usuarios: debido a la enorme complejidad de las pruebas los usuarios deberán estar muy relacionados con la metodología de las librerías de enlace dinámico del G.M.S y el ambiente de trabajo en PETREL, además de conocer las nociones básicas explicadas en la capacitación del uso del componente BridgeGMS.

Restricciones:

- Por políticas empresariales se deberá tener la licencia activa de PETREL sin esta licencia el componente no funcionara.
- Los requisitos hardware de este módulo son los mismos que los requisitos de PETREL ver anexo A.
- Debido a la comunicación con una plataforma completamente ajena a PETREL se debe tener las licencias pertinentes del G.M.S.

Requisitos específicos:

Funcionales:

- El actor usara una lista de despliegue para seleccionar el pozo con el que desean trabajar, los pozos disponibles serán los que se encuentren registrados en el campo del proyecto activo en PETREL.

- El actor usara las listas de despliegue con los registros contenidos en el pozo seleccionado, de acuerdo con la información brindada por la central de datos, la cual tiene almacenado que biblioteca de enlace dinámico del G.M.S. se va a usar y cuáles son sus entradas.
- El actor elegirá el paso con el cual desea se ejecute la prueba este paso es la distancia en pies (ft) de la profundidad del pozo, por defecto este paso es cada 0.5 pies (ft)
- El actor podrá elegir que resultados son los que desea crear en PETREL una vez a ejecutado la biblioteca de enlace dinámico del G.M.S. por medio de una lista de chequeo podrá seleccionar que registros crear, y por medio de un cuadro de texto le dará nombre a ese registro

No funcionales:

- El sistema deberá mostrar la información de las listas de pozos y registros que se crean una vez el actor selecciona una biblioteca de enlace dinámico G.M.S.
- Si la licencia petrel no existe o no esta activa el sistema no se podrá abrir.
- Si no hay un proyecto petrel actualmente activo el sistema no mostrara datos de pozos ni registros.
- El sistema deberá mostrar los datos de los registros seleccionados por el actor, de esta forma el actor podrá si lo desea observar errores en los datos que desea ingresar en la prueba antes de iniciar dicho proceso.
- El sistema deberá mostrarle a l usuario por medio de una barra de proceso el estado actual de dicha prueba, esto debido al gran tiempo de ejecución de las pruebas esta visualización es importante.
- El sistema deberá mostrar las salidas del proceso una vez ha finalizado junto con los datos de entrada que se ingresaron.
- El sistema debe mostrar un mensaje de registros creados exitosamente.

5.4.2 Análisis de requisitos (interprete de las bibliotecas de enlace dinámico G.M.S.).

Introducción:

Propósito: crear un módulo que permita interpretar las bibliotecas de enlace dinámico G.M.S en la plataforma software PETREL y así poder ejecutar sus procesos directamente sin necesidad de migrar información de una plataforma a la otra.

Ámbito del sistema: Este módulo se llamara IntermediarioOCEAN estará encargado de establecer la comunicación con G.M.S. y crear los objetos necesarios para la ejecución de las bibliotecas de enlace dinámico G.M.S. además de comunicarse con la central de datos y así crear los objetos necesarios para

almacenar la información de entrada y salida de dichas bibliotecas. El objetivo principal del módulo será solo la conexión e interpretación de las bibliotecas de enlace dinámico G.M.S. esto beneficiara en gran medida a la organización y mantenimiento de código además permitirá aislar errores más rápido al ser un módulo especializado en una sola tarea.

Descripción general:

Perspectiva del producto: Este producto será una parte importante del producto final debido a que se encargara de uno de los dos principales objetivos de componente software el cual es comunicarse con G.M.S. y así poder ejecutar sus bibliotecas de enlace dinámico G.M.S. directamente en el escritorio de trabajo PETREL. Como modulo del sistema completo BridgeGMS, será el encargado de suministrarle a la central de datos la información necesaria para que los nuevos registros puedan ser creados en PETREL.

Funciones del producto: Como funciones principales serán suministrarle a la central de datos la información de cuantas bibliotecas de enlace dinámico cumplen con todos los requisitos necesarios para que se puedan ejecutar en BridgeGMS, además de este será la encargada de suministrar la información de que datos de entrada requiere y que datos de salida producirá la ejecución de esta biblioteca, igualmente creara los objetos necesarios para la ejecución de estas bibliotecas y almacenara los resultados en la central de datos para su traspaso a PETREL

Características de los usuarios: debido a la enorme complejidad de las pruebas los usuarios deberán estar muy relacionados con la metodología de las librerías de enlace dinámico del G.M.S y el ambiente de trabajo en PETREL, además de conocer las nociones básicas explicadas en la capacitación del uso del componente BridgeGMS.

Restricciones:

- Por políticas legales se deberá tener la licencia activa de G.M.S. sin esta licencia el componente no funcionara.
- Los requisitos hardware de este módulo son los mismos que los requisitos de PETREL ver anexo A.
- Debido a la comunicación con una plataforma completamente ajena a G.M.S. se debe tener las licencias pertinentes de PETREL.

Requisitos específicos:

Funcionales:

- El actor usara una lista de despliegue para seleccionar la biblioteca de enlace dinámico en la que desee trabajar, estas bibliotecas serán las que se encuentren instaladas en ese momento en la plataforma G.M.S.

- El actor una vez finaliza la interpretación de las bibliotecas de enlace dinámico G.M.S. podrá seleccionar que salidas desea convertir a información en PETREL por medio de una lista de chequeo.

No funcionales:

- El sistema deberá verificar la existencia de G.M.S. por medio de una búsqueda a su llave de registro.
- El sistema deberá verificar la existencia de la base de datos del sistema G.M.S. para así poder acceder a la información de las bibliotecas existentes en la plataforma.
- El sistema deberá validar que la licencia G.M.S. se encuentre vigente y así evitar perjudicar la autoría del G.M.S. y sus módulos.
- El sistema deberá crear un enlace con el núcleo del sistema G.M.S. y así poder ejecutar sus métodos sin la necesidad de ejecutar directamente G.M.S.
- El sistema deberá validad cada una de las bibliotecas existentes en G.M.S. y mostrar solo las disponibles para la ejecución de ecuaciones.
- El sistema deberá crear un enlace con estas ecuaciones dinámicas (dlls) por medio de núcleo del sistema para así poder ejecutarlas sin la necesidad de tener abierto la plataforma G.M.S.
- El sistema deberá ejecutar las ecuaciones y comunicarse con la central de datos para entregarle los resultados de dicha ejecución.

5.5 MODELADO UML

5.5.1 Diagramas de Casos de uso principales del sistema

Se definen los casos de uso como la descripción de las iteraciones que se producen entre los usuarios y el sistema. Cuando el usuario usa el sistema para llevar a cabo una tarea específica, se define los procedimientos del sistema independientemente de la implementación, y se representa mediante una elipse con el nombre en su interior. El nombre correspondiente debe reflejar la tarea específica que el usuario desea llevar a cabo usando el sistema y las acciones que acompañan dicha tarea, estos nombres están acompañados mediante un verbo que indica la acción a efectuar.

Se ha seleccionado un proceso característico que la herramienta software ejecuta y permite entender las acciones realizadas por el software a partir de las decisiones de usuario.

Caso de usos Seleccionar Entradas

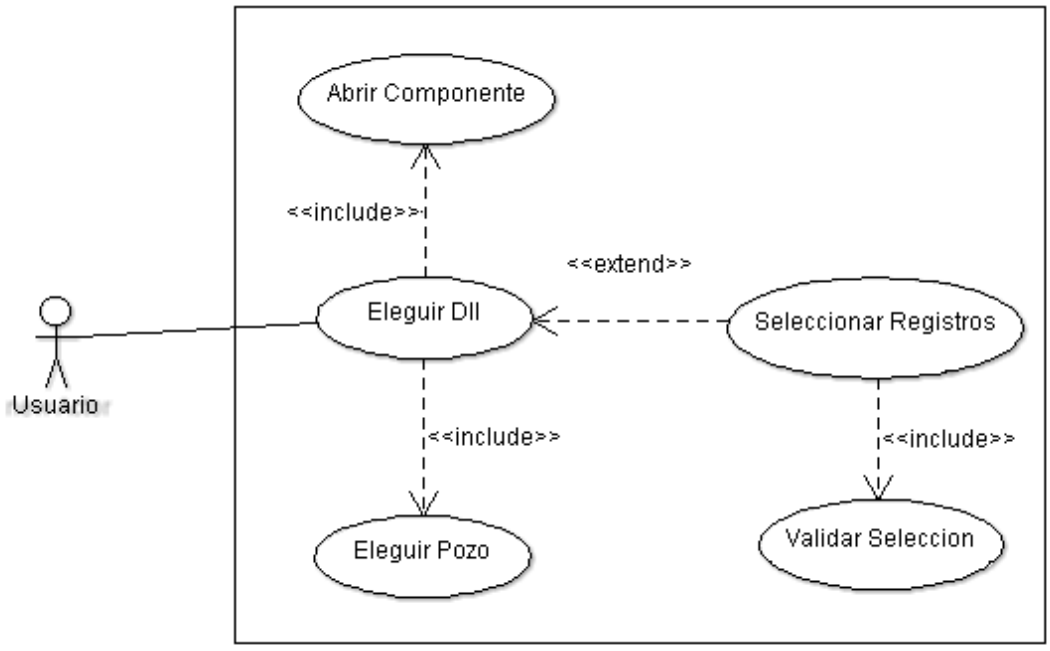


Figura 17 Caso de usos Seleccionar Entradas

Fuente: Autor de este proyecto

Caso de uso	**Descripción**
Elegir biblioteca de enlace dinámico	El usuario selecciona de una lista una de las bibliotecas de enlace dinámico disponibles del G.M.S. ya incluidas de manera automática por el intérprete de bibliotecas, el sistema valida esta biblioteca y construye las listas de registros de entrada dinámicamente requeridas para la ejecución del dll
Elegir pozo	El usuario selecciona de una lista uno de los pozos disponibles en el proyecto Petrel actualmente en ejecución incluida automáticamente por el intérprete de datos Petrel, el sistema valida este pozo y llena las listas de registros con los registros disponibles en ese pozo

Seleccionar Registros (Logs)	El usuario selecciona un registro específico para cada una de las listas creadas dinámicamente por el sistema o elige la opción de otro he ingresa el valor constante de ese registro
Validar selección	El usuario válida la selección que hiso de registros visualizando las entradas y sus datos una vez se encuentra conforme con su selección procede a ejecutar la dll seleccionada.

Caso de uso Ejecutar prueba

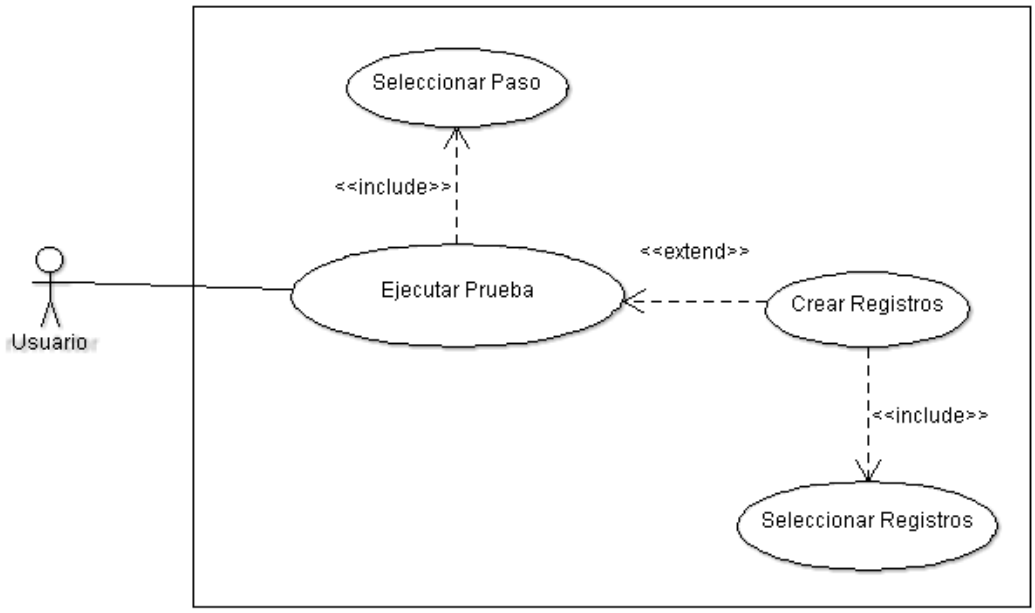


Figura 18 Caso de uso Ejecutar prueba

Fuente: Autor de este proyecto

Caso de uso	Descripción
Ejecutar Prueba	El usuario una vez a seleccionado los registros adecuados para la ejecución de la biblioteca de enlace dinámico procede a ejecutar la dll.
Seleccionar paso	El usuario digita el paso o intervalo con el que desea se filtre los datos este intervalo es aplicable a MD y solo se evaluaran los datos pertenecientes a este intervalos, como intervalo por defecto seria 0.5 pies
Crear registros	Una vez el usuario ha seleccionado el registro o los registros que desea crear en Petrel, se procede a realizar la creación de estos en la base de datos por medio de un botón que anida todo el procedimiento de creación de un WellLog en Petrel.
Seleccionar Registros	El usuario selecciona los registros que desea crear en Petrel por medio de cuadros de chequeo visualizados a lado de cada nombre de registro de salida

5.5.2 Diagramas de Secuencia principales del sistema

El diagrama de secuencia es un tipo de diagrama usado para modelar interacción entre objetos en un sistema según UML. Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso. Un diagrama de secuencia muestra los objetos que intervienen en el escenario con líneas discontinuas verticales, y los mensajes pasados entre los objetos como flechas horizontales.

Diagrama de secuencia para Caso de Uso Seleccionar Entradas

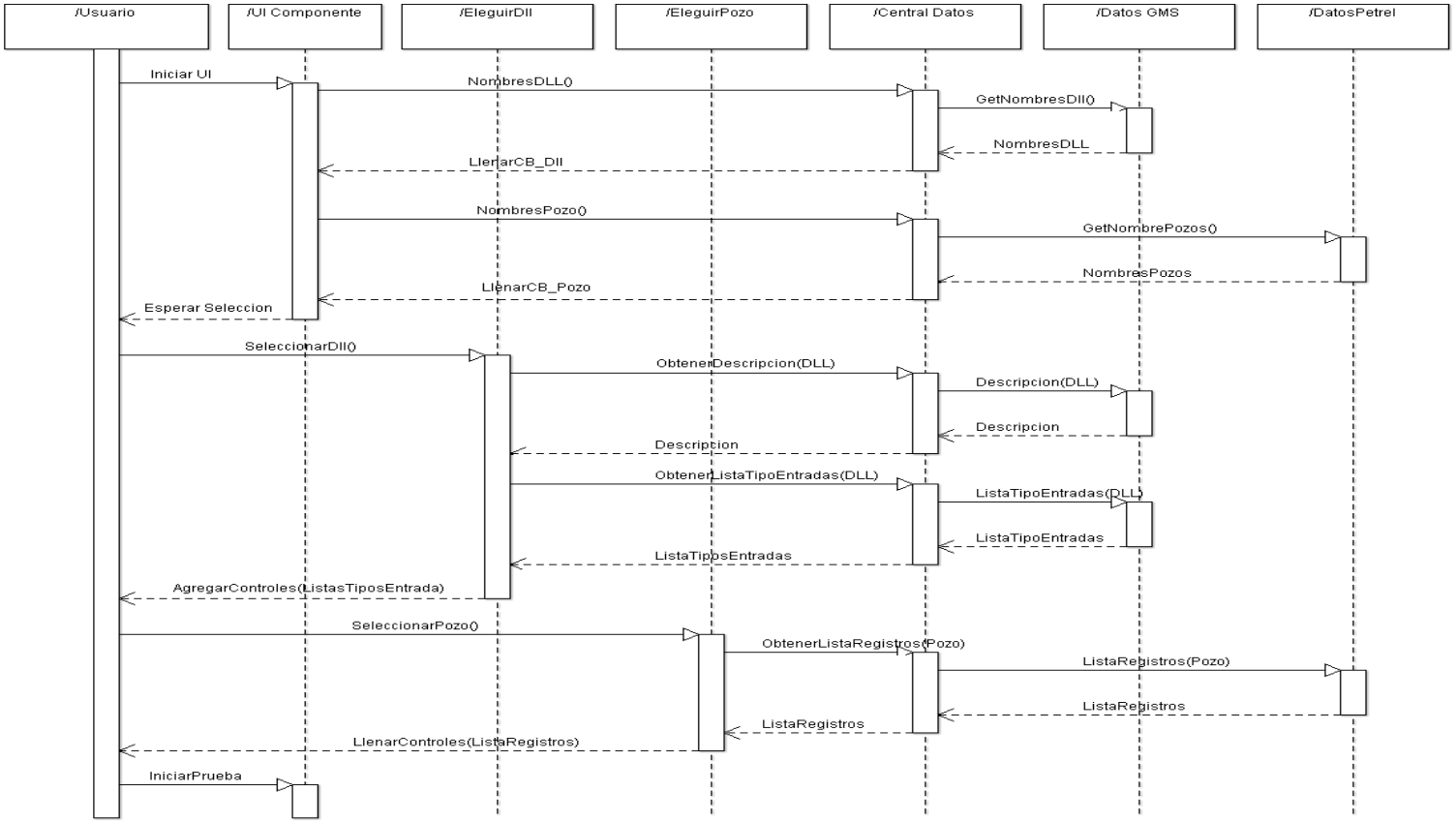


Figura 19 Diagrama de secuencia caso de uso Seleccionar Entradas

Fuente: Autor de este proyecto.

Diagrama de secuencia para el caso de uso Ejecutar Prueba

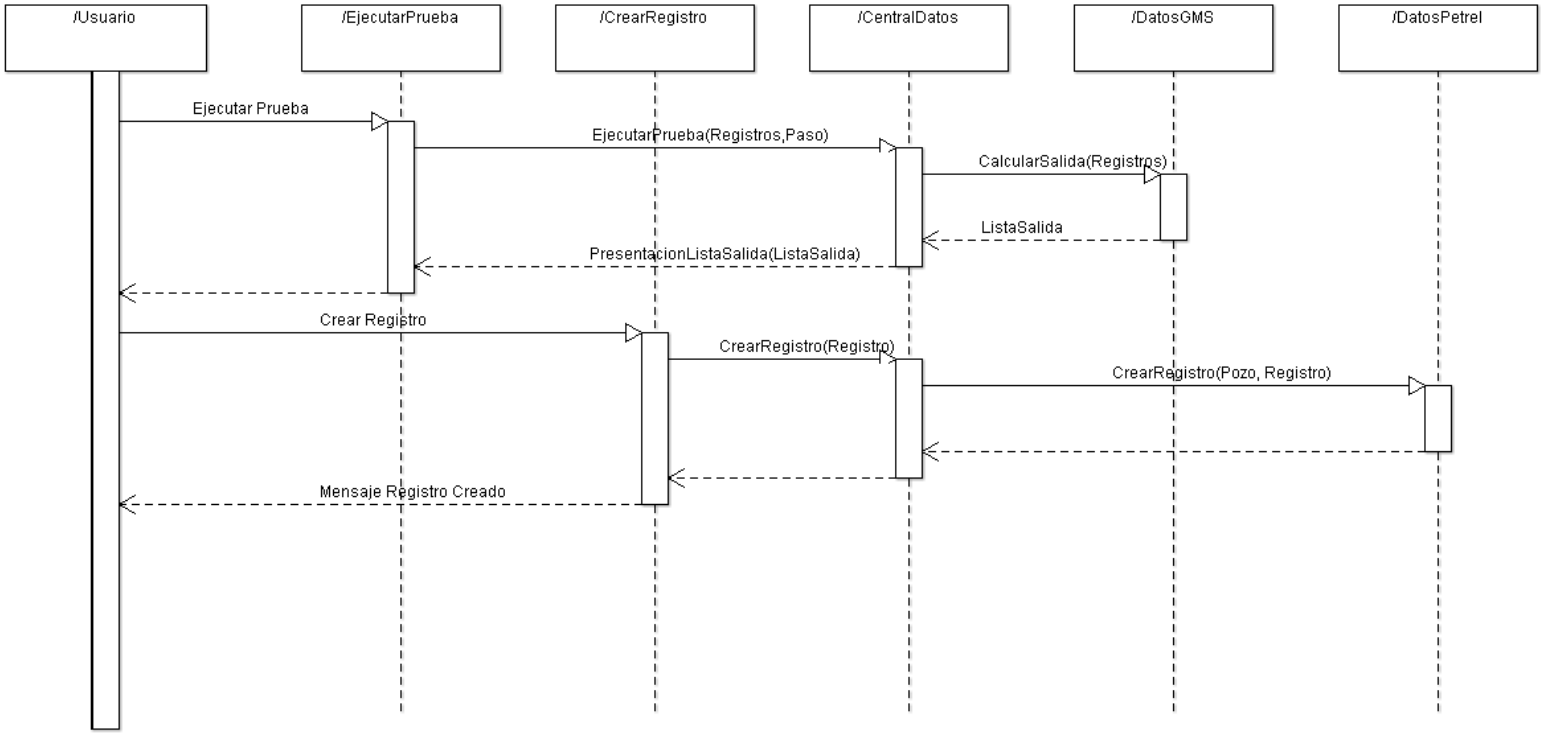


Figura 20 Diagrama de secuencia caso de uso Ejecutar Prueba

Fuente: Autor de este Proyecto

6 DESARROLLO DE LOS OBJETIVOS DEL PROYECTO

Diseñar un complemento software para un área de conocimiento en específico, como es en este caso el entorno petrolero, puede prolongar un proyecto si no se tienen objetivos y tareas claras del mismo. Debido a esto, es necesario determinar las tareas a resolver, y asi implementar la línea de desarrollo para el complemento software.

Para el cumplimiento de este objetivo se define los pasos de planificación, definición, recopilación de datos e interpretación. Para cada uno de los pasos se determinaron unos indicadores, que permitieron evaluar la claridad que se tenía del tema.

Paso 1: identificar el objetivo general del negocio, en este caso se determinó que el fin principal es el complementos software Bridge-GMS

Paso 2: identificar lo que se quiere conocer o aprender, dentro de este paso es necesario cuestionarse ¿Qué actividades se tienen que gestionar o realizar? Y es aquí donde esos objetivos son enlazados a las inquietudes en el proceso de preguntas. Esto se maneja como modelo mental debido a que no siempre se lleva un esquema total de las preguntas, y como fin se determinan las actividades a realizar. Aquí se encuentra el contenido del complemento, la identificación de las actividades que se requieren e ir enlazando todo a un modelo mental de objetos, variables y conexiones entre otras.

Paso 3: se identifican los objetivos específicos, esto permite identificar claramente la relación entre los resultados obtenidos y el objetivo plateado, los cuales son nombrados y desarrollados en este proyecto.

Paso 4: identificar las entidades y atributos relacionados con los objetivos específicos. En este paso es donde el modelo mental debe ser claro. De aquí se obtienen las preguntas más relevantes, las cuales van directamente asociadas a las entidades y atributos, pues se parte de una situación bajo diseño y se puede convertir en un proceso iterativo para refinar las preguntas.

Paso 5: formalizar los objetivos del negocio. En este paso se traducen los objetivos del negocio en objetivos de medición, que son los objetivos específicos en este proyecto, y que permitieran finalmente la medición de los resultados.

6.1 IDENTIFICACION DE LAS VARIABLES DE LA PLATAFORMA G.M.S. Y DISEÑO DEL INTERMEDIARIO SOFTWARE G.M.S.

En esta sección se describirá de manera general las variables y métodos que un componente software debe implementar para que el G.M.S. lo acepte como suyo, identificando estas variables se podrá trazar un mapa de los métodos que se requieren implementar para que un componente en Petrel acepte las bibliotecas de enlace dinámico del G.M.S.

En este paso se hiso un estudio completo de la Proyecto de grado titulada: DISEÑO, ESTRUCTURACIÓN E IMPLEMENTACIÓN DE UN ESTÁNDAR DE DESARROLLO SOFTWARE PARA EL MODELADO DE FENÓMENOS GEOEMCANICOS EN EL GIEP, mas en específico de su estándar de desarrollo y la arquitectura de su plataforma.

Este paso dio como resultados una biblioteca de enlace dinámico “IntermediarioOcean” la cual tiene como principal objetivo realizar el enlace con la plataforma G.M.S. y de allí conectares con el core de esta para así usar sus bibliotecas de enlace dinámico sin la necesidad de utilizar recursos innecesarios (Hardware).

Utilizando los métodos de referencias dinámicas que permite utilizar el lenguaje de programación C# fue posible crear una biblioteca de enlace dinámico desvinculada completamente del DataCore del G.M.S esto permite mantener el objetivo principal con el que se creó el G.M.S el cual era seguir creciendo y modificándose a medida que los requisitos cambiaran, por este motivo fue necesario buscar un método de vinculación dinámico.

```
COREAssembly.GetType("NombreTipo");

COREAssembly.GetType().GetMethod("NombreMétodo");
```

Estos llamados a referencias externas son el centro de llamado de los métodos no solo del DataCore del G.M.S sino de las bibliotecas de enlace dinámico que se utilizaran más adelante en Petrel la forma es que se realizó la conexión se describe en el diagrama de flujo presentado a continuación.

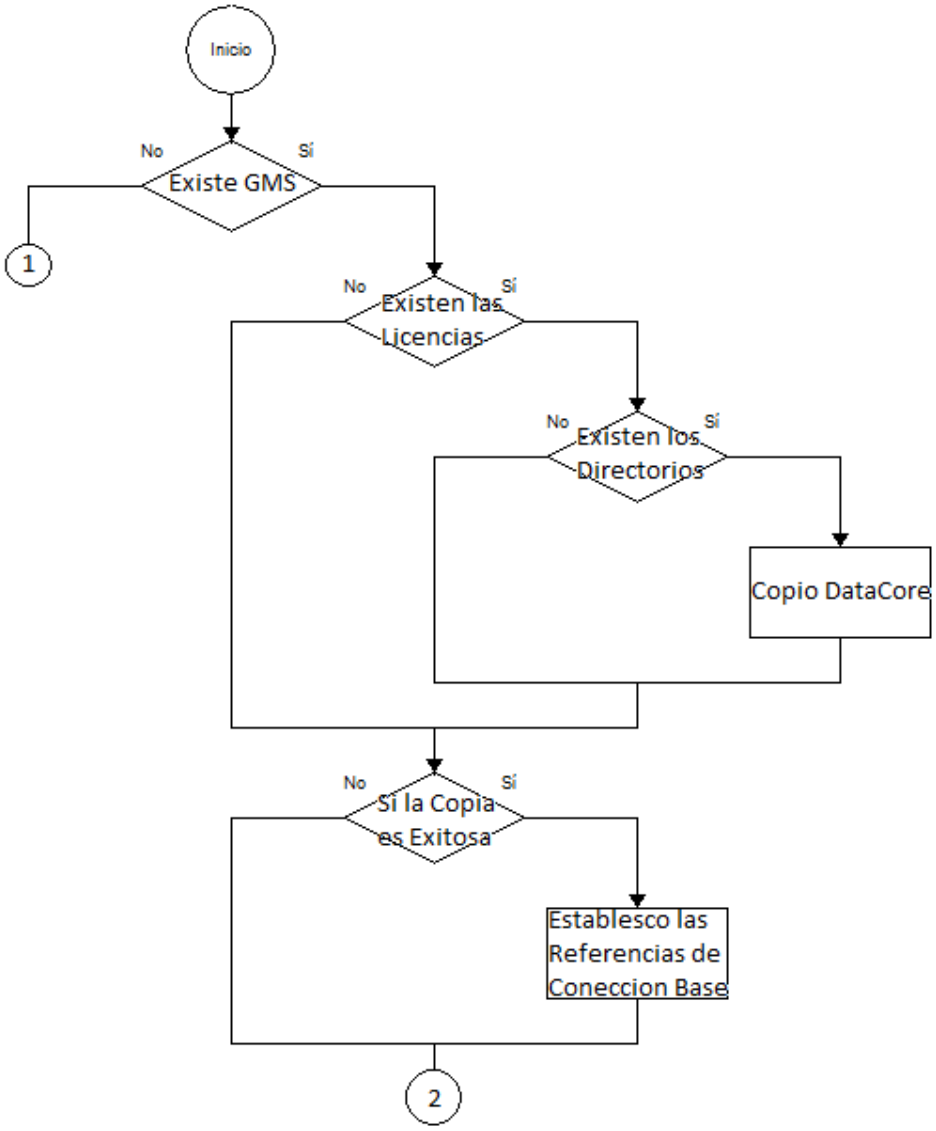


Figura 21 Parte A diagrama de flujo IntermediariOcean

Fuente: Autor de este proyecto

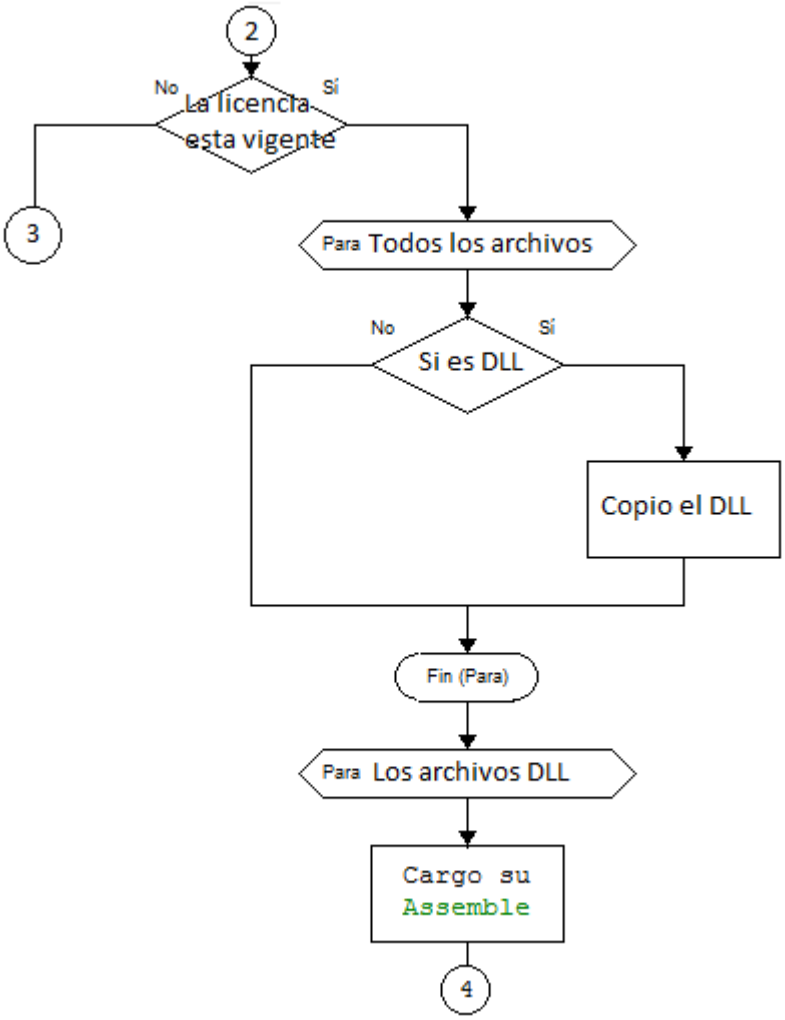


Figura 22 Parte B diagrama de flujo IntermediariOcean

Fuente: Autor de este proyecto

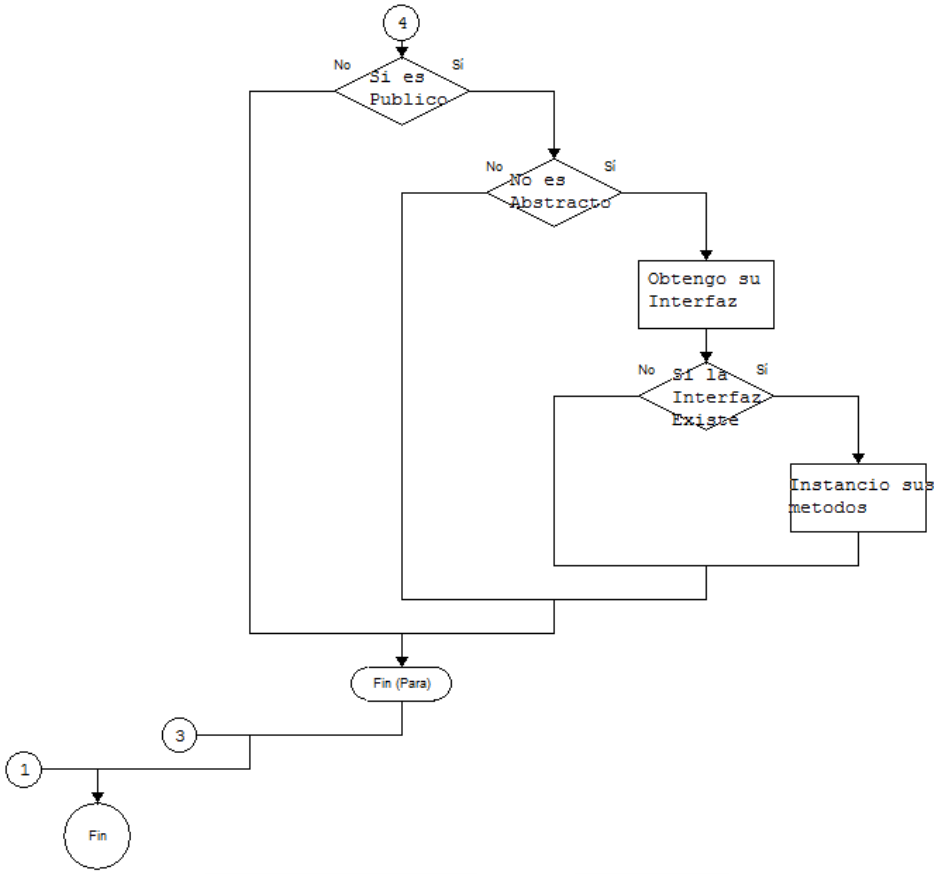


Figura 23 Parte C diagrama de flujo IntermediariOcean

Fuente: Autor de este proyecto

Una vez se realiza la conexión el uso de las bibliotecas de enlace dinámico disponibles se realiza de la misma forma por medio de verificaciones y usando las referencias dinámicas se construyen los métodos, en el actual flujo de trabajo para así disponer de ellas en los futuros análisis directamente en Petrel.

Los siguientes son los métodos identificados e implementados dinámicamente en el BridgeGMS

- getSubGrupos
- getEcuaciones
- getAssembly
- getTextoEcuacion
- getNombre
- getTiposRelacionados

- getParametrosRelacionados
- getTiposSalidas

6.2 DISEÑO DE LA INTERFAZ GRAFICA DE USUARIO Y VISUALIZACION DE RESULTADOS

Una vez se ha creado el intermediario entre Petrel y G.M.S, lo que queda por realizar es el procedimiento de manipulación de los datos de las dos plataformas, como datos de entrada se tienen la información de la biblioteca de enlace dinámico seleccionada por el usuario y los datos almacenados en la base de datos del proyecto Petrel, y como datos de salida se tienen los resultados de la ejecución de la dll del G.M.S y los WellLogs creados por el usuario, usando los resultados de la dll ejecutada.

Datos de entrada

Para el proceso de ingresos de datos el componente software configura una visualización compacta de los datos de entrada, la siguiente imagen muestra el formulario de datos de entrada como lo encontraría el usuario una vez inicia el componente

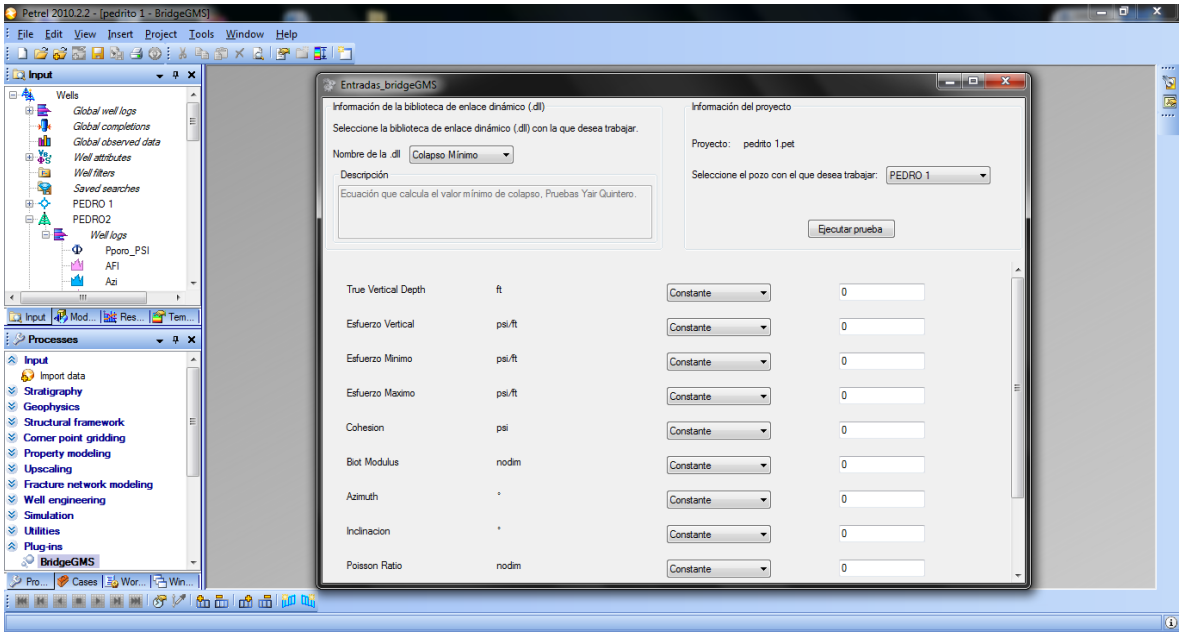


Figura 24 Ventana datos de entrada BridgeGMS

Fuente: Autor de este proyecto

Como se puede ver en la parte superior izquierda se encuentra la información de las bibliotecas de enlace dinámico del G.M.S., esta información se divide en la lista de bibliotecas disponibles de la cual el usuario selecciona una de ellas y un recuadro de texto donde se muestra la información de la biblioteca, todo este proceso es automático y el usuario no necesita realizar complicadas iteraciones con el componente para realizar una conexión con el G.M.S.

En la parte superior derecha se encuentra la información del proyector actualmente en ejecución en Petrel y por medio de una lista el usuario selecciona el pozo en el que desea trabajar, en este pozo será donde se extraigan los registros y donde el usuario creara los nuevos registros producto de la ejecución de la biblioteca seleccionada, nuevamente esta conexión es automática y el usuario no tiene que realizar complicadas iteraciones con el componente para realizar esta conexión.

En la parte inferior se encuentra la lista de entradas la cual dependerá de la biblioteca seleccionada, los controles se agregaran de manera dinámica, al igual que las listas se llenaran de manera dinámica una vez el usuario ha seleccionado un pozo con el cual trabajar, cada lista contendrá los registros del pozo seleccionado más una opción para agregar un registro de constante.

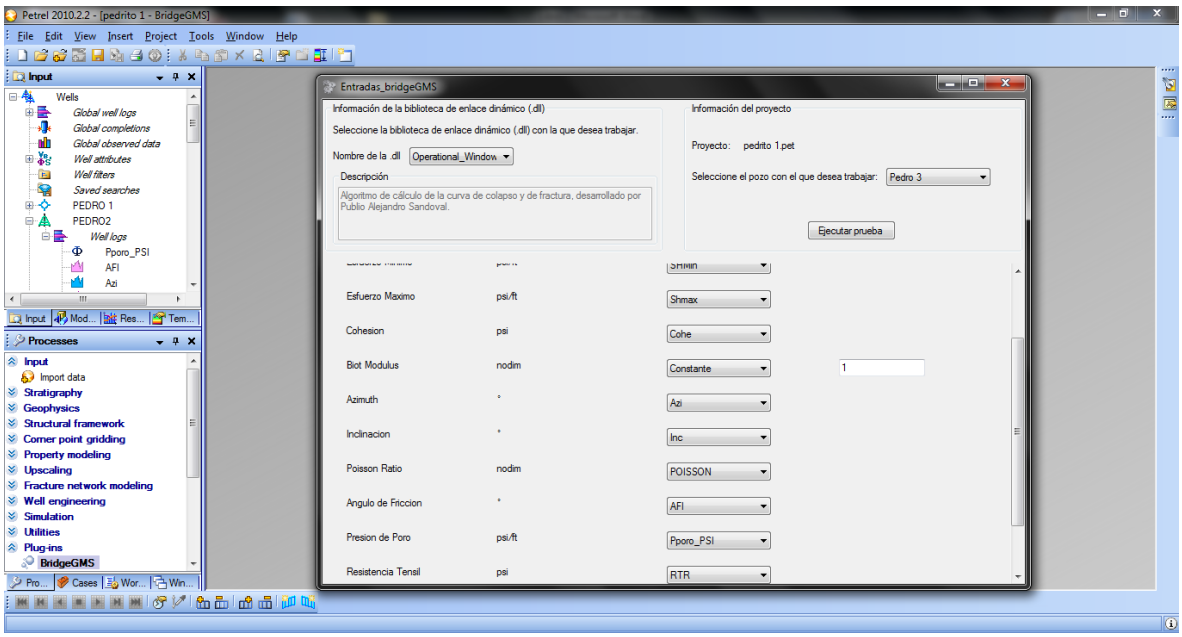


Figura 25 Ventana datos de entrada BridgeGMS

Fuente: Autor de este proyecto

Una vez el usuario ha seleccionado los registros necesarios para ejecutar la biblioteca G.M.S. se procede con la visualización de entradas.

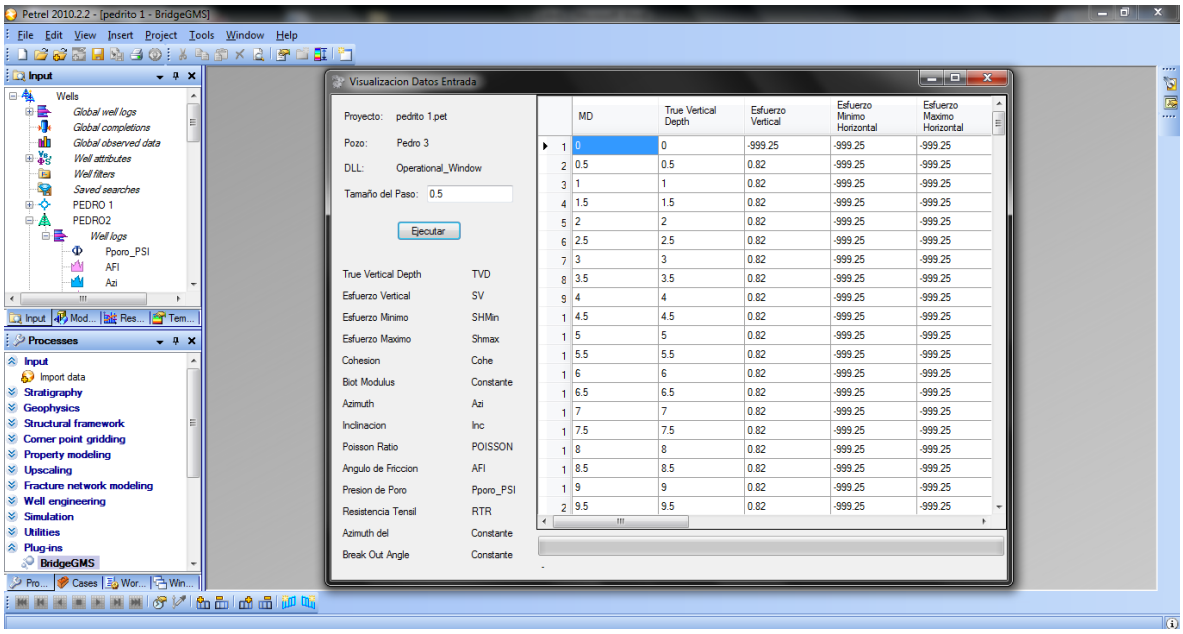


Figura 26 Ventana Visualización datos de entrada BridgeGMS

Fuente: Autor de este proyecto

En esta ventana el usuario puede visualizar los datos de las entradas que requiere la biblioteca del G.M.S., en esta ventana el usuario también elige el paso con el que desea que los datos se ejecuten por defecto este paso es 0.5 pies, una vez seleccionado el paso el usuario procede a ejecutar, la barra de progresión situada en la parte inferior derecha de la ventana le mostrará al usuario cuantos datos han sido analizados, debido a que la ejecución de una prueba puede tardar minutos es necesario que el usuario visualiza el proceso.

Finalmente la ventana de salidas se visualiza con los datos de salida y unas listas de cajas de chequeo que el usuario puede usar para elegir que registros desea crear en Petrel.

Una vez el usuario elige que registros desea crear en petrel procede a dar click en botón crear el cual importara los registros seleccionados a el proyecto Petrel y específicamente al pozo en el que se está trabajando.

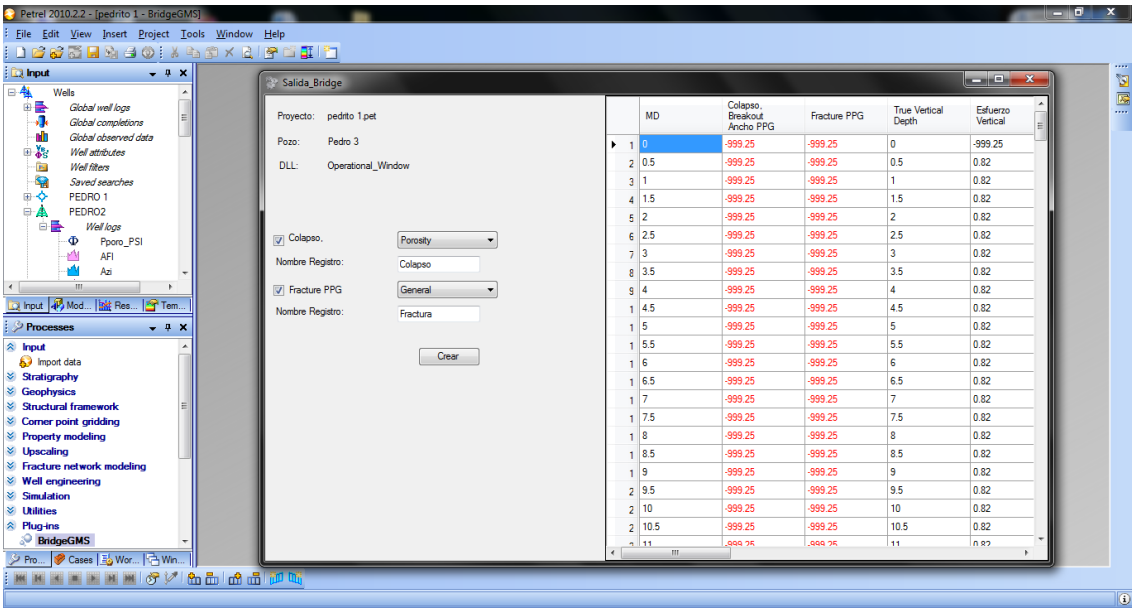


Figura 27 Ventana datos de resultados BridgeGMS

Fuente: Autor de este proyecto

Finalmente los registros se crearan y al usuario le será informado por medio de una ventana emergente este acción.

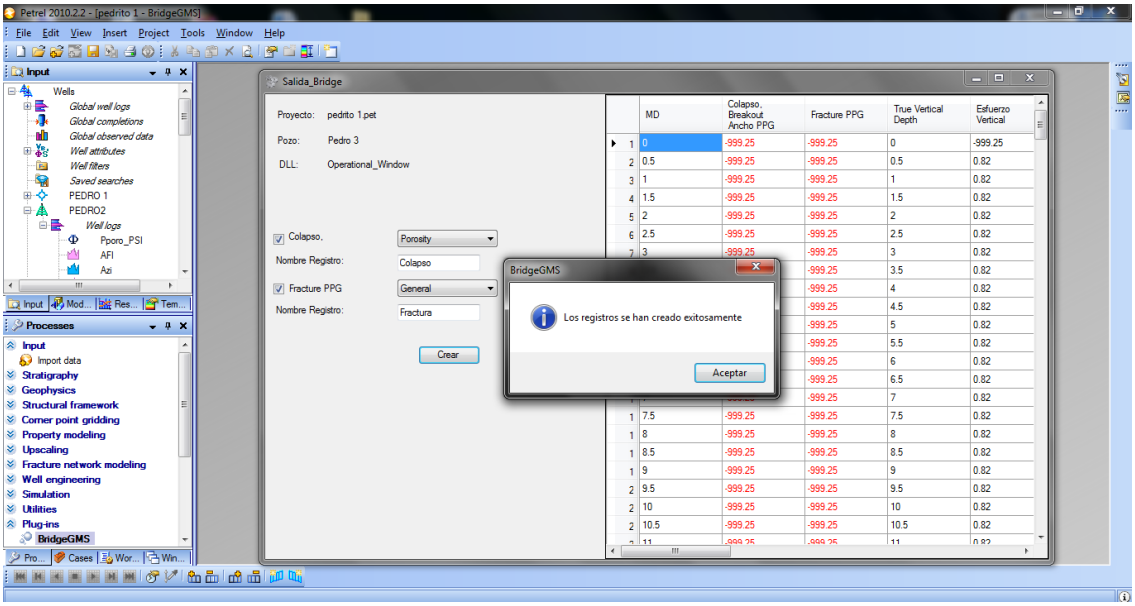


Figura 28 Ventana emergente de BridgeGMS

Fuente: Autor de este proyecto.

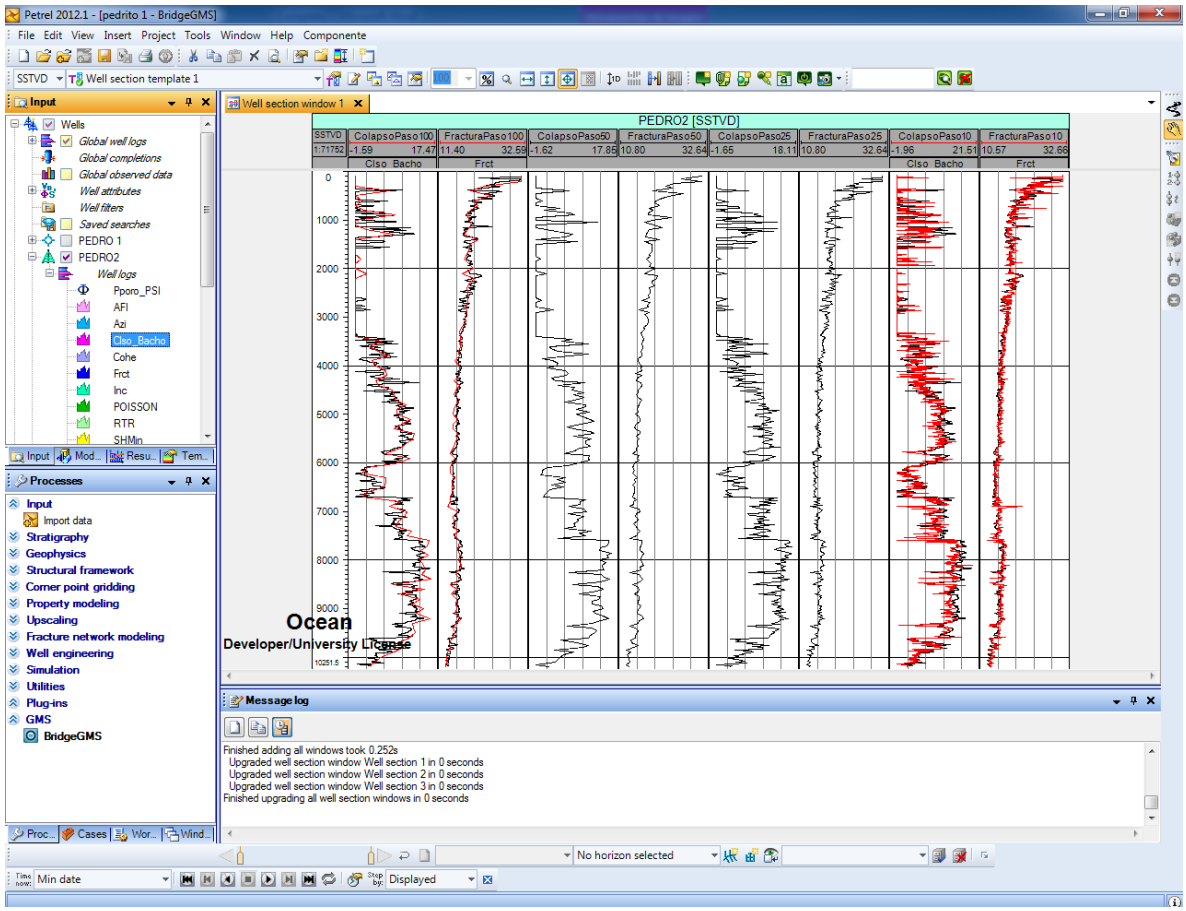


Figura 29 Resultados de ejecución

Fuente: Autor de este proyecto.

6.3 CREACIÓN DEL MANUAL DE REFERENCIA.

Para el cumplimiento de este objetivo se creó un manual de referencias que incluye tanto las experiencias adquiridas en este proyecto como un ejemplo que complementa los ya existentes en el manual que ofrece la empresa Schlumberger Getting Started with Ocean, ver Anexo A.

7 CONCLUSIONES

- Se cumplió a cabalidad con cada uno de los objetivos estipulados de este proyecto, así como cada una de las pautas detectadas durante la extracción de requerimientos realizada, resaltando los objetivos referentes a la identificación de las variables que podrían llegar a interferir en la conexión de dos plataformas software completamente diferentes.
- Gracias a los diferentes estándares, normas y modelos que se usaron en el desarrollo de estas dos plataformas fue posible realizar la conexión entre ellas siguiendo los lineamientos, y recomendaciones de la documentación individual de cada plataforma.
- Gracias al acople que finalmente se obtuvo con las dos plataformas se reduce el tiempo no productivo que tiene un examinador en cada una de las pruebas lo que conlleva a un significante incremento en la cantidad de pruebas que puede realizar un usuario y eso finalmente repercute en una disminución significativa de los costos por cada prueba.
- Con el desarrollo de un componente software bajo el entorno de programación visual Studio 2008 y el lenguaje de programación C#, que permite una integración dinámica de dos plataformas, no solo se cumplió con el objetivo general sino que se hizo un agregado al vincular referencias dinámicas al proyecto, esto ocasiona que las futuras actualizaciones de las dos plataformas no altere el buen funcionamiento de BridgeGMS, permitiendo así a las dos plataformas continuar con su crecimiento y expansión.
- Las funcionalidades y características incorporadas en BridgeGMS permiten que los operarios del sistema tengan una iteración óptima, cómoda y sencilla con el componente software.
- Se elaboró un manual de desarrollo que envuelve los conocimientos principales obtenidos en el desarrollo de esta Proyecto, por lo tanto se cumple con el último objetivo el cual es dejar unas memorias de lo que se puede hacer, cómo hacerlo y que errores se pueden presentar, para que así futuros tesistas puedan usarlo como base para desarrollos software utilizando el SDK Ocean para la plataforma software Petrel.

8 RECOMENDACIONES

- Para garantizar el buen funcionamiento del componente es necesario que los operarios del sistema no solo estén familiarizados con las dos plataformas sino que hayan tenido una correcta inducción de los procedimientos que se deben seguir para el uso de BridgeGMS, debido a que las ecuaciones embebidas en las dlls del G.M.S. requieres cierto grado de familiaridad para su correcto uso.
- Se sugiere expandir las capacidades de BridgeGMS aprovechando la sección Grafica en 2D y 3D que ofrece Petrel y así conseguir que el operario del sistema pueda sacar un mayor resultado a los datos procesados en las bibliotecas de enlace dinámico G.M.S. y es así como se recomienda crear un pre visualizador de los registros que se van a crear, así se evita crear registros que finalmente se tendrá que borrar por su poca contribución a la evaluación o test que se esté realizando.
- Se recomienda proseguir con la investigación del SDK Ocean para Petrel ya que el contenido abarcado en esta Proyecto es solo una de las muchas aplicaciones que ofrece esta plataforma, y quedan aún muchas más por explorar debido a que el área de la geomecánica es muy amplia.

9 BIBLIOGRAFÍA

GAMMA, Erich; Helm, Richard; Johnson; Ralph; Vlissides, Jhon, Design Patterns.China 2004.

GOODMAN, Richard E., Introduction To Rock Mechanics, 2a Ed. New York: Wiley, 1989.

Hoek, Evert, Rock Mechanics - An Introduction For The Practical Engineer. En: Mining Magazine, 1966.

IEEE, Especificación De Requisitos Según El Estándar De Ieee 830. 2008. Gamma, Erich; Helm, Richard; Johnson; Ralph; Vlissides, Jhon, Design Patterns.China 2004.

MANTILLA, Lilian; TOLOZA Juliet, Modelo De Maduracion Para Proyectos De Investigación. Trabajo De Grado Ingenieria Industrial Bucaramanga: Universidad Industrial De Santander, Facultad Fisico Mecánicas 2009.

RAMIRES, Julia, Integración Petrel – Ocean: Herramienta De Avanzada Para Análisis De Datos Del Subsuelo, En: Revista Digital Apuntes De Investigación Vol.3: Bucaramanga 2012.

RATIONAL SOFTWARE CORPORATION, Rational Unified Process Best practices For Software Development Teams, 1998.

ROGER, Pressman, Ingenieria Del Software Un Enfoque Práctico, 6a Ed. Mcgraw Hill, 2005.

ROJAS, Jorge, SOLER, Diego, Diseño, Estructuración E Implementación De Un Estándar De Desarrollo Software Para El Modelado De Fenómenos Geomecánicos En El Giep (Grupo De Investigación De Estabilidad De Pozo).Trabajo De Grado Ingenieria De Sistemas E Infromatica: Universidad Industrial De Santander, Facultad Fisico Mecánicas, 2011.

SCHLUMBERGER , Welcome To The Petrel, Vol. 1, Houston Texas: Software Development Teams, 2010.

SCHLUMBERGER, Getting Started With Ocean, Vol. 1, Houston Texas: Schlumberger Information Solutions. 2010.

SCHLUMBERGER, Ocean Application Development Framework, Houston Texas: Software Development Teams, 2011.

SCHLUMBERGER, Ocean Basics, Vol. 3. Houston Texas: Software Development Teams, 2010.

SCHLUMBERGER, Ocean For Petrel Data Access, Vol. 2. Houston Texas: Software Development Teams, 2010.

10 ANEXOS

Anexo A Documento Manual de Usuario SDK Ocean

Tabla de Contenido

1. INTRODUCCION

En este documento se describirá los objetivos e información clara y concisa de cómo utilizar el SDK OCEAN para la plataforma PETREL.

El SDK OCEAN fue creado por la empresa SCHLUMBERGER con el objetivo de brindar facilidades a las personas involucradas en la perforación, seguimiento y cierre de zonas que poseen alguna relación con la explotación de Gas, Petróleo y sus derivados.

La importancia de este manual se centra en las experiencias obtenidas en la elaboración de la Proyecto de grado titulada "**Desarrollo de un componente software para integrar las librerías del software de modelado geomecánico (G.M.S) en la plataforma PETREL",** desarrollada por el estudiante de ingeniería de sistemas e informática Pedro Andrés Rodríguez Sánchez vinculado a la Universidad Industrial de Santander (UIS).

2. OBJETIVOS DE ESTE MANUAL

El objetivo primordial de este Manual es ayudar a los estudiantes de ingeniería de sistemas e informática y carreras afines que deseen elaborar herramientas software utilizando el SDK OCEAN para PETREL, por medio de un manual que desglosa no solo en pasos sencillos lo que debe saber una programador de este SDK, sino que se menciona los requisitos básicos tanto en software como en hardware y se muestran algunos errores comunes.

- Requisitos mínimos del sistema tanto en hardware como en software.
- Ejemplo completo de acceso y manipulación de datos en PETREL utilizando el SDK OCEAN.

3. CONOCIMIENTOS NECESARIOS

Los conocimientos mínimos que deben de tener las personas que deseen utilizar no solo este manual sino el SDK OCEAN para PETREL son:

- Conocimiento en el lenguaje de programación C SHARP (C#)
- Amplio conocimiento y dominio de programación orientada a objetos
- Idioma ingles en un nivel aceptable de lectura (para los manuales y ayudas de OCEAN)
- Conocimientos básicos de temas relacionados con la perforación, extracción, y cierre de plataformas petroleras y sus derivados
- Conocimiento y manejo de la plataforma PETREL.

4. ESPECIFICACIONES TECNICAS

Las especificaciones técnicas para el uso óptimo del SDK OCEAN para PETREL son:

4.1. HARDWARE

- Intel© Pentium© o procesadores compatibles de al menos 1.5 GHz .
- 2 GB RAM (recomendado 4GB RAM para las versiones de 64- bit).
- 2.5 GB de espacio libre en el disco duro.
- Tarjeta gráfica dedicada compatible con OpenGL.
- Pantalla con una resolución de 1024x768 y 32 bis de colores.7

4.2. SOFTWARE

- Windows 7 (32 o 64 - bit), Microsoft© Windows© XP Professional (32-bit (SP2 o mayor)).
- Microsoft© .NET Framework Versión 3.5.
- Macrovision Secure Flexnet (Llave física PETREL).
- Visual Studio 2008, idioma inglés, cualquier versión.
- SDK OCEAN versión 2010.1, 2010.2, 2011.1 o 2011.2
- PETRE Versión 2010.1, 2010.2, 2011.1 o 2011.2 (la misma versión que el SDK OCEAN)

5. INSTALACION

Antes de proceder con la instalación del SDK OCEAN es necesario cumplir algunos requisitos previos

- Asegúrese de tener instalada la versión en inglés de Visual Studio 2008, es muy importante que la versión sea en ingles debido a algunos problemas encontrados en la compatibilidad de la versión en español con el SDK OCEAN hace difícil instalar este en una versión diferente a la del idioma origen, por lo tanto para evitar futuros errores instalar la versión en inglés de cualquiera de las presentaciones de Visual Studio 2008 Estándar/Profesional.
- Asegurarse que la versión del Microsoft© .NET Framework sea la correcta para versiones del SDK OCEAN 2010 y 2011 es necesario la versión 3.5.
- Asegúrese que la versión que tiene instalada de petrel sea la misma versión que va a instalar del SDK OCEAN, para el ejemplo se trabajará con PETREL 2010.2 por lo tanto se instalará la versión 2010.2 del SDK OCEAN
- Asegúrese de que el software de verificación de licencias CodeMeterRuntime este correctamente instalado en el ordenador que se va a utilizar.

5.1. SDK OCEAN

La instalación del SDK OCEAN se puede realizar por dos medios, uno directamente del ejecutable que se puede descargar de la página de Schlumberger http://www.ocean.slb.com/Pages/developers.aspx utilizando nuestro usuario y contraseña adquiridos una vez se compra el paquete de programador en cualquiera de sus dos versiones, la otra opción sería usar el DVD que nos envían junto con la llave física una vez se adquiere el paquete de desarrolladores, en las dos opciones el procedimiento es más o menos el mismo.

5.1.1. Instalación del SDK OCEAN

Se inicia la instalación dando doble click al ejecutable o seleccionando la opción “install Ocean” del autorun del DVD, mostrará la siguiente pantalla.

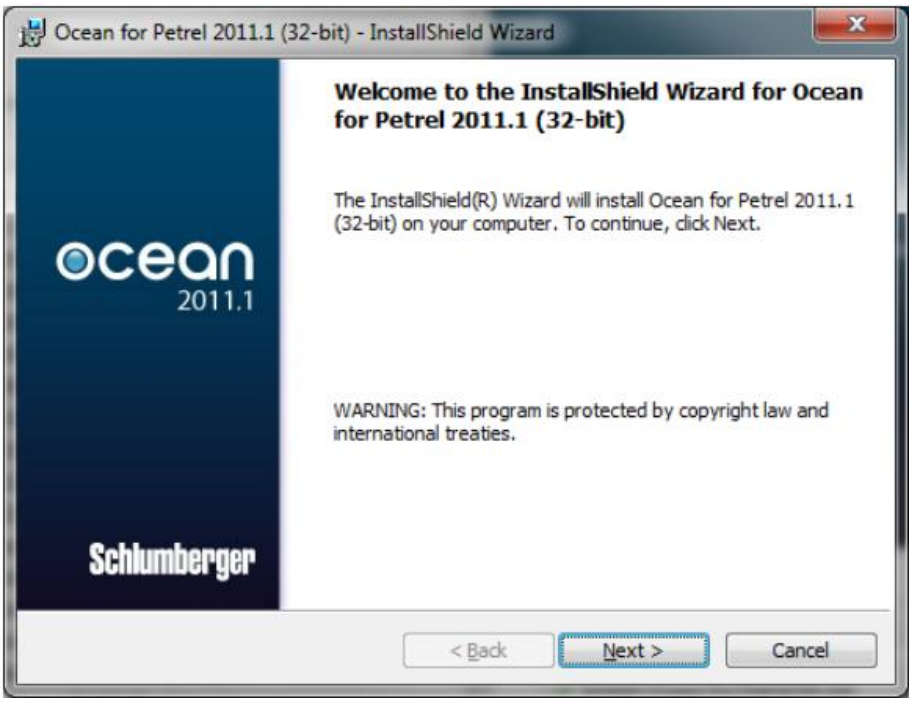


Figura 30 Ventana de instalación SDK Ocean

Fuente: Autor de este proyecto

Una vez se presente esta pantalla se da click en Next, y mostrará la siguiente pantalla.

Figura 31 Ventana de instalación SDK Ocean

Fuente: Autor de este proyecto

En esta pantalla se debe leer, entender y aceptar los términos del contrato, al darle click en Next mostrará la siguiente pantalla

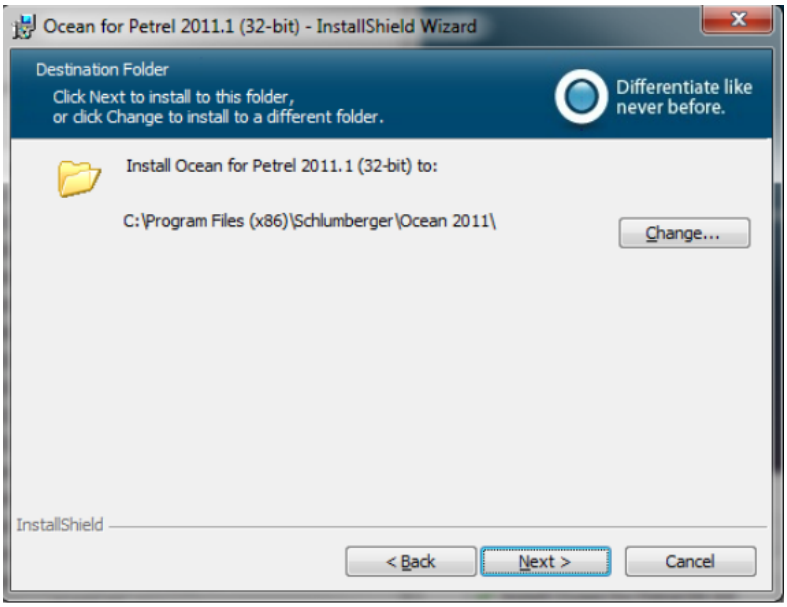


Figura 32 Ventana de instalación SDK Ocean

Fuente: Autor de este proyecto

Si de alguna forma PETREL no se encuentra instalado en su dirección por defecto se debe buscar la carpeta Schlumberger en donde esté instalada la plataforma PETREL, por consejo de los mismos desarrolladores de esta plataforma lo mejor es no cambiar las rutas por defecto.

Al darle click en Next se mostrará la siguiente pantalla

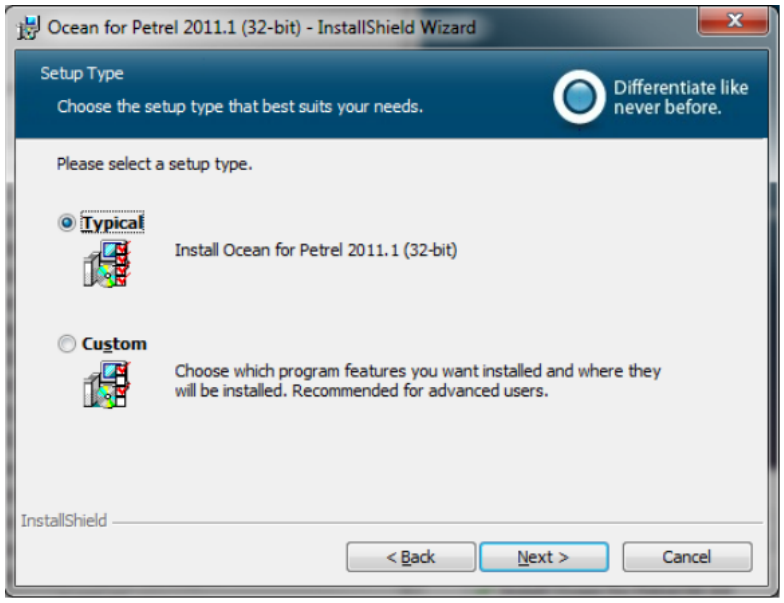


Figura 33 Ventana de instalación SDK Ocean

Fuente: Autor de este proyecto

Se elige el modo típico para la instalación y se procede a dar click en Next, se verá la siguiente pantalla.

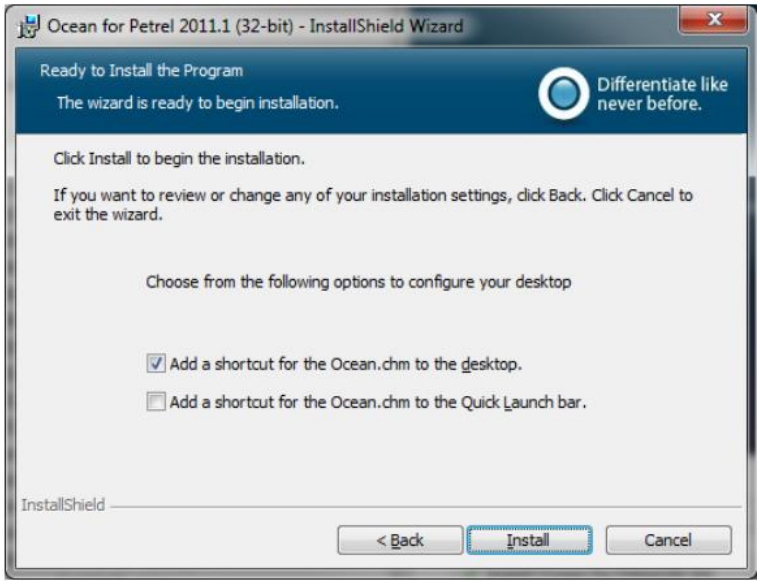


Figura 34 Ventana de instalación SDK Ocean

Fuente: Autor de este proyecto

Por último se procede a crear un acceso directo en el escritorio y a darle click en Install, tras lo cual se deberá esperar entre unos 3 a 8 minutos a que finalice la instalación y actualización de Visual Studio 2008.

5.1.2. Licenciamiento de PETREL y OCEAN

Una vez se tiene instalado la plataforma PETREL se procede a la activación de su licencia, se introduce la llave física en un puerto USB y se espera a que el software CodeMeterRuntime reconozca y active la llave

Figura 35 Llave física plataforma PETREL

Fuente: Autor de este proyecto

Una vez lo ha reconocido se ejecuta el programa SchlumbergerLicensing2010 o 2011 dependiendo de qué versión se está instalando.

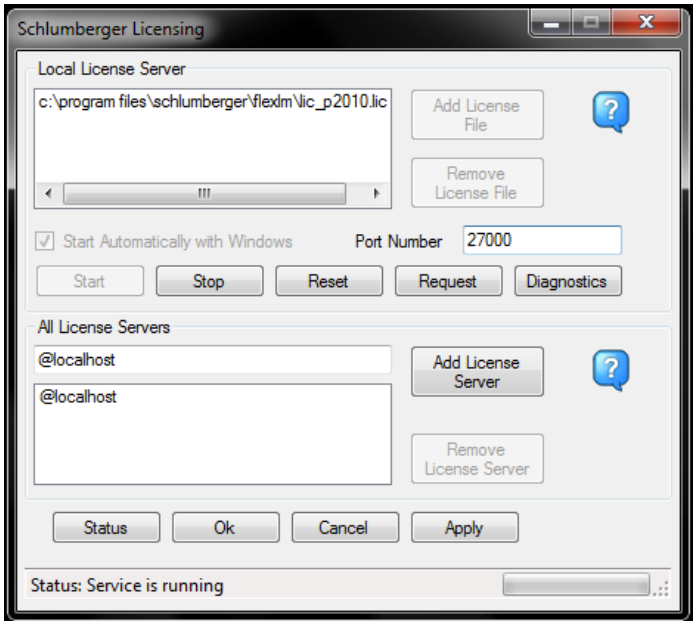


Figura 36 Ventana de licenciamiento SDK Ocean y Petrel

Fuente: Autor de este proyecto

Se da click en el botón Add License File para buscar el archivo .lic proporcionado por la empresa Schlumberger una vez se adquiere el kit de programador, tras esto se da click al botón Start y finalmente al botón Ok, si el archivo .lic contiene la autorización de programación se verá el módulo de OCEAN al final de la lista de plug-in disponible en PETREL.

Una vez finalizado este procedimiento lo único que falta es verificar que Visual Studio 2008 tiene el témplate OCEAN, si no es así lo más seguro es que si Visual Studio no sea el correcto verifique requisitos del software.

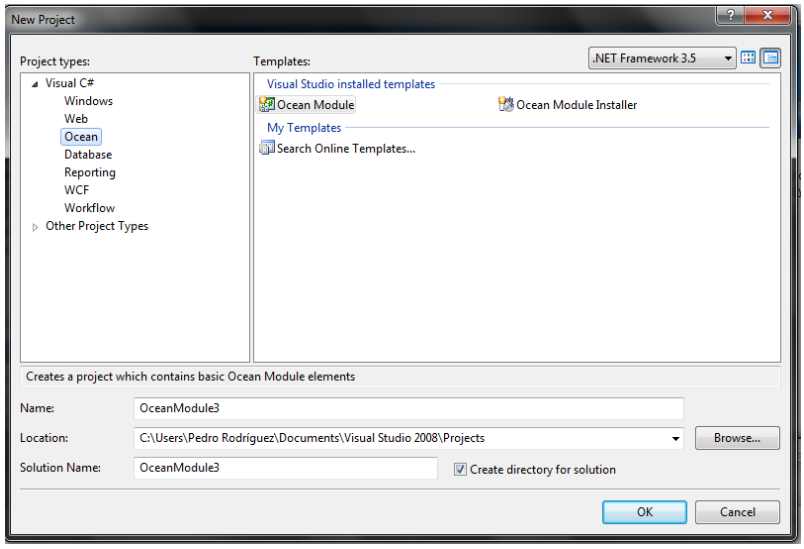


Figura 37 Ventana nuevo proyecto Ocean

Fuente: Autor de este proyecto

6. PRUEBA DE FUNCIONAMIENTO

Para realizar una prueba basta con iniciar Visual Studio 2008

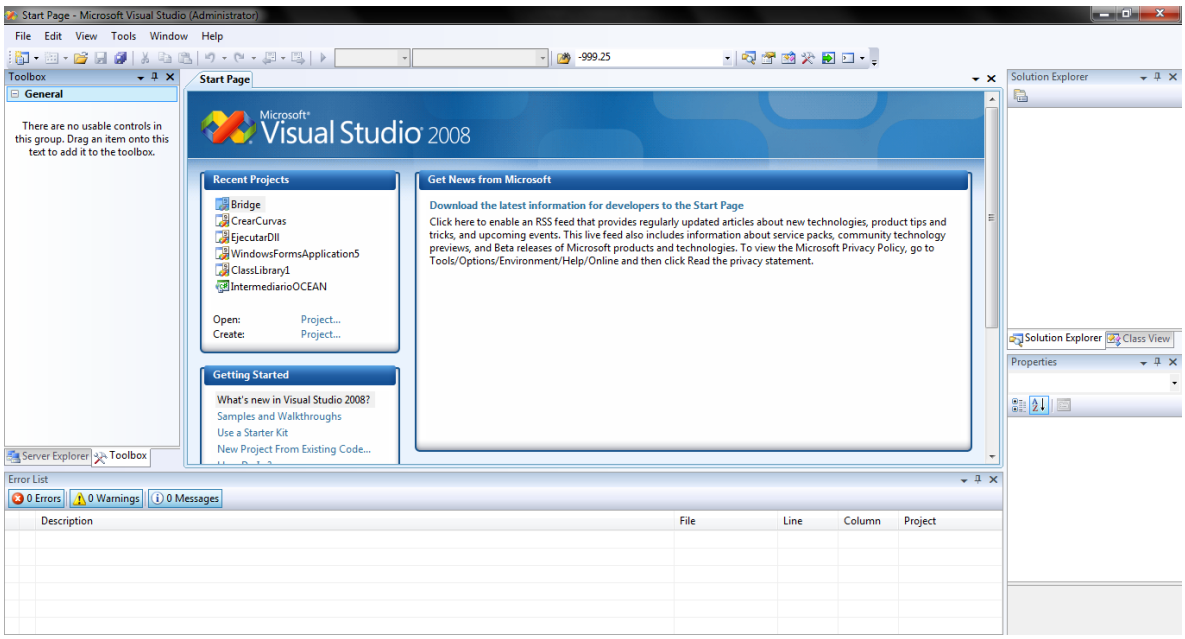


Figura 38 Visual Studio 2008

Fuente: Autor de este proyecto

Elegir File-> new Project

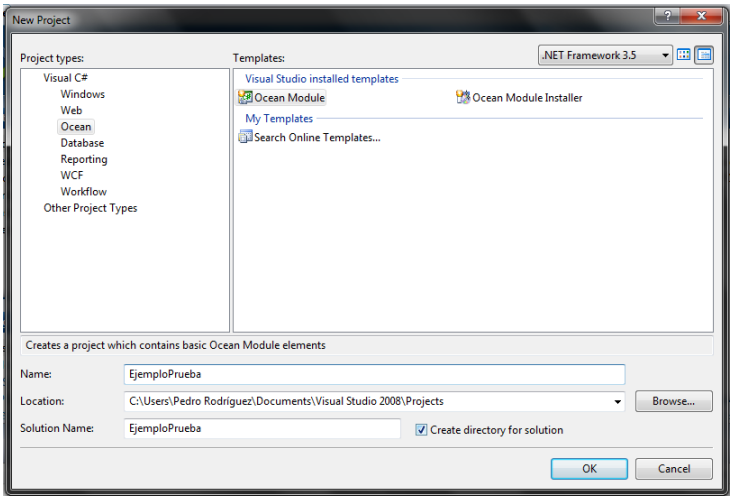


Figura 39 Ventana nuevo módulo Ocean

Fuente: Autor de este proyecto

En la ventana desplegable seleccionar el template Ocean y en él, Ocean Module, darle nombre al proyecto que para este ejemplo será EjemploPrueba y click en Ok.

6.1. Asistente de nuevo proyecto Ocean

Tras iniciar un nuevo proyecto Ocean un asistente aparecerá y pedirá en pasos muy sencillos información básica del componente software que se va a implementar

Figura 40 Ventana paso 1 del Asistente Ocean

Fuente: Autor de este proyecto

En esta ventana muestra el nombre del proyecto, aquí se debe seleccionar la opción new Workstep que será la clase básicas para que un proyecto se comunique con petrel, este workstep tendrá los métodos sobre-escribibles que requiere el core de petrel para admitir un componente como suyo, se le dara un nombre y se procede a dar click en Next, si es la primera vez que usa OCEAN deberá agregar la ruta donde está el ejecutable de PETREL.

Figura 41 Ventana paso 2 del Asistente Ocean

Fuente: Autor de este proyecto

En esta ventana se elege la versión de OCEAN que se va a implementar para este ejemplo seria 2010.1 and 2010.2. click en Next

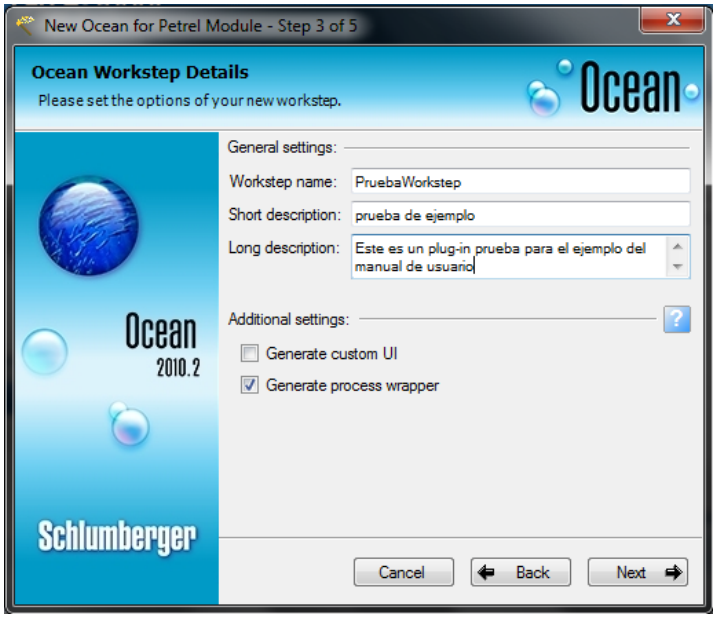


Figura 42 Ventana paso 3 del Asistente Ocean

Fuente: Autor de este proyecto

En esta ventana se llenara la información que se mostrará una vez se inicia el Workstep en petrel, si lo desea se puede generar un UI por defecto, sin embargo esta práctica no permite libre manejo de la UI por lo tanto se recomienda crear los formularios como clases por aparte y anexarlos por medio de código en el Workstep, una vez terminado la descripción se da click en Next

Figura 43 Ventana paso 4 del Asistente Ocean

Fuente: Autor de este proyecto

En esta ventana se puede agregar variables de diferentes tipos predefinidos por petrel si es un componente el cual se conoce las variables de entrada y salida se puede usar esta ventana para agregarlas directamente a la clase DomainObject de Petrel y así aprovechar su uso seguro sin preocuparce por conversiones de unidades, o templates, para finalizar dar click en Next.

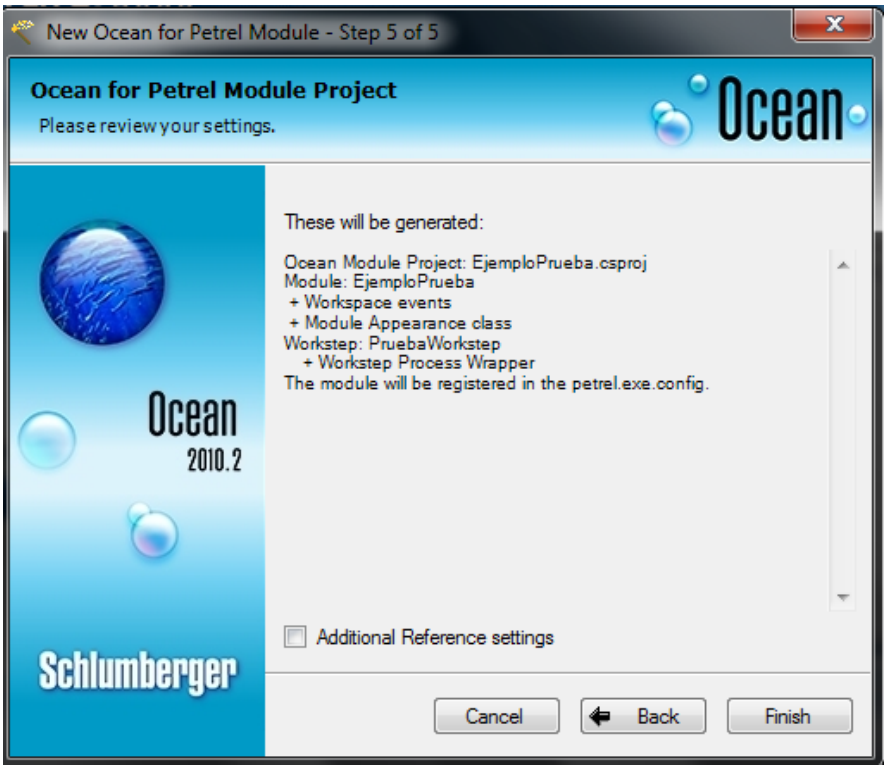


Figura 44 Ventana 5 del Asistente Ocean

Fuente: Autor de este proyecto

Esta sería la última ventana mostrará en resumen las elecciones que se han realizado a lo largo de los pasos anteriores acá se encuentra la opción de agregar nuevas referencias sin embargo esto se puede agregar más adelante, click en Finish y se espera a que el asistente termine de generar el proyecto.

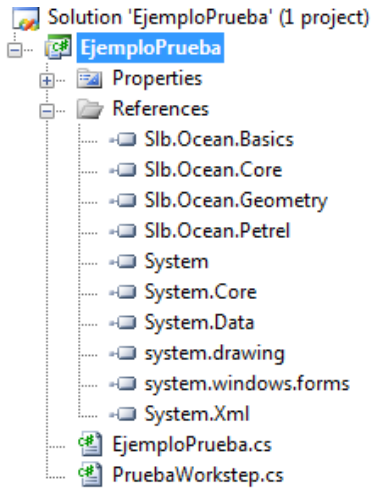


Figura 45 Árbol de clases y referencias del proyecto

Fuente: autor de este proyecto

Este será el árbol de proyecto básico generado por el asistente tendrá dos clases y las dll básicas para que el Workstep pueda ser abierto por PETREL, se inicia el compilado dando click en Star Debugging, se observa que se abre la aplicación Petrel, esto es debido a que la compilación de código se hará directamente sobre PETREL y no en el compilador por defecto de C#.

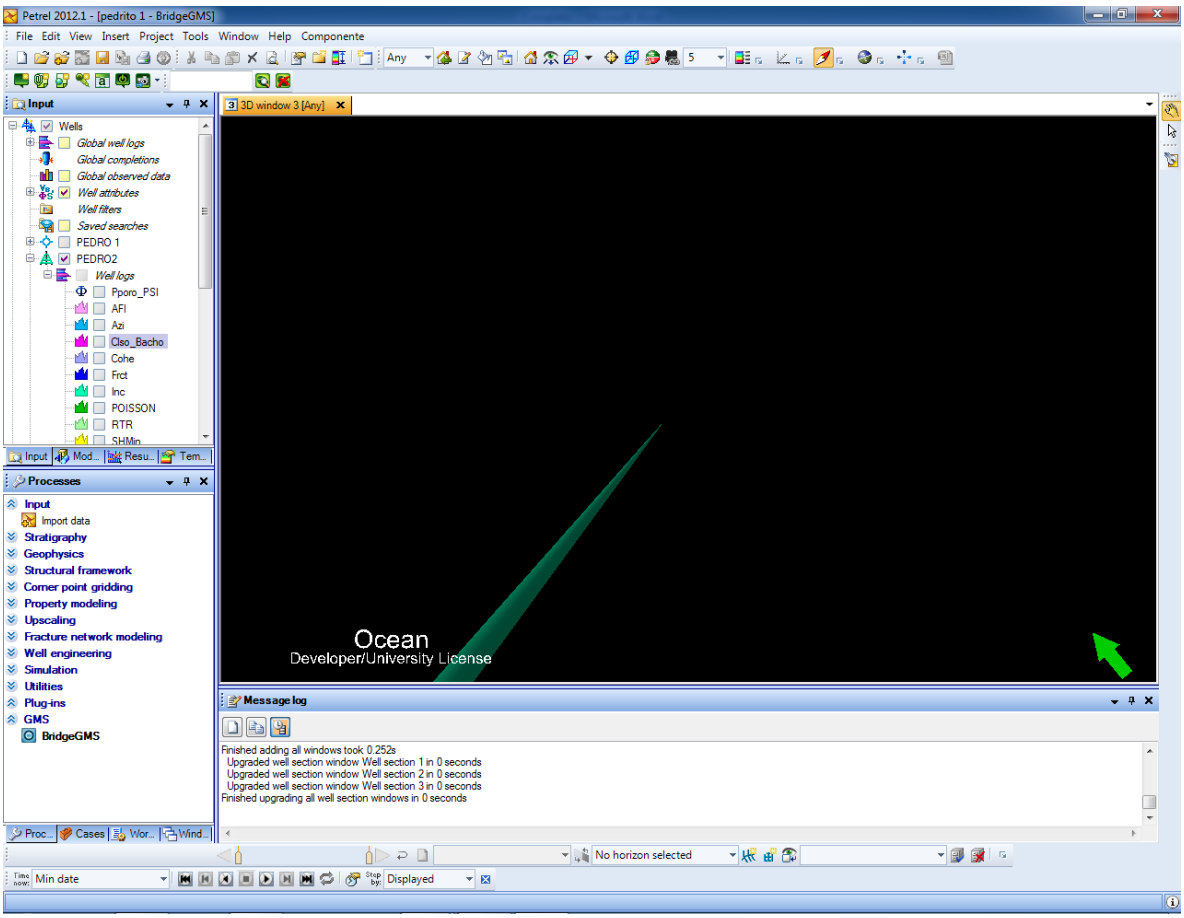


Figura 46 Escritorio de trabajo Petrel

Fuente: Autor de este proyecto

Una vez en PETREL en la ventana de processes se despliega la lista Plug-ins y en ella se encontrara el nuevo plug-in listo para usarse para este caso sería PruebaWorkstep se da doble click y debe mostrar la siguiente ventana.

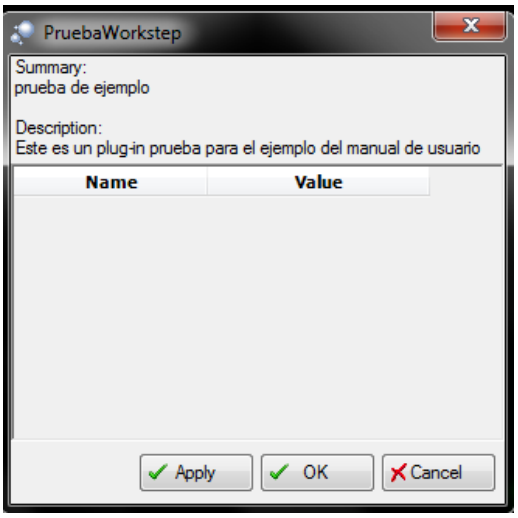


Figura 47 Ventana información Plug-in

Fuente: Autor de este proyecto

La información mostrada en esta ventana proviene de la llenada en el asistente al darle click a Apply o a OK ejecutara el código que se haya puesto en el método ExecuteSimple y al darle click en Cancel cerrara el plug-in.

7. MI PRIMER COMPONENTE (PLUG-IN)

Una vez se ha realizado una completa lectura a la documentación de ayuda de OCEAN y a sus libros de acceso de datos, se pueden usar los métodos y clases que Ocean posee, para manejar los tipos de datos de PETREL y así obtener y enviar información a PETREL directamente por código, en el siguiente ejemplo se realizara una conexión al proyecto activo de petrel, se creara un formulario y se conectara con el workstep, por medio de unas listas se mostrará los pozos y registros que dichos pozos poseen para finalmente mostrar los datos en una tabla.

Se Inicia con abrir el EjemploPrueba creado en el capítulo anterior.

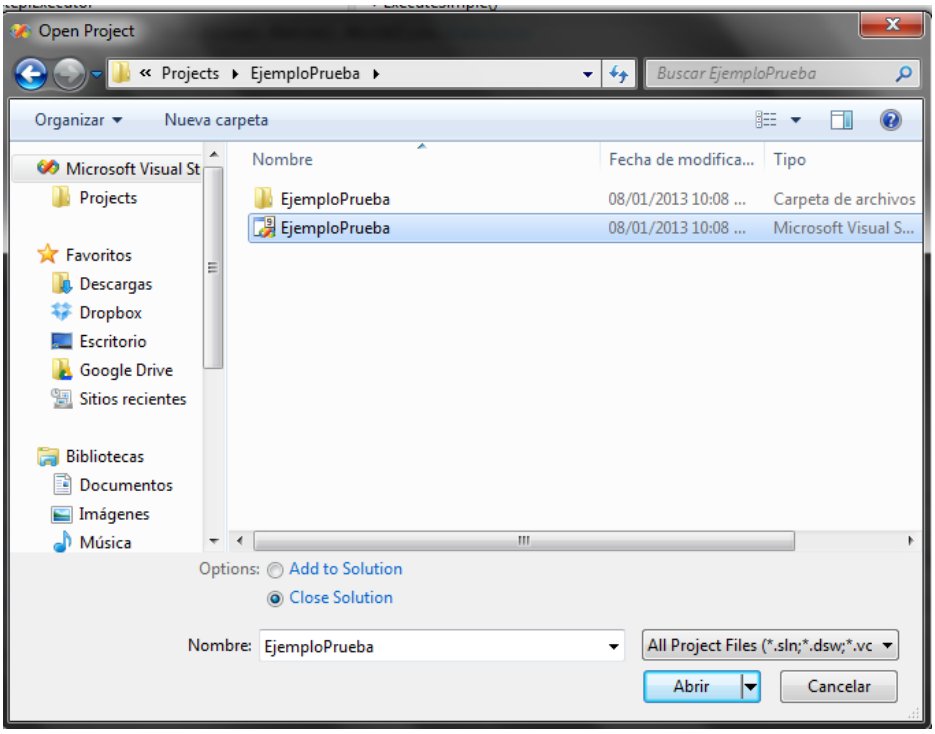


Figura 48 Ventana abrir proyecto Ocean

Fuente: Autor de este proyecto

7.1. Escribiendo el código

Una vez abierto se accede al árbol del proyecto y posicionando el mouse sobre el proyecto, se agregará un nuevo ítem que para este caso será un formulario tipo windowsform

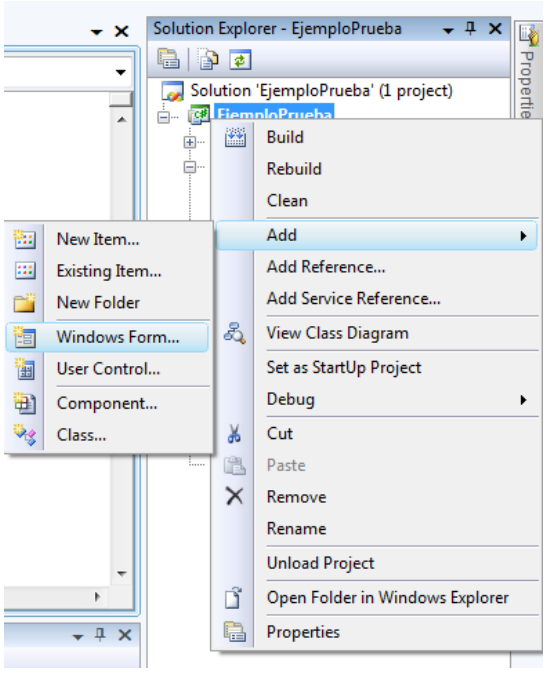


Figura 49 Agregar nuevo formulario

Fuente: autor de este proyecto

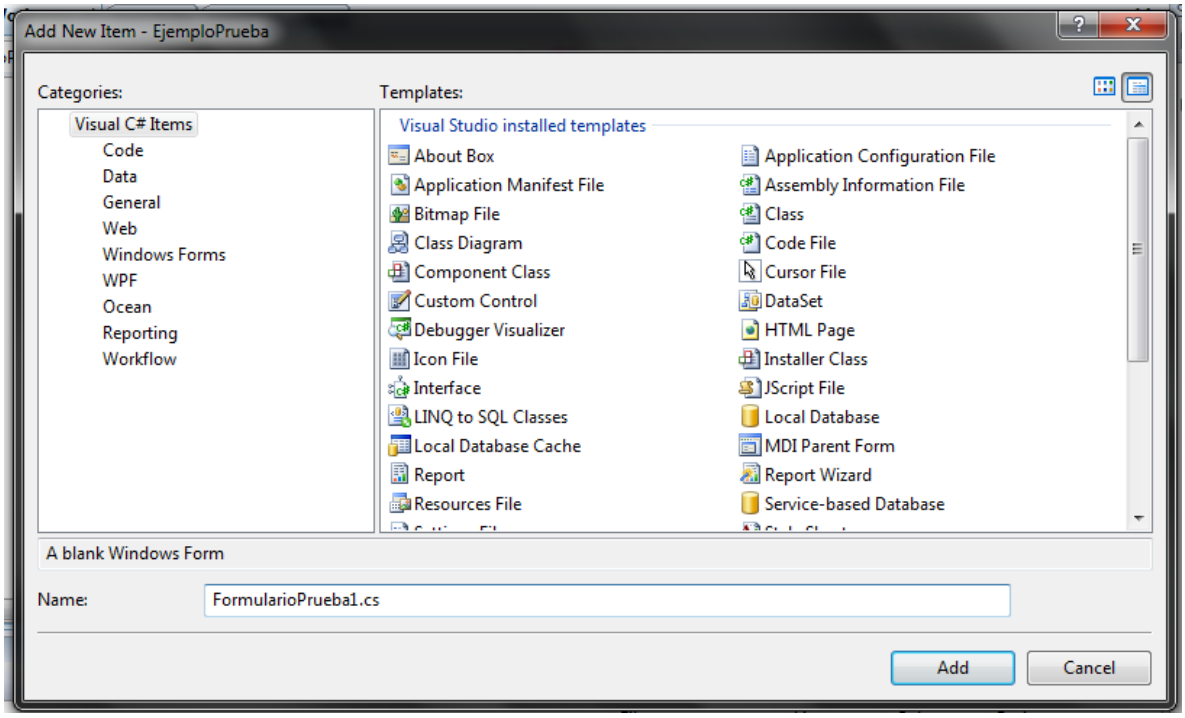


Figura 50 Nuevo WindowsForm

Fuente: autor de este proyecto

Se le asignará nombre a nuestro formulario que para este ejemplo será FormularioPrueba1 y se procede a darle click a Add.

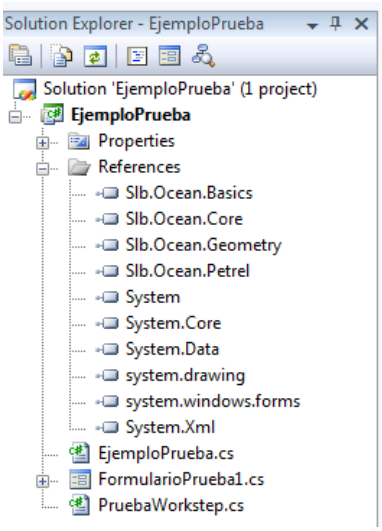


Figura 51 Árbol de clases del proyecto

Fuente: autor de este proyecto

En el árbol del proyecto se visualiza el nuevo formulario creado, sin embargo aún no está enlazado con el proceso de la clase PruebaWorkstep, para enlazar este formulario y poderlo visualizar en PETREL será necesario ir hasta la clase de PruebaWorkstep y buscar el método

```
public override void ExecuteSimple()

{

// TODO: Implement the workstep logic here.

}
```

Este método que es override lo que hace es sobreescribir el método del core de PETREL y asi es como se comunica petrel con sus componentes anexos.

En este método se agregan las siguientes lineas

```
FormularioPrueba1 FormPrueba = new FormularioPrueba1();
```

FormPrueba.Show();

De esta forma se visualiza el formulario que se acabó de crear pero aun no hace nada, agregamos los controles pertinentes y luego se procede a crear la clase AccesoDatosPetrel.

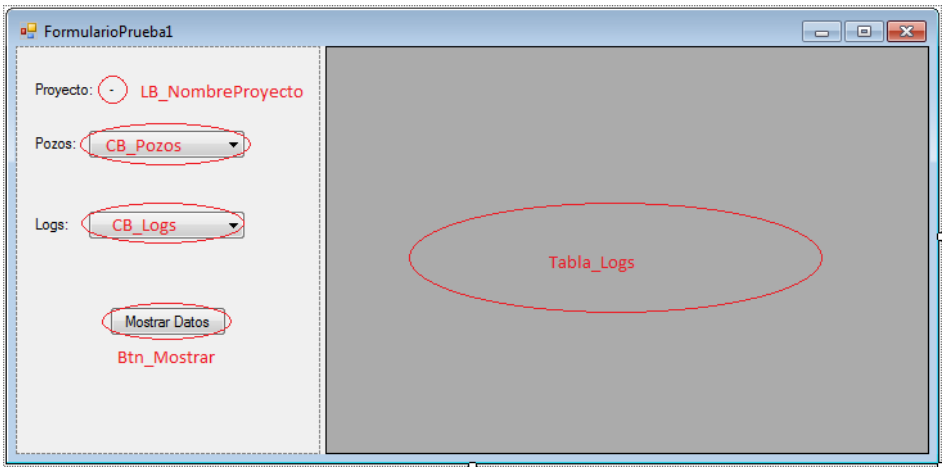


Figura 52 Diseño formulario

Fuente: autor de este proyecto

Ya en este punto se crea una clase que será la encargada del acceso a los datos de PETREL “DatosPetrel.cs”, y otra que será la encargada de almacenar esos datos en memoria para su uso “CentralDatos.cs”, esta última aplicara el patrón de diseño Singleton para que sus datos siempre sean los mismos no importa desde donde sean utilizados.

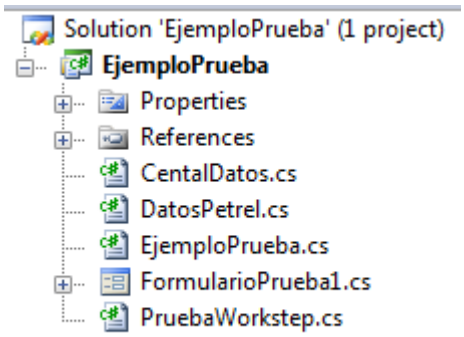


Figura 53 Árbol solución Ejemplo prueba

Fuente: autor de este proyecto

La clase CentralDatos será una clase a la que se aplica el patrón de diseño singleton por lo tanto su declaración será especial y en ella se manejan las variables necesarias para resolver este ejemplo.

```
using System.Text;


using Slb.Ocean.Petrel;

using Slb.Ocean.Petrel.DomainObject;

using Slb.Ocean.Petrel.DomainObject.Well;

using Slb.Ocean.Core;

using Slb.Ocean.Petrel.UI;


namespace EjemploPrueba

{

 public class CentalDatos

 {

 private static CentalDatos datos;


 private CentalDatos()

 {


 }


 public static CentalDatos Datos

 {

 get
```

```
{
if (datos == null)
{
datos = new CentalDatos();
}
return datos;
}
}
public int TamañoDatos;
public string NombreProyecto;
public List<String> ListaNombrePozos = new List<string>();
public List<String> ListaNombreLogs = new List<string>();
public List<double[]> ValorCurva = new List<double[]>();
public Project proyecto;
public WellRoot raiz;
public BoreholeCollection BhC;
}
}
```

En la clase DatosPetrel se agregan las referencias a los using básicas que se necesitan para acceder a los datos de un proyecto PETREL

```
using Slb.Ocean.Petrel;
using Slb.Ocean.Petrel.DomainObject;
using Slb.Ocean.Petrel.DomainObject.Well;
using Slb.Ocean.Core;
using Slb.Ocean.Petrel.UI;
```

dentro de la clase se inicia una conexión con la clase CentralDatos

```
CentalDatos Datos = CentalDatos.Datos;
```

Por ser una clase singleton no se le dará new a su declaracion.

Se crea el método AccesarDatos () el cual sera público

```
public void AccesarDatos()
{
}
```

En este método se agregan las siguientes lineas de codigo.

```
//busco el proyecto principal de PETREL
Datos.proyecto = PetrelProject.PrimaryProject;
//Guardo en un string el nombre del proyecto
Datos.NombreProyecto = Datos.proyecto.Name;
//Me ubico en la raiz del proyecto
Datos.raiz = WellRoot.Get(Datos.proyecto);
//copio las colecciones de pozo en el proyecto
Datos.BhC = Datos.raiz.BoreholeCollection;
```

Con estas lineas acceso a al proyecto actualmente abierto en PETREL y se copian sus colecciones de datos a una instancias de BoreholeCollection llamada BhC, lo siguiente es limpiar la lista de nombres de los pozos.

```
//limpio la lista de pozos
Datos.ListaNombrePozos.Clear();
```

Ahora si puedo empesar a recorrer las colecciones y almacenar en ListaNombrePozos los nombres de los pozos del proyecto actual de PETREL.

```
//recorro los pozos del proyecto
foreach (Borehole bh in Datos.BhC)
```

```
{
        Datos.ListaNombrePozos.Add(bh.Name);
}
```

Por medio de un foreach accedo a los borehole y por medio de su método name obtengo y almaceno el nombre de cada uno de los pozos disponibles en el proyecto PETREL en la variable ListaNombrePozos de la clase singleton CentralDatos.

En el código de FormularioPrueba crearemos un método el cual será convocado desde el constructor del formulario para que sea el encargado del llenar automáticamente con los nombres de los pozos el ComboBox CB_Pozos.

```
private void CargarDatosPozo()
{
        DPetrel.AccesarDatosPozo();
 CB_Pozos.Items.Clear();
 for (int i = 0; i < Datos.ListaNombrePozos.Count; i++)
 {
         CB_Pozos.Items.Add(Datos.ListaNombrePozos[i]);
 }
 if (CB_Pozos.Items.Count>0)
         CB_Pozos.SelectedIndex = 0;
}
```

DPetrel es una instancia de la clase DatosPetrel, Datos es una instancia de la clase CentralDatos, ya en este punto se tienen los nombres de los pozos en una lista y el usuario podrá elegir entre que pozo desea trabajar, seleccionando uno se reescribirá la lista de CB_Logs la que contiene todos los log de dicho pozo seleccionado.

En la clase DatosPetrel se crea un método para extraer los nombre de todos los logs pertenecientes a ese pozo, como dato de entrada se necesita el índice seleccionado del CB_Pozos para así comprobar en que pozo se va a trabajar.

```
public void AccesarDatosLogs(int indice)

{

}
```

Dentro de este método lo primero que se hará será limpiar la lista en la que vamos a guardar los nombres de los Logs.

```
//Limpio la lista de Logs

Datos.ListaNombreLogs.Clear();
```

Se accede a los pozos disponibles del proyecto, y se rrecorren por medio de un foreach.

```
foreach (Borehole bh in Datos.BhC)

{

}
```

Por medio del método bh.name obtengo el nombre del pozo en el que se encuentra el ciclo, verificamos por medio de un if si el nombre del pozo es el mismo que el nombre seleccionado en el CB_Pozos

```
if (bh.Name == Datos.ListaNombrePozos[indice])

{

}
```

El ciclo del foreach continuará hasta encontrar el pozo seleccionado en CB_Pozos, en este momento se procede a extraer los nombres de los welllogs o los registros del pozo seleccionado, por medio de otro foreach se recorren cada uno de los logs y por medio del método log.name se obtienen el nombre de cada uno de los registros para almacenarlos en la clase CentralDatos.

```
foreach (WellLog log in bh.Logs.WellLogs)

{

        Datos.ListaNombreLogs.Add(log.Name);

}
```

Al final del ciclo for each se pondrá un break para salir de estos ciclos y no seguir haciendo comparaciones inecesarias.

```
break;
```

una vez tenemos este método terminado, en la lista ListaNombresLogs de la clase CentralDatos quedara guardado los nombres de los registros del pozo seleccionado, ahora en el código del formulario creamos un evento para que se active cuando el índice seleccionado de la lista CB_Pozos es modificado en este método usaremos la instancia DPetrel para llamar el método que acabamos de crear enviándole el índice del pozo seleccionado.

```
private void CB_Pozos_SelectedIndexChanged(object sender, EventArgs e)

{

        DPetrel.AccesarDatosLogs(CB_Pozos.SelectedIndex);

}
```

Ahora solo falta llenar el CB_Logs con los nombres de los logs del pozo seleccionado, pero antes borraremos cualquier ítem existente en el CB_Logs, se usa un ciclo for para recorrer la lista y por medio del método Add agregamos uno a uno los registros del pozo seleccionado además se almacenará el valor del tamaño de los datos de registro.

```
CB_Logs.Items.Clear();

for (int i = 0; i < Datos.ListaNombreLogs.Count; i++)

{

        CB_Logs.Items.Add(Datos.ListaNombreLogs[i]);

        Datos.TamañoDatos = log.SampleCount;

}

if (CB_Logs.Items.Count > 0)

        CB_Logs.SelectedIndex = 0;
```

En este punto ya tenemos las dos listas disponibles, la de los pozos disponibles y la de los registros disponibles del pozo seleccionado ahora codificaremos el método de extraer los datos del registro seleccionado en DatosPetrel, este método tendrá como datos de entrada los índices de los elementos seleccionados de las dos listas CB_Pozos y CB_Logs.

```
public void ExtraerDatosLogs(int indicePozo, int indiceLogs)

{

}
```

Lo primero que haremos será borrar la lista de ValorCurva

```
Datos.ValorCurva.Clear();
```

Procedemos ha realizar la busqueda del pozo y el log seleccionado en los combo box CB_Pozos y CB_Logs por medio de los indices seleccionados.

```
foreach (Borehole bh in Datos.BhC)

{

        if (bh.Name == Datos.ListaNombrePozos[indicePozo])

        {

                foreach (WellLog log in bh.Logs.WellLogs)

                {

                        if (log.Name == Datos.ListaNombreLogs[indiceLogs])

                        {

                        }

                }

        }

}
```

Una vez el ciclo se cumple encontraremos el registro que es usuario esta pidiendo observar por lo tanto procedemos a extraer los datos de petrel y almacenarlos en una lista en la clase CentralDatos.

```
double[] Vector = new double[Datos.TamañoDatos];

double[] Vector2 = new double[Datos.TamañoDatos];

int indice = 0;
```

```
foreach (WellLogSample Sample in log.Samples)

{

        Vector2[indice] = Math.Round(Sample.MD, 2);

        Vector[indice] = Math.Round(Sample.Value, 2);

        indice++;

}

Datos.ValorCurva.Add(Vector2);

Datos.ValorCurva.Add(Vector);

break;
```

una vez tenemos el método terminado, se programa el evento click del boton BT_Mostrar

```
private void BT_Mostrar_Click(object sender, EventArgs e)

{

}
```

En este método mostraremos en un datagridview los datos de la curva que hemos seleccionado y que estan almacenados en la clase CentralDatos, pero andes deberemos invocar el método de DatosPetrel que me extrae los datos de la curva seleccionada

```
DPetrel.ExtraerDatosLogs(CB_Pozos.SelectedIndex, CB_Logs.SelectedIndex);
```

Ahora si mostramos los datos en una tabla.

```
Tabla_Logs.Rows.Clear();

Tabla_Logs.ColumnCount = 2;

string[] Fila = new string[Datos.ValorCurva.Count];

for (int i = 0; i < Datos.TamañoDatos; i++)

{

        for (int j = 0; j < Datos.ValorCurva.Count; j++)
```

```
            {
                Fila[j] = Datos.ValorCurva[j][i].ToString("0.##");
            }
            Tabla_Logs.Rows.Add(Fila);
            Tabla_Logs.Rows[i].HeaderCell.Value = (i + 1).ToString();
        }
        Tabla_Logs.AutoResizeRowHeadersWidth(DataGridViewRowHeadersWidthSizeMode.AutoSizeToAllHeaders);
```

7.2. Observando los resultados

Una vez terminado el plug-in se compila el codigo y se ejecuta una vez la compilacion sea exitosa, cuando inicie petrel crearemos o abriremos un proyecto con pozos y datos del registos en los pozos, despues se comprueba que el plug-in este funcionando correctamente, le damos doble click al plug-in y esperamos nos muestre el formulario de informacion.

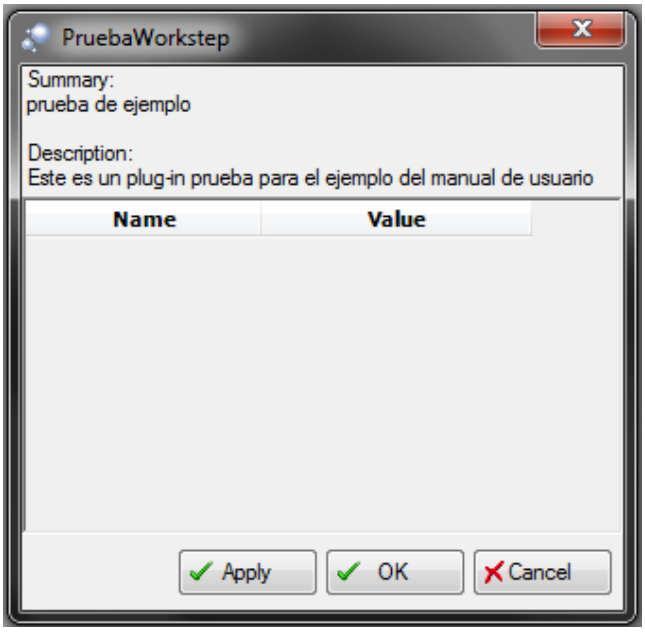


Figura 54 Ventana información Ejemplo prueba

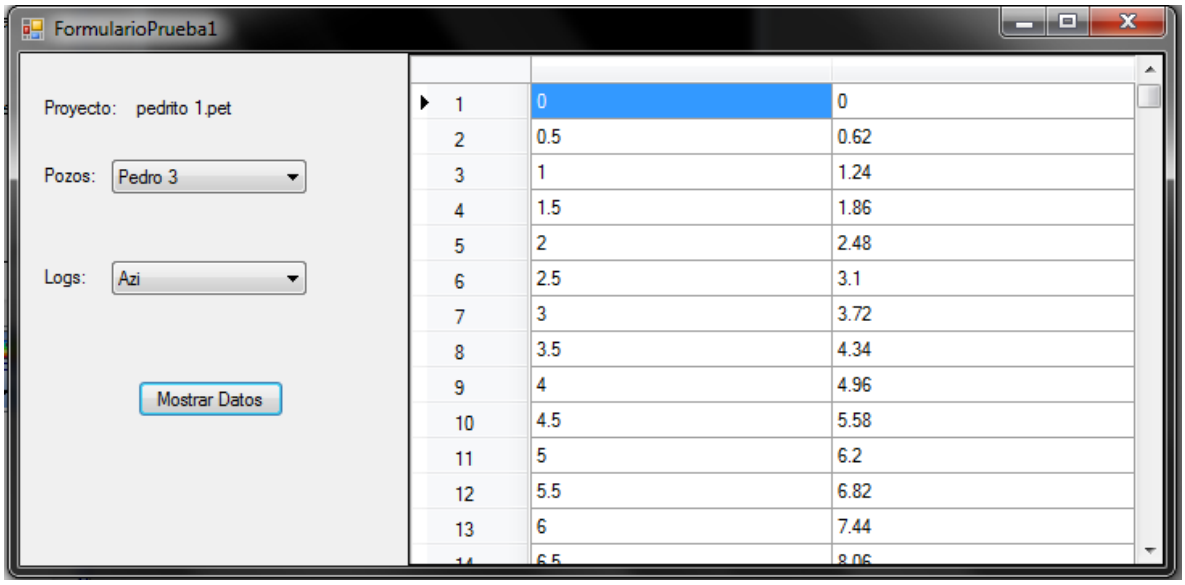
Fuente: autor de este proyecto

Damos click en el botos OK para iniciar el plug-in veremos la siguiente pantalla, ya con los datos de el proyecto actulmente abierto en petrel.

Figura 55 Ventana Formulario prueba

Fuente: autor de este proyecto

Solo basta eleguir el pozo y el log que deseamos visualizar y dar click en mostrar datos.



1	0	0
2	0.5	0.62
3	1	1.24
4	1.5	1.86
5	2	2.48
6	2.5	3.1
7	3	3.72
8	3.5	4.34
9	4	4.96
10	4.5	5.58
11	5	6.2
12	5.5	6.82
13	6	7.44

Figura 56 Ventana datos salida

Fuente: autor de este proyecto