UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 9

Trabajo de diploma para optar por el Título de

Ingeniero en Ciencias Informáticas.

Subsistema de configuración y seguridad para el sistema de control de la Empresa de Perforación y Extracción de Petróleo de Occidente.

Autor: Antonio Antelo Vázquez.

Tutor: Ing. Yunier Alexander Pimienta Fernández.

Co-tutor: Msc. Antonio Antelo Hunt.

Ciudad de la Habana, 2010.
Año 52 de la Revolución.

PENSAMIENTO

VALE MÁS SABER ALGUNA COSA DE TODO, QUE SABERLO TODO DE UNA SOLA COSA.

Blaise Pascal.

AGRADECIMIENTOS

Un agradecimiento especial a mi familia que me ha apoyado en todas las decisiones importantes de mi vida.

Deseo dejar plasmado mi reconocimiento a todos los profesores que participaron en mi formación profesional.

Mi gratitud y reconocimiento al profesor Sardiñas que fue la primera persona que descubrió mi amor por la Matemática, con su apoyo y ayuda he logrado éxitos en la misma.

Agradecerle a mi padre por los magníficos consejos que me ha dado, los cuales he utilizado en el desarrollo de esta investigación.

A mi tutor Pimienta el cual me ha apoyado y me ha ayudado en la realización de este trabajo.

Mi agradecimiento a todas las personas que de una manera u otra me han ayudado y me han apoyado en la realización de este trabajo.

A todos

MUCHAS GRACIAS.

DEDICATORIA

Dedico especialmente este trabajo a la Revolución Cubana y a nuestro Comandante en Jefe Fidel Castro Ruz.

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmamos la presente a los ___ días del mes de _____ del año 2010.

_____	_____
Antonio Antelo Vázquez (Autor)	Ing. Yunier Alexander Pimienta (Tutor)

RESUMEN

En la Empresa de Perforación y Extracción de Petróleo de Occidente (EPEPO) perteneciente a la Unión CUBAPETROLEO (CUPET), se operan un gran volumen de datos e informaciones que se generan en los procesos productivos con métodos pocos eficientes que no responden a la complejidad, ni a la precisión que requiere el flujo informativo en la actualidad, lo que ha provocado inconvenientes en la organización y evaluación de la misma. Ante estas dificultades se contactó con la UCI para lograr el diseño de un sistema que permita el control de la información que se genera en los procesos de extracción y producción de la entidad EPEPO; el cual aún carece del subsistema de configuración y seguridad para lograr la funcionalidad del control informático automatizado.

El presente trabajo de diploma contiene un estudio de los principales enfoques sobre la implementación de subsistemas de configuración y seguridad. El desarrollo de esta implementación en particular se sustentada en los principales requerimientos del sistema de control de la información concebido, lo que ha permitido obtener como resultado la propuesta de un subsistema que permita la gestión de todos los parámetros de configuración y seguridad de dicho sistema, el cual permitirá optimizar el proceso de control de información. La solución planteada esta delimitada en la implementación de un subsistema de Configuración y Seguridad utilizando el lenguaje de programación Java y gestor de base de datos Postgres. El éxito de la misma depende en gran medida de los patrones de diseños implementados en su desarrollo, así como, del cumplimiento de la arquitectura concebida por el arquitecto del proyecto. Esta implementación será tomada como punto de partida para la los demás subsistema, debido a que con esta se sustentarán las bases de la arquitectura.

Palabras claves: **Implementación, Configuración y Seguridad.**

ÍNDICE

Introducción.

Hoy día los procesos de gestión y sistemas de la información han adquirido un papel protagónico en todas las esferas de la sociedad y con singular énfasis en los flujos de informaciones que se generan en el proceso productivo de las empresas, lo que permite la rápida recopilación, procesamiento, análisis y evaluación de la información. Gracias al desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) estos procesos son concebidos en sistemas automatizados. Por tanto, la ciencia de la computación, está presente en toda la esfera de la sociedad y ya sin ella no es posible ningún avance en los procesos tecnológicos empresariales.

En Cuba el avance tecnológico también ha adquirido un papel primordial debido a todos los esfuerzos de la Revolución para garantizar y promover el uso de la tecnología en escuelas e instituciones de la sociedad cubana. La creación de la Universidad de las Ciencias Informáticas (UCI), entidad destinada al logro del desarrollo informático en la creación y utilización del software. La UCI hace realidad el uso de la tecnología en los procesos de gestión y producción en diversas empresas de gran impacto en la economía nacional.

La Unión Cuba Petróleo (CUPET), que vinculada al Ministerio de la Industria Básica (MINBAS), es la encargada de satisfacer las necesidades del mercado nacional de hidrocarburos de forma competitiva, a partir del incremento de la producción y la optimización del uso de los combustibles nacionales. Esta cuenta con tres Empresas de Perforación y Extracción de Petróleo (EPEP): la Empresa de Perforación y Extracción de Petróleo del Centro (EPEPC), la Empresa de Perforación y Extracción de Petróleo de Occidente (EPEPO) y la EPEP-Majagua destinadas al control de los procesos de extracción, producción y comercialización del crudo nacional.

Las EPEP tienen como misión la exploración y explotación de yacimientos de petróleo en aras de lograr un aumento de reservar de hidrocarburos, el vertiginoso crecimiento de la producción, la expansión de los procesos de tratamiento y transportación de petróleo y gas para ofertar productos de calidad que cubran las necesidades del mercado actual.

A pesar de los avances, la incorporación de los sistemas automatizados en las empresas no ha sido el más adecuado, como es el caso de la Empresa de Perforación y Extracción de Petróleo de Occidente (EPEPO). En esta entidad se manipula gran cantidad de información referente a los indicadores de producción de dicha empresa mediante la utilización de herramientas ofimáticas tales como: Microsoft Word, Microsoft Excel y Microsoft Access. Lo anteriormente planteado trae consigo que se necesiten gran cantidad de recursos para el control de los indicadores de producción y que existan errores en la realización de los cálculos sobre la producción. El uso de este tipo de herramientas ofimáticas imposibilita el conocimiento real de la producción para la toma de decisiones en momento determinado. Por lo que se contactó con la UCI para lograr el diseño de un sistema que permita el control de la información que se genera en los procesos de extracción y producción de la entidad EPEPO. El mismo aún carece del subsistema de configuración y seguridad para lograr la funcionalidad del control informático automatizado. Por consiguiente, se plantea el siguiente **problema a resolver:** ¿Cómo lograr la funcionalidad y la seguridad del sistema de control informático de la Empresa de Perforación y Extracción de Petróleo de Occidente?

Según el problema identificado anteriormente se plantea como **objeto de estudio:** El proceso de control del flujo de información que se genera durante la etapa de producción en la Empresa de Perforación y Extracción de Petróleo de Occidente.

Para resolver el problema se propone el siguiente **objetivo general:** Implementar el Subsistema de configuración y seguridad para el sistema de control informático del proyecto de la Empresa de Producción y Extracción de Petróleo de Occidente.

El objetivo trazado delimita el siguiente **campo de acción:** La implementación del subsistema de configuración y seguridad en el sistema de control informático del proyecto de la Empresa de Perforación y Extracción de Petróleo de Occidente.

Todo lo antes expuesto conduce a plantear la siguiente **hipótesis:** Si se implementa el subsistema de configuración y seguridad en el sistema de control informático de la Empresa de Perforación y Extracción de Petróleo de Occidente, entonces, se logrará la funcionalidad y la seguridad del mencionado sistema.

Para cumplir con los objetivos de esta investigación y resolver la situación problemática planteada, se proponen las siguientes tareas de la investigación:

1- Estudio del estado del arte.
2- Evaluar las herramientas y tecnologías útiles para el desarrollo de subsistemas de configuración y seguridad del sistema.
3- Implementar los subsistemas de Configuración y Seguridad.
4- Validar los subsistemas diseñados.

Los métodos teóricos utilizados en el proceso de investigación fueron los siguientes: el histórico lógico en la determinación de la evolución histórica del proceso de innovación, la inducción y deducción, el análisis y la síntesis en la caracterización y análisis del objeto y el campo de investigación, y la abstracción científica para arribar a conceptos y generalizaciones.

También se implementó como métodos empíricos la observación directa durante todo el proceso de búsqueda de información, entre las técnicas utilizadas se emplearon: La consulta a miembros del proyecto y la revisión documental.

Este trabajo constituye el resultado de una investigación sustentada en un estudio de caso enfocado hacia la mejora.

Se espera como **aporte práctico**: El establecimiento de un subsistema completo y flexible que logre la configuración y la seguridad del sistema de control informático en las Empresa de Perforación y Extracción de Petróleo de Occidente conforme lo previsto en el proyecto, lo que resuelve la carencia del subsistema.

La presente tesis se estructura en tres capítulos. Un primer capítulo, que está dirigido al análisis de los antecedentes, evolución, así como a las limitaciones que se presentan en la implementación de un subsistema de configuración y seguridad.

El segundo capítulo está encaminado a la conformación y descripción del subsistema de configuración y seguridad en la Empresa de Perforación y Extracción de Petróleo de Occidente.

El tercer capítulo se orienta a la validación de los resultados donde se registra las pruebas que le son realizadas al software en el desarrollo y que el resultado obtenido cumpla los requisitos especificados.

Capítulo I. Fundamentación teórica.

Es indudable que el desarrollo y la consolidación de los sistemas de control informático en las empresas han de estar sustentados en los elementos teóricos y en los principios y criterios contemporáneos de funcionalidad opcional-flexible y segura. Por ello, para resolver el problema en esta investigación, se realizó un estudio del arte desde sus antecedentes hasta la contextualización del mismo. Por consiguiente, en este capítulo se pretende como objetivo:

- ✓ Exponer los principales enfoques y criterios teóricos que sustentan un sistema de control informático para los flujos de información que se generan en las empresas, atendiendo las tendencias globales, así como las limitaciones que se manifiestan en la implementación de la configuración y seguridad en el manejo del mismo.

1.1.- Antecedentes y evolución de los sistemas informáticos en el control de las informaciones. Conceptos y definiciones.

Los cambios y transformaciones acontecidas en el orbe en los últimos años en las tecnologías de la información y las comunicaciones, la necesidad del hombre de encontrar vías rápidas y efectivas en el manejo de la información, ha conducido a un desarrollo progresivo en los sistemas de información y en el control de los mismos. Se impone a su vez la necesidad que los nuevos sistemas de control informáticos en las empresas cuenten con facilidades en su funcionalidad opcional-flexible y a la vez seguros acorde a las necesidades de los clientes.

La tecnología nunca hubiera existido de no ser por el desarrollo del ordenador o computadora. Toda la sociedad utiliza estas máquinas, en distintos tipos y tamaños, para el almacenamiento y manipulación de datos. Los equipos informáticos han abierto una nueva era gracias a las técnicas de automatización, y han permitido mejorar los sistemas modernos de comunicación. Las computadoras son herramientas esenciales prácticamente en todos los campos de investigación y en tecnología aplicada. [1]

Innegablemente en la década de los años 60 del siglo XX, la información adquiere un rol relevante a partir de que la información científica, fue definida por *Borko*: en " *Information Science: What is it?*", en 1968,

donde ofreció definiciones medulares y guías para el progreso de la nueva ciencia. *Borko*, definió la *Information Science* como una "ciencia interdisciplinaria que investiga las propiedades y el comportamiento, flujo y uso de la información, así como las técnicas, manuales y mecánicas, del proceso informativo para su más eficaz almacenamiento, recuperación, análisis y diseminación". [2]

Evidentemente cualquier análisis teórico de los sistemas informáticos, necesariamente han requerido de la definición de términos y conceptos que han tenido lugar en su progreso evolutivo, los cuales se expondrán progresivamente, así como los manejados en la implementación del subsistema para la solución del problema planteado.

Hoy, muchos de los autores argumentan que este éxito de los sistemas informáticos y su control está basado en la creatividad humana ante la diversidad de complejidades, la cual a su vez, está condicionado por la capacidad innovadora y de desarrollo de las organizaciones, porque es obvio que el progreso de los sistemas informáticos ha evolucionado gracias a los avances de la ciencia y el desarrollo tecnológico de las informática y el conocimiento los cuales han generado cambios progresivos favorables.[3]

Los antecedentes en la experiencia internacional y en el contexto cubano demuestran que los diseños de los sistemas informáticos cada día presentan más complejidad y demandan mayores requerimientos funcionales y de seguridad en la operatividad de las organizaciones. Es por ello se hace vital que el manejo automatizado de las informaciones sea integral y responda a las exigencias actuales.

En el período actual se ha evolucionado en lo referente a la manipulación de la información. Una etapa llamada la sociedad de la información, donde el factor fundamental del adelanto es el conocimiento. Una nueva sociedad, con organizaciones fundamentadas en el aprendizaje, cuyo capital más preciado es el ser humano, sustentada en un desarrollo científico-tecnológico sin precedentes provocado por el cambio en la informática y las comunicaciones en el progreso del desarrollo económico y social del mundo.

El cambio es la base del progreso, es un salto, que está dado por la acumulación de acontecimientos cuantitativos y cualitativamente superior a los anteriores, por ello todo cambio es un proceso de avance en forma de espiral previamente concebido. Por tanto todo sistema está sujeto al perfeccionamiento, a la mejora, a la adaptación, o sea, al cambio conforme las expectativas y demandas; entiéndase por Sistema al conjunto organizado de cosas o partes interdependientes, que se relacionan formando un todo unitario y

complejo, que siempre estará relacionado con el contexto que lo rodea. Mientras que los Subsistemas son las partes o cosas del sistema que conforman un todo. Por tanto, las partes que componen al sistema, no se refieren al campo físico (objetos), sino más bien al funcional. De este modo las cosas o partes pasan a ser funciones básicas realizadas por el sistema. Estas se pueden enumerar en: entradas, procesos y salidas.

La información es un elemento fundamental para el desarrollo. La misma ocupa, cada vez más, un espacio mayor en la economía de los países a escala mundial. No solo en la economía sino en las instituciones, negocios o empresas. De aquí la importancia de un sistema de información, el que constituye conjunto de elementos que interactúan entre sí con el fin de apoyar las actividades de una empresa o negocio. Por tanto un sistema de información realiza cuatro actividades básicas: entrada, almacenamiento, procesamiento y salida de información, o sea, tienen lugar varias actividades o acciones en un proceso. [2]

Por sistema se puede comprender también a un conjunto de funciones que operan armónicamente con un fin en el que interactúan sus elementos o subsistemas componentes. Todo sistema posee sus normas y requerimientos de funcionamiento que posibilita la coherencia operativa.

Lógicamente el hombre en el contexto de la sociedad de la información esta precisado del perfeccionamiento cada día, del control del flujo de informaciones que se generan en el sector empresarial. Por ende, la utilización de sistemas que permitan adquirir, producir, analizar y transmitir, al menor costo posible, datos e información con una calidad, seguridad y exactitud suficientes, a modo de garantizar el funcionamiento exitoso de la organización en cuestión.

El control de la información constituye un proceso de gran importancia, que permite organizar y corregir oportunamente con validez y seguridad los procesos y comportamientos, atendiendo las metas de la empresa o negocio. Si se tiene en cuenta que las principales funciones de dirección son; la planificación, la organización, la dirección o mando y el control, entonces, se puede comprender el papel esencial que tiene el control y sobre todo, los sistemas de control informático en la autorregulación, análisis de los flujos de información que tienen lugar en las empresas y que de hecho inciden en la toma de decisión adecuada.[4]

El control ha sido definido como la verificación, monitoreo y seguimiento de los objetivos previamente concebidos. Todo esto lleva a pensar que el control es un mecanismo que permite corregir desviaciones a través de indicadores cualitativos y cuantitativos dentro de un contexto de desempeño organizacional. El concepto de control informático parte de este concepto general y específicamente es al referido al monitoreo y evaluación de los sistemas informáticos y puede ser utilizado en el contexto organizacional para evaluar el desempeño y proporcionar los elementos para la toma de decisión. El control se basa en la consecución de las siguientes actividades: planear y organizar, hacer, evaluar y mejorar. [5]

1.2.- El proceso de configuración y seguridad en los sistemas informáticos.

El proceso de configuración y seguridad en los sistemas informáticos que tienen por misión la recopilación y evaluación de las informaciones que se generan en los flujos productivos, están llamados a garantizar la funcionalidad opcional-flexible y seguridad operativa del sistema en cuestión. En la actualidad no es posible la utilización de un sistema informático que no cuente con un subsistema de configuración y seguridad, ya que los mismos deben favorecer con facilidad operativa del sistema al cliente.

1.2.1.- La configuración en los sistemas de control informático

En su diseño e implementación la configuración debe concebirse como un proceso. En el cual se debe tener en cuenta las reglas y requerimientos del sistema que va a formar parte, para garantizar el funcionamiento operativo con facilidades al cliente o usuario.

La configuración está constituida por varios componentes y elementos que proporcionan la operatividad y vías opcionales de necesidades de uso. Por tanto, es un conjunto de datos que determina el valor de algunas variables de un programa o sistema de software. Las opciones generalmente son cargadas en su inicio y en algunos casos se deberá reiniciar para poder ver los cambios, ya que el programa no podrá cargarlos mientras se esté ejecutando, si la configuración aún no ha sido definida por el usuario (personalizada), el programa o sistema cargara la configuración por defecto (predeterminada). Estas son consideraciones a tener en cuenta en la implementación de la configuración en un sistema. [6]

También hay que considerar que la configuración puede ser el conjunto de ítems, elementos o productos que se generan en un proyecto, como parte de la solución diseñada; mientras que la gestión de la

Configuración es el proceso de administrar el nacimiento y evolución de los ítems que se generan en un proyecto. Este proceso incluye la identificación, registro, resguardo, contabilización, control de cambios y auditoría de los ítems de la configuración; por ello el Gestor de la configuración es la persona del equipo del proyecto que se encarga de [6]:

- ✓ Identificar los ítems de la configuración
- ✓ Asignarles nombres de acuerdo a ciertas reglas
- ✓ Archivar los documentos y productos en la carpeta o directorio del proyecto
- ✓ Asegurar la integridad física de los ítems mediante respaldos y otros mecanismos de seguridad.

La configuración puede estar predeterminada o personalizada, compréndase que la Configuración predeterminada, es la que no se ha definido aún, generalmente no es la más recomendada, ya que por ese mismo motivo se le da la posibilidad al usuario de modificarla, una configuración predeterminada tiene que estar preparada para [6]:

- ✓ Usuarios de todas las edades y ambos sexos.
- ✓ Generalmente en inglés.
- ✓ Nivel gráfico medio.
- ✓ Seguridad media.

En el diseño e implementación de esta configuración predeterminada, debe preverse que la misma cumpla con la condición que sea lo más adaptable posible, pero siempre es mejor poseer una configuración personalizarla para adaptarla a las necesidades de usuarios.[6] Una configuración personalizada es la definida especialmente por el usuario, esta es guardada generalmente en un archivo o en una base de datos, puede estar cifrada para que solo se pueda modificar por el programa a configurar, o puede ser texto plano para que también se pueda modificar sin depender del programa (esto sucede más frecuentemente en sistemas Unix).

Otro de los aspectos a tener en cuenta son los errores de configuración, los cuales son generados por una escritura incorrecta de las líneas del archivo de configuración o que el hardware este limitado a una configuración que no requiera de tantos recursos como esta precisa. Esto conlleva a una ejecución defectuosa del programa o sistema operativo o a la imposibilidad de ejecutarse. Para evitar errores de

configuración, es importante leer los requerimientos mínimos de una configuración y que estos igualen o estén por debajo de los del hardware. [7]

1.2.2. La seguridad en los sistemas de control informático

Durante las primeras décadas de la existencia de las computadoras, estas fueron usadas principalmente por investigadores y otros sin una gran masificación. En estas condiciones, la seguridad no recibió mucha atención. Hoy día, cuando millones de ciudadanos comunes usan la computadora y el intercambio de información a través de las redes en todas las latitudes del mundo, la seguridad en los sistemas computacionales e informáticos aparece como un problema potencial de grandes proporciones.

En su forma más sencilla, la seguridad se ocupa de garantizar que las personas no puedan leer o modificar información. Tiene por propósito proteger e impedir el acceso no autorizados, así como verifica la autenticidad de la información. La mayoría de los problemas de seguridad son causados intencionalmente por gente maliciosa que intenta ganar algo o hacerle daño a alguien. [8]

La seguridad de los sistemas de control informáticos accionan en cuatros direcciones fundamentalmente interrelacionadas: confidencialidad, autenticación, no repudio y control de integridad. La confidencialidad consiste en mantener la información fuera del alcance de los usuarios no autorizados. La autenticación se encarga de determinar quién recibirá, antes de revelar la información confidencial. El no repudio se encarga de las firmas, o sea, confirmación del que utilizó o manipuló la información. La integridad se encarga de comprobar la veracidad de la información.

Por lo que se puede deducir que “Seguridad Informática”, es la disciplina, técnicas y herramientas para ayudar a proteger la confidencialidad, integridad y disponibilidad de la información y los sistemas, o sea, garantizar que los recursos informáticos de una compañía susceptible de robo, pérdida o daño, ya sea de manera personal, grupal o empresarial, estén disponibles para cumplir sus propósitos, que no estén dañados o alterados por circunstancias o factores externos. [9]

La seguridad en un sistema operativo es el conjunto de garantías que brinda el mismo para proteger la confidencialidad, integridad y disponibilidad de la información y los sistemas. [10]

Criptografía: Es un método más para proteger partes de ficheros, o el fichero entero.[10]

La seguridad de computadoras y redes aborda los siguientes cuatro requisitos:[11]

- ✓ **Secreto:** Exige que la información de un sistema de computadoras sea accesible para lectura solamente por partes autorizadas. Este tipo de acceso incluye impresión, visualización y otras formas de revelación.
- ✓ **Integridad:** Exige que los elementos de un sistema de computadoras puedan ser modificados solo por partes autorizadas. La modificación incluye escritura, cambio de estado, borrado y creación.
- ✓ **Disponibilidad:** Exige que los elementos de un sistema de compradores estén disponibles para las partes autorizadas.
- ✓ **Autenticidad:** Requiere que un sistema de computadoras sea capaz de verificar la identidad de un usuario.

Los tipos de amenazas se dividen en cuatro categorías:[11]

Interrupción: Se destruye un elemento del sistema o se hace inaccesible o inútil. Este es un ataque a la disponibilidad. Como ejemplo se incluyen la destrucción de una pieza del hardware, como un disco duro.

Interceptación: Una parte no autorizada consigue acceder a un elemento. Este es un ataque al secreto. La parte no autorizada puede ser una persona o una computadora. Como ejemplo se incluyen la intervención de las conexiones telefónicas para conseguir datos de una red y la copia ilícita de archivos o programas.

Modificación: Una parte no autorizada no sólo consigue acceder, sino que falsifica un elemento. Este es un ataque a la integridad. Como ejemplo se incluyen el cambio de valores en un archivo de datos, la alteración de un programa para que se comporte de manera diferente y la modificación del contenido de los mensajes transmitidos en una red.

Invención: Una parte no autorizada inserta falso objeto en el sistema. Este es también un ataque a la autenticidad. Como ejemplo se incluyen la inserción de mensajes falsos en una red o la adición de registros a un archivo.

Se suelen dividir las amenazas que existen sobre los sistemas informáticos en tres grandes grupos, en función del ámbito o la forma en que se pueden producir [11]:

Desastres del entorno. Dentro de este grupo se incluyen todos los posibles problemas relacionados con la ubicación del entorno de trabajo informático o de la propia organización, así como con las personas que de una u otra forma están relacionadas con el mismo. Por ejemplo, se han de tener en cuenta desastres naturales (terremotos, inundaciones...), desastres producidos por elementos cercanos, como los cortes de fluido eléctrico, y peligros relacionados con operadores, programadores o usuarios del sistema.

Amenazas en el sistema: Bajo esta denominación se contemplan todas las vulnerabilidades de los equipos y su software que pueden acarrear amenazas a la seguridad, como fallos en el sistema operativo, medidas de protección que éste ofrece, fallos en los programas, copias de seguridad.

Amenazas en la red: Cada día es menos común que una máquina trabaje aislada de todas las demás; se tiende a comunicar equipos mediante redes locales, intranets o la propia Internet, y esta interconexión acarrea nuevas y peligrosas amenazas a la seguridad de los equipos, peligros que hasta el momento de la conexión no se suelen tener en cuenta. Por ejemplo, es necesario analizar aspectos relativos al cifrado de los datos en tránsito por la red, a proteger una red local del resto de Internet, o a instalar sistemas de autenticación de usuarios remotos que necesitan acceder a ciertos recursos internos a la organización (como un investigador que conecta desde su casa a través de un módem).

Se pueden identificar tres clases de intrusos [11]:

Suplantador: Un individuo que no está autorizado a usar la computadora y que penetra en un sistema de control de acceso para explotar una legítima entrada de usuario. El suplantador es probablemente una persona ajena, el abusador es generalmente alguien interno y el usuario clandestino puede ser tanto externo como interno.

Abusador: Un usuario legítimo que accede a datos, programas o recursos a los que no está autorizado, o que está autorizado pero hace mal uso de sus privilegios.

Usuario Clandestino: Un individuo que está a cargo del control de supervisión del sistema y utiliza este control para evadir la auditoría y el control de acceso o para suprimir la recopilación de datos auditados.

Los ataques de intrusos varían desde benignos hasta serios. En el benigno, hay mucha gente que simplemente desea explorar las redes y ver que hay ahí fuera. En el extremo serio están los individuos que intentan leer datos privilegiados, realizar modificaciones no autorizados a los datos o trastornar el sistema.

Algunas de las técnicas que suelen utilizar los hackers son [11]:

- ✓ Probar las contraseñas por omisión empleadas en las cuentas estándares que se suministran junto al sistema.
- ✓ Probar exhaustivamente todas las contraseñas cortas (de uno a tres caracteres).
- ✓ Probar palabras del diccionario del sistema o de una lista de contraseñas probables.
- ✓ Reunir información sobre los usuarios, como sus nombres completos, los nombre de su esposa e hijos, cuadros de la oficina y libros de su oficina relativos a aficiones.
- ✓ Probar los números de teléfono de los usuarios, sus números de DNI y números de habilitación.
- ✓ Probar todos los números legales de matrículas del estado.
- ✓ Emplear un caballo de Troya para saltarse las restricciones de acceso.
- ✓ Intervenir la línea situada entre un usuario remoto y el sistema anfitrión.

Regularmente los problemas de seguridad vienen dados por atacantes internos a la organización afectada. En organismos los atacantes suelen ser personal interno, así como por piratas externos a la entidad que aprovechan de una negligencia.

Las amenazas no siempre deben ser vistas como actos intencionados contra el sistema: muchos de los problemas pueden ser ocasionados por accidentes.

1.3. Tendencias, Herramientas y Tecnologías útiles para la implementación del subsistema de configuración y seguridad en los sistemas de control informáticos de empresas.

Existen diversas herramientas y metodologías que sirven como apoyo en la implementación de cualquier software informático. Seleccionar la más adecuada y útil para un propósito es tarea de las personas vinculadas al desarrollo del mismo. Para la implementación del subsistema de configuración y seguridad el arquitecto del proyecto (Véase la tesis del Arquitecto) realizó una investigación donde seleccionó un conjunto de tendencias, herramientas y tecnologías para hacer que el desarrollo de este subsistema sea homogéneo.

1.3.1. Tendencias, Herramientas y Tecnologías Comparadas. Valoración de la selección.

En la investigación realizada por el arquitecto (Véase la tesis del Arquitecto) se establecen cuatro elementos los cuales son:

- ✓ Metodología de desarrollo.
- ✓ Herramientas CASE.
- ✓ Sistemas Gestores de Base de Datos (SGBD o DBMS).
- ✓ Lenguajes de Programación.

Se puede decir que de las metodologías se estableció la comparación entre Microsoft Solutions Framework (MSF), Rational Unified Process (RUP) y Programación Extrema (Extreme Programming, XP) siendo estas las tecnologías más famosas de su tipo. Dentro de las Herramientas CASE se compararon el Visual Paradim y Rational Rose, formando estas un subconjunto limitado pero a la vez el más usado en la universidad y en el mundo. De los gestores de Base de Datos se contrastaron tres gestores fundamentales SQL Server, Oracle y Postgres. Y por último se realizó un estudio sobre Lenguajes de Programación entre los cuales se encuentra C++, C# y Java.

Vale destacar además el estudio el Lenguaje Extensible de Marcas (XML) en el transcurso del desarrollo de esta investigación ya que es utilizado en ficheros de configuración mayormente, entre otras cosas más.

Una vez mostrado todo el conjunto de tendencias, herramientas y tecnologías que se compararon, es hora de mostrar la selección realizada para la creación del sistema. Entre tantas metodologías se decidió escoger RUP debido a que esta es una metodología usada y probada ya por la universidad y además por su ventaja ante las demás, esta selección es correcta debido a que en caso utilizar la metodología Programación Extrema se debe tener a un integrante por parte del cliente trabajando con el equipo de desarrollo, lo cual se ha querido pero ha sido imposible.

Para apoyar a la metodología RUP seleccionada se escogió como herramienta CASE el Visual Paradigm considerando que esta herramienta contiene una buena semántica y una organización dirigida a diagramas lo cual permite una mejor organización de la documentación. También permite generar código y documentar los diagramas. Esta además, presenta una fuerte integración con el Gestor de Base de Datos Postgres el cual fue seleccionado entre los demás gestores. Postgres es libre óptimo, contiene un conjunto de posibilidades que lo hace una alternativa viable para el desarrollo del sistema.

De los lenguajes de programación se seleccionó el Java el cual es orientado a objeto. Para un mejor desarrollo del sistema existen framework implementados en este lenguaje los cuales suelen ser implementaciones de patrones de diseño conocidos, aderezados con funciones que asisten al desarrollador y brindan una base sólida sobre la cual desarrollar aplicaciones concretas permitiendo obviar los componentes más triviales y genéricos del desarrollo. Tal es el caso del framework Spring [1]que permite separar la lógica del negocio del acceso a datos y de la presentación. En la presentación se decidió utilizar el framework Swing[2], ya viene acoplado con el NetBeans[3], herramienta también escogida para el desarrollo del sistema. En el acceso a datos se escogió Hibernate[4].

De los patrones se utilizarán todos los que estos framework anteriormente mencionado nos provean para lograr un subsistema altamente reutilizable y adaptable a la necesidades existentes.

[1] Plataforma líder para la construcción y ejecución de aplicaciones empresariales implementadas en Java, incrementando la productividad de desarrollo y el rendimiento en tiempo de ejecución.

[2] Elemento de *Sun Microsystems Java Foundation Classes* (JFC), que constituye API para proporcionar una interfaz de usuario gráfica (GUI) para programas de Java.

[3] Plataforma utilizada en la implementación de aplicaciones a partir de un conjunto de componentes de software llamados módulos.

[4] Framework facilita el almacenamiento y la recuperación de objetos persistentes de java en una base de datos a través del mapeo objeto/relaciona.

Las tendencias, tecnologías y herramientas seleccionadas cumplen con las necesidades establecidas para la implementación del Subsistema de configuración y seguridad.

1.4. Enfoques y limitaciones predominantes en la implementación de la configuración y seguridad en los sistemas de control informático de empresas.

Existen disímiles enfoques para la implementación de la configuración y seguridad en los sistemas de control informático de empresas. Cada entidad presenta un conjunto de características a configurar por lo cual provoca limitaciones en la implementación. Se puede decir que los enfoques no son más que las formas que se definen para lograr la implementación de la configuración y seguridad. Las limitaciones no son más un conjunto de características que presenta la empresa, que hace necesario que los enfoques utilizados sean adaptables.

1.4.1. Tendencias actuales en la implementación de la configuración y la seguridad en los sistemas informáticos en las empresas.

Entre las tendencias actuales de la seguridad informática en el mundo y en Cuba, se manifiesta el empleo de las directivas de seguridad informática, la cual se ejecuta mediante la revisión de la siguiente lista [12]:

1. Directiva de seguridad informática física, como los controles de acceso físico.
2. Directivas de seguridad de la red (por ejemplo, las referentes al correo electrónico y a Internet).
3. Directivas de seguridad de los datos (control de acceso y controles de integridad).
4. Planes y pruebas de contingencias.
5. Planes de recuperación de desastres.
6. Conocimiento y formación en seguridad informática.
7. Directivas de administración y coordinación de la seguridad informática.
8. Documentos de control de acceso.
9. Otras contraseñas de administración de dispositivos.

También se emplean: [13]

Estrategias proactivas, reactivas y pruebas.

- ✓ Proactiva o de previsión de ataques es un conjunto de pasos que ayuda a reducir al mínimo la cantidad de puntos vulnerables existentes en las directivas de seguridad y al desarrollar planes de contingencia. La determinación del daño que un ataque va a provocar en un sistema, las debilidades y puntos vulnerables explotados.
- ✓ Reactiva o estrategia posterior al ataque ayuda al personal de seguridad a evaluar el daño que ha causado el ataque, a repararlo o a implementar el plan de contingencia desarrollado en la estrategia proactiva, a documentar y aprender de la experiencia, y a conseguir que las funciones comerciales se normalicen lo antes posible.
- ✓ Pruebas, el último elemento de las estrategias de seguridad, las pruebas y el estudio de sus resultados, se lleva a cabo después de que se han puesto en marcha las estrategias reactiva y proactiva. Para asegurar los fondos necesarios para las pruebas, es importante que los directivos sean conscientes de los riesgos y consecuencias de los ataques, así como de las medidas de seguridad que se pueden adoptar para proteger al sistema, incluidos los procedimientos de las pruebas.

Estudio pronósticos de posibles ataques y riesgos

Lo primero es determinar los ataques que se pueden esperar y las formas de defenderse contra ellos. Es imposible estar preparado contra todos los ataques; por lo tanto, hay que prepararse para los que tiene más probabilidad de sufrir la organización. Siempre es mejor prevenir o aminorar los ataques que reparar el daño que han causado.

Para mitigar los ataques es necesario conocer las distintas amenazas que ponen en peligro los sistemas, las técnicas correspondientes que se pueden utilizar para comprometer los controles de seguridad y los puntos vulnerables que existen en las directivas de seguridad. El conocimiento de estos tres elementos de los ataques ayuda a predecir su aparición e incluso, su duración o ubicación. La predicción de los ataques trata de pronosticar su probabilidad, lo que depende del conocimiento de sus distintos aspectos. Los diferentes aspectos de un ataque se pueden mostrar en la siguiente ecuación: [12]

Amenazas+ Motivos + Herramientas y técnicas + Puntos vulnerables = Ataque

Considere todas las amenazas posibles que causan ataques en los sistemas. Entre éstas se incluyen los agresores malintencionados, las amenazas no intencionadas y los desastres naturales. Amenazas como empleados ignorantes o descuidados, y los desastres naturales no implican motivos u objetivos; por lo tanto, no se utilizan métodos, herramientas o técnicas predeterminadas para iniciar los ataques. Casi todos estos ataques o infiltraciones en la seguridad se generan internamente; raras veces los va a iniciar alguien ajeno a la organización. [12]

Para estos tipos de amenazas, el personal de seguridad necesita implementar estrategias proactivas o reactivas.

Para cada tipo de método de ataque

Para iniciar un ataque, se necesita un método, una herramienta o una técnica para explotar los distintos puntos vulnerables de los sistemas, de las directivas de seguridad y de los controles.

La siguiente lista muestra algunas de estas técnicas:

- ✓ Ataques de denegación de servicio
- ✓ Ataques de invasión
- ✓ Ingeniería social
- ✓ Modificación de paquetes
- ✓ Repetición de paquetes
- ✓ Adivinación de contraseñas

Control de Seguridad de datos y del sistema [11]:

- ✓ ¿Qué controles de acceso, controles de integridad y procedimientos de copias de seguridad existen para limitar los ataques?
- ✓ ¿Hay directivas de privacidad y procedimientos que deban cumplir los usuarios?
- ✓ ¿Qué controles de acceso a los datos (autorización, autenticación e implementación) hay?
- ✓ ¿Qué responsabilidades tienen los usuarios en la administración de los datos y las aplicaciones?

- ✓ ¿Se han definido técnicas de administración de los dispositivos de almacenamiento con acceso directo? ¿Cuál es su efecto en la integridad de los archivos de los usuarios?
- ✓ ¿Hay procedimientos para controlar los datos importantes?

Elaborar planes de contingencia [14]

Un plan de contingencia es un plan alternativo que debe desarrollarse en caso de que algún ataque penetre en el sistema y dañe los datos o cualquier otro activo, detenga las operaciones habituales. El plan se sigue si el sistema no se puede restaurar a tiempo. Su objetivo final es mantener la disponibilidad, integridad y confidencialidad de los datos (es el "Plan B").

Utilización de métodos para la recuperación de la información o restauración del sistema: métodos que garanticen la restauración de la información, bases de datos o el sistema en sí, al producirse alguna falla técnica, accidente, ataque o interrupción de la energía eléctrica. [15]

1.4.2. Limitaciones actuales en la implementación del subsistema de configuración y seguridad en los sistemas de control informáticos.

A pesar del avance existente se pueden encontrar muchas limitaciones en la implementación de la configuración y seguridad de los sistemas, o sea, no existe una herramienta que pueda lograr la implementación de la configuración y la seguridad en un sistema. Además, existen diversos criterios de diferentes autores en la literatura sobre cómo implementar la configuración y seguridad en los sistemas de control informático. Se hallan maneras de configurar los elementos de un sistema de una forma individual pero cuando se quiere lograr una integración de estos se hace necesaria la construcción de un subsistema de configuración y seguridad para automatizar todos estos procesos en la aplicación.

Actualmente la Empresa de Perforación y Extracción de Petróleo de Occidente, utiliza herramientas ofimáticas para el control de la información de algunos procesos, lo cual se realiza de forma asilada. Sin ningún tipo de configuración o seguridad se puede provocar, que haya información susceptible de repercusión económica para la empresa repetida y en ocasiones accedida por personas no autorizadas. Existen diversas empresas que han desarrollado sistemas destinados para el control de información entre los que se encuentra:

InfoProd desarrollado por la Empresa InfOil, es un sistema destinado al petróleo y gas de producción de base de datos y análisis del sistema. La información puede ser visualizada gráficamente, y los reportes pueden ser generados o visualizados en tablas. Permite una traza de seguimiento a las terminaciones de pozos y la actividad de reacondicionamiento. Los datos se pueden exportar fácilmente a disimiles formatos como son Excel, Páginas Web (HTML) y Documentos de Texto. Los usuarios también pueden definir informes personalizados. Está programado en Smalltalk [5]lenguaje de programación orientado a objetos y utiliza como gestor de base de datos Microsoft SQL Server.

DATAMA es un producto desarrollado por la Empresa Fiel Development, contiene base de datos geo-referenciada de reservorios programada en Visual Basic sobre plataforma SQL Server. Permite la organización de la información de su equipo de trabajo de reservorios, disminuyendo de los tiempos de búsqueda de la información, permitiendo un mayor rendimiento de los procesos de exportación de información hacia otros software de la industria y la recuperación de la información de otras aplicaciones eliminando, la duplicación de procesos de carga y actualización de la información.

A pesar de todas estas funcionalidades de los productos descritos anteriormente todavía no permiten una configuración y seguridad personalizada a las necesidades actuales que presenta la EPEPO, debido a que aunque presenta una forma de configuración existen parámetros que todavía no son configurables en el sistema. Además, son productos costosos, propietarios y desarrollados con herramientas propietarias que no permiten su modificación, distribución y estudio dejando de ser opciones factibles para la empresa.

En las actuales condiciones que se enfrenta el sistema de control informático de la Empresa de Perforación y Extracción de Petróleo de Occidente, aún carece de un subsistema de configuración y seguridad en el control de la información que se genera para que responda a las exigencias contemporáneas y a las necesidades de los usuarios. Por tanto, se hace indispensable el establecimiento de la configuración y seguridad al sistema para garantizar la funcionalidad y protección adecuada del mismo, ya que la literatura internacional y la cubana, no hacen referencia a la implementación de ellos en las condiciones de las empresas petroleras. Es por ello que no hay reporte de dichos procederes para el establecimiento de dicho subsistema.

[5]Lenguaje puro de programación orientada a objetos, siendo este simple y uniforme.

Conclusiones del Capítulo.

En este capítulo se hizo un análisis de los antecedentes y evolución de los sistemas de control informático. También se realizó un análisis a los elementos teóricos importantes para la correcta comprensión del problema. Se muestran las distintas herramientas, metodologías y tecnologías a utilizar en la realización del subsistema.

Se hizo un estudio de las tendencias y limitaciones actuales en la implementación de la configuración y la seguridad, mostrando los diferentes enfoques que existen en Cuba y el mundo. También se realizó un estudio de las empresas competidoras, las cuales tienen sistemas muy competitivos los cuales realizan la misma función que el sistema a realizar, pero los productos ofrecidos todavía no cumplen con las condiciones existentes que rigen el sistema de producción de EPEPO.

Capítulo II. Descripción de la solución. Propuesta del subsistema.

En este capítulo se abundan puntos importantes en cuanto a la concepción y desarrollo del subsistema de configuración y seguridad. Se muestran las características que debe poseer el subsistema para permitir una mayor eficiencia en la Empresa de Perforación y Extracción de Petróleo de Occidente, entidad que tiene determinadas características que hace necesario la configuración de parámetros importantes para poder realizar el control de la información.

Se estarán abordando los principales elementos que sustentan el subsistema diseñado, donde se realizará una profunda valoración de los artefactos generados por el analista y el arquitecto del proyecto. Luego se realizará un esbozo de como está aprovechada la arquitectura, las posibilidades propiciadas por los framework y las librerías utilizadas en la programación del subsistema, con el objetivo de facilitar la comprensión del funcionamiento de los componentes implementados. Igualmente se describirán y examinarán ciertos patrones los que permitirán lograr una mejor implementación del subsistema.

2.1. Características de la entidad y del sistema de control informático del proyecto de las Empresa de Perforación y Extracción de Petróleo de Occidente. Necesidad de su configuración y seguridad para su eficiencia.

El macro-proceso de producción de petróleo y gas en las empresas petroleras cubanas está compuesto por una serie de sub-procesos, los cuales son: **extracción, recolección, medición, separación, tratamiento y comercialización**. Aunque cada empresa configura el proceso de acuerdo con las características de sus instalaciones o del crudo que procesan. Los sub-procesos de extracción, recolección y separación de petróleo y gas tienen como misión, garantizar el incremento de la producción a partir del empleo de los mejores métodos de recuperación secundaria y terciaria. Estos son empleados en la explotación de los yacimientos petrolíferos, así como el control de la medición eficaz del 100 % del total de producción del fondo de pozos activos. A través de estos sub-procesos se recolecta el fluido total de la producción del yacimiento, separando el agua libre y el gas acompañante. Con el sub-proceso de medición entre otras cosas se chequea el BSW (% de agua en sedimento) de los pozos, lo que permite hacerle un seguimiento a los mismos, evitando que el incremento de dicho BSW por la declinación del propio pozo no pase desapercibida y afecte el proceso tecnológico.

Los sub-procesos de tratamiento y comercialización de crudo son los encargados de gestionar la mejora continua del sistema de tratamiento, almacenamiento y comercialización de crudo garantizando el 100% del tratamiento de la producción de los yacimientos planificado en el plan de producción anual. Todo esto se realiza bajo el estricto cumplimiento de los indicadores técnicos, económicos y de calidad establecidos para el cumplimiento de las políticas de gestión ambiental y política de la calidad con vista a lograr un producto final que satisfaga a los clientes.

El sistema de control informático concebido para la EPEPO (SISPEP) es un sistema que surge en la universidad de las Ciencias Informáticas con el objetivo de controlar todo el flujo de información que se genera a partir del proceso de producción en la EPEPO. Entre las características de la empresa que el sistema de control tiene en cuenta se encuentra las instalaciones que conforman el esquema de producción, tales como:

- ✓ Planta de tratamiento de agua
- ✓ Batería
- ✓ Centro Colector
- ✓ Gasoducto
- ✓ Estación de Re-bombeo
- ✓ Instalación Básica de Medición
- ✓ Cargadero
- ✓ Punto de Inyección
- ✓ Centro de Medición de Gas
- ✓ ENERGAS
- ✓ Estación Cabecera de Oleoducto/Oleoducto
- ✓ Procesadora de Gas
- ✓ Unidad generadora de electricidad
- ✓ Dirección Empresa

El sistema debe permitir agregar instalaciones a la entidad. Estas instalaciones tendrán definidas una serie de comportamientos y parámetros que se controlarán. A continuación se describen las

características de las instalaciones importantes para el desarrollo del sistema que intervienen directamente en el proceso de producción en la EPEPO.

Instalación Básica de Medición (IBM).

Las IBM son las instalaciones que se encargan de realizar mediciones a los pozos, estas mediciones deben de hacerse de forma periódica debido a que el dato de esa medición del pozo es lo que se utiliza para calcular la producción real del pozo. Esta medición es aprobada o no por el jefe a cargo de la planta en ese momento. En esta instalación también se le da un primer tratamiento al fluido o crudo, el gas que es separado pasa a través de gaseoducto a la planta de tratamiento de gas (ENERGAS) y el crudo pasa hacia los centros colectores (CC).

Centro colectores (CC)

En caso de que la entidad no posea IBM los pozos estarían conectados a un CC, los cuales se encargarían de realizar las mediciones de los mismos siguiendo el mismo proceso de las IBM.

En caso de que los pozos estén muy distantes de los CC su producción se centraliza en un tanque donde cada cierto tiempo se recoge el crudo en una paila o camión cisterna. Esta producción pasa a formar parte del CC también.

El centro colector se encarga de realiza un primer tratamiento al fluido, el gas que es separado en el proceso pasa a través de gaseoducto a la planta de tratamiento de gas (ENERGAS). Estas instalaciones pueden o no tener FWKO[6] (*Free Water Knock Out*) en caso tener el tratamiento al fluido se realiza mediante la separación por medio calor, utilizando un dispositivo o tanque se lleva un control de sus parámetros.

Si esta entidad no tiene FWKO, se les inyecta un conjunto de sustancias terso-activas (STA), sustancias reductoras de viscosidad (SRV) y nafta, para reducir la viscosidad del crudo, cuya cantidad se ha de restar a la producción del centro al terminar el día. De estas sustancias se lleva también un control estricto del consumo.

[6] Separador trifásico usado para darle tratamiento al crudo.

En caso de que el CC envié su producción a una instalación de re-bombeo es preciso que controle además los parámetros de este bombeo que realiza cada cierto tiempo (4hrs).

Estación de re-bombeo ER

Esta es la instalación que se encarga básicamente de recibir el crudo que es enviado desde los CC y trasegarlo a las instalaciones de tratamiento de Crudo (baterías). En caso de ser necesario se le inyecta nafta, SRV y STA al crudo para disminuir su espesura. También se lleva un control estricto de lo que se recibió de cada CC y de lo que se bombea para las Baterías, así como de las sustancias que le son inyectadas al crudo.

Plantas de tratamiento de crudo o baterías (BAT)

Es la instalación donde se realiza el tratamiento final del crudo ya sea mediante calor o mediante el empleo de sustancias químicas. Una de las formas de aplicar este tratamiento puede ser mediante la utilización de calderas si las poseen. Las baterías controlan todas y cada una de las calderas, en caso de tenerlas, así como el gas que las mismas consumen. Otra de las formas es mediante la utilización de productos químicos. Esta última también se controla estrictamente.

Las baterías poseen un conjunto de tanques con diferentes volúmenes los cuales se utilizan para el proceso de puesta de calidad del crudo. Estos tanques se controlan y de esta manera se conoce la producción real de la empresa al final del día.

Esta instalación se encarga además, de vender el crudo por lo que se lleva un control de las ventas que se realizan a las diferentes instituciones propias del país así como a súper-tanqueros. Ya sea por oleoductos o por pailas (camiones cisternas).

Entre otras de las características principales del sistema de control informático al cual se le insertará este subsistema de configuración y seguridad propuesto, se evidencia en el mismo que: los estilos se encuentran en el centro de la arquitectura y constituyen buena parte de su sustancia. Son el nivel de abstracción mayor para estructurar el sistema. En la solución propuesta se utilizará el estilo de arquitectura *Layers* (Capas), específicamente 3 capas el cual enfatiza la modificabilidad y escalabilidad de

los datos del software. Este estilo soporta un diseño basado en niveles de abstracción crecientes, lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales. Además, admite muy naturalmente optimizaciones, refinamientos y proporciona amplia reutilización o sea al igual que los tipos de datos abstractos se puede reutilizar un mismo nivel en varias aplicaciones. Permite también la estandarización, ya que el cambio de nivel no afecta el resto.[16]

SISPEP se compone por subsistemas, y estos a su vez por módulos. Los subsistemas no son independientes entre sí, y para el desarrollo de la aplicación se tienen concebidos los siguientes:

- ✓ Subsistema de configuración y seguridad: Contiene la realización de los casos de uso del sistema (CUS) correspondientes con la gestión de codificadores, configuración del sistema. Además de la gestión de usuarios, roles y permisos sobre el sistema.
- ✓ Subsistema de centro colector (CC): Contiene la realización de los CUS correspondientes a la gestión de parámetros de pozos, sus mediciones, su producción y el cálculo de la producción en general del CC.
- ✓ Subsistema de batería (Bat): Contiene la realización de los CUS correspondientes al control de la producción proveniente de los CC asociados, así como, las ventas realizadas a los clientes de la EPEPO y los parámetros propios de la Batería.
- ✓ Subsistema de despacho de producción (DP): Contiene la realización de los CUS correspondientes al control de la producción diaria de la EPEPO y la gestión de las afectaciones.
- ✓ Subsistema de despacho central (DC): Contiene la realización de los CUS correspondientes al control de la producción diaria, cierre operacional y anual de la producción de crudo y gas de la EPEPO.
- ✓ Subsistema de yacimiento (Yac): Contiene la realización de los CUS correspondientes a la confección de los planes operativos y anuales de la empresa.

La siguiente figura 2.1 muestra la división del sistema en subsistemas. Cualquier cambio en la lógica de negocio, solo perturbaría al subsistema al cual le corresponde la responsabilidad. Suministra y ayuda el trabajo en equipo, permitiendo la opción de desarrollar en paralelo y garantizar una culminación ágil y eficaz de la aplicación.

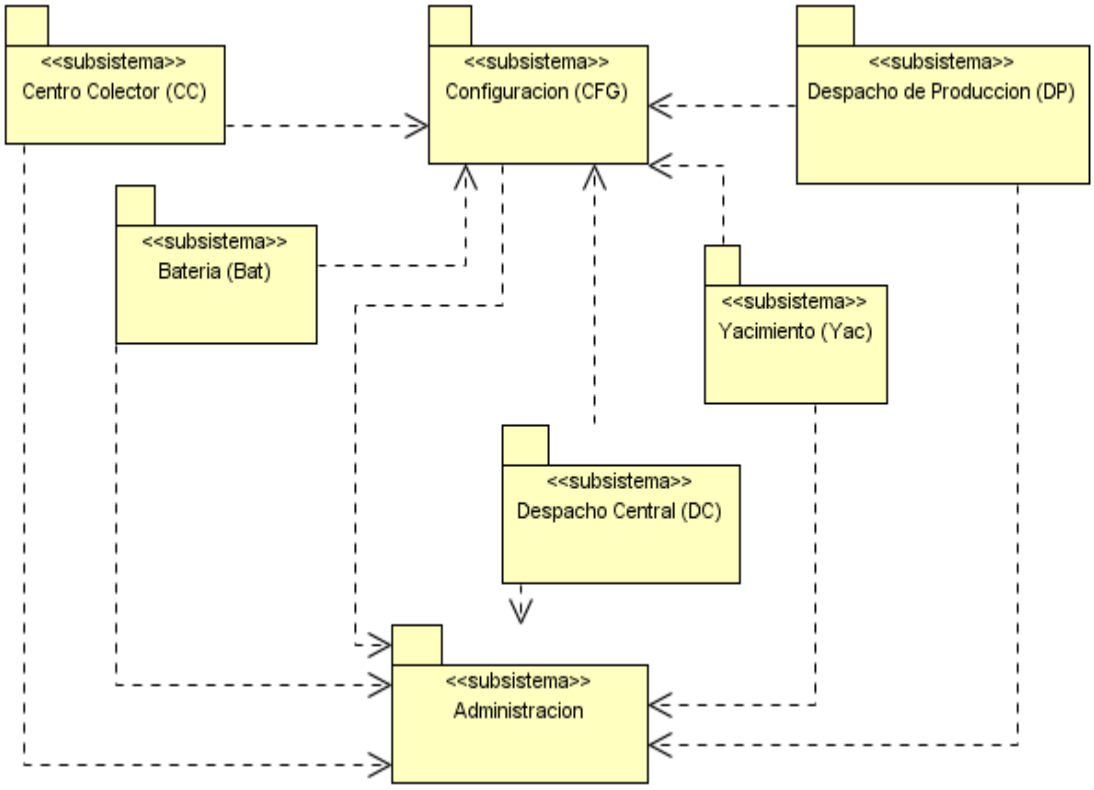


Figura 2.1: División del Sistema en Subsistemas. [16]

En la siguiente figura 2.2 se muestra una vista general de la arquitectura para los subsistemas.

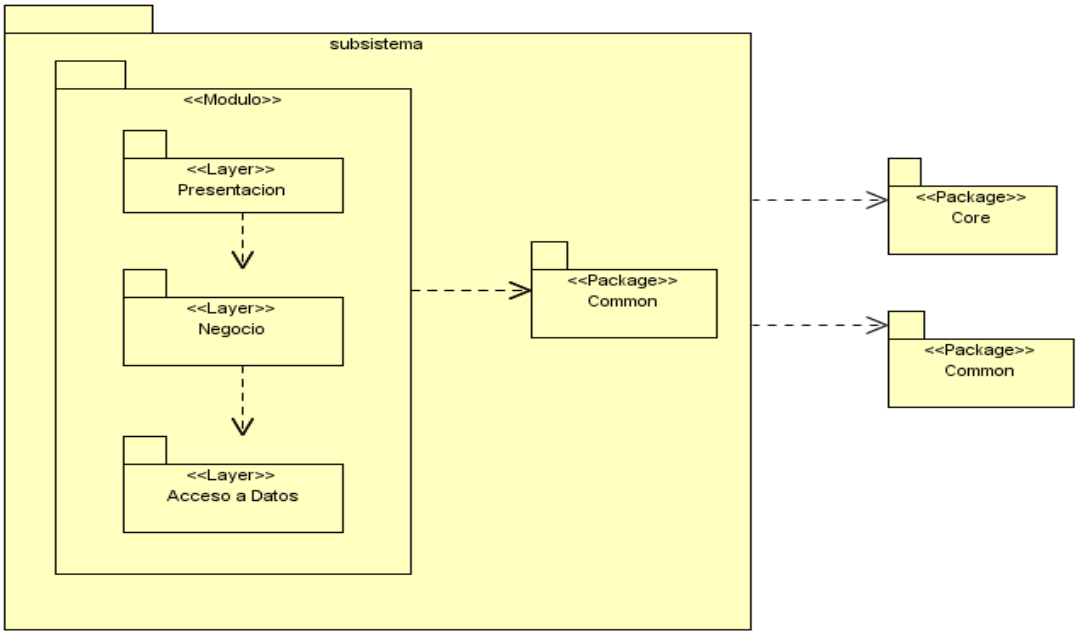


Figura 2.2: Visión General de la Arquitectura. [16]

La vista general de la arquitectura para los subsistemas define tres capas verticales [16]:

Presentación: contiene las clases, formularios y componentes del framework Swing que componen el sistema.

Lógica de aplicaciones: contiene las clases que controlan la lógica del negocio, componentes del framework Spring.

Acceso a datos: clases que controlan la lógica transaccional y la interacción con la base de datos, componentes del framework Hibernate.

También se puede observar en la figura anterior la utilización por parte del subsistema de dos paquetes que servirán de apoyo a la interacción entre subsistemas que son el Common y el Core.

Common: paquete que encierra clases e interfaces comunes de la configuración de los módulos. Permite eliminar dependencias directas entre los módulos específicos de la aplicación y se extiende también para establecer este mecanismo de colaboración entre subsistemas.[16]

Core: contiene los paquetes y clases de configuración del sistema, presenta un mecanismo genérico de acceso a datos.[16]

Restricciones de acuerdo con la estrategia de diseño

- ✓ El diseño de las aplicaciones se hará utilizando La Programación Orientada a Objetos (POO).
- ✓ Se utilizarán las tecnologías que brindan los framework definidos para cada una de las capas de la aplicación:
    - Para la capa de presentación: framework Swing.
    - Para la capa de lógica del negocio: framework Spring.
    - Para la capa de Acceso a Datos: framework Hibernate.

Aspectos generales

Las interacciones entre las capas ocurren generalmente por invocación de métodos, por definición los niveles más bajos no deben poder utilizar funcionalidad ofrecida por las capas superiores. [16]

Los requerimientos funcionales o condiciones que el sistema debe cumplir son: [17]

RF1. Gestionar pozo.

1.1 El sistema debe permitir insertar pozo.

1.2 El sistema debe permitir modificar pozo.

1.3 El sistema debe permitir eliminar pozo.

RF2. Gestionar yacimiento.

2.1 El sistema debe permitir insertar yacimiento.

2.2 El sistema debe permitir modificar yacimiento.

2.3 El sistema debe permitir eliminar yacimiento.

RF3. Gestionar método de exploración.

3.1 El sistema debe permitir insertar métodos de exploración.

3.2 El sistema debe permitir modificar métodos de exploración.

3.3 El sistema debe permitir eliminar métodos de exploración.

RF4. Gestionar capa.

4.1 El sistema debe permitir insertar capa.

4.2 El sistema debe permitir modificar capa.

4.3 El sistema debe permitir eliminar capa.

RF5. Gestionar compañía.

5.1 El sistema debe permitir insertar compañía.

5.2 El sistema debe permitir modificar compañía.

5.3 El sistema debe permitir eliminar compañía.

RF6. Gestionar tipo de pozo.

6.1 El sistema debe permitir insertar tipo de pozo.

6.2 El sistema debe permitir modificar tipo de pozo.

6.3 El sistema debe permitir eliminar tipo de pozo.

RF7. Gestionar instalación.

7.1 El sistema debe permitir insertar instalación.

7.2 El sistema debe permitir modificar instalación.

7.3 El sistema debe permitir eliminar instalación.

RF8. Gestionar tipo de instalación.

8.1 El sistema debe permitir insertar tipo de instalación.

8.2 El sistema debe permitir modificar tipo de instalación.

8.3 El sistema debe permitir eliminar tipo de instalación.

RF9. Gestionar producción fundamental.

9.1 El sistema debe permitir insertar producción fundamental.

9.2 El sistema debe permitir modificar producción fundamental.

9.3 El sistema debe permitir eliminar producción fundamental.

RF10. Gestionar categoría de pozo.

10.1 El sistema debe permitir insertar categoría de pozo.

10.2 El sistema debe permitir modificar categoría de pozo.

10.3 El sistema debe permitir eliminar categoría de pozo.

RF11. Gestionar parámetros de pozo del centro colector (CC).

11.1 El sistema debe permitir insertar parámetros de pozo del CC.

11.2 El sistema debe permitir modificar parámetros de pozo del CC.

11.3 El sistema debe permitir eliminar parámetros de pozo del CC.

RF12. Gestionar parámetros del separador de entrada (PSE) del CC.

12.1 El sistema debe permitir insertar PSE del CC.

12.2 El sistema debe permitir modificar PSE del CC.

12.3 El sistema debe permitir eliminar PSE del CC.

RF13. Gestionar parámetros del separador de medición (PSM) del CC.

13.1 El sistema debe permitir insertar PSM del CC.

13.2 El sistema debe permitir modificar PSM del CC.

13.3 El sistema debe permitir eliminar PSM del CC.

RF14. Gestionar parámetros del F.W.K.O del CC.

14.1 El sistema debe permitir insertar parámetros del F.W.K.O del CC.

14.2 El sistema debe permitir modificar parámetros del F.W.K.O del CC.

14.3 El sistema debe permitir eliminar parámetros del F.W.K.O del CC.

RF15. Gestionar dosificación de productos químicos (DPQ) del CC.

15.1 El sistema debe permitir insertar DPQ del CC.

15.2 El sistema debe permitir modificar DPQ del CC.

15.3 El sistema debe permitir eliminar DPQ del CC.

RF16. Gestionar entidad.

16.1 El sistema debe permitir insertar entidad.

16.2 El sistema debe permitir modificar entidad.

16.3 El sistema debe permitir eliminar entidad.

RF17. Gestionar tipo de producto químico (TPQ).

17.1 El sistema debe permitir insertar TPQ.

17.2 El sistema debe permitir modificar TPQ.

17.3 El sistema debe permitir eliminar TPQ.

RF18. Gestionar trasiego.

18.1 El sistema debe permitir insertar trasiego.

18.2 El sistema debe permitir modificar trasiego.

18.3 El sistema debe permitir eliminar trasiego.

RF20. El sistema debe permitir generar el parte de producción del CC.

RF21. El sistema debe permitir graficar la información.

RF22. El sistema debe permitir autenticar usuarios.

RF23. Gestionar tipo de destino (TD).

23.1 El sistema debe permitir insertar TD.

23.2 El sistema debe permitir modificar TD.

23.3 El sistema debe permitir eliminar TD.

RF24. Gestionar ciclos de producción (CP).

24.1 El sistema debe permitir insertar CP.

24.2 El sistema debe permitir modificar CP.

24.3 El sistema debe permitir eliminar CP.

RF25. Gestionar permiso.

25.1 El sistema debe permitir insertar permiso.

25.2 El sistema debe permitir modificar permiso.

25.3 El sistema debe permitir eliminar permiso.

RF26. Gestionar rol.

26.1 El sistema debe permitir insertar rol.

26.2 El sistema debe permitir modificar rol.

26.3 El sistema debe permitir eliminar rol.

RF27. Exportar información.

27.1 El sistema debe permitir exportar información a formato Excel.

27.2 El sistema debe permitir exportar información a formato PDF.

RF28. Gestionar destino.

28.1 El sistema debe permitir insertar destino.

28.2 El sistema debe permitir modificar destino.

28.3 El sistema debe permitir eliminar destino.

RF29. Gestionar tipo de afectación (TA).

29.1 El sistema debe permitir insertar TA.

29.2 El sistema debe permitir modificar TA.

29.3 El sistema debe permitir eliminar TA.

RF30. Gestionar usuario.

30.1 El sistema debe permitir insertar usuario.

30.2 El sistema debe permitir modificar usuario.

30.3 El sistema debe permitir eliminar usuario.

Entre los requerimientos no funcionales se encuentran: [16]

Requerimientos de hardware

Estaciones de trabajo

- ✓ Periféricos Mouse y Teclado PS 2 o USB.
- ✓ Tarjeta de red.
- ✓ 1Mb de caché L2, 256 Mb de memoria RAM.
- ✓ 5 GB de espacio libre en disco.
- ✓ CPU Pentium II a 800 MHz o superior.
- ✓ Impresora.

Servidor de base de datos

- ✓ Periféricos: Mouse y Teclado PS 2 o USB.
- ✓ Tarjeta de Red.
- ✓ 1GB MB de RAM.
- ✓ Capacidad de Almacenamiento 20 GB.
- ✓ Arquitectura hardware Intel.
- ✓ Sistema operativo Windows 2000 o superior, GNU-Linux (recomendado), Unix.

Requerimientos de software

Estaciones de trabajo

- ✓ Sistema operativo: Windows 2000 o superior, GNU-Linux, Unix.
- ✓ Java Runtime Environment (Máquina Virtual de Java) versión 1.6.

Servidor de base de batos

- ✓ Sistema operativo: Windows 98 o superior, GNU-Linux (recomendado), Unix.
- ✓ Gestor de base de datos: PostgreSQL. v8.2.

Redes

- ✓ La red existente en las instalaciones debe soportar la transacción de paquetes de información de al menos unas 10 máquinas a razón de 2 ó 3 Mb/s.
- ✓ Para hacer más fiable la aplicación la misma debe estar protegida contra fallos de corriente y conectividad, para lo que se deberá parametrizar los tiempos para realizar copias de seguridad. La

réplica de datos se realizará utilizando la herramienta DEW[7]. En esta se especificará que primero se deben replicar los Centros Colectores, luego las Baterías y por último el Despacho de Producción; el Despacho Central trabajará directamente con la Base de Datos (BD) central. Lo que se realizará teniendo en cuenta el tiempo de mejor tráfico en la red.

Seguridad

- ✓ La seguridad se tratará desde el modelamiento del sistema, pasando por las primeras fases de desarrollo del sistema y profundizando en las finales.
- ✓ Se utilizarán las reglas de la "programación segura", se deberá hacer un fuerte tratamiento de excepciones desde la capa de presentación hasta la capa de acceso a datos incluyendo el trabajo con la base de datos. Para reforzar este aspecto los desarrolladores se apoyarán en los framework seleccionados para el desarrollo del sistema.
- ✓ Los usuarios de la BD solamente tendrán acceso a las tablas de la base de datos a las cuales se les designe según el rol que desempeña, no se utilizarán usuarios con privilegios administrativos para realizar las conexiones entre servidores.
- ✓ Se registrarán y auditarán las trazas dejadas por los usuarios al realizar las diferentes operaciones en el sistema. La programación de las auditorías será según corresponda.
- ✓ La asignación de usuarios y sus opciones sobre el sistema se garantizará desde el subsistema de administración.

Portabilidad, escalabilidad y reusabilidad

- ✓ El sistema deberá ser utilizado desde cualquier plataforma de software (Sistema operativo).
- ✓ El sistema permitirá hacer un uso racional de los recursos de hardware de la máquina, sobre todo en los PC clientes.
- ✓ La aplicación se construirá utilizando estándares internacionales y patrones, para facilitar su integración futura con componentes desarrollados por cualquier empresa y garantizar posibilidades de un mantenimiento ágil.

[7] Software de réplica de datos, que se basa en la copia de datos de una localización a otra. Intenta cubrir las más importantes insuficiencias vinculadas con la distribución de datos entre los gestores más populares como la protección, recuperación, sincronización de datos, transferencia de datos entre diversas localizaciones o la centralización de la información en una única localización.

- ✓ De acuerdo a las condiciones económicas del país, el sistema deberá minimizar la cantidad de gastos de despliegue y contar con la capacidad de ajustarse a las necesidades de las EPEP.
- ✓ Se desarrollará cada pieza del sistema en forma de componentes (subsistemas) con el fin de re-utilizarlos para futuras versiones.
- ✓ La documentación de la arquitectura deberá ser reutilizable para poder documentarla como una familia de productos.

Como se puede apreciar el flujo de producción de la empresa depende fundamentalmente de los pozos los cuales son el eje principal de la EPEPO. Los mismos pueden estar conectados a CC o a una IBM cuando se le necesita hacer una medición. También en caso de estar conectados a un tanque aislado de cualquier instalación, la producción saldría por los CC al que pertenece el tanque. De los pozos se debe controlar una serie de parámetros que son fundamentales a la hora del cálculo de la producción de la empresa.

La entidad EPEPO deberá tener asociado uno o más yacimientos sobre los cuales tendrá a su vez los pozos para obtener el petróleo. Los pozos extraen el petróleo de una o varias capas de la corteza terrestre.

2.2. Elementos que sustentan la implementación del subsistema de configuración y seguridad.

En las actuales condiciones del contexto global y del país se requieren que los sistemas de control informático en las empresas cuenten con un subsistema de configuración y seguridad adecuado, capaz de integrar todo el proceso y el sistema, por tanto, este subsistema que se concibe para su implementación y se sustenta en los siguientes criterios:

- ✓ Lógica del proceso de producción y control descrita en el epígrafe anterior.
- ✓ El ordenamiento de operaciones tiene en cuenta los factores clave a controlar por los directivos, eje clave en el subsistema ya que es el encargado de vigilar e informar de todas las acciones realizadas en el sistema.
- ✓ El subsistema está concebido a partir del esquema de producción de EPEPO ya que lo construye una vez iniciado el sistema y se encarga de definir las instalaciones que contiene la

entidad así como permitir la creación o reestructuración de elementos en el esquema de producción con el objetivo de reducir el impacto que los nuevos cambios puedan provocar.

Siguiendo cada uno de los criterios anteriores, se sustentó el diseño y la conformación de la arquitectura del subsistema de configuración y seguridad, con el objetivo de lograr una implementación que reúna todos los requisitos.

El subsistema se sustenta en las funcionalidades necesarias para la puesta en marcha de la aplicación, además de que constituyen la base para la seguridad de la aplicación. A partir del se puede gestionar la traza de las operaciones realizadas en el sistema, los codificadores que se utilizan en varios o todos los módulos del sistema, así como los datos del personal y la producción, de los cuales dependen varias operaciones fundamentales para el sistema.[16]

Las funcionalidades que se encuentran en el subsistema agrupadas en caso de uso del sistema (CUS). Los requerimientos funcionales son mencionados en el epígrafe anterior.

Gestionar usuario (RF30): Permite al administrador del sistema gestionar los usuarios que luego interactuarán con el sistema. [16]

Autenticar usuario (RF22): Este caso de uso permite la autenticación de los usuarios que acceden al sistema, controlando de esta manera la seguridad del mismo. [16]

Gestionar rol (RF26): Permite la gestión de roles para los diferentes usuarios. [16]

Gestionar permiso (RF25): Permite la gestión de los permisos de los usuarios, para controlar de esta manera el nivel de acceso a las diferentes partes del sistema. [16]

Gestionar codificadores (RF1, RF2, RF5, RF7, RF16): Permite gestionar los codificadores del sistema los cuales se utilizarán en varios o todos los módulos de la aplicación. [16]

En la siguiente figura 2.3 se muestra la vista de casos de uso.

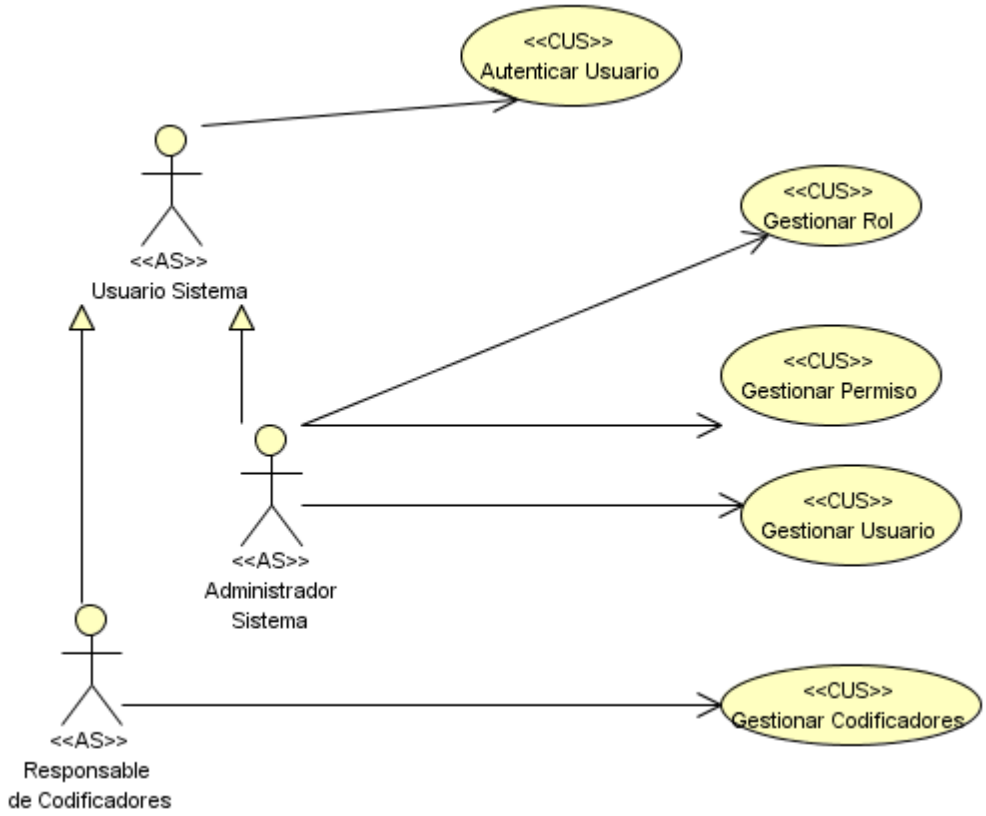


Figura 2.3 Vista de casos de uso del sistema del subsistema de configuración y seguridad. [16]

En esta vista se puede observar los requerimientos del subsistema en casos de usos y además agrupados con la utilizan del patrón de caso de uso CRUD.

Vista de implementación

Esta vista muestra las organizaciones y dependencias lógicas entre componentes software, sean estos componentes de código fuente, binarios o ejecutables. En este diagrama se tiene en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. Con la representación de esta vista se ayuda a los desarrolladores a visualizar el camino de la implementación y permite tomar decisiones respecto a las tareas de la implementación.[16]

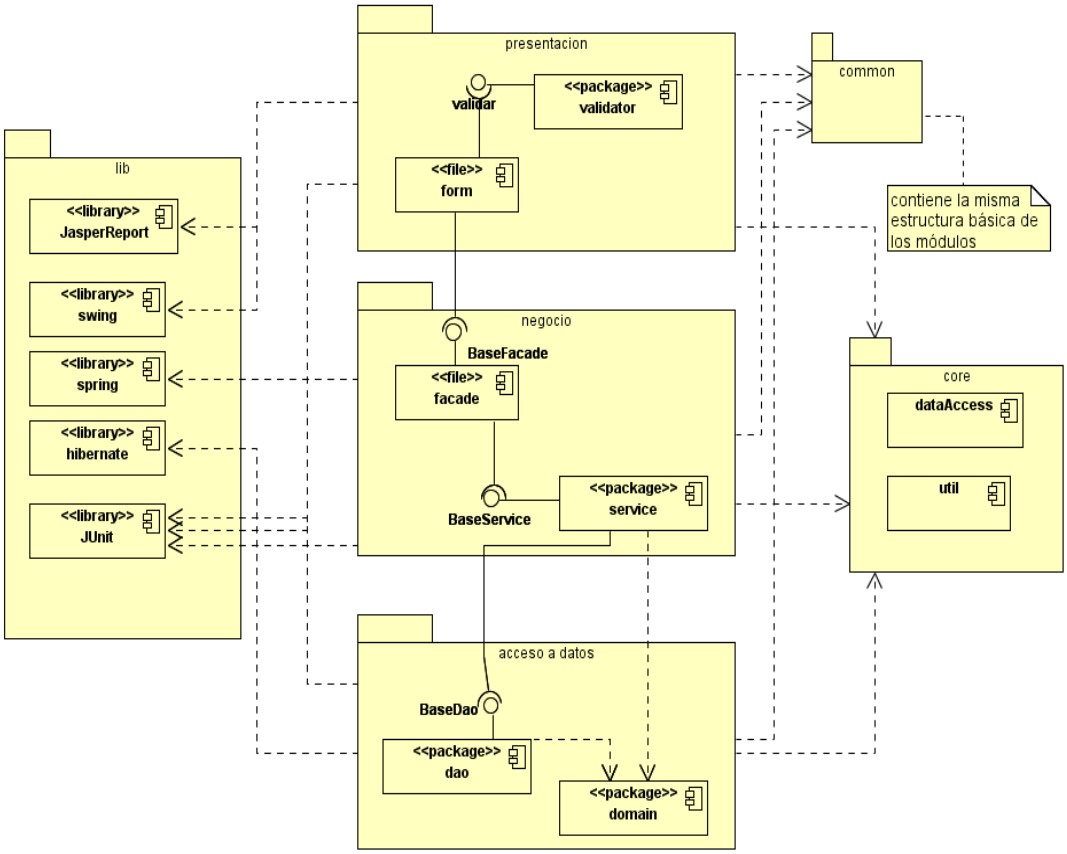


Figura 2.4: Vista de implementación. [16]

Cada paquete encapsula uno o más componentes que se interrelacionan entre ellos para darle solución a la aplicación y todos depende de la JRE[8] que debe de estar instalada en la estación de trabajo que decida ser instalada la aplicación.[16]

A continuación se muestra como se implementan los componentes físicos del sistema agrupándolos en paquetes. [16]

- ✓ core: Contiene los componentes fundamentales para el trabajo en cada una de las capas, así como los componentes de acceso a datos.

[8] Acrónimo de *Java Runtime Environment*, es un grupo de utilidades que permite la ejecución de programas codificados en java.

- ✓ common: Contiene la misma estructura básica de los módulos y encierra clases e interfaces comunes.
- ✓ presentación: Contiene todos los formularios, los cuales están constituidos por los componentes que brinda el framework swing, así como las clases de validación y las que controlan los eventos lanzados en la vista. Permite visualizar los reportes mediante el generador de reporte *JasperReport* en cualquiera de los formatos que este soporta.
- ✓ negocio: Contiene todas las interfaces y servicios requeridos para la densa lógica de negocio identificada.
- ✓ acceso a datos: Contiene las clases u objetos de acceso a datos de la aplicación, ficheros de mapeo generados por hibernate y conjunto de XML[9] con las consultas a la BD en HQL[10] y clases persistentes.
- ✓ lib: Contiene todas las librerías a utilizar en el desarrollo del proyecto. Las mismas serán utilizadas para el acceso a datos y para el trabajo con aspectos, la seguridad, la cache del sistema, los reportes en diversos formatos, entre otros.

Cada uno de estos elementos, son la base para establecer un conjunto de etapas lógicas en la implementación. Elementos que sustentan la implementación mostrando un subsistema configurable, extensible y adaptable a las necesidades actuales.

2.3. Descripción del subsistema implementado.

La implementación práctica de este subsistema consta de cinco etapas secuenciales lógicas, las que deben cumplirse en el orden que se describe a continuación para su aplicación. Ellas son:

- ✓ Primera etapa: Generación de las clases que pertenecen al dominio del problema.
- ✓ Segunda etapa: Creación e implementación de las clases e interfaces de la capa de acceso a datos.
- ✓ Tercera etapa: Creación e implementación de las clases e interfaces de la capa de negocio.
- ✓ Cuarta etapa: Creación e implementación de la capa de presentación.

9 Lenguaje de Etiquetado Extensible que desempeña un rol principal en el intercambio de una gran diversidad de datos.
10 Acrónimo de *Hibernate Query Language*. Poderoso lenguaje para la realización de consultas orientadas a objetos a un base datos modelo/relacional.

- ✓ Quinta etapa: Integración de las capas.

Cada una de estas etapas contiene pasos, los cuales se deben cumplir. Algunos de estos pasos se realizan de manera automatizada permitiendo reducir considerablemente el tiempo de implementación. La tarea de la documentación del código fuente o de los artefactos generados en la implementación se realizará en todas las etapas, es decir, en todo el ciclo de vida de la construcción del subsistema.

2.3.1 Generación de las clases del dominio del problema.

La programación orientada a objeto es uno de los conceptos más utilizados en el desarrollo de software actual. Este paradigma permite modelar un problema en término de objetos de la vida real y llevarlo a la computadora mediante clases que producen una abstracción del objeto. Las clases solo se tendrán los atributos y métodos que son relevantes para el desarrollo del software.

En el contexto del subsistema a implementar existen un conjunto de objetos del dominio del problema, los cuales se encuentran ubicados en la base de datos utilizando el modelo entidad-relación. La tarea principal es convertir el modelo entidad-relación de la base de datos a un modelo objeto-relacional en Java, el cual sea conceptualmente más sencillo y reutilizable.

La entidad EPEPO contiene un conjunto de instalaciones, las cuales a su vez tienen asociado un conjunto de pozos y tanques. Esta entidad es una estructura constructiva la cual tiene paredes, baños, piso y otros elementos. Como se puede apreciar esta posee un gran conjunto de atributos los cuales no se podrían modelar en una sola clase debido al tamaño que tendrá. Por lo tanto, es en este punto donde entra la delimitación de las propiedades y métodos que tendrán las clases dentro del domino del problema. Estas propiedades y métodos deben ser suficientes para la realización del subsistema y del sistema de manera generar para permitir el control de la información generada por mediciones de las instalaciones y los pozos. Por ende, no es necesario tener atributos referentes a la estructura constructiva de la entidad, sino atributos que reflejen la estructura organizativa referente a las instalaciones y los pozos que la componen para lograr un almacenamiento de la información necesaria para realizar el control.

Para la generación de las clases que va a representar objetos en el sistema se utilizará la herramienta hibernate-tools. Esta permite generar las clases del mapeo del modelo entidad-relación al modelo de objeto relacional. Lo primero que se realiza es la configuración de la conexión del fichero XML

(hibernate.cfg.xml), el cual se muestra en el **Anexo #1.** Se puede observar que en este fichero se especifica driver de conexión. También se define la URL de conexión donde se precisa el servidor, el puerto y el nombre de la base de datos, así como, se encuentra el usuario y la contraseña para la realización de la conexión de la base de datos.

Una vez creada la configuración, se procede a la creación del XML (hibernate.reveng.xml). Este permite especificar que tablas de la base de datos se han de mapear. El mismo debe quedar definido de esta manera: **Véase Anexo #2.** Es importante aclarar que este proceso se realiza en el IDE de programación de NetBeans el cual tiene incorporado este plug-in por defecto y además provee de buenas opciones en el tratamiento de ficheros XML.

Una vez creado este fichero se puede pasar a la generación de los ficheros .hbm.xml y los pojos, los primeros son ficheros que permiten establecer una relación entre el pojo generado y su tabla en la base de datos, por lo tanto, es de importancia que se tenga conocimiento de los elementos que constituyen este fichero. El segundo es un fichero de extensión **.java** que contiene las clases que va a representar en el modelo de objetos a la tabla. La estructura del paquete (principal.common.domain.sispep) luego de mapear la BD quedaría como se aprecia en el **Anexo #3**.

Una vez mapeada la BD se procede a la construcción de las clases que permitan consultar, salvar, actualizar y eliminar los datos de la base de datos utilizando las clases mapeadas.

2.3.2 Creación e implementación de las interfaces de la capa de acceso a datos.

Luego de haber mapeado la BD ya se cuenta con todas las clases persistentes del negocio y sus relaciones. Ahora se hace necesario la creación de un conjunto de clases que permitan realizar consultas a los datos que se encuentran en la base de datos. Se utiliza para esto el patrón *Data Access Object(DAO)* el cual plantea la creación de una clase que permitan el trabajo con los datos de la base de datos en un objeto específico. También es interesante destacar que todas las clases o interfaces que se implementen en esta capa tendrán en su nombre la sub-cadena Dao.

Por cada clase de mapeo de la base de datos se crea una clase Dao que extiende de la clase BaseDaoImpl y su interface que implementa la interface BaseDao. Para ilustrar mejor esto se utilizará la

funcionalidad gestionar pozos porque básicamente esto se realiza para todas las demás funcionalidades. La interface BaseDao es un conjunto de métodos básicos para la realización de consultas a la base de datos con Hibernate.

En el **Anexo #4 y #5** se muestra el código de la interfaz BaseDao y el de la clase BaseDaoImpl, recalcar que el uso de las interfaces permite que los implementadores no conozcan los detalles de la implementación de los métodos. Se nota como la clase BaseDaoImpl hereda de la clase HibernateDaoSupport, la cual es una clase del Framework Spring que contiene el objeto de hibernateTemplate. El objeto hibernateTemplate permite la realización de consultas en la base de datos con una simple llamada al método necesario. La ventaja de este objeto es que el programador no tiene que abrir y cerrar sesiones de Hibernate, lo cual facilita la integración entre el Framework Spring y el Hibernate.

De la manera descrita anteriormente se crean las clases e interfaces Daos de cada una de las entidades en ficheros independientes. Cada clase Dao creada contiene una interface la cual extiende de la interface IBaseDao y una implementación la cual extiende de BaseDaoImpl con el objetivo de reutilizar los métodos simples de consulta a la base de datos ya implementados por si fuera necesario redefinir el comportamiento de cualquiera de esto en la clase hija. Para la implementación del subsistema en cuestión, no se hizo necesario redefinir ninguno de estos métodos pero no significa que esto no es necesario, sino que hasta ahora no fue preciso, en caso de que se redefina el comportamiento no habría impacto de cambio en las otras capas.

El siguiente código muestra como quedaría el fichero de la interface ICodPozoDao y el de la clase que la implementa CodPozoDaoImpl. Así es como queda en todas las demás clases Daos realizadas en el subsistema **Véase Anexo #6 y #7.** Se ha terminado la capa de acceso a datos. Se puede ver una claridad en la implementación y como las clases de esta capa realizan solamente la responsabilidad que le fue otorgada. Solo quedaría la pregunta ¿Se implementa aquí la lógica de nuestro de negocio? , la respuesta es clara, no, estas clases solo permiten acceder a los datos, se necesitan de otras clases que utilizando las funcionalidades que les brindan estas clases puedan tomar los datos procesarlos según la lógica de negocio.

2.3.3 Creación e implementación de las interfaces de la capa de negocio.

Esta capa contiene los servicios que implementan la lógica de negocio del subsistema o sistema en cuestión. En el caso particular de este subsistema se necesitan los servicios que permitan gestionar todos los parámetros concebidos para su realización.

Los servicios contienen uno o varios atributos que hacen referencia al Dao de la información necesitada. En el subsistema se utiliza un servicio por cada clase Dao que exista, lo que posibilita una relación donde cada servicio que se crea tenga asociado un Dao, sino se incumple con el diseño y además tratar de que la filosofía que se utilice no quebrante el patrón en 3 capas propuesto por el arquitecto del sistema donde las capas superiores no pueden tener instancias de las capas inferiores. Se puede ver como quedaría la implementación de la interface y la clase ICodPozoService y CodPozoServiceImpl. **Véase los Anexos #8 y #9**

Se tiene un punto de acceso a la lógica de negocio de la aplicación. La cara de esta lógica de negocio va a ser conducida por un conjunto de interfaces que permitirán a la capa de presentación realizar las operaciones que reporten un resultado visible. La claridad y la reutilización son las características que tiene cada uno de los servicios del negocio. Estos deben, al igual que el acceso a datos, ser accedidos mediante interfaces que abstraigan al usuario de su implementación. En este punto también se lanzan las excepciones referentes a reglas del negocio. En este subsistema no se lanzan excepciones porque se decidió trabajar mejor con las excepciones lanzadas por Hibernate en la presentación ya que las excepciones lanzadas resuelven las necesidades de los errores que actualmente se podrían tener.

2.3.4 Creación e implementación de la capa de presentación.

Hasta este punto se tiene el núcleo del subsistema, un conjunto de servicios que realizan todas las operaciones concebidas por la lógica del negocio en cuestión. Se hace necesario mostrarle al usuario final (Cliente no informático) una forma mediante la cual pueda utilizar estos servicios sin tener que conocer detalles del código. Permitiéndole realizar todo el conjunto de funcionalidades mediante una interfaz visual agradable y fácil de usar.

Desde la invención del mouse la mayoría de los sistemas de computadoras utilizan este dispositivo para permitir a los usuarios el acceso a las operaciones mediante la utilización de los eventos que ocurren en el sistema. La creación de menús donde el usuario pueda seleccionar la opción que él desea o la utilización del click derecho para realizar determinadas aplicaciones se han vuelto de moda, porque reducen la cantidad de interacciones entre el usuario y el sistema.

Este subsistema utiliza un conjunto de menús, diálogos y formularios que le permitirán al usuario la accesibilidad de esto con solamente un click izquierdo o derecho. Por ejemplo, para los servicios de crear y actualizar pozos se creó en el subsistema un diálogo, el cual le permite al usuario entrar todos los datos referentes al pozo tales como: nombre, instalación a la que pertenece, ciclo de producción, método de producción entre otras. El código que se muestra en el **Anexo #10** responde a la implementación de un DataEditor, componente Swing similar a los llamados grids de la web, que permite una gestión fácil y reutilizable en cada una de las vistas. Se utilizó este componente, en todas las vistas del subsistema en general.

Se puede apreciar u observar que el DataEditor se le pasa un objeto de tipo Form, el cual es quien contiene realmente los datos del objeto. Para mostrar los datos tanto para adicionar como para modificar los datos con un simple click en el objeto que se desee aplicar la operación. Además se puede constatar como hay ciertos parámetros visuales como: el título del DataEditor y el Form de detalles; es configurado en un fichero **mesage.properties (Véase Anexo #11).** Esto permite la utilización de varios idiomas en el subsistema con solo cambiar el contenido del fichero, lo cual ahora no es objetivo, pero en futuras versiones puede ser de gran consideración. Se puede ver ahora como quedaría el código del objeto Form. **Véase Anexo #12**

Se puede percibir como la clase hereda de la clase TabbedForm del Spring Rich Cient. Esto hace que sea necesario redefinir el método getTabs () el cual devuelve un arreglo de componentes visuales Tab, donde cada uno de estos contienen un formulario. Dentro de este método se utiliza un objeto de tipo TableFormBuilder el cual es el contenedor de los componentes Swing. Cada componente de formulario Swing se asocian a una propiedad de la clase entidad Pozo. Esta asociación entre un atributo de la clase y un componente visual es lo que permite la llamada automática de los métodos de acceso de escritura y/o lectura, para inicializar y/u obtener los datos del objeto creado con los datos entrados por el usuario.

2.3.5 Integración de las capas.

Una vez que todas las capas ya estén concluidas se pasa al proceso de integración entre estas. Este se realiza según las restricciones que imponen los framework de desarrollos utilizados. En este caso específico el framework Spring el cual se está utilizando en la capa de negocio permite una elevada integración con el framework Hibernate y Swing. Toda esta integración se maneja a través de archivos XML, los cuales permiten la creación de objetos en su contexto evitando la creación de múltiples instancias en el subsistema, es decir, implementando de una manera automática el patrón Singleton.

En el fichero conexión.xml (**Véase Anexo 13)** se configura la conexión a la base de datos especificando mediante un DataSource los parámetros de conexión que se encuentran en el fichero **hibernate.cfg.properties**. **Véase Anexo #14**

En el fichero de nombre dao.xml se declaran todos los Daos del subsistema (**Véase el Anexo #15**). Se puede notar como el codPozosDao extiende del BaseDao en la declaración de estos *Beans* en este fichero, lo cual fue especificado además en la programación de estas clases explicado en el epígrafe 2.3.2**.**

Una vez que se tiene el Dao se procede la configuración del servicio (**Véase el Anexo #16**) y como el servicio tiene una propiedad Dao se le asigna el Dao correspondiente anteriormente declarado en el fichero Dao.xml

En este punto se debe configurar la parte visual de nuestra aplicación para que esta se integre a la lógica de negocio. Esto se realiza mediante el fichero richclient-context.xml (**Véase Anexo #17**). Este inicializa el contexto de la aplicación que se utiliza para configurar los componentes y servicios dentro de la plataforma. La plataforma utiliza una serie de servicios en tiempo de ejecución, y estos deben estar configurados aquí. Los dos granos clave para esto son los ServiceLocator y ApplicationServices.

Aparte de los servicios, se definen los diferentes elementos que componen su aplicación, al igual que el descriptor de aplicación, puntos de vista, asesor del ciclo de vida, reglas de validación, etc. Por lo general, si se hiciera necesario cambiar ciertos parámetros de los que vienen por defecto que fue el caso del subsistema se tendría que cambiar el texto siguiente:

1. El startingPageId en el lifecycleAdvisor.
2. El eventExceptionHandler en el lifecycleAdvisor
3. Especificar la ubicación del paquete de recursos en el messageSource.
4. Especificar las propiedades de los archivos de mapeo de imágenes en imageResourcesFactory.
5. Especificar la clase rulesSource.
6. Configurar las vistas.

Una vez terminada la integración de las capas se procede a configurar todo el conjunto de acciones que se podrán realizar en el subsistema de manera general en el fichero command-context.xml **(Véase Anexo #18)**. Este es el encargado de declarar los menús que tendrá la aplicación, y los comandos que se ejecutarán de manera global en el sistema. Entienda por comando como una acción o evento que puede realizar el usuario en el subsistema.

Como se puede ver se ha recreado como fue la implementación de cada una de las capas asociadas en la aplicación. Además se muestra la integración de todas estas capas utilizando como ejemplo base el CUS Gestionar pozo, el cual como se mencionaba en epígrafes anteriores es el más importante a implementar debido a que los pozos son los que rigen todo el proceso de producción de EPEPO. Es válido aclarar que las demás implementaciones se encontrarán en el código fuente del sistema las cuales se encuentran bien documentadas y cumplen con los estándares establecidos.

Conclusiones del capítulo.

En el desarrollo de este capítulo se describió la realización de la implementación. Se puso en práctica el uso de los patrones de diseño. Se mostró la estrategia trazada para lograr la implementación del subsistema. Se enumeran además un conjunto de etapas que se deben seguir de manera secuencial, cualquier variación de los pasos puede traer consigo errores que luego pueden llegar a ser difíciles de encontrar.

Se evidenció la integración mediante ficheros XML de los framework usados. Las ventajas que esto proporciona y su necesidad. También se muestra en cada una de las etapas el cumplimiento con la arquitectura propuesta y la codificación, cumpliendo con el estándar de código establecido. La implementación es la materialización de todos los artefactos generados por los demás integrantes del proyecto en un producto usable para el cliente.

Capítulo III. Validación del subsistema a implementar.

En el presente capítulo se expone la validación el subsistema de configuración y seguridad. A medida que se va desarrollando un sistema el desarrollador o implementador le realiza un conjunto de pruebas con el objetivo de validar su solución. Con estas pruebas el desarrollador trata de corregir errores en el código. En este epígrafe se evidencian las pruebas realizadas al subsistema. En tal sentido se utilizaron las pruebas de unidad y la validación mediante especialistas. Además, se muestra un análisis de los resultados obtenidos en cada una de las pruebas de validación aplicada.

3.1. Pruebas realizadas.

Las pruebas que se realizaron serán a partir de la capa de servicio utilizando el Framework JUnit el cual fue seleccionado en una investigación precedente por el Arquitecto del proyecto. JUnit es un framework para escribir pruebas de unidad o de integración automatizadas en Java. Este es Open Source y está programado por Erich Gamma y Kent Beck, utiliza aserciones para comprobar resultados esperados. Tras la ejecución de las pruebas genera un informe indicando el número de pruebas ejecutadas y cuáles no se han ejecutado satisfactoriamente.

Existen dos tipos de fallos diferentes para una prueba

- ✓ Fallo: Indica que ha fallado una aserción, es decir, el código que se está probando no devuelve los resultados esperados, por lo que falla la comparación de los resultados.

- ✓ Error: Indica que ha ocurrido una excepción no esperada y que por tanto no se está capturando (ejemplo: NullPointerException o ArrayIndexOutOfBoundsException).

El objetivo de realizar las pruebas con JUnit Framework es que sean automáticas repetibles. Por tanto, cuando se ejecuten se deben crear los datos que sean precisos y luego recuperar el estado inicial, es decir, eliminar los datos de prueba para que puedan volver a ejecutarse más adelante.

El modo en que se aplicaron las pruebas fue creando una BD para pruebas, la cual es diferente de la BD utilizada en el entorno de producción y de esta manera los datos de las pruebas no entren en conflicto con los datos de ejecución. Se emplearon las anotaciones **@Before** y **@After** para definir los métodos a ejecutar antes y después de la ejecución de cada una de las pruebas de una clase. Se utilizan las anotaciones **@BeforeClass** y **@AfterClass** para definir los métodos a ejecutar antes y después de la ejecución del conjunto de pruebas de una clase.

Las pruebas se aplicaron a cada uno de los servicios implementados en el sistema para medir el funcionamiento de estos, tanto para validez como para tiempo de respuesta.

3.1.2 Ejecución de las pruebas.

A continuación, se mostrará el proceso de crear las pruebas con JUnit para el CUS Gestionar Pozo el cual también fue el escogido para la descripción de su implementación. Aclarar que este proceso se le realizó a cada uno de los CUS implementados en el subsistema y está reflejado en el código fuente del sistema de control. El objetivo es evidenciar como se realiza este proceso mediante un ejemplo.

La estructura de paquetes que plantea Java para sus proyectos es genérica. Esta estructura ya fue estudiada y comprobada para la realización del sistema de control, por lo cual es la que sigue en la implementación de este subsistema. En esta estructura existe un paquete llamado Test, el cual es donde se encontrarán todas las clases de pruebas del sistema. **Véase Anexo #19**

El paquete *Test* está formado por el paquete útil, este a su vez está constituido por clases que se utilizarán en todas las pruebas del sistema y son clases que se emplearán en todos los paquetes de prueba. El otro paquete de importancia es el **config.gestion.service** donde se encontrarán las clases de pruebas para cada uno de los servicios del subsistema de Configuración y Seguridad.

Para la realización de las pruebas del subsistema se estableció el criterio de una clase de prueba por servicio. Por ejemplo en el caso del servicio ICodPozoService se creó una clase PozoServiceTest, la cual tiene los métodos necesarios para la realización de las pruebas. Esta clase contiene varios métodos entre los que se encuentran los de inicialización y borrado de datos como el caso del populateDb y el cleanDb respectivamente, los cuales son utilizados para inicializar la base de datos y eliminar los datos de prueba

de la base de datos tratando de que esta mantenga el estado que tenía antes de realizar las pruebas. Recordar que se utilizará una nueva base de datos para realizar las pruebas, la cual es diferente de la que se encuentra orientada para la producción con el fin de evitar colisiones de los datos, aunque esto se considera también, ya que se trabaja con la eliminación de los datos de prueba. Los otros métodos que componen a la clase de prueba son los que tengan la anotación @Test encima que representarán casos de prueba.

La clase PozoServiceTest tiene métodos de inicialización y borrado de datos, además de los de prueba. **Véase Anexo 20**

Se puede observar que existen dos anotaciones que no han sido explicada **@BeforeClass** y **@AfterClass.** La primera anotación está asociada al método de inicialización de los juegos de datos para la prueba en la base de datos y es un método que se ejecuta antes de los métodos de prueba. La segunda es luego de terminar las pruebas ejecutar un método, el cual se encarga del borrado de datos generados por las pruebas realizadas. Con estas anotaciones el proceso de pruebas se realiza de forma automática sin tener que crear ningún objeto explícitamente en el código fuente.

La codificación del método populateDb en cada una de las clases de pruebas tiene que estar acorde con las pruebas que se desean realizar y este debe posibilitar la creación de datos de pruebas que van a ser utilizados. **Véase Anexo #21**

Además se puede observar la utilización de un método de la clase DbUtil, la cual es una clase que contiene métodos estáticos para la inicialización del contexto de la aplicación, utilizando la clase GlobalsName que tiene constantes con la localización de los ficheros XML de prueba para levantar el contexto de la aplicación. **Véase Anexo #22 y #23**

Una vez que los datos de prueba se encuentren insertados en la base de datos es necesario eliminar estos datos y los que puedan surgir en la ejecución de los métodos de pruebas, para esto se implementa el método cleanDb (**Véase Anexo #24**). Este método realiza la eliminación de los datos que fueron creados en el método anterior y en las pruebas realizadas, por esto realiza un recorrido por todos los pozos creados y los elimina. Luego se eliminan los datos de prueba que se crearon en el método populateDb, los cuales fueron almacenados en atributos estáticos.

Para la realización del método cleanDb se tienen en cuenta todos los datos que se generen en las pruebas realizadas por lo que se debe tener bien concebidas las pruebas, sobre los métodos que se realizarán y los datos nuevos que generan estos métodos. En el caso particular que se está analizando el PozoServiceTest, cuenta con 3 métodos de pruebas, los cuales responden a las necesidades actuales del subsistema.

Los métodos de pruebas que se conciben tienen que ver con crear, actualizar y eliminar un pozo. Debido a que todos los CUS son gestionar información básicamente la realización de las pruebas en gran medida se realizan de la misma manera, solo con cambio en los datos que se va a insertar. Cada uno de los métodos responde a un caso de prueba por lo que sí se desea agregar un nuevo caso de prueba se agregaría un nuevo método con el caso de prueba y la línea necesaria en el método cleanDb para borrar los datos que se puedan generar y que este método no tenga en cuenta. **Véase Anexo #25**

Luego se ejecutan las pruebas en el NetBeans que es el IDE utilizado, el cual permitió ejecutar las pruebas de manera automática devolviendo los resultados de estas. **Véase Anexo #26**

3.2. Evaluación por criterio de especialista.

Una vez realizadas las pruebas de unidad, se decidió utilizar un conjunto de especialistas, los cuales utilizando un conjunto de parámetros, evaluaron el subsistema. La utilización de este método de evaluación, además de las tradicionales pruebas de software, se lleva a cabo debido a que el subsistema aún no puede ser utilizado, ya que el sistema de control SISPEP, se encuentra aún en desarrollo.

Los principales elementos que se midieron fueron la usabilidad y operatividad del subsistema, utilizando el punto de vista de otros usuarios. Cabe aclarar la utilización de usuarios con conocimientos en la informática, los cuales basarán su respuesta en conocimiento técnico.

Los especialistas (**Véase Anexo #27**) hicieron un análisis del subsistema apoyándose en tres parámetros fundamentales, donde cada uno de estos parámetros tuvo un conjunto de puntos a medir (**Véase el Anexo #28**).

Los resultados de la validación (**Véase Anexo #29, #30, #31**) mostraron de manera general los porcentajes obtenidos por cada uno de los parámetros analizados. Obteniendo la siguiente puntuación final en cada uno de los parámetros expresados en porciento:

- ✓ Parámetros de la interfaces gráficas a medir: 95,71%
- ✓ Parámetros del subsistema enfocado a la solución informática: 100%
- ✓ Parámetros enfocados en la seguridad e integridad de los datos: 100%

3.3. Análisis y discusión de los resultados de la validación.

En el epígrafe 3.1 se muestra la realización de un conjunto de pruebas al software utilizando JUnit. Este conjunto de pruebas demostró que las funcionalidades se ejecutan de manera correcta y en un tiempo razonable, nunca pasándose del segundo. En cada uno de los casos de pruebas se trató de buscar deficiencias en los métodos, lo cual no fue encontrado. La realización de las pruebas no limita que los servicios estén exentos de errores, por lo que se recomienda pasar a otros tipos de pruebas más rigurosas las cuales pueden mostrar posibles fallos no encontrados en las pruebas aplicadas.

También se escogió a un conjunto de especialistas informáticos, los cuales emitieron sus criterios en una encuesta realizada a los mismos, referente al subsistema desarrollado. Luego del análisis realizado en el epígrafe 3.2, se obtuvo un resultado satisfactorio general del subsistema de un 97.57% que representa el promedio entre todos los parámetros analizados, permitiendo que el subsistema se encuentre listo para la gestión de la configuración y la seguridad del sistema de control.

Los métodos utilizados para la validación del subsistema son métodos empleados a nivel mundial. Las pruebas utilizadas son sustentadas por el papel actual que se desempeña y la etapa en cuestión para la realización del subsistema. Se prefirió además, utilizar el método de criterio de especialista para lograr una mayor certeza, permitiendo sustentar sólidamente el subsistema desarrollado.

Conclusiones del capítulo.

En este capítulo se logró validar el subsistema de control utilizando las pruebas unitarias con el Framework JUnit y el método de Especialistas. Se establecieron las bases para la realización de pruebas al subsistema y luego al sistema que aún está en construcción, se mostró además, mediciones no solo de validez de datos sino de tiempo de respuesta de los datos. Por lo tanto, se garantiza la validación del subsistema.

En este apartado, se puede ver como se integró de manera práctica el Framework Spring y el Framework JUnit para la realización de pruebas. Igualmente se demuestra el funcionamiento correcto de la lógica de las funcionalidades implementadas.

Conclusiones Generales.

Este subsistema de Configuración y Seguridad tiene como principal propósito la reducción o simplificación del esfuerzo destinado para la construcción del esqueleto de producción y la gestión de los parámetros de seguridad del sistema. Con este se trata de lograr un aumento en la calidad y en la eficiencia de actividades administrativas dentro del sistema de control SISPEP. Al mencionado subsistema se le realizó un conjunto de pruebas, a través de la cuales se pudo concluir que la creación del esquema de producción, la gestión de usuarios y la gestión de permisos se realiza de una manera más rápida y centralizada permitiendo un mejor control en la información.

Por lo que se puede arribar entonces a las siguientes conclusiones:

- ✓ El estudio detallado del estilo seleccionado por el arquitecto y su uso en la implementación logró una adecuada distribución de los componentes dentro del subsistema, los cuales también pueden ser usados en otros subsistemas que los necesiten.
- ✓ Con la implementación adecuada de los patrones de diseño se logró un subsistema de mayor calidad, claridad y sencillez tanto en la estructura como en el mantenimiento de código del mismo.
- ✓ Con la utilización de las herramientas, tendencias y tecnologías seleccionadas se logró un subsistema configurable y reprogramable.

Por lo tanto, se considera que el resultado final obtenido cumple con los requisitos planteados, constituyendo una solución viable al problema en análisis.

Recomendaciones

Como recomendaciones se establecen los siguientes puntos:

- ✓ Efectuar un continuo refinamiento de la implementación durante cada ciclo de desarrollo.
- ✓ Mejorar la concepción del módulo de seguridad, donde se establezca una manera más dinámica en la gestión de los permisos a la utilizada actualmente.
- ✓ Agregarle nuevas funcionalidades que hagan falta según nuevos cambios y concepciones por parte del cliente.
- ✓ Evaluar nuevamente los elementos que componen la base de datos y que las nuevas estructuraciones vayan enfocadas al framework Hibernate el cual es el que se utiliza para el desarrollo del sistema de manera general.
- ✓ Valorar la posibilidad de generalizar la utilización de este subsistema en las demás EPEP del país o en otras aplicaciones referentes a la industria petrolera.

Trabajos Citados

1. Peralta, M. *Sistema de Información*. 2009 [cited 1 de noviembre del 2009]; Available from: http://www.monografias.com/trabajos7/sisinf/sisinf.shtml.
2. Mesa, Y.R. *De la gestión de información a la gestión del conocimiento* 16 de febrero del 2006 [cited 1 de noviembre del 2009]; Available from: http://bvs.sld.cu/revistas/aci/vol14_1_06/aci02106.htm.
3. Europea, C. (2002) *La innovación en un nuevo panorama. Revista de Innovación y transferencia de tecnología.* Volume,
4. Aja, Q. 1 de noviembre 2009 [cited; Available from: http://www.GestionConocimientoInformacion.es/GestiónInformacion.pdf.
5. Borghello, C. *Auditoría y Control*. 2000-2009 [cited 11 de noviembre del 2009]; Available from: http://www.segu-info.com.ar/politicas/auditoria.htm.
6. Wikipedia. 7 de Octubre del 2009 [cited 1 de Noviembre del 2009]; Available from: http://es.wikipedia.org/wiki/Configuraci%C3%B3n_%28inform%C3%A1tica%29.
7. Creativecommons. [cited 1 de Noviembre del 2009]; Available from: http://creativecommonsorg/licenses/by-sa/3.0/deed.es.
8. Tanenbaum, A.W., *Redes de Computadoras Cuarta Edicion*, s.l. : Pearson.
9. Microsoft. *Seguridad computacional.* 20 de enero de 2009 [cited 1 de noviembre del 2009]; Available from: http://www.microsott.com.
10. Agrimensura, F.d.C.E.y.N.y. *Seguridad de los Sistemas Operativos*. 14 de julio de 2002 [cited 1 de noviembre del 2009]; Available from: http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/SO14.htm.
11. Agrimensura, F.d.C.E.y.N.y. *Sistema Operativo. Seguridad*. 14 de julio del 2002 [cited 11 de noviembre del 2009]; Available from: http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/MonogSO/SEG02.html.
12. Christopher Benson, I.C.P.L. *Estrategias de seguridad* 20 de julio de 2004 [cited 1 de noviembre del 2004]; Available from: http://www.microsoft.com/spain/technet/recursos/articulos/2005.aspx.
13. Borghello, C. *Estrategia de Seguridad*. 2000-2009 [cited 1 de noviembre del 2009; Available from:

http://www.segu-info.com.ar/politicas/seguridad.htm.

14. Borghello, C. *Plan de Contingencia*. 2000 - 2009 [cited 11 de noviembre del 2009]; Available from: http://www.segu-info.com.ar/politicas/contingencia.htm.
15. Borghello, C. *Backups / Copias de Seguridad*. 2000 - 2009 [cited 11 de noviembre del 2009]; Available from: http://www.segu-info.com.ar/politicas/backups.htm.

16. Fernández, Y.A.P., *Propuesta de Arquitectura de Software para la Empresa de Perforación y Extracción de Petróleo de Occidente*. 2009, Universidad de las Ciencias Informáticas: Cuba.

17. Andino, H.M., *Diseño de base de datos para el Sistema de Producción de la Empresa de Perforación y Extracción de Petróleo de Occidente (EPEPO)*. 2009, Universidad de las Ciencias Informáticas: Cuba.

Bibliografía.

1. Wikipedia. 7 de Octubre del 2009 [cited 1 de Noviembre del 2009]; Available from: http://es.wikipedia.org/wiki/Configuraci%C3%B3n_%28inform%C3%A1tica%29.
2. Tanenbaum, A.W., Redes de Computadoras Cuarta Edicion, s.l.: Pearson.
3. SpringHispano. 2009 [cited 1 de noviembre del 2010]; Available from: http://springhispano.org/.
4. Source, S. 2010 [cited 1 de noviembre del 2009]; Available from: http://www.springsource.org/.
5. PostgresSQl. 2010 [cited 1 de noviembre del 2009]; Available from: http://www.postgresql.org/.
6. Peralta, M. Sistema de Información. 2009 [cited 1 de noviembre del 2009]; Available from: http://www.monografias.com/trabajos7/sisinf/sisinf.shtml.
7. Monografia. Auditoría Informática. 16 de enero de 2008 [cited 1 de noviembre del 2009]; Available from: http://www.monografias.com/trabajos/auditoinfo/auditoinfo.shtml.
8. Microsoft. Seguridad computacional. 20 de enero de 2009 [cited 1 de noviembre del 2009]; Available from: http://www.microsott.com.
9. Mesa, Y.R. De la gestión de información a la gestión del conocimiento 16 de febrero del 2006 [cited 1 de noviembre del 2009]; Available from: http://bvs.sld.cu/revistas/aci/vol14_1_06/aci02106.htm.
10. María Eugenia Bupo, E.G., Gabriel Mauri, Gustavo Peman , Nicolás Rossi, Lucía Sanz La evolución del pensamiento administrativo: un análisis práctico. Mayo del 2004 [cited 1 de noviembre del 2009]; Available from: http://www.gestiopolis.com/recursos2/documentos/fulldocs/ger/evopensadm.htm.
11. LANVIN, D.F. Universidad de Oviedo. 17 de noviembre del 2009 [cited 1 de noviembre del 2009]; Available from: http://www.di.uniovi.es/~dflanvin/home/?download=SpringRCP.pptx.
12. Kriptopolis. Seguridad: El traje nuevo del emperador. 27 de julio de 2008 [cited 1 de noviembre del 2009].
13. Jimenez, J.A. Seguridad en un sistema de Información. Monografías. 2008 [cited 1 de noviembre del 2009]; Available from: http://www.monografias.com/trabajos/seguinfo/seguinfo.shtml.
14. International, V.P. Visual Paradigm. 2005 [cited 1 de noviembre del 2009]; Available from: http://www.visual-paradigm.com/.
15. Infoil. 2009 [cited 1 de noviembre del 2009]; Available from: http://www.infoil.com.ar/.
16. IBM. IBM Rational Unified Process (RUP). 2009 [cited 1 de noviembre del 2010]; Available from: http://www-01.ibm.com/software/awdtools/rup/.
17. Fernández, Y.A.P., Propuesta de Arquitectura de Software para la Empresa de Perforación y Extracción de Petróleo de Occidente. 2009, Universidad de las Ciencias Informáticas: Cuba.
18. Feamster, N. La seguridad en la Ingeniería de Software. Software Engineering Summer. 2001 [cited 1 de noviembre del 2009]; Available from: http://www.acm.org/crossroads/espanol/xrds7-4/onpatrol74.html.

19. Europea, C. (2002) La innovación en un nuevo panorama. Revista de Innovación y transferencia de tecnología. Volume,

20. Development, F. 2009 [cited 1 de noviembre de 2009]; Available from: http://www.fielddevelopment.com/.

21. Definición.de. Definición de seguridad informática. 2008 [cited 1 de noviembre del 2009]; Available from: http://definicion.de/seguridad-informatica/.

22. Creativecommons. [cited 1 de Noviembre del 2009]; Available from: http://creativecommonsorg/licenses/by-sa/3.0/deed.es.

23. Christopher Benson, I.C.P.L. Estrategias de seguridad 20 de julio de 2004 [cited 1 de noviembre del 2004]; Available from: http://www.microsoft.com/spain/technet/recursos/articulos/2005.aspx.

24. Bupo, M.E., y otros. [cited; Available from: http://www.gestiopolis.com/recursos2/documentos/fulldocs/ger/evopensadm.htm.

25. Borghello, C. Plan de Contingencia. 2000 - 2009 [cited 11 de noviembre del 2009]; Available from: http://www.segu-info.com.ar/politicas/contingencia.htm.

26. Borghello, C. Backups / Copias de Seguridad. 2000 - 2009 [cited 11 de noviembre del 2009]; Available from: http://www.segu-info.com.ar/politicas/backups.htm.

27. Borghello, C. Auditoría y Control. 2000-2009 [cited 11 de noviembre del 2009]; Available from: http://www.segu-info.com.ar/politicas/auditoria.htm.

28. Borghello, C. Implementación de una Política de Seguridad. 2000-2009 [cited 1 de noviembre del 2009]; Available from: http://www.segu-info.com.ar/politicas/implementacion.htm.

29. Borghello, C. Estrategia de Seguridad. 2000-2009 [cited 1 de noviembre del 2009; Available from:

http://www.segu-info.com.ar/politicas/seguridad.htm.

30. Alfasa. Módulo de Seguridad. 2008 [cited 1 de noviembre de 2009]; Available from: http://www.alfasa.com/seguridad.htm.

31. Aja, Q. 1 de noviembre 2009 [cited; Available from: http://www.GestionConocimientoInformacion.es/GestiónInformacion.pdf.

32. Agrimensura, F.d.C.E.y.N.y. Sistema Operativo. Seguridad. 14 de julio del 2002 [cited 11 de noviembre del 2009]; Available from: http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/MonogSO/SEG02.html.

33. Agrimensura, F.d.C.E.y.N.y. Seguridad de los Sistemas Operativos. 14 de julio de 2002 [cited 1 de noviembre del 2009]; Available from: http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/SO14.htm.

34. (RCP), S.R.C.P. 2009 [cited 1 de noviembre del 2009]; Available from: http://spring-rich-c.sourceforge.net/1.1.0/index.html.

35. (RCP), S.R.C.P. 25 de agosto de 2007 [cited 1 de noviembre del 2009]; Available from: http://opensource.atlassian.com/confluence/spring/display/RCP/User+Documentation.

Anexos.

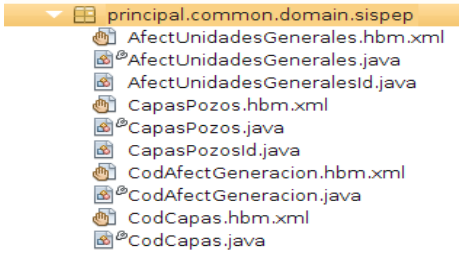
Anexo 1 Imagen del fichero XML hibernate.cfg.xml

```
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property>
    <property name="hibernate.connection.driver_class">org.postgresql.Driver</property>
    <property name="hibernate.connection.url">jdbc:postgresql://10.34.1.50:5800/sispep</property>
    <property name="hibernate.connection.username">postgres</property>
  </session-factory>
</hibernate-configuration>
```

Anexo 2 Imagen del fichero hibernate.reveng.xml

```
<hibernate-reverse-engineering>
  <schema-selection match-catalog="sispep" match-schema="public"/>
  <table-filter match-name="param_fwko"/>
  <table-filter match-name="capas_pozos"/>
  <table-filter match-name="planes_prod"/>
  <table-filter match-name="cod_metodos"/>
  <table-filter match-name="cod_prod_fund"/>
  <table-filter match-name="cod_tipo_instalacion"/>
  <table-filter match-name="cod_categ_prod"/>
  <table-filter match-name="cod_tanque"/>
  <table-filter match-name="cod_entidades"/>
```

Anexo 3 Imagen del paquete principal.common.domain.sispep



Anexo #4 Imagen del código IBaseDaoImpl

```
public interface IBaseDao {
    public void persist(Object object);

    public void save(Object object);

    public void saveOrUpdate(Object object);

    public void update(Object object);

    public Object load(Class clazz, long id);

    public Object get(Class clazz, long id);

    public Object get(Class  clazz, Serializable id);

    public Object get(String clazzString, long id);

    public Object load(Class clazz, String id);

    public Object uniqueByField(Class clazz, String fieldName, Object value) throws IndexOutOfBoundsException ;

    public List allByField(Class clazz, String fieldName, Object value);

    public List loadAll(Class clazz);

    public void flush();

    public void refresh(Object obj);

    public void delete(Object obj);
```

Anexo #5 Imagen del código BaseDaoImpl

```
public class BaseDaoImpl extends HibernateDaoSupport implements  IBaseDao{

    public void persist(Object object) {
        getHibernateTemplate().persist(object);
    }
    public void save(Object object) {
        getHibernateTemplate().save(object);
    }
    public void saveOrUpdate(Object object) {
        getHibernateTemplate().saveOrUpdate(object);
    }
    public void update(Object object) {
        getHibernateTemplate().update(object);
    }
    public Object load(Class clazz, long id) {
        return getHibernateTemplate().load(clazz, id);
    }
    public Object get(Class clazz, long id) {
        return getHibernateTemplate().get(clazz, id);
    }
    public Object get(String clazzString, long id) {
        Class clazz = null;
        try {
            clazz = Class.forName(clazzString);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
            return null;
        }
    public List allByField(Class clazz, String fieldName, Object value) {
        return getSession().createCriteria(clazz).add(Restrictions.eq(fieldName, value)).list();
    }
    public List loadAll(Class clazz) {
        return getHibernateTemplate().loadAll(clazz);
    }
    public void flush() {
        getHibernateTemplate().flush();
    }
    public void  refresh(Object obj) {
          getHibernateTemplate().refresh(obj);
    }
    public void delete(Object obj){
      getHibernateTemplate().delete(obj.getClass().getName(), obj);
    }
    public Object get(Class clazz,Serializable  id){
       return getHibernateTemplate().get(clazz, id);
   }
}
```

Anexo #6 Imagen del fichero ICodPozoDao

```
public interface ICodPozosDao extends IBaseDao{


}
```

Anexo #7 Imagen del fichero CodPozoDaoImpl

```
public class CodPozosImpDao extends BaseDaoImpl implements ICodPozosDao{

}
```

Anexo #8 Imagen de la interface ICodPozoService

```
public interface ICodPozosService {
    public void salvarActualizar(CodPozos obj);
    public void salvar(CodPozos obj);
    public void actualizar(CodPozos obj);
    public void eliminarPorId(CodPozosId id);
    public CodPozos getCodPozos(CodPozosId id);
    public List getAllCodPozos();
    public ICodPozosDao getCodPozosDao();
    public void setCodPozosDao(ICodPozosDao codPozosDao);
}
```

Anexo #9 Imagen de la clase CodPozoServiceImpl

```
public class CodPozosImpService implements ICodPozosService{
    private IBaseDao codPozosDao;
    public void salvar(CodPozos obj){
        codPozosDao.save(obj);
    }
    public void actualizar(CodPozos obj){
        codPozosDao.update(obj);
    }
    public void eliminarPorId(CodPozosId id) {
        codPozosDao.delete(getCodPozos(id));
    }
    public CodPozos getCodPozos(CodPozosId id) {
       return (CodPozos) codPozosDao.get(CodPozos.class, id);
    }
    public List getAllCodPozos() {
        return codPozosDao.loadAll(CodPozos.class);
    }
    public ICodPozosDao getCodPozosDao() {
        return (ICodPozosDao) codPozosDao;
    }
    public void setCodPozosDao(ICodPozosDao codPozosDao) {
        this.codPozosDao = codPozosDao;
    }
}
```

Anexo #10 Imágenes del Código del data editor de pozos.

```
public class PozoDataEditor extends DefaultDataEditorWidget
{
    public PozoDataEditor(PozoDataProvider itemDataProvider)
    {
        super("pozoDataEditor", itemDataProvider);

        setDetailForm(new PozoForm());
        setFilterForm(new PozoFilterForm());

        PropertyColumnTableDescription tableDescription = new PropertyColumnTableDescription("pozoDataEditor",
CodPozos.class);
        tableDescription.addPropertyColumn("id.nombre");
        //tableDescription.addPropertyColumn("descripcion");
       tableDescription.addPropertyColumn("id.idInst");
       tableDescription.addPropertyColumn("codInstalaciones.codEntidades.siglas");
       tableDescription.addPropertyColumn("codYacimientos.nombre");
       tableDescription.addPropertyColumn("codTipoPozo.descripcion");
       tableDescription.addPropertyColumn("codCategProd.descripcion");
       tableDescription.addPropertyColumn("codCompaniasByIdPropietario.nombre");
       tableDescription.addPropertyColumn("codCompaniasByIdOperario.nombre");
       tableDescription.addPropertyColumn("codMetodos.descripcion");
       tableDescription.addPropertyColumn("codCiclo.descripcion");
       tableDescription.addPropertyColumn("codProdFund.descripcion");
       tableDescription.addPropertyColumn("strCoefA");
       tableDescription.addPropertyColumn("strCoefP");

        //tableDescription.addPropertyColumn("supplier.name");
        setTableWidget(tableDescription);

    }
```

Anexo #11 Imagen del fichero message.properties

```
#pozos form
codCiclo.label=Ciclo de Prod.
codCategProd.label=Categoria
codProdFund.label=Prod. Fund.
codCompaniasByIdPropietario.label=Propietario
codCompaniasByIdOperario.label=Operador
codYacimientos.label=Yacimiento
codMetodos.label=Metodo
```

Anexo #12 Imagen del código del Objeto Form

```
public class PozoForm extends TabbedForm {

    private CodEntidades entidad;
    private JComponent focus;
    private List someList;
    RefreshableValueHolder yacimientoPropListHolder = new RefreshableValueHolder(new Closure() {

        public Object call(Object o) {
            return someList;
        }
    }, true, true);

    public PozoForm() {
        super(FormModelHelper.createFormModel(new CodPozos(), "pozoForm"));
        YacimientoService servicio = (YacimientoService) Application.services().getService(YacimientoService.class);
        someList = servicio.findYacimiento(new YacimientoFilter());
    }

    @Override
    protected Tab[] getTabs() {
        SwingBindingFactory bind = (SwingBindingFactory) this.getBindingFactory();
        TableFormBuilder tableForm = new TableFormBuilder(bind);
        tableForm.addSeparator("General");
        tableForm.row();

        //cod de instalacion
        RefreshableValueHolder instalacionListHolder = new RefreshableValueHolder(new Closure() {

            public Object call(Object o) {

                InstalacionService servicio = (InstalacionService) Application.services().getService
(InstalacionService.class);
                return servicio.findInstalacion(new InstalacionFilter());
            }
        }, true, false);
```

```
        //       categorias de produccion
        RefreshableValueHolder catProdListHolder = new RefreshableValueHolder(new Closure() {

            public Object call(Object o) {
                CategProdService servicio = (CategProdService) Application.services().getService(CategProdService.class);
                return servicio.getAllCategProd();

            }
        }, true, false);
        //ciclo de producción
        RefreshableValueHolder cicloListHolder = new RefreshableValueHolder(new Closure() {

            public Object call(Object o) {
                CicloService servicio = (CicloService) Application.services().getService(CicloService.class);
                return servicio.getAllCiclos();
            }
        }, true, false);

        //produccion Fundamental
        RefreshableValueHolder prodFundListHolder = new RefreshableValueHolder(new Closure() {

            public Object call(Object o) {
                ProdFundamentalService servicio = (ProdFundamentalService) Application.services().getService
(ProdFundamentalService.class);
                return servicio.getAllCodProdFundamentales();
            }
        }, true, false);
        //company propietario
        final RefreshableValueHolder companyPropListHolder = new RefreshableValueHolder(new Closure() {

            public Object call(Object o) {
                CompaniaService service = (CompaniaService) Application.services().getService(CompaniaService.class);
                return service.getAllCodCompany();
            }
        }, false, false);

     tableForm.add(bind.createBoundComboBox("codTipoPozo", tipoPozoPropListHolder, "descripcion"));
     tableForm.add(bind.createBoundComboBox("codCategProd", catProdListHolder, "descripcion"));
     tableForm.row();

     Binding binding = bind.createBoundComboBox("codCompaniasByIdPropietario", companyPropListHolder, "nombre");
     JComboBox combo1 = (JComboBox) binding.getControl();
     combo1.addPopupMenuListener(new PopupMenuListener() {

         public void popupMenuWillBecomeVisible(PopupMenuEvent pme) {
             companyPropListHolder.refresh();
             //throw new UnsupportedOperationException("Not supported yet.");
         }

         public void popupMenuWillBecomeInvisible(PopupMenuEvent pme) {
             //throw new UnsupportedOperationException("Not supported yet.");
         }

         public void popupMenuCanceled(PopupMenuEvent pme) {
             //throw new UnsupportedOperationException("Not supported yet.");
         }
     });
     tableForm.add(binding);
     tableForm.add(bind.createBoundComboBox("codMetodos", metodoPropListHolder, "descripcion"));
     tableForm.row();
     Binding binding1 = bind.createBoundComboBox("codCompaniasByIdOperario", companyOperListHolder, "nombre");
      JComboBox combo2 = (JComboBox) binding1.getControl();
     combo2.addPopupMenuListener(new PopupMenuListener() {
```

```
        // metodo de explotacion
        RefreshableValueHolder metodoPropListHolder = new RefreshableValueHolder(new Closure() {

            public Object call(Object o) {
                MetodoExplotacionService servicio = (MetodoExplotacionService) Application.services().getService
(MetodoExplotacionService.class);
                return servicio.getAllCodMetodoExplotacion();

            }
        }, true, false);

        //tipo de Pozo
        RefreshableValueHolder tipoPozoPropListHolder = new RefreshableValueHolder(new Closure() {

            public Object call(Object o) {
                TipoPozoService servicio = (TipoPozoService) Application.services().getService(TipoPozoService.class);
                return servicio.getAllCodTipoPozo();
            }
        }, true, false);


        focus = tableForm.add("id.nombre", "colSpan=1 align=left")[1];
        tableForm.row();
        tableForm.add("codInstalaciones");
        //tableForm.add(bind.createBoundComboBox("codInstalaciones", instalacionListHolder, "id.idInst"));
        //addFormValueChangeListener("codInstalaciones", new ChangedValueListener());


        tableForm.add("codYacimientos");
       // tableForm.add(bind.createBoundComboBox("codYacimientos", yacimientoPropListHolder, "nombre"));
        tableForm.row();


        tableForm.row();
       }
    });
    tableForm.add(binding);
    tableForm.add(bind.createBoundComboBox("codMetodos", metodoPropListHolder, "descripcion"));
    tableForm.row();
    Binding binding1 = bind.createBoundComboBox("codCompaniasByIdOperario", companyOperListHolder, "nombre");
     JComboBox combo2 = (JComboBox) binding1.getControl();
    combo2.addPopupMenuListener(new PopupMenuListener() {

        public void popupMenuWillBecomeVisible(PopupMenuEvent pme) {
            companyOperListHolder.refresh();
            //throw new UnsupportedOperationException("Not supported yet.");
        }

        public void popupMenuWillBecomeInvisible(PopupMenuEvent pme) {
            //throw new UnsupportedOperationException("Not supported yet.");
        }

        public void popupMenuCanceled(PopupMenuEvent pme) {
            //throw new UnsupportedOperationException("Not supported yet.");
        }
    });
    tableForm.add(binding1);
    tableForm.add(bind.createBoundComboBox("codCiclo", cicloListHolder, "descripcion"));
    tableForm.row();


    tableForm.add(bind.createBoundComboBox("codProdFund", prodFundListHolder, "descripcion"), "colSpan=3 align=left");


       return new Tab[]{new Tab("primerTab", tableForm.getForm())};//tableForm.getForm();

   }

   @Override
   public void commit() {
       super.commit();
       CodPozos pozo = (CodPozos) getFormObject();
       pozo.getId().setEntidad(pozo.getCodInstalaciones().getId().getEntidad());
       pozo.getId().setIdInst(pozo.getCodInstalaciones().getId().getIdInst());

   }

   public boolean requestFocusInWindow() {
       return focus.requestFocusInWindow();
   }
```

Anexo #13 Imagen del Fichero conexión.xml

```
<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName">
        <value>org.postgresql.Driver</value>
    </property>
    <property name="url">
        <value>jdbc:postgresql://10.34.1.50:5800/sispep</value>
    </property>
    <property name="username">
        <value>postgres</value>
    </property>
    <property name="password">
        <value>proyecto</value>
    </property>
</bean>
<bean id="sessionFactory" class="org.springframework.orm.hibernate3.LocalSessionFactoryBean
    <property name="dataSource" ref="dataSource"/>
    <property name="mappingResources">
        <list>
            <value>principal/common/domain/sispep/CodCiclo.hbm.xml</value>
```

Anexo #14 Imagen del fichero hibernate.cfg.properties.

```
hibernate.connection.url=jdbc\:postgresql\://10.34.1.50\:5800/sispep
hibernate.connection.driver_class=org.postgresql.Driver
hibernate.connection.username=postgres
hibernate.connection.password=proyecto
hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
hibernate.show_sql=true
hibernate.hbm2ddl.auto=true
```

Anexo #15 Imagen del Fichero Dao.xml

```
<bean id="baseDao"  class="principal.common.dao.BaseDaoImpl">
    <property name="hibernateTemplate" ref="hibernateTemplate"/>
</bean>
<bean id="codPozoDao" parent="baseDao" class="principal.config.gestion.dao.CodPozosImpDao">
</bean>
<bean id="codAfectacionesDao" parent="baseDao" class="principal.config.gestion.dao.CodAfectacionesImpDao">
</bean>
```

Anexo #16 Imagen del Fichero service.xml

```
<bean id="codPozoService"  class="principal.config.gestion.service.CodPozosImpService">
    <property name="codPozosDao" ref="codPozoDao"></property>
</bean>
<bean id="codAfectacionesService"  class="principal.config.gestion.service.CodAfectacionesImpService">
    <property name="objAfectaciones" ref="codAfectacionesDao"></property>
</bean>
```

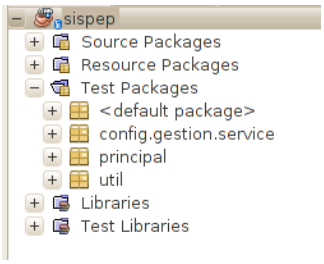
Anexo #17 Imagen del Fichero richclient-context.xml

```
<bean id="lifecycleAdvisor" class="principal.SimpleLifecycleAdvisor">
    <property name="windowCommandBarDefinitions" value="principal/common/presentacion/command/commands-cc
    <property name="windowCommandManagerBeanName" value="windowCommandManager" />
    <!--<property name="startingPageId" value="proxyPage" />-->
    <property name="menubarBeanName" value="menuBar" />
    <property name="toolbarBeanName" value="toolBar" />
    <property name="statusBar" ref="statusBar1" />

</bean>

<bean id="applicationPageFactory" depends-on="serviceLocator"
    class="org.springframework.richclient.application.docking.flexdock.FlexDockApplicationPageFactory">
    <property name="floatingEnabled" value="true"/>
    <property name="defaultPerspective" value="defaultPerspective"/>
    <property name="perspectiveFactory" ref="perspective"/>
</bean>
<bean id="perspective" class="principal.common.presentacion.layout.DemoPerspectiveFactory">
    <property name="dockableIds">
        <list>
            <value>inicioView</value>
        </list>
    </property>
</bean>

<bean id="serviceLocator" class="org.springframework.richclient.application.ApplicationServicesLocator">
    <property name="applicationServices" ref="applicationServices" />
</bean>
```

Anexo #18 Imagen del Fichero command-context.xml

```
<bean id="windowCommandManager"
          class="org.springframework.richclient.application.support.ApplicationWindowCommandManager">
    <property name="sharedCommandIds">
        <list>
            <value>propertiesCommand</value>
            <value>deleteCommand</value>
            <value>cutCommand</value>
            <value>copyCommand</value>
                        <!--<value>miComandoGlobalCommand</value>-->
            <value>newCommand</value>
        </list>
    </property>

</bean>
```

Anexo #19 Imagen del paquete Test.



Anexo #20 Imagen estructura de la clase PozoServiceTest

```
public class PozoServiceTest {

    private static ICodPozosService pozoService ;
    private static CodCiclo testciclo;
    private static CodCompanias testOCompannia;
    private static CodCompanias testPCompannia;
    private static CodMetodos testMetodo;
    private static CodInstalaciones testInstalaciones;
    private static CodPozosId testCodPozosId;
    private static CodCategProd testCodCategoria;
    private static CodPozos testpozo;
    private static CodYacimientos testCodYacimiento;
    private static CodEntidades testEntidades;
    private static CodProdFund testProdFundamental;
    private static CodTipoPozo testCodTipoPozo;

    @BeforeClass
    public static void populateDb() {}

    @Test
    public void testCreatePozo() throws InstanceNotFoundException {}

    @Test
    public void testUpdatePozo() throws InstanceNotFoundException {}

    @Test
    public void testDeletePozo() throws InstanceNotFoundException {}

    @AfterClass
    public static void cleanDb() throws Exception {}

    public ICodPozosService getPozoService() {}

    public void setPozoService(ICodPozosService pozoService) {}
}
```

Anexo #21 Imagen del método populateDb.

```
    @BeforeClass
    public static void populateDb() {
        DbUtil.populateDb();
        pozoService = DbUtil.getCodPozosService();
        IBaseDao dao = DbUtil.getDAO();
        testEntidades  = new CodEntidades(1, "EPEPOTEST", "Entidad de Prueba");
        testciclo = new CodCiclo("Cic", "Ciclo de Prueba");
        testOCompannia = new CodCompanias("CUPETTEST");
        testPCompannia = new CodCompanias("CHERRY");
        testMetodo     = new CodMetodos("Met", "Metodo1");
        testInstalaciones = new CodInstalaciones(new CodInstalacionesId(testEntidades.getEntidad(),
"Ins"),testEntidades,new CodTipoInstalacion('c', "Centro Colector Test"), "InstPrueba");
        testCodYacimiento  = new CodYacimientos("YAC", testEntidades, "Boca de Jaruco");
        testCodCategoria  =  new CodCategProd("CAT", "CategoriaPrueba");
        testProdFundamental = new CodProdFund('A', "AguaTest");
        testCodTipoPozo = new CodTipoPozo('P', "ExploracionTest");
        dao.save(testciclo);
        dao.save(testCodTipoPozo);
        dao.save(testOCompannia);
        dao.save(testPCompannia);
        dao.save(testMetodo);
        dao.save(testEntidades);
        dao.save(testInstalaciones);
        dao.save(testCodCategoria);
        dao.save(testCodYacimiento);
        dao.save(testProdFundamental);
        testpozo = new CodPozos();
        testpozo.setId(new CodPozosId("pozo2test", testInstalaciones.getId().getEntidad(), testInstalaciones.getId
().getIdInst()));
        testpozo.setCodCiclo(testciclo);
        testpozo.setCodCompaniasByIdOperario(testOCompannia);
        testpozo.setCodCompaniasByIdPropietario(testPCompannia);
        testpozo.setCodMetodos(testMetodo);
        testpozo.setCodInstalaciones(testInstalaciones);
        testpozo.setDescripcion("Pozo de Prueba");
```

```
        testpozo.setCodTipoPozo(testCodTipoPozo);
        testpozo.setCodYacimientos(testCodYacimiento);
        testpozo.setCodProdFund(testProdFundamental);
        testpozo.setCodCategProd(testCodCategoria);
        testpozo.setCoefA(true);
        testpozo.setCoefP(false);
        dao.save(testpozo);
        testCodPozosId = testpozo.getId();
    }
```

Anexo #22 Imagen de la clase DbUtil.

```
public class DbUtil {

        private static IBaseDao DAO;
        private static ApplicationContext context;

    public static void populateDb() {
        context = new ClassPathXmlApplicationContext(new String[]{SPRING_CONFIG_TEST_FILE_CONEXION,
                        SPRING_CONFIG_TEST_FILE_DAO, SPRING_CONFIG_TEST_FILE_SERVICE});
        setDAO((IBaseDao) context.getBean("baseDao"));
    }
    public static ICodPozosService getCodPozosService(){
        return (ICodPozosService) context.getBean(ApplicationUtil.BEAN_COD_POZO_SERVICE);
    }

     public static IBaseDao getDAO() {
        return DAO;
    }
    public static void setDAO(IBaseDao aDAO) {
        DAO = aDAO;
    }
}
```

Anexo #23 Imagen de la clase GlobalsName.

```
public final class GlobalNames {

    public static final String SPRING_CONFIG_TEST_FILE_CONEXION =
    "classpath:/conexion.xml";

    public static final String SPRING_CONFIG_TEST_FILE_DAO =
    "classpath:daos.xml";

    public static final String SPRING_CONFIG_TEST_FILE_SERVICE =
    "classpath:/service.xml";

private GlobalNames () {}


}
```

Anexo #24 Imagen del método cleanDb.

```
 @AfterClass
 public static void cleanDb() throws Exception {


     IBaseDao dao = DbUtil.getDAO();
     List pozos = dao.loadAll(CodPozos.class);
     for (Iterator it = pozos.iterator(); it.hasNext();) {
         CodPozos object = (CodPozos) it.next();
         dao.delete(object);

     }


     dao.delete(testInstalaciones);
     dao.delete(testCodYacimiento);
     dao.delete(testEntidades);
     dao.delete(testciclo);
     dao.delete(testCodTipoPozo);
     dao.delete(testOCompannia);
     dao.delete(testPCompannia);
     dao.delete(testMetodo);
     dao.delete(testCodCategoria);
     dao.delete(testProdFundamental);


 }
```

Anexo #25 Imagen de la implementación de los métodos de prueba de la clase PozoServiceTest.

```
@Test
public void testCreatePozo() throws InstanceNotFoundException {

    CodPozosId idPozo = new CodPozosId();
    idPozo.setNombre("pozotest");
    idPozo.setEntidad(testEntidades.getEntidad());
    idPozo.setIdInst(testInstalaciones.getId().getIdInst());
    CodPozos pozo1 = new CodPozos();
    pozo1.setId(idPozo);
    pozo1.setCodCiclo(testciclo);
    pozo1.setCodCompaniasByIdOperario(testOCompannia);
    pozo1.setCodCompaniasByIdPropietario(testPCompannia);
    pozo1.setCodMetodos(testMetodo);
    pozo1.setCodInstalaciones(testInstalaciones);
    pozo1.setDescripcion("Pozo de Prueba");
    pozo1.setCodTipoPozo(testCodTipoPozo);
    pozo1.setCodYacimientos(testCodYacimiento);
    pozo1.setCodProdFund(testProdFundamental);
    pozo1.setCodCategProd(testCodCategoria);
    pozo1.setCoefA(true);
    pozo1.setCoefP(true);

    if (getPozoService() != null){
    getPozoService().salvar(pozo1);
    CodPozos pozo2 = getPozoService().getCodPozos(pozo1.getId());

    Assert.assertEquals(pozo1.getId(), pozo2.getId());
    }
}
```

```
@Test
public void testUpdatePozo() throws InstanceNotFoundException {

    testpozo.setDescripcion("Pozo de Prueba Actualizado");
    getPozoService().actualizar(testpozo);
    CodPozos pozo2 = getPozoService().getCodPozos(testpozo.getId());
    Assert.assertEquals(testpozo.getId(), pozo2.getId());
}
```

```
@Test
public void testDeletePozo() throws InstanceNotFoundException {
    getPozoService().eliminarPorId(testCodPozosId);
    CodPozos pozo2 = getPozoService().getCodPozos(testCodPozosId);
    Assert.assertEquals(pozo2, null);
}
```

Anexo #26 Tabla de resultados de las pruebas con el Framework JUnit al CUS Gestionar Pozos.

Método	Tiempo de Ejecución en Segundos	Resultado del Caso de Prueba
testCreatePozo	0,098	exitoso
testUpdatePozo	0,05	exitoso
testDeletePozo	0,073	exitoso

Anexo #27 Tabla de especialista

No.	Nombre del Especialista
1	Ing. Rolando Toledo Fernández
2	Ing. Alejandro Orgelio Hernández Cebrián
3	Ing. Adrian Gracia Águila
4	Ing. Maykel López Oliva
5	Ing. Carlos Enrique Ramírez Martin

6	Ing. Yancy Martínez Pérez
7	Ing. Armando Ortíz Cabrera

Anexo #28 Plantilla de Validación del Subsistema de Configuración y Seguridad para el sistema SISPEP.

Nombre del Especialista: _____ Facultad___

Cargo que Ocupa_____

Cada uno de estos parámetros recibe una puntuación de 1 a 5 puntos. Siendo 5 puntos la puntuación más alta que puede tener un parámetro.

Parámetros de la interfaces gráficas a medir:

- ✓ Tiempo de respuesta de la interface para una acción determinada.
- ✓ Validación de los datos de entrada por el usuario.
- ✓ Funcionalidades de edición: copiar, cortar y pegar.
- ✓ Visibilidad de los datos mostrados.
- ✓ Mensajes de errores personalizados.
- ✓ Personalización de las vistas mostradas.
- ✓ Iconos representativos.

Parámetros del subsistema enfocado a la solución informática:

- ✓ Búsqueda de datos dentro del subsistema.
- ✓ Concordancia entre los parámetros gestionados y la base de datos actual.
- ✓ Descomposición informática del problema.
- ✓ Cumplimiento de la arquitectura propuesta por el arquitecto.

Parámetros enfocados en la seguridad e integridad de los datos:

- ✓ Integridad en los datos insertados en la base de datos.
- ✓ Algoritmo de encriptación utilizado para la contraseña de los usuarios.

Anexo #29 Tabla de resultados de la validación del Subsistema de Configuración y Seguridad para el sistema SISPEP, análisis de los parámetro de las Interfaces gráficas.

	Especialistas							
Puntos a medir en las interfaces	1	2	3	4	5	6	7	
Tiempo de respuesta de las interfaces para una acción determinada.	5	5	5	5	5	5	5	

Validación de los datos de entrada por el usuario.	5	5	5	3	5	5	5	
Funcionalidades de edición.	5	5	5	5	5	5	5	
Visibilidad de los datos mostrados.	5	5	5	4	5	5	5	
Mensajes de errores personalizados.	5	5	5	5	5	5	5	
Personalización de las vistas mostradas.	5	5	5	5	5	5	5	
Iconos representativos.	5	5	5	5	5	5	5	

Anexo #30 Tabla de resultados de la validación del Subsistema de Configuración y Seguridad para el sistema SISPEP, análisis de los parámetros enfocados en la solución informática.

	Especialistas							
Puntos a medir enfocados a la solución informática	1	2	3	4	5	6	7	8
Búsqueda de datos dentro del subsistema.	5	5	5	5	5	5	5	
Validación de los datos de entrada por el usuario.	5	5	5	5	5	5	5	
Concordancia entre los parámetros gestionados y la base de datos actual.	5	5	5	5	5	5	5	
Descomposición informática del problema.	5	5	5	5	5	5	5	
Cumplimiento de la arquitectura propuesta por el arquitecto.	5	5	5	5	5	5	5	

Anexo #31 Tabla de resultados de la validación del Subsistema de Configuración y Seguridad para el sistema SISPEP, análisis de los parámetros enfocados a la seguridad e integridad de los datos.

	Especialistas							
Puntos a medir enfocados en la seguridad e integridad de los datos	1	2	3	4	5	6	7	8
Integridad en los datos insertados en la base de datos.	5	5	5	5	5	5	5	
Algoritmo de encriptación utilizado para la contraseña de los usuarios.	5	5	5	5	5	5	5	

Glosario de términos.

Petróleo: Líquido natural oleaginoso e inflamable, constituido por una mezcla de hidrocarburos, que se extrae de lechos geológicos continentales y marítimos y tiene múltiples aplicaciones químicas e industriales.

Ítems: Elementos simples que se encuentran contenidos como parte de un dato o de un elemento compuesto.

Configuración: Disposición y forma de las partes que componen un todo.

Funcionalidad: Conjunto de características que hacen que algo sea práctico y utilitario.

Flexible: Elemento que se acomoda con facilidad a distintas situaciones o a las propuestas de otros.

Yacimiento: Sitio donde se halla naturalmente una roca madre, un mineral, un fósil, o restos arqueológicos.

Componente: Pieza o elemento que forma parte un todo u otro elemento.

Crudo: Petróleo sin refinar.

Consulta: Búsqueda de datos o información.

Viscosidad: Propiedad de los fluidos que se gradúa por la velocidad de salida de aquellos a través de tubos capilares.

Autenticar: Identificar la existencia de un elemento que forma parte de un ámbito y está autorizado a realizar determinadas acciones en ese ámbito.

Mapeo: Correlación de conceptos que se encuentran concebidos de distinta forma en diferentes medios.

Medición: No es más que el control que se le realiza a los pozos de parámetros determinados para de esta manera saber cuánto en realidad está produciendo. Estas mediciones se realizan también en los Centro Colectores.

Framework: Conjunto de clases, funciones y librerías, las cuales están implementadas en un lenguaje de programación específico.

Sistema: Conjunto de piezas relacionadas entre sí que conforman un todo.

Lenguaje de programación: Conjunto de instrucciones u órdenes que pueden ser comprendidas por el ordenador para la ejecución de una acción determinada.

Traza: El registro de las acciones realizadas por un ente.

Carpeta: Es la agrupación virtual de ficheros de datos en la computadora siguiendo una organización o criterio establecido por el usuario.

Fichero: Es un conjunto de información que se guarda de manera virtual en un computador para ser accedido.