Universidad de las Ciencias Informáticas

Facultad 5

Título: Módulo de Seguridad para SIPP v 2.0.

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

Autor(es): Joyce Suárez Fabre

Tutor: Ing. Yidian Y. Castellanos Sabarí

Co-tutores: Ing. Omar Martínez Díaz

Ing. David Tavares Cuevas

La Habana, 2013.

"Año 55 de la Revolución"


Seamos realistas, soñemos lo imposible.
Ernesto Che Guevara

Agradecimientos:

Le agradezco a la Revolución y en especial a nuestro Comandante en Jefe Fidel Castro por haberme dado la oportunidad de convertirme en Ingeniera en Ciencias Informáticas.

Le agradezco a mi abuela por confiar siempre en mí, por los sacrificios, por el inmenso amor que me ofreció, siempre estarás en mi corazón viejiviri, sin ti no hubiera llegado tan lejos.

Le agradezco a mi mami por ser mi mejor amiga, mi confidente y consuelo, por escucharme, entenderme y apoyarme siempre, por ser quien seca mis lágrimas y disfruta mis triunfos, gracias por todo lo que me has dado.

A mi papá por ser ejemplo e inspirarme a ser siempre una mejor persona, por sus consejos y su paciencia, por ser el mejor hombre que conozco.

A mi hermano Nelson por confiar en mí en todo momento y ser mi guía.

A mi hermano Noel David por la alegría que trajo a mi vida, por tenerme de ejemplo e inspirarme a hacer siempre lo correcto para mostrarle el camino.

A Neisy y Giraldo por haber entrado en mi familia para causarnos alegrías. Gracias por quererme tanto.

A mi tía Jania por recibirme siempre alegre, por escucharme, por el inmenso amor que profesa, sin ti estos 5 años hubieran sido mucho más difíciles.

A mi familia habanera por el apoyo, por estar cuando se necesitaba, por ser la mejor familia del mundo.

Agradezco al mejor amigo que uno puede soñar tener, al que cumple con todos los requisitos, ese que me ha visto llorar, reír, soñar, desmoronarme y levantarme, no hubiera llegado a ser lo que soy hoy si no me hubiera encontrado con él, a Livan, por los buenos y los malos momentos, por los sueños y por los perros azules con pintas amarillas.

A Orson por el apoyo, por los buenos momentos, por la sonrisa, por la paciencia, por los nenishis, por convencerme de que podía lograr mucho aun cuando yo creía que ya estaba vencida, por hacerme volar lo más lejos que he volado.

A mi tutor Yidian por presionarme, por confiar en mí, por ser el amigo que es, por enseñarme mucho más allá de la tesis.

Agradezco a la FEU, que me enseñó que no hay tareas imposibles y que puedes estar en dos lugares al mismo tiempo y que no hay nada mejor que hacer las cosas sin esperar nada a cambio. Al piquete de la FEU, a los que compartieron conmigo noches, madrugadas y días completos de carreras y locuras. A los viejos que ya algunos no están por aquí pero que nos dejaron su huella: Jose Augusto, Carlo Carlo, Abel, Olia, Carlos y Jorge Arce, Nilo, a los

contemporáneos que venimos juntos dando tropiezos desde hace mucho: Victor, José Joel, Lester, a los nuevos, los que están hoy, a todos gracias por lo que he me han enseñado.

Agradezco a la familia del teatro, que tantos buenos momentos pasamos, tantos ensayos, tantas risas, tanto esfuerzo compartido, en especial a Cesar que me enseñó a amar en serio el teatro.

Agradezco a todos aquellos que compartieron conmigo en el aula, esos que tantas veces me salvaron para las pruebas y seminarios. A las muchachitas del apartamento 92108, en especial Liuba, que se convirtió en mi consuelo y mi mano derecha.

A todos los que de alguna manera colaboraron en esta tesis o me ayudaron en estos 5 años,

A la UCI por forjarme,

A todos muchas gracias.

Dedicatoria:

A mi abuela Cuchi por ser siempre la inspiración de mis triunfos y acompañarme en cada paso de mi vida, mis mayores logros siempre estarán dedicados a ella.

A mi mamá por su comprensión y apoyo incondicional.

A mi papá por convertirme en la persona que soy hoy.

A mi hermano Nelson por ser mi guía y mostrarme el camino correcto siempre.

A mi hermano Noel David por inspirarme y ser de mis más grandes orgullos.

Declaración Jurada de Autoría

Declaro ser autora de la presente tesis y cedo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____

Joyce Suárez Fabre

_____ _____

Ing. Yidian Y. Castellanos Sabarí Ing. Omar Martínez Díaz

_____

Ing. David Tavares Cueva

Datos de Contacto

Ing. Yidian Yosbel Castellanos Sabarí

Universidad de Ciencias Informáticas, Ciudad de la Habana, Cuba.

Email: yycastellanos@uci.cu

Ing. Omar Martínez Correa

Universidad de Ciencias Informáticas, Ciudad de la Habana, Cuba.

Email: omarmd@uci.cu

Ing. David Tavares Cuevas

Universidad de Ciencias Informáticas, Ciudad de la Habana, Cuba.

Email: dtavares@uci.cu

Resumen:

El trabajo consiste en el desarrollo de un módulo de seguridad para SIPP v2.0 que elimine los conflictos de acceso a la información y permita la gestión de usuarios y roles por funcionalidades garantizando así el acceso seguro a la información almacenada en la aplicación.

En el presente documento se exponen los resultados de la investigación realizada y se presenta la fundamentación teórica que sustenta la realización de este trabajo. Se hace un estudio del estado del arte de los procesos de administración de usuarios y roles por funcionalidades y se analizan los permisos a funcionalidades en SIPP.

El módulo de seguridad brinda la posibilidad de gestionar los usuarios y roles así como los permisos de acceso a las distintas funcionalidades otorgados a cada rol. Permite además la autenticación del usuario en el sistema y el acceso del mismo a la información según el rol asignado. La utilización de este módulo constituye una mejora considerable de la seguridad de la información y dinámica de actualización del Sistema de Manejo Integral de Perforación de Pozos.

Índice de contenidos

Índice de tablas:

Índice de figuras:

Introducción:

El desarrollo acelerado de las Tecnologías de la Información y las Comunicaciones (TIC) y las ventajas tanto de seguridad como de organización y gestión que ellas poseen impulsan el incremento de su utilización. Las TIC posibilitan el fácil acceso a una inmensa fuente de información, pueden procesar rápidamente y de forma fiable todo tipo de datos, constituyen canales de información inmediata, poseen una amplia capacidad de almacenamiento y automatización de trabajos así como gran interactividad y digitalización de toda la información.

Cuba es consciente de la necesidad de alcanzar un mayor desarrollo tecnológico e informático. El 12 de enero de 2000 se crea el Ministerio de la Informática y las Comunicaciones, hoy Ministerio de las Comunicaciones. Este ministerio se encarga de fomentar el uso de las TIC en función del desarrollo de la economía nacional, la sociedad y al servicio del ciudadano. Como parte de las acciones para lograrlo se crea en el año 2002 la Universidad de las Ciencias Informáticas (UCI) que tiene como misión: formar profesionales comprometidos con su Patria y altamente calificados en la rama de la Informática, producir aplicaciones y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación y servir de soporte a la industria cubana de la informática.(1)

En la Universidad de las Ciencias Informáticas se encuentra el Centro de Desarrollo de Informática Industrial (CEDIN). Este centro cuenta con varios proyectos encaminados a la informatización de la industria petrolera. Uno de estos proyectos es el Sistema de Manejo Integral de Perforación de Pozos (SIPP). El SIPP es el encargado de controlar una serie de datos representativos relacionados con la perforación de pozos realizados por la Unión Cuba-Petróleo (CUPET), que es la unión de empresas responsable del desarrollo y mantenimiento de la industria del petróleo en Cuba, en conjunto con otras compañías extranjeras que operan y prestan servicios a esta importante y vital rama de la economía. Está estructurado modularmente y responde al producto MaximusDrillPro; entre sus módulos se encuentra el módulo de seguridad, encargado de delimitar el acceso a la información orientado a los niveles de permisos definidos en el negocio así como de monitorear la actividad de los usuarios.En el módulo de seguridad de SIPP el registro de los usuarios es realizado por el Administrador de Sistema y no por los usuarios como convencionalmente se realiza. Las contraseñas viajan y se almacenan encriptados.

El módulo de seguridad de SIPP v 1.0 cuenta con un área de seguridad y un área de administración. En el área de seguridad se encuentran las funcionalidades Autenticar usuario (encargada de proporcionar el acceso inicial al sistema solo a aquellos usuarios registrados) y Cambiar contraseña (encargada de dar opción para que los usuarios puedan cambiar su contraseña de acceso a la aplicación). El área de administración cuenta con las funcionalidades Gestionar usuario (manipula los datos de los usuarios autorizados a trabajar con el sistema y la información que le compete a cada uno), Gestionar rol (manipula los

datos de los usuarios autorizados, les otorga un papel dentro de la aplicación y les da acceso a un entorno determinado), Gestionar permiso (gestiona los permisos de cada usuario y rol, que no es más que delimitar el horizonte de trabajo del especialista solamente a los lugares donde su responsabilidad lo requiera) y Guardar registros de seguridad (provee una vigilancia para poder tener elementos en caso de que sea burlada la seguridad, trabaja con los registros que quedan de las acciones de lectura y escritura, ya sea para consultarlos o guardarlos).

El negocio de SIPP, debido a la dinámica comercial y a las necesidades que determina la perforación de los pozos, cambia con mucha frecuencia a la hora de decidir qué usuario y con qué rol accederá a las distintas funcionalidades, provocando que existan cambios en la aplicación que requieren de tiempo y recursos para darle solución. Cuando en el módulo de seguridad se crea un rol, este viene con permisos definidos en un paquete de funcionalidades de forma general. Esto genera conflictos de acceso a la información ya que dicho rol no necesita acceder al paquete completo y además en caso de surgir nuevos roles en el negocio no se pueden agregar a la aplicación. Es por eso que existe la necesidad de que el SIPP cuente con un módulo de seguridad donde el administrador del sitio pueda gestionar los permisos de los roles y que no permita las incursiones no autorizadas.

De la problemática anterior se deriva la siguiente interrogante como **problema científico**: ¿Cómo garantizar la seguridad del sistema SIPP?

Una vez determinado el problema científico queda definido como **objetivo general** de la investigación: Desarrollar un módulo de seguridad para SIPP v2.0 que elimine los conflictos de acceso a la información. Los procesos de administración de usuarios y roles por funcionalidades constituyen el **objeto de estudio** enmarcando el **campo de acción** en la administración de permisos a funcionalidades en SIPP. En la presente investigación se defiende la idea de que con el desarrollo del módulo de seguridad de SIPP v2.0 se logrará administrar los usuarios, roles y permisos por funcionalidades del sistema.

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

1. Descripción de las soluciones informáticas existentes a nivel nacional e internacional para conformar el estado del arte de la investigación.

2. Caracterización de los distintos tipos de herramientas para conformar el perfil tecnológico del módulo a implementar.

3. Identificación y caracterización de las metodologías utilizadas en el desarrollo de módulos para escoger la metodología a utilizar en el desarrollo del componente.

5. Modelación del módulo para la administración de la seguridad en el sistema SIPP.

6. Implementación del módulo.

7. Validación de la solución propuesta mediante pruebas de caja negra y caja blanca e integración con los restantes módulos.

Para alcanzar los resultados esperados en esta investigación se aplicaron **métodos de investigación científica**. Estos métodos proporcionan una serie de pasos sistemáticos y constituyen instrumentos que conllevan a un conocimiento científico.

Del nivel teórico, ha sido seleccionado el método Histórico-Lógico con el fin de analizar la evolución histórica de los fenómenos y la proyección lógica de su comportamiento futuro. Esto posibilita conocer el estado actual de los procesos de seguridad en sistemas web tanto nacionales como internacionales y conocer algunos sistemas existentes que puedan servir de ayuda al desarrollo del módulo de seguridad que se propone. Permite además observar en el tiempo la evolución del proyecto SIPP desde su primera versión.

Otro método seleccionado dentro de este nivel es el Analítico-Sintético. Con el análisis se divide el fenómeno en sus múltiples relaciones y componentes para facilitar su estudio. En el proceso de síntesis se establece mentalmente la unión entre las partes previamente analizadas y de esta forma se descubren sus características generales y las relaciones esenciales entre ellas. Este método es utilizado en el transcurso de la investigación en las distintas fuentes citadas.

Dentro de los métodos empíricos se ha seleccionado como técnica de recopilación de información la entrevista. Con la confección previa de un guión de preguntas, enmarcadas en el problema objeto de estudio, se logró una mayor comprensión de las necesidades y objetivos del sistema a desarrollar.

El presente trabajo está distribuido en tres capítulos:

Capítulo 1: En este capítulo se explican conceptos fundamentales que ayudan a comprender mejor el trabajo. Se sistematiza la teoría y metodología a utilizar y se hace un estudio de los procesos de gestión de seguridad en aplicaciones web, así como, de las características de las principales herramientas y tecnologías a utilizar en el desarrollo del módulo de seguridad.

Capítulo 2: En este capítulo se describen los artefactos que se generan durante el diseño del módulo. Se realiza además la definición de los requerimientos del sistema y se generan las Historias de Usuario detallándose cada una de las tareas de ingeniería para su solución.

Capítulo 3: Se hace una valoración a partir del análisis e interpretación de los datos y resultados alcanzados del sistema así como la validación del mismo mediante pruebas de aceptación.

Capítulo 1: Fundamentación Teórica.

1.1 Introducción

En este capítulo se analiza la situación actual en los procesos de seguridad de las aplicaciones web. Se detallan las herramientas y tecnologías utilizadas para dar solución al problema planteado y se establece una comparación cualitativa entre las diferentes opciones existentes y las tendencias actuales en cuanto a seguridad a nivel mundial.

1.2 Seguridad en aplicaciones web

Cualquier organización que expone sus servicios informáticos a redes de acceso tendrá que realizar un esfuerzo significativo para asegurar que la información y recursos están protegidos.

En una aplicación web, se puede dividir la seguridad en:

- **Disponibilidad**: Propiedad o característica de los activos, consistente en que las entidades o procesos autorizados tienen acceso a los mismos cuando lo requieren.
- **Autenticidad**: Propiedad o característica consistente en que una entidad es quien dice ser o bien que garantiza la fuente de la que proceden los datos
- **Integridad**: Propiedad o característica consistente en que el activo de información no ha sido alterado de manera no autorizada.
- **Confidencialidad**: Propiedad o característica consistente en que la información ni se pone a disposición, ni se revela a individuos, entidades o procesos no autorizados.
- **Trazabilidad**: Propiedad o característica consistente en que las actuaciones de una entidad pueden ser imputadas exclusivamente a dicha entidad.(2)

El análisis de las vulnerabilidades potenciales es un objetivo clave para el incremento de la seguridad en las aplicaciones web. Muchas veces, condicionados por la premura de tiempo para el desarrollo de los sistemas y enfrascados en resolver las exigencias del cliente, los desarrolladores descuidan los aspectos básicos de seguridad en la implementación de la aplicación.

El proyecto de las diez mayores vulnerabilidades de OWASP[1] es una lista de vulnerabilidades que requieren inmediata solución en un sistema web. El objetivo del proyecto es crear conciencia sobre la seguridad en aplicaciones web mediante la identificación de algunos de los riesgos más críticos que enfrentan las organizaciones. A continuación se muestra la lista definida en el 2010:

[1] Proyecto abierto de seguridad en aplicaciones Web (OWASP por sus siglas en inglés).

Fallos de Inyección:

Los fallos de inyección, tales como *SQL y LDAP*(Protocolo Ligero de Acceso a Directorios), ocurren cuando datos no confiables son enviados a un intérprete como parte de un comando o consulta. Los datos hostiles del atacante pueden engañar al intérprete y ejecutar comandos no intencionados o acceder a datos sin autorización. (2)

Secuencia de Comandos en Sitios Cruzados (XSS):

Los fallos XSS ocurren cada vez que una aplicación toma datos no confiables y los envía al navegador web sin una validación y codificación apropiada. XSS permite a los atacantes ejecutar secuencia de comandos en el navegador de la víctima los cuales pueden secuestrar las sesiones de usuario, destruir los sitios web, o dirigir al usuario hacia un sitio malicioso. (2)

Perdida de Autenticación y Gestión de Sesiones:

Las credenciales de cuentas y los testigos de sesión (*session token*) frecuentemente no son protegidos adecuadamente. Los atacantes obtienen contraseñas, claves, o testigos de sesión para obtener identidades de otros usuarios. (2)

Referencia Insegura y Directa a Objetos:

Una referencia directa a objetos ("*direct object reference*") ocurre cuando un programador expone una referencia hacia un objeto interno de la aplicación, tales como un fichero, directorio, registro de base de datos, o una clave tal como una URL o un parámetro de formulario Web. Un atacante podría manipular este tipo de referencias en la aplicación para acceder a otros objetos sin autorización. (2)

Falsificación de Petición en Sitios Cruzados (CSRF):

Un ataque CSRF fuerza al navegador validado de una víctima a enviar una petición a una aplicación Web vulnerable, la cual entonces realiza la acción elegida por el atacante a través de la víctima. CSRF puede ser tan poderosa como la aplicación que está siendo atacada. (2)

Configuración defectuosa de seguridad:

Una buena seguridad requiere tener definida e implementada una configuración segura para la aplicación, marcos de trabajo, servidor de aplicación, servidor web, base de datos, y plataforma. Todas estas configuraciones deben ser definidas, implementadas, y mantenidas ya que por lo general no son seguras por defecto. Esto incluye mantener todo el software actualizado, incluidas las librerías de código utilizadas por la aplicación. (2)

Almacenamiento Criptográfico Inseguro:

Las aplicaciones Web raramente utilizan adecuadamente funciones criptográficas para proteger datos y credenciales. Los atacantes usan datos débilmente protegidos para llevar a cabo robos de identidad y otros crímenes, tales como fraude de tarjetas de crédito. (2)

Fallos de restricción de acceso a URL:

Frecuentemente, una aplicación solo protege funcionalidades delicadas previniendo la visualización de enlaces o URLs a usuarios no autorizados. Los atacantes utilizan esta debilidad para acceder y llevar a cabo operaciones no autorizadas accediendo a esas URLs directamente. (2)

Comunicaciones Inseguras:

Las aplicaciones frecuentemente fallan al autenticar, cifrar, y proteger la confidencialidad e integridad de tráfico de red sensible. Cuando esto ocurre, es debido a la utilización de algoritmos débiles, certificados expirados, inválidos, o sencillamente no utilizados correctamente.(2)

Redirecciones y Reenvíos no validados:

Las aplicaciones web frecuentemente redirigen y reenvían a los usuarios hacia otras páginas o sitios web, y utilizan datos no confiables para determinar la página de destino. Sin una validación apropiada, los atacantes pueden redirigir a las víctimas hacia otros sitios, o utilizar reenvíos para acceder páginas no autorizadas.(2)

En la implementación del módulo de seguridad de SIPP v2.0 se tendrán en cuenta estas vulnerabilidades y las implicaciones que provocan. Durante el desarrollo se irán tratando cada una con el objetivo de obtener un módulo de seguridad libre de peligros potenciales.

1.3 Procesos de administración de usuarios y roles por funcionalidades

Una de las formas de garantizar la seguridad de acceso en un software es utilizando la administración de usuarios y roles por funcionalidades. Un requisito típico de los sitios web es permitir que solo algunos usuarios (usuarios autenticados) puedan acceder a ciertas páginas. Para lograr esto se pueden usar roles que administren el acceso de los usuarios al sistema. Se entiende por usuario una persona o sistema que utiliza una computadora o servicio informático y los roles constituyen un conjunto de derechos y funciones que serán otorgados a los usuarios con el fin de agruparlos según los permisos que posean. Para utilizarlos, después de crearlos, se asignan usuarios individuales a uno o varios roles y se conceden permisos de acceso al rol. A cada usuario del rol se le conceden los permisos definidos para el mismo.(3)De esta forma se garantiza la disponibilidad de la información a los usuarios con permisos suficientes, la autenticación para acceder al sistema, la integridad

de la información almacenada y la confidencialidad de aquella información que no debe ser compartida con todos los usuarios.

1.3.1 Sistemas que utilizan administración de usuarios y roles por funcionalidades

La administración de usuarios y roles por funcionalidades es una técnica muy utilizada en la actualidad. Los sistemas en los que la seguridad de la información constituye aspecto primordial tienen en cuenta este procedimiento con el fin de garantizar solo el acceso necesario de los usuarios a la información. A continuación se presentan algunos sistemas que utilizan esta técnica para garantizar su seguridad:

DotNetNuke

El software DotNetNuke lo utilizan desarrolladores, diseñadores y personas de negocio para crear rápidamente y mantener fácilmente páginas web de todo tipo. Es la solución de gestión de contenidos web número uno en el ecosistema de Microsoft: más de 700.000 páginas web han sido desplegadas en todo el mundo utilizando estos productos. Ha habido hasta el momento más de 7 millones de descargas del proyecto de código abierto. DotNetNuke Corp. tiene su sede central en San Mateo, (California), y oficinas en Vancouver y Amsterdam.

Una de las opciones más interesantes de DotNetNuke es la gestión de usuarios y roles, por su funcionalidad y su facilidad de uso. Posee fundamentalmente, dos opciones de Administración de usuarios: Roles de Seguridad y Cuentas de Usuario. Los roles de seguridad permiten crear grupos de usuarios con permisos específicos para dicho grupo. Un usuario puede pertenecer a tantos grupos como se considere necesario. Respecto a las cuentas de usuario el sistema permite decidir si cualquiera puede crear una cuenta o si las cuentas se validan por parte de los usuarios administradores. (4)

ArcGIS

Es un completo sistema para trabajar con mapas e información geográfica. Permite a usuarios de cualquier parte recopilar, organizar, administrar, analizar, compartir y distribuir información geográfica. ArcGIS se puede considerar un sistema para hacer que los mapas y la información geográfica estén disponibles en un departamento, en toda una empresa, entre diversas organizaciones y en comunidades de usuarios en Internet. Con el uso de este sistema se puede trabajar con mapas de Sistemas de Información Geográfica (SIG) inteligentes, compilar y administrar información geográfica, resolver problemas con el análisis espacial y compartir información aprovechando el potencial de los mapas.

En el Administrador de ArcGIS se puede visualizar y administrar los roles. El módulo de seguridad contiene una página Roles donde se pueden ver y administrar los permisos de los mismos. Las entidades disponibles en esta página varían en función de su identidad a almacenar. Si está utilizando el almacenamiento integrado de ArcGIS, puede agregar, modificar y eliminar roles en el Administrador.(5)

1.3.2 Modelo de Control de Acceso Basado en Roles (RBAC)

Está basado en la definición de un conjunto de elementos y de relaciones entre ellos. Describe un grupo de usuarios que pueden estar actuando bajo un conjunto de roles y realizando operaciones en las que utilizan un conjunto de recursos. En una organización, un rol puede ser definido como una función que describe la autoridad y responsabilidad dada a un usuario en un instante determinado. Incluye un conjunto de sesiones donde cada sesión es la relación entre un usuario y un subconjunto de roles que son activados en el momento de establecer dicha sesión. Cada sesión está asociada con un único usuario. Mientras que un usuario puede tener una o más sesiones asociadas. Los permisos disponibles para un usuario son el conjunto de permisos asignados a los roles que están activados en todas las sesiones del usuario, sin tener en cuenta las sesiones establecidas por otros usuarios en el sistema. RBAC añade la posibilidad de modelar una jerarquía de roles de forma que se puedan realizar generalizaciones y especializaciones en los controles de acceso y se facilite la modelización de la seguridad en sistemas complejos.
Otro aspecto importante en el modelo RBAC es la posibilidad de especificar restricciones. Estas restricciones son un fuerte mecanismo para establecer políticas organizacionales de alto nivel. Las restricciones pueden ser de dos tipos: estáticas o dinámicas. Las restricciones estáticas nos permiten solucionar conflictos de intereses sobre relaciones usuario/rol (*Static Separation of Duty*, SSD) y reglas de cardinalidad de roles (*Role Cardinality*), desde una perspectiva de política de seguridad. Por otro lado, las restricciones dinámicas (*Dynamic Separation of Duty*, DSD) al igual que las SSD, limitan los permisos que son disponibles para un usuario. Sin embargo DSD difieren de las SSD por el contexto en el cual estas limitaciones son impuestas. Las DSD limitan la disponibilidad de los permisos aplicando las restricciones sobre los roles que pueden ser activados durante una sesión de usuario.

Principios a tener en cuenta al implementar RBAC:

- Un usuario tiene acceso a los objetos de acuerdo al rol que tenga asignado.
- Los roles son definidos en base a funciones de trabajo.
- Los permisos son definidos en función de la autoridad y responsabilidad que se asume en una función de trabajo.
- Las operaciones sobre un objeto son invocadas de acuerdo a los permisos.

La arquitectura de RBAC se puede representar como se muestra en la Figura 1:

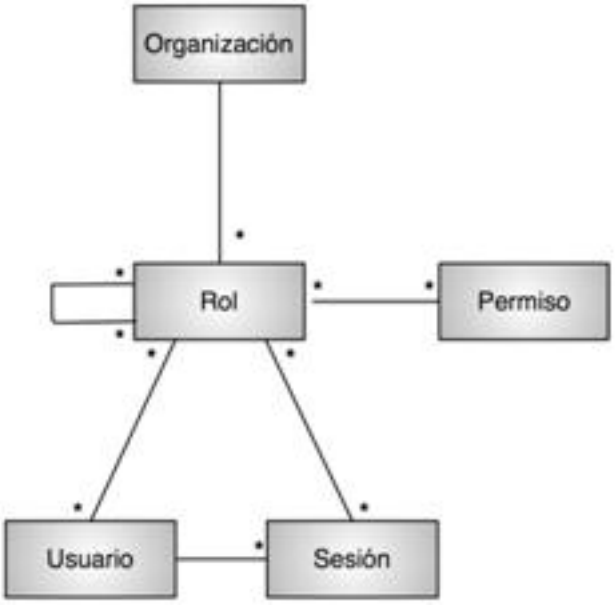

Figura 1 Arquitectura de RBAC

Ventajas:

-Dada la alta integración entre los roles y las responsabilidades de los usuarios, se pueden seguir los principios del mínimo privilegio y de la separación de responsabilidades.
-Es ampliamente aceptado como una práctica recomendada.

El problema fundamental de este modelo es la dificultad en la administración de sistemas con gran cantidad de acciones y permisos. Este control muchas veces se hace a través de ACL (Listas de Control de Acceso), al asociar usuarios con permisos, en caso de existir un usuario con muchos permisos sobre varios recursos implicaría una suma importante de ACL. Este modelo es además una solución todo o nada a un problema que en algunos casos necesita una respuesta más granulada. El modelo RBAC estático por lo general no se puede utilizar para manejar de manera eficiente la explosión de roles, en el que se requieren múltiples funciones estrechamente relacionadas, en otras palabras, provoca aumento exponencial en las reglas de acceso en comparación con el acceso a las variables. Este enfoque a menudo da como resultado en la aplicación más roles que usuarios reales, causando un severo cuello de botella administrativa.

Un ejemplo de la utilización de RBAC puede ser el típico flujo de trabajo (*Workflow*) que se genera en la aprobación de una factura de un proveedor en cualquier sistema de planificación de recursos empresariales o ERP (por sus siglas en inglés). En este caso, dicha factura pasará por distintos estados intermedios en los que distintos usuarios (Jefe de compras, Jefe de un departamento y otros) con rol “Revisor” le darán el visto bueno hasta que finalmente sea aprobada por Dirección. Durante ese flujo dichos usuarios jugarán el Rol “Revisor” de forma consecutiva, pero una vez esté aprobada la factura para su pago, dichos

usuarios adquieren el Rol “Lector”, que a diferencia del Rol anterior, solo permitirá consultar la factura. (6)

1.4 Sistema de Manejo Integral de Perforación de Pozos (SIPP) v1.0

El SIPP automatiza los procesos de gestión y control de información, resultado del proceso de perforación en los pozos de petróleo, obteniendo como resultado reportes de estado de la perforación del pozo, dirección del pozo (inclinometría), reportes de estado de las barrenas, bombas, herramientas, costos, lodo, cementación, encamisado, entre otros. Posee un modelo de base de datos robusto, con alto rendimiento y con altos volúmenes de peticiones, que responde al proceso de manera efectiva, posibilitando manejar todos los datos necesarios para la toma de decisiones, en caso de contingencias (complejidades en la perforación), por los directivos de la empresa a la que pertenezca el pozo. Actualmente este sistema es utilizado por CUPET para la gestión y el control de la información resultado de los procesos realizados en los pozos de petróleo en perforación.

Otras de las fortalezas que posee SIPP son que es un sistema sobre plataforma web integrada con sistemas de escritorio, permite importar y exportar archivos .pdf, .txt, .xls y XML, se desarrolló utilizando solo tecnologías y herramientas bajo licencias libres, permite la visualización de los datos a través de gráficas 2D y funciona bajo estándares y normas utilizados en todo el mundo para la perforación de pozos en tierra.

El sistema trabaja a través de módulos, los cuales responden a las necesidades de cada uno de los usuarios del sistema. El módulo de seguridad es el encargado de garantizar la seguridad del sistema, agrupando las funcionalidades de gestión de usuarios, credenciales (roles) de los mismos, además de solo garantizar el acceso a aquellos usuarios registrados.(7)

El módulo de seguridad de SIPP v1.0 no permite al administrador crear nuevos roles ni gestionar los permisos de los roles existentes. Esto provoca que no se garantice la integridad y confidencialidad de la información ya que una vez definido un rol en los archivos de configuración, solo el equipo de desarrollo puede agregar nuevos roles al sistema. Teniendo esta particularidad, se da el caso en que manipulan los datos usuarios que no poseen la responsabilidad para hacerlo. Igualmente, al no tener la posibilidad de gestionar los permisos de los roles existentes, al ocurrir cambios en el negocio y variar las necesidades del cliente referentes a la gestión de la información surgen dificultades que no pueden ser solventadas con la versión actual. En esta primera versión las interfaces son poco amigables al usuario, y no aportan en algunos casos toda la información requerida para realizar la acción correspondiente.

1.5 Metodologías de desarrollo de software

Para la elaboración de sistemas de cierta envergadura es necesario apoyarse en una metodología de desarrollo que guíe el proceso de creación del software. Las metodologías

imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente.

Existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. En los últimos años ha aumentado la utilización de las metodologías ágiles. En una reunión celebrada en febrero de 2001 en Utah-EEUU, nace el término “ágil” aplicado al desarrollo de software. (8)

1.5.1 SXP

Para el desarrollo del módulo se ha seleccionado SXP[2], una metodología ágil compuesta por las metodologías SCRUM[3] y XP (*eXtreme Programming*, Programación Extrema). A continuación se expondrán algunas características de ambas por separado para lograr un mejor entendimiento de la unión de las mismas.

SCRUM

Es una metodología ágil de administración de proyectos, entre ellos, proyectos de desarrollo de software. Con SCRUM los proyectos progresan a través de una serie de iteraciones que duran entre una o dos semanas y un mes, llamadas "Sprint". No es una metodología de análisis, ni de diseño, como podría ser RUP (Proceso Unificado de Desarrollo), es una metodología de gestión del trabajo.

Esta metodología está basada en el modelo espiral pues consiste en una serie de iteraciones que cumplen los objetivos primariamente definidos, que permite ver los avances que se hacen gradualmente y preparando los elementos que serán utilizados en los siguientes pasos. Esto permite que sean evaluados permanentemente los riesgos como la cancelación de un proyecto por superar el tiempo o los recursos asignados. Este modelo “espiral” se aplica cuando no se conoce el dominio de aplicación del problema, cuando los desarrolladores y usuarios tienen poca experiencia en el tema, hay falta de precisión sobre los problemas a resolver, los requisitos son inestables y hay condicionamientos estrictos en cuanto a tiempo y costo. Sus prácticas implementan la visibilidad, adaptación e inspección. Los roles en esta metodología se dividen en dos categorías: los implicados (usuarios finales, áreas comerciales, áreas contables, marketing, entre otros) y los comprometidos (jefe del equipo, dueño del producto y el equipo) para garantizar que las personas que tienen la responsabilidad poseen además la autoridad necesaria para poder lograr el éxito del proceso y que quienes no la posean no produzcan interferencias innecesarias.(9)

XP

A decir de algunos autores como la ingeniera en Ciencias Informáticas Gladys M. Peñalver Romero en su trabajo: SXP, metodología de desarrollo de software y la Ingeniera Informática María A. Mendoza Sánchez en el artículo Metodologías De Desarrollo De Software, XP es

[2] Metodología de desarrollo de software compuesta por las metodologías SCRUM y XP.
[3] Metodología de desarrollo de software ágil que se basa en la iteración y entrega incrementales de desarrollo de un producto o servicio.

una de las metodologías de desarrollo de software más exitosas en la actualidad. La creciente utilización de esta metodología en el desarrollo de software a nivel mundial apoya esta afirmación. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

La metodología XP se basa en:

Pruebas Unitarias: se realizan a los principales procesos, de tal manera que adelantándose en algo hacia el futuro, se puede hacer pruebas de las fallas que pudieran ocurrir. Es como adelantarse a obtener los posibles errores.

Refabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.

Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa. (8)

Fases de XP:

- Exploración.
- Planificación de la Entrega.
- Iteraciones
- Producción
- Mantenimiento
- Muerte del Proyecto

SXP

Está especialmente indicada para proyectos pequeños, que presentan un rápido cambio de requisitos donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados y una alta flexibilidad. Ofrece una estrategia tecnológica, a partir de la introducción de procedimientos ágiles que permitan actualizar los procesos de software para el mejoramiento de la actividad productiva fomentando el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo, ayudando al líder del proyecto a tener un mejor control del mismo.

Consta de 4 fases principales:

- Planificación-Definición donde se establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto.
- Desarrollo, es donde se realiza la implementación del sistema hasta que esté listo para ser entregado.
- Entrega, puesta en marcha.

- Mantenimiento, donde se realiza el soporte para el cliente.

De cada una de estas fases se realizan numerosas actividades tales como el levantamiento de requisitos, la priorización de la Lista de Reserva del Producto, definición de las Historias de Usuario, diseño, implementación, pruebas, entre otras; de donde se generan artefactos para documentar todo el proceso. Las entregas son frecuentes, y existe una refactorización[4] continua, lo que permite mejorar el diseño cada vez que se le añada una nueva funcionalidad. Con la utilización de SCRUM para la gestión, se logra una planificación y organización inigualable; mientras que XP respalda con sus prácticas todo el proceso de desarrollo, obteniéndose de esta forma un proceso de software completo.(10)

1.6 Lenguaje de programación

Un lenguaje de programación es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Por lo tanto, un lenguaje de programación es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo. (11)

PHP v5.3.1

PHP, por sus siglas en inglés, es un acrónimo recursivo que significa Pre-Procesador de Hipertexto. Fue creado originalmente por Rasmus Lerdorf en 1994. Es un lenguaje de programación utilizado para la creación de sitios web.

PHP es un lenguaje de *script* interpretado en el lado del servidor utilizado para la generación de páginas web dinámicas, embebidas en páginas de Lenguaje de Marcado de Hipertexto (HTML, por sus siglas en inglés) y ejecutadas en el servidor. PHP no necesita ser compilado para ejecutarse. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas. Los archivos cuentan con la extensión php.

Ventajas:

- Muy fácil de aprender.
- Se caracteriza por ser un lenguaje muy rápido.
- Soporta la orientación a objeto.
- Es un lenguaje multiplataforma: Linux, Windows, entre otros.
- Capacidad de conexión con la mayoría de los manejadores de base de datos: entre ellos PostgreSQL, que es el que se utiliza en el desarrollo del módulo.
- Capacidad de expandir su potencial utilizando módulos.
- Su página oficial posee documentación en la cual incluye descripción y ejemplos de cada una de sus funciones.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Incluye gran cantidad de funciones.
- No requiere definición de tipos de variables ni manejo detallado del bajo nivel.

[4] La **refactorización** (del inglés *refactoring*) es una técnica de la ingeniería de software para reestructurar un código fuente, alterando su estructura interna sin cambiar su comportamiento externo.

Desventajas:

- Se necesita instalar un servidor web.
- La legibilidad del código puede verse afectada al mezclar sentencias de HTML y PHP.(12)

Basado en las ventajas antes descritas se decidió seleccionarlo para la implementación y desarrollo del SIPP y de esta forma también para el módulo de seguridad. Las desventajas que presenta este lenguaje pueden ser solventadas de manera fácil, al instalar Apache como servidor web y evitando mezclar las sentencias HTML y PHP.

1.7 Marco de trabajo

Un marco de trabajo (conocido por *framework* por su traducción al inglés) es una plataforma para la implementación de una aplicación. Simplifica el desarrollo mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, proporciona estructura al código fuente, forzando al desarrollador a adicionar código más legible y estético. Por último, facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas.

Ventajas del uso de un *framework*:

- El programador no necesita plantearse una estructura global de la aplicación, sino que el *framework* le proporciona un esqueleto que hay que rellenar.
- Facilita la colaboración (todo lo que sea definir y estandarizar va a ahorrar tiempo y trabajo a los desarrollos colaborativos).
- Es más fácil encontrar herramientas (utilidades, bibliotecas) adaptadas al *framework* concreto para facilitar el desarrollo. (13)

1.7.1 Symfony v1.2.8

Symfony es un *framework* diseñado para la realización de aplicaciones web de forma óptima desde las más sencillas hasta las más robustas, enfocado principalmente en la arquitectura MVC (Modelo, Vista, Controlador). Además cuenta con herramientas y clases que aseguran la rapidez y la disminución del tiempo de desarrollo. Automatiza acciones comunes como realizaciones de CRUD (*Create* - Crear, *Update* - Actualizar, *Delete* - Eliminar) sobre una o varias tablas de la base de datos en dependencia del Modelo de Objeto Relacional (ORM – *Object Relational Model*) utilizado.

Symfony está desarrollado completamente en PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL Server. Se puede utilizar tanto en plataformas *nix (Unix, Linux, entre otras) como en plataformas Windows.

Symfony fue diseñado para ajustarse a los siguientes requisitos:

- Fácil de instalar y configurar.
- Independiente del sistema gestor de bases de datos. Su capa de abstracción y el uso de Propel[5], permiten cambiar con facilidad de SGBD (Sistema Gestor de Base de Datos) en cualquier fase del proyecto.
- Utiliza programación orientada a objetos, de ahí que sea imprescindible PHP 5.
- Sencillo de usar en la mayoría de casos, aunque es preferible para el desarrollo de grandes aplicaciones web que para pequeños proyectos.
- Aunque utiliza MVC, tiene su propia forma de trabajo en este punto, con variantes del MVC clásico como la capa de abstracción de base de datos, el controlador frontal y las acciones.
- Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador sólo debe configurar aquello que no es convencional.
- Sigue la mayoría de las mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales, adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios de *phpDocumentor*[6] y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con las bibliotecas de otros fabricantes (13)

Seguridad en Symfony:

La posibilidad de ejecutar una acción puede ser restringida a usuarios con ciertos privilegios. Las herramientas proporcionadas por Symfony para este propósito permiten la creación de aplicaciones seguras, en las que los usuarios necesitan estar autenticados antes de acceder a alguna característica o a partes de la aplicación. Añadir esta seguridad a una aplicación requiere dos pasos: declarar los requerimientos de seguridad para cada acción y autenticar a los usuarios con privilegios para que puedan acceder estas acciones seguras.

Antes de ser ejecutada, cada acción pasa por un filtro especial que verifica si el usuario actual tiene privilegios de acceder a la acción requerida. En Symfony, los privilegios están compuestos por dos partes:

• Las acciones seguras requieren que los usuarios estén autenticados.

• Las credenciales son privilegios de seguridad agrupados bajo un nombre y que permiten organizar la seguridad en grupos.

Para restringir el acceso a una acción se crea y se edita un archivo de configuración YAML llamado *security.yml* en el directorio *config/* del módulo. En este archivo, se pueden especificar los requerimientos de seguridad que los usuarios deberán satisfacer para cada acción o para todas (*all*) las acciones. (13)

[5] ORM utilizado por Symfony.
[6] Es un generador de documentación de código abierto escrito en PHP.

Producto a que el módulo será desarrollado en PHP, Symfony es una buena opción debido a que por el uso de su ORM, orienta toda la base de datos a objetos, lo cual elimina las inyecciones SQL, soporta las base de datos más usadas hoy en día y en caso de ocurrir algún cambio en el gestor de base de datos esto no afectaría en nada el funcionamiento de la aplicación. Además de ser libre, fue diseñado principalmente para sistemas web de alta complejidad y cuenta también con una gran comunidad de desarrollo.(13)Por todas estas ventajas el SIPP lo utiliza como *framework* de desarrollo.

Existen otros *frameworks* como Zend Framework 2 que es un marco de código abierto para el desarrollo de aplicaciones y servicios web utilizando PHP 5.3. Zend Framework 2 utiliza código orientado a objetos 100% y utiliza la mayor parte de las nuevas características de PHP 5.3. Posee componentes, como *Zend_Auth* que provee la autenticación de usuarios y *Zend_Acl* que proporciona una lista de control de acceso (ACL) a la aplicación para la gestión de privilegio, o sea los roles solicitan acceso a los recursos. Para crear un recurso se utiliza *Zend_Acl_Resource_Interface*, una clase sólo necesita implementar esta interfaz, que consta de un único método, *getResourceId* (), para que *Zend_Acl* reconozca el objeto como un recurso. Al igual que con los recursos, la creación de un rol también es muy simple en este framework. Todos los roles deben implementar *Zend_Acl_Role_Interface*. Esta interfaz se compone de un único método, *getRoleId* (), además, *Zend_Acl_Role* es proporcionado por *Zend_Acl* como una implementación básica del rol para los desarrolladores que puede ser extendida según sea necesario. También existen componentes que implementan bibliotecas de cliente para acceder de forma sencilla a los servicios web más populares.(14)

Por las ventajas que proporciona este *framework* sobre todo para la administración de usuarios y roles sería provechoso estudiar la aplicación de este *framework* en posteriores desarrollos del módulo y del SIPP.

1.7.2 Interfaz de usuario

Existen varios *frameworks* utilizados para facilitar la programación visual de páginas y sistemas. En la presente investigación se tienen en cuenta algunos de los más populares mundialmente con el fin de seleccionar el más efectivo para el módulo a implementar.

Dojo

Es un *framework* que contiene APIs[7] y controles para facilitar el desarrollo de aplicaciones web que utilicen AJAX[8]. Contiene un sistema de empaquetado inteligente, efectos de interfaz

7 Interfaz de programación de aplicaciones (IPA) o API (del inglés *Application Programming Interface*) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

de usuario, abstracción de eventos, almacenamiento de APIs en el cliente, e interacción de APIs con AJAX.

Resuelve asuntos de usabilidad comunes como pueden ser la navegación y detección del navegador, soportar cambios de URL en la barra de URLs para luego regresar a ellas (*bookmarking*), y la habilidad de degradar cuando *AJAX/JavaScript* no es completamente soportado en el cliente. Es conocido como "la navaja suiza del ejército de las bibliotecas *Javascript*". Proporciona una gama amplia de opciones en una sola biblioteca *JavaScript* y es compatible con navegadores antiguos.(15)

jQuery

Es una biblioteca *JavaScript cross-browser*[9] que facilita el recorrido de DOM (Modelo de Objetos del Documento), manejo de eventos, animación, y las interacciones AJAX. jQuery fue creado por John Resig, proporciona abstracciones comunes para el lado del cliente y realiza tareas como manejo de eventos, animación y AJAX . jQuery también proporciona una plataforma para crear plugins que extienden sus capacidades más allá de las previstas por el núcleo.(16)

ExtJS v3.2

Es un framework diseñado para la creación de páginas web dinámicas del lado del cliente basado en lenguaje JavaScript, permite una gran reutilización de componentes, personalización de los mismos y utiliza AJAX, además de lo ventajoso que puede representar la similitud de su aspecto con el de una aplicación de escritorio. Sus características principales son: gran desempeño, componentes de interfaz de usuario personalizables con buen diseño y documentación.

Una de las grandes ventajas de utilizar ExtJS es que nos permite crear aplicaciones complejas utilizando componentes predefinidos así como un manejador de *layouts* similar al que provee Java Swing, gracias a esto facilita una experiencia consistente sobre cualquier navegador, evitando el tedioso problema de validar que el código escrito funcione bien en cada uno. (17)

En la implementación del módulo de seguridad se ha decidido utilizar ExtJS. Este *framework* es orientado a objetos, tiene controles predefinidos y adaptadores que permiten utilizar otros *frameworks* Javascript como por ejemplo, Yui, Prototype, Jquery y otros. Es además muy utilizado a nivel internacional y posee buena documentación y soporte.

1.8 Gestor de base de datos

En la actualidad los SGBD (Sistema de Gestión de Base de Datos) facilitan el uso de técnicas para gestionar convenientemente la información a almacenar o recuperar, según el caso, de forma fácil de interpretar y útil para el usuario, con facilidad y fiabilidad. Es por ello

[8] Es una técnica de desarrollo web para crear aplicaciones interactivas. Intercambio asíncrono entre los datos del cliente y el servidor web.

[9] Las soluciones que se adaptan a todo tipo de navegadores se dice que son *cross-browser*.

que se han convertido en el instrumento o soporte básico más ampliamente usado en la gestión de los sistemas informáticos.

Un sistema gestor de base de datos se define como el conjunto de programas que administran y gestionan la información contenida en una base de datos. Se trata de un conjunto de programas no visibles al usuario final que se encargan de la privacidad, la integridad, la seguridad de los datos y la interacción con el sistema operativo. Proporciona una interfaz entre los datos, los programas que los manejan y los usuarios finales. (18)

1.8.1 PostgreSQL v9.1

PostgreSQL es un servidor de base de datos objeto-relacional libre. Como en muchos otros proyectos *Open Source (código abierto)*, el desarrollo de PostgreSQL no es manejado por una sola compañía, sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales que trabajan en su desarrollo, dicha comunidad es denominada el PGDG (Grupo Global de Desarrollo de PostgreSQL). Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido más tarde en otros sistemas de gestión comerciales. PostgreSQL incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional.

Ventajas:

- Soporte de protocolo de comunicación encriptado por SSL.
- Extensiones para alta disponibilidad, nuevos tipos de índices, datos espaciales, minería de datos.
- Incorpora una estructura de datos de arreglo.
- Incorpora funciones de diversa índole como manejo de fechas y geométricas.
- Permite la declaración de funciones propias, así como la definición de disparadores
- Soporta el uso de índices, reglas y vistas.
- Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.
- Corre en casi todos los sistemas operativos: Linux, Unix, Windows.
- Cuenta con soporte nativo para los lenguajes más populares como son PHP, C, C++, Perl, Python.

Este gestor puede ser utilizado, modificado y distribuido por cualquiera gratuitamente, para cualquier propósito ya sea con fines privados, comerciales o académicos.(19)

Además de las facilidades antes mencionadas se decidió escoger este gestor para el desarrollo del SIPP porque es altamente escalable tanto en la cantidad de datos que puede manipular como en la cantidad de usuarios concurrentes que puede atender, lo cual se

ajusta a las características del sistema. Es por tanto el SGBD utilizado en la implementación del módulo de seguridad.

1.9 Servidor Web

El servidor web es un programa que se aloja en una computadora que escucha peticiones HTTP de *HyperText Transfer Protocol* (Protocolo de Transferencia de Hipertexto) las cuales son atendidas y procesadas por este programa. Atendiendo a las peticiones hechas, gestiona las páginas web o ejecuta códigos en el servidor para dar respuesta, ya sea información o un mensaje detallado del error.

1.9.1 Apache v2.2

Apache es un servidor de páginas web de código abierto, multiplataforma y modular, se desarrolla dentro del proyecto Servidor HTTP de la Fundación de Software Apache. Presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración. Se ha adaptado siempre a una gran variedad de entornos a través de su diseño modular, que permite a los administradores de sitios web elegir qué características van a ser incluidas en el servidor, y seleccionar qué módulos se van a cargar, ya sea al compilar o al ejecutar el servidor.

La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales, y no remotamente. Sin embargo, algunas se pueden accionar remotamente en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache.

Características principales:

- Trabaja sobre múltiples plataformas (Unix, Linux, MacOSX, Win32, OS2).
- Incluye módulos que se cargan de forma dinámica.
- Soporta CGI, Perl, PHP.
- Posee soporte para bases de datos.
- Soporta SSL para transacciones seguras.
- Incluye soporte para host virtuales.
- Soporta HTTP 1.1.
- Es rápido y eficiente.

Ventajas:

- Ayuda en la mejora del posicionamiento.
- Este servidor junto con el módulo *mod_rewrite* puede convertirse en una herramienta muy útil para adicionar páginas con enlaces amigables para los buscadores.
- Es un software libre.
- *Open Source.*
- Modular.

- Extensible.
- Presenta mensajes de error altamente configurables. (20)

Apache es el servidor web definido para el SIPP y por tanto el servidor a utilizar en el desarrollo del módulo.

1.10 Entorno de Desarrollo Integrado

Existen diferentes formas de introducir un programa Java en una computadora. Se puede utilizar un Ambiente de Desarrollo Integrado (IDE, por sus siglas en inglés) o un editor de textos plano. Un IDE es más que una larga pieza de software, que permite introducir, compilar y ejecutar programas. La introducción, compilación y ejecución son parte del desarrollo de un programa y están integradas juntas en un ambiente, de ahí el nombre ambiente de desarrollo integrado. Algunos IDE son gratuitos y otros muy caros. (21)

Netbeans v7.2:

Permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (*manifestfile*) que lo identifica como módulo. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software. Es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo.

El IDE NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Es un producto libre y gratuito sin restricciones de uso. (22)

Debido a las funcionalidades que aporta NetBeans ha sido seleccionado para la implementación del SIPP y por tanto es utilizado también en el desarrollo del módulo de seguridad. Netbeans permite crear aplicaciones Web con PHP 5, posee un potente debugger integrado y además viene con soporte para Symfony. Al tener también soporte para AJAX, es una muy buena opción para desarrollar aplicaciones LAMP o WAMP.

1.11 Modelado de software

El modelado de sistemas de software es una técnica para tratar con la complejidad inherente a estos sistemas. El uso de modelos ayuda al desarrollador a visualizar el sistema a construir, los modelos de un nivel de abstracción mayor pueden utilizarse para la comunicación con el cliente. Además, las herramientas de modelado y las de Ingeniería de Software pueden ayudar a verificar la corrección de modelos que constituyen una fuente de documentación insustituible en el proceso de desarrollo del software.

1.11.1 El Lenguaje Unificado de Modelado

(UML, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (*Object Management Group,* Grupo de Gestión de Objetos).Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un modelo del sistema, incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. Su evolución se ha visto confirmada con el UML 2.0, que incluye 13 tipos de diagramas y diversas funcionalidades que no se veían en la versión anterior, entre ellas, darle respaldo a BPMN (*Business Process Modeling Notation*, Notación para el Modelado de Procesos de Negocio).(23)

1.11.2 Herramienta para el modelado, Visual Paradigm para UML v8.0

Visual Paradigm para UML es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software. Permite modelar todo tipo de diagramas UML, generar código desde diagramas, generar documentación, realizar ingeniería tanto directa como inversa, entre otras funciones. Este software permite crear modelos de gran calidad en poco tiempo y a un menor costo, ya que presenta una licencia gratis para uso no comercial.(24)

Visual Paradigm para UML fue seleccionado como herramienta CASE para el modelado del sistema debido a que a diferencia de otras herramientas de este tipo es multiplataforma y posee una interfaz amigable que permite un desenvolvimiento mayor por parte del equipo de desarrollo, que además ya se ha familiarizado con ella, pues tiene un papel protagónico en el estudio de la Ingeniería de Software como asignatura en la UCI y es además la herramienta utilizada en el modelado del SIPP.

1.12 Conclusiones del capítulo

Teniendo en cuenta el estudio realizado anteriormente sobre las tendencias actuales en cuanto a los procesos de gestión de la seguridad en aplicaciones web, las vulnerabilidades principales y las herramientas y técnicas más utilizadas en el mundo de la Informática para sistemas de este tipo, se decide desarrollar el módulo de seguridad de SIPP empleando SXP como metodología, PHP como lenguaje de programación, Netbeans como IDE de desarrollo, PostgreSQL como gestor de bases de datos y usando los *frameworks* Symfony y ExtJS.

Se determinó utilizar UML como lenguaje de modelado y la herramienta CASE para representar todos los modelos y diagramas que surjan como artefactos del proceso de desarrollo será Visual Paradigm para UML.

Con el estudio del modelo RBAC se han podido identificar los requisitos fundamentales con que debe contar el módulo y cómo debiera ser la implementación del mismo en función de garantizar la seguridad en el acceso a la información. A pesar de las ventajas que aporta

este modelo, en la implementación del módulo de seguridad de SIPP v1.0 se ha decido utilizar las opciones que ofrece el marco de trabajo Symfony para configurar la seguridad, ya que las mismas siguen los principios básicos de RBAC y facilitan el trabajo del desarrollador en cuanto a organización y entendimiento, considerando que además en este caso se cuenta con poca experiencia tanto en el trabajo con Symfony como en el trabajo con el *framework* para la interfaz de usuario Extjs.

Capítulo 2: Descripción y análisis de la solución propuesta

2.1 Introducción

En el presente capítulo se hace una descripción detallada del diseño del sistema. Se muestran las Historias de Usuarios y las tareas de ingeniería asignadas a cada una describiendo en detalle las características del sistema definidas previamente. Además se hace un estudio del patrón arquitectónico a utilizar así como de los patrones de diseño presentes en la implementación del módulo de seguridad.

2.2 Propuesta de sistema

El sistema a desarrollar debe garantizar la seguridad de la aplicación SIPP, así como la administración de usuarios y roles por funcionalidades. Para ello se contará con una interfaz que permitirá realizar estas acciones de manera sencilla e intuitiva. El sistema mostrará una lista con todas las aplicaciones existentes, de cada aplicación los módulos con los que cuenta y finalmente las acciones disponibles de cada módulo. De esta forma el administrador podrá seleccionar qué permisos otorgar a cada rol. La interfaz de modificación y actualización de los parámetros de una conexión será idéntica a la interfaz para insertar con la diferencia que ya aparecerán en los campos los antiguos datos, de forma tal que no haya que entrar todos los datos nuevamente. De esta forma estaríamos realizando el proceso de configuración de una forma mucho más eficiente y más sencilla que como se hace actualmente, respetando los distintos niveles de usuarios existentes.

2.3 Especificación de los requisitos de software

Requisitos funcionales:

Los requisitos funcionales definen el comportamiento interno del sistema, son las funcionalidades específicas que muestran cómo los casos de uso serán llevados a la práctica.(25)

RF1: Gestionar Usuario, este requisito es uno de los encargados de la seguridad del sistema; tiene como funcionalidad el trabajo con los datos de los usuarios autorizados a la manipulación de la información en el sistema.

- **Insertar Usuario:** para ingresar un nuevo usuario al sistema se requiere de un formulario que contenga los campos usuario, nombre, CI, sexo y el rol a desempeñar dentro del sistema, que estará en dependencia del área de trabajo del administrador, pues el administrador del pozo solo puede insertar usuarios en el pozo, no así el de la DIPP (Dirección de Investigación de Perforación de Pozos) que puede insertar tanto de la propia DIPP como de cualquier pozo.
- **Actualizar Usuario:** Para modificar un usuario existente se debe cargar un formulario con los datos registrados del usuario.

- **Eliminar Usuario:** Para eliminar un usuario debe existir un formulario que muestre a todos los existentes en el sistema y que pueda ser seleccionado aquel que desea eliminarse.
- **Listar Usuario:** Se listarán todos los usuarios del área de trabajo en que se encuentre.
- **Buscar Usuario:** Para buscar un usuario se debe introducir el nombre o el usuario y buscar la correspondencia en la lista de usuarios.

RF2: Autenticar Usuario, proporciona el acceso inicial al sistema, solo a aquellos usuarios registrados en el mismo, para lo cual, se necesita un formulario con los campos usuario y contraseña. Para poder acceder al sistema todos los usuarios deben estar autenticados. En el caso de que el usuario intente acceder a alguna información no autorizada, el sistema debe denegar el acceso.

RF3: Cambiar Contraseña, posibilita cambiar la contraseña por parte del mismo usuario, para lo cual, se necesita un formulario con los campos usuario, contraseña anterior, nueva contraseña, confirmar nueva contraseña.

RF4: Gestionar Rol, tiene como funcionalidad la creación de nuevos roles, actualización de los ya creados, así como la eliminación de roles que sean desechados por parte del Administrador del Sistema.

- **Adicionar Rol:** Para crear un nuevo rol en el sistema se requiere de un formulario que contenga los campos rol, nombre del rol y lugar (Pozo, DIPP). Además un formulario para adicionar los permisos a las funcionalidades de ese rol.
- **Actualizar Rol:** para actualizar un rol existente se debe cargar un formulario para seleccionar el rol a modificar así como los permisos de acceso a las diferentes funcionalidades del sistema.
- **Eliminar Rol:** para eliminar un rol, debe existir un formulario que muestre a todos los existentes en el sistema y que pueda ser seleccionado aquel que desea eliminarse. Si hay usuarios asignados a ese rol no se podrá eliminar hasta que no sean eliminados esos usuarios o movidos a otros roles.
- **Listar Rol:** Se listarán todos los roles del área de trabajo en que se encuentre.

Requisitos no funcionales:

Los requisitos no funcionales son aquellos que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos. Son todos aquellos requisitos que ni describen información a guardar ni funciones a realizar.(25)

Requerimientos de apariencia o interfaz externa: La apariencia del sistema propuesto cumplirá con los estándares de diseño establecidos por el proyecto SIPP. Se utilizan colores claros, con un contraste agradable a la vista y un tamaño de fuente apropiado para una

lectura fácil. La interfaz será amigable, fácil de usar, sencilla, asequible para todo tipo de usuarios, de forma tal que los usuarios sin conocimientos básicos de informática puedan interactuar con ella de forma intuitiva.

Requerimientos de usabilidad: Los usuarios con privilegios de administración deberán tener conocimientos avanzados en los temas referentes a las acciones que se realizan en los pozos y los accesos a las mismas. El resto de los usuarios se recomienda impartirle un curso básico sobre el manejo e interacción con el sistema que se propone.

Requerimientos de rendimiento: Para poder lograr un rendimiento óptimo del sistema en cuestión, se recomienda que las PC clientes posean 256 de RAM y un Navegador que soporte *Javascript*. Además debe permitir trabajar conectados concurrentemente 250 usuarios como mínimo y necesita un servidor de base de datos de (4 o superior) gigas de RAM.

Requerimientos de seguridad: La información manejada por el sistema no será pública, de ahí la necesidad de protegerla de usuarios no autorizados, manipulación inadecuada y estados de inconsistencia. Se implementará de forma tal que cada usuario tenga acceso a la información que esté prevista para su rol. En todo momento la información debe estar disponible para los usuarios con acceso autorizado, y debe preservarse la integridad de los datos.

Requerimientos de soporte: De acuerdo con el equipamiento que sea adquirido para el soporte de nuestra aplicación dependerá el mantenimiento del mismo teniendo en cuenta la tecnología y características.

Restricciones de diseño: Como Framework de trabajo contaremos con Symfony y Lenguaje de Programación PHP 5, o sea utilización de tecnología orientada a objetos y uso de la Arquitectura MVC con sus ventajas y desventajas.

Requisitos de Hardware: El hardware de la estación de trabajo del servidor Web donde se ejecutará el sistema deberá ser un Procesador Pentium D 2x2 cache, 3.2 GHz y con memoria RAM entre 4 y 8 GB, el servidor de Base de Datos será Procesador Pentium D 2x2 cache, 3.2 GHz con memoria RAM entre 4 y 8 GB y almacenamiento en discos SCSI[10] en espejo y capacidad igual o superior a los 120 GB cada disco. Además deberá contar con tecnología de respaldo de datos históricos. En la estación de trabajo del cliente se requiere un Procesador Pentium 3 (o superior), con memoria RAM mínima de 256 MB.

Requisitos de software: La PC del cliente debe contar con un Sistema Operativo tanto Windows como Linux (cualquiera de sus distribuciones) y el Navegador Web compatible con HTML 2.0 y CSS (*Cascading Style Sheets*, hojas de estilo en cascada), podrá ser Netscape 3 (o superior), Internet Explorer 4.2 (o superior) y compatibles. En el Servidor Web deberá

[10] SCSI, acrónimo inglés de *Small Computers System Interface* (Interfaz de Sistema para Pequeñas Computadoras), es una interfaz estándar para la transferencia de datos entre distintos dispositivos del bus de la computadora.

instalarse Servidor Web Apache 2.2.X o superior y en el Servidor de Base de Datos se necesita PostgreSQL.

Requisitos de Licencia: Licencia bajo código abierto: El sistema debe cumplir con los lineamientos necesarios para la producción de software libre y la comunidad de desarrollo y soporte, debe ser con Licencia Pública General para Bibliotecas de GNU (LGPL por sus siglas en inglés) no puede incluir elementos de Licencia Pública General de GNU (GPL por sus siglas en inglés) dentro del desarrollo.

2.4 Validación de requisitos

La validación de requisitos tiene por finalidad comprobar que los requisitos de software son consistentes, completos, precisos, realistas, verificables y definen lo que el usuario desea del producto final. La validación se realiza a través de diversos métodos. Uno de estos métodos es el prototipado que consiste en construir una maqueta del futuro sistema de software a partir de los requisitos recogidos en la especificación. Estos prototipos serán evaluados por el cliente y usuarios para comprobar su corrección y completitud.(26)(Ver Anexos 2, 3, 4,5, 6, 7.)

2.5 Historias de usuarios

Las Historias de Usuarios (HU) es uno de los principales artefactos que propone la metodología SXP, son tarjetas en las que se recoge de forma esquemática y en un lenguaje claro qué es lo que se quiere hacer. Son redactadas desde las perspectivas del cliente con ayuda de los programadores. Las Historias de Usuario deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. A continuación cada HU con su respectiva descripción:

<table>
<tr><td colspan="2">Historia de Usuario</td></tr>
<tr><td>Número:1</td><td>Nombre Historia de Usuario: Gestionar usuario</td></tr>
<tr><td colspan="2">Modificación de Historia de Usuario Número: ninguna</td></tr>
<tr><td>Usuario: Administrador</td><td>Iteración Asignada: 1</td></tr>
<tr><td>Prioridad en Negocio: Muy Alta</td><td>Puntos Estimados: 2</td></tr>
<tr><td>Riesgo en Desarrollo: Medio</td><td>Puntos Reales: -</td></tr>
<tr><td colspan="2">Descripción:
• Adicionar Usuario: para ingresar un nuevo usuario al sistema se requiere de un formulario que contenga los campos usuario, nombre, CI, sexo y el rol a desempeñar dentro del sistema.
• Actualizar Usuario: Para modificar un usuario existente se debe cargar un</td></tr>
</table>

formulario con los datos registrados del usuario.

- **Eliminar Usuario:** Para eliminar un usuario debe existir un formulario que muestre a todos los existentes en el sistema y que pueda ser seleccionado aquel que desea eliminarse.
- **Listar Usuario:** Se listarán todos los usuarios del área de trabajo.
- **Buscar Usuario:** Para buscar un usuario se debe introducir el nombre y buscar la correspondencia en la lista de usuarios.

Tabla 1 HU Gestionar usuario

Historia de Usuario	
Número:2	**Nombre Historia de Usuario:** Autenticar usuario
Modificación de Historia de Usuario Número: ninguna	
Usuario: Todos los usuarios que accedan al sistema.	**Iteración Asignada:** 3
Prioridad en Negocio: Media	**Puntos Estimados:** 1
Riesgo en Desarrollo: Medio	**Puntos Reales:** -
Descripción: Proporciona el acceso inicial al sistema, solo a aquellos usuarios registrados en el mismo, para lo cual, se necesita un formulario con los campos usuario y contraseña. Para poder acceder al sistema todos los usuarios deben estar autenticados. En el caso de que el usuario intente acceder a alguna información no autorizada, el sistema debe denegar el acceso.	

Tabla 2 HU: Autenticar usuario

Historia de Usuario	
Número: 3	**Nombre Historia de Usuario:** Gestionar Rol
Modificación de Historia de Usuario Número: ninguna	
Usuario: Administrador	**Iteración Asignada:**2
Prioridad en Negocio: Alta	**Puntos Estimados:** 2
Riesgo en Desarrollo: Alto	**Puntos Reales:** -

Descripción:

- **Adicionar Rol:** Para crear un nuevo rol en el sistema se requiere de un formulario que contenga los campos rol, nombre del rol, área de trabajo (Pozo, DIPP), así como las acciones de cada módulo para otorgarle los permisos.
- **Actualizar Rol:** para actualizar un rol existente se debe cargar un formulario para seleccionar el rol a modificar así como los permisos de acceso a las diferentes funcionalidades de los módulos.
- **Eliminar Rol:** para eliminar un rol, debe existir un formulario que muestre a todos los existentes en el sistema y que pueda ser seleccionado aquel que desea eliminarse.
- **Listar Rol:** Se listarán todos los roles existentes en el sistema.

Tabla 3 HU: Gestionar rol

Historia de Usuario	
Número:4	**Nombre Historia de Usuario:** Cambiar contraseña
Modificación de Historia de Usuario Número: ninguna	
Usuario: Todos los usuarios del sistema.	**Iteración Asignada:**3
Prioridad en Negocio: Media	**Puntos Estimados:** 1
Riesgo en Desarrollo: Medio	**Puntos Reales:** -
Descripción: Posibilita cambiar la contraseña por parte del mismo usuario, para lo cual, se necesita un formulario con los campos usuario, contraseña anterior, nueva contraseña, confirmar nueva contraseña.	

Tabla 4 HU: Cambiar contraseña

2.6 Plan de iteraciones

Una iteración es el acto de repetir un proceso con el objetivo de alcanzar una meta deseada, objetivo o resultado. La planificación de sprint (iteraciones) es una reunión crítica de gran importancia para la metodología. Una planificación mal ejecutada puede arruinar por completo todo el sprint. Este plan tiene como propósito proporcionar al equipo suficiente información facilitando que el equipo trabaje con calma y evitar interrupciones durante unas pocas semanas, además de ofrecerle al cliente suficiente confianza. Una planificación de

sprint produce concretamente: una meta de sprint, una lista de miembros y su nivel de dedicación y una lista de HU incluidas en el sprint.

Iteración 1

Es el sprint que permite tener la primera versión de prueba, la cual será mostrada al cliente con el objetivo de obtener una retroalimentación para el grupo de trabajo. Será implementada la HU de alta prioridad que constituye la base del módulo brindándole al sistema las funcionalidades básicas.

Iteración 2

En este sprint será analizada otra HU de alta prioridad que no fue analizada en la primera iteración. Con la culminación de la misma se tendrá implementada otra de las peticiones del cliente descritas en las HU.

Iteración 3

En este sprint serán implementadas las funcionalidades de baja prioridad. Estas funciones tienen el propósito de brindar al cliente comodidad en la gestión de otras tareas asociadas a las de alta prioridad como la autenticación y el cambio de contraseña de los usuarios.

2.6.1 Plan de duración de las iteraciones

Para una mayor organización del trabajo y como parte del ciclo de vida de un proyecto utilizando la metodología SXP, se crea el plan de duración de cada una de las iteraciones, en este caso se hace para el único equipo de desarrollo con el cual se cuenta. Este plan tiene como finalidad reflejar la duración de cada iteración, así como el orden en que serán implementadas las HU en cada una de las mismas, lo que ayuda a obtener una idea aproximada del tiempo que durará la confección del sistema en su totalidad. La tabla 5 muestra el Plan de duración de las iteraciones en la implementación del módulo:

Iteraciones	HU a implementar	Duración total de las Iteraciones
Iteración 1	HU1: Gestionar usuario	3 semanas
Iteración 2	HU3: Gestionar Rol	3 semanas
Iteración 3	HU2:Autenticar Usuario HU4:Cambiar Contraseña	2 semanas

Tabla 5 Plan de duración de las iteraciones

2.7 Tareas de ingeniería

Para dar solución a las HU es necesario realizar un grupo de acciones llamadas tareas de ingeniería. Estas son descritas una vez terminada toda la documentación necesaria para generar el código, en fichas como las HU, pero en el lenguaje de los programadores y permiten una mejor organización del proceso de implementación del módulo además de posibilitar que sea conocido el grado de complejidad de cada HU teniendo en cuenta la

cantidad de tareas asociadas a ella. Se efectúan al final de cada iteración. A continuación se muestran las tareas de ingeniería definidas por cada HU:

Tareas asignadas a la iteración 1:

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario: HU 1**
Nombre Tarea: Desarrollar formulario para adicionar usuario.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.70
Fecha Inicio: 28/03/2013	**Fecha Fin:2/04/2013**
Programador Responsable: Joyce Suárez Fabre	
Descripción: Se crea el código que permite crear el formulario para la inserción del usuario en el sistema. El formulario que se crea contiene una serie de datos.	

Tabla 6 Desarrollar formulario para adicionar usuario

Tarea de Ingeniería	
Número Tarea: *2*	**Número Historia de Usuario: HU 1**
Nombre Tarea: Desarrollar formulario para actualizar datos de usuario.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.70
Fecha Inicio: 3/04/2013	**Fecha Fin:7/04/2013**
Programador Responsable: Joyce Suárez Fabre	
Descripción: Es desarrollado el formulario para modificar los datos de los usuarios que se encuentran en el Sistema.	

Tabla 7 Desarrollar formulario para actualizar datos de usuario

Tarea de Ingeniería	
Número Tarea: *3*	**Número Historia de Usuario: HU 1**
Nombre Tarea: Implementar la funcionalidad de eliminar usuarios.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.42
Fecha Inicio: 8/04/2013	**Fecha Fin:10/04/2013**
Programador Responsable: Joyce Suárez Fabre	
Descripción: Se crea el código que permite eliminar un usuario en el sistema.	

Tabla 8 Implementar la funcionalidad para eliminar usuarios

Tarea de Ingeniería	
Número Tarea: *5*	**Número Historia de Usuario: HU 1**
Nombre Tarea: Implementar la funcionalidad de listar usuarios.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.56
Fecha Inicio: 11/04/2013	**Fecha Fin:14/04/2013**
Programador Responsable: Joyce Suárez Fabre	
Descripción: Se crea el código que permite listar un usuario en el sistema.	

Tabla 9 Implementar la funcionalidad de listar usuarios

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario: HU 1**
Nombre Tarea: Implementar la funcionalidad de buscar usuarios.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.70
Fecha Inicio: 14/04/2013	**Fecha Fin: 18/04/2013**
Programador Responsable: Joyce Suárez Fabre	
Descripción:	
Se crea el código que permite buscar un usuario en el sistema.	

Tabla 10 Implementar la funcionalidad de buscar usuarios

Tareas asignadas a la iteración 2:

Tarea de Ingeniería	
Número Tarea: *6*	**Número Historia de Usuario: HU 3**
Nombre Tarea: Desarrollar el formulario para adicionar un nuevo rol	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.70
Fecha Inicio: 19/04/2013	**Fecha Fin:23/04/2013**
Programador Responsable: Joyce Suárez Fabre	
Descripción: Se crea el código que permite crear el formulario para la inserción del rol en el sistema. El formulario que se crea contiene una serie de datos.	

Tabla 11 Desarrollar formulario para adicionar rol

Tarea de Ingeniería	
Número Tarea: *7*	**Número Historia de Usuario: HU 3**

Nombre Tarea: Implementar la funcionalidad para actualizar los datos de un rol.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.70
Fecha Inicio: 24/04/2013	**Fecha Fin:27/04/2013**
Programador Responsable: Joyce Suárez Fabre	
Descripción: Se crea el código que permite actualizar los datos de un rol en el sistema.	

Tabla 12 Implementar la funcionalidad para actualizar datos del rol

Tarea de Ingeniería	
Número Tarea: 8	**Número Historia de Usuario: HU 3**
Nombre Tarea: Implementar la funcionalidad para eliminar un rol.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.56
Fecha Inicio: 28/04/2013	**Fecha Fin:1/05/2013**
Programador Responsable: Joyce Suárez Fabre	
Descripción: Se crea el código que permite eliminar un rol en el sistema.	

Tabla 13 Implementa la funcionalidad para eliminar rol

Tarea de Ingeniería	
Número Tarea: 10	**Número Historia de Usuario: HU 3**
Nombre Tarea: Implementar la funcionalidad para listar los roles.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.56
Fecha Inicio: 2/05/2013	**Fecha Fin:9/05/2013**
Programador Responsable: Joyce Suárez Fabre	
Descripción: Se crea el código que permite listar los roles del sistema.	

Tabla 14 Implementar la funcionalidad para listar los roles

Tareas asignadas a la iteración 3:

Tarea de Ingeniería	
Número Tarea: 11	**Número Historia de Usuario: HU 2**
Nombre Tarea: Implementar la funcionalidad para autenticar los usuarios.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 10/05/2013	**Fecha Fin: 16/05/2013**
Programador Responsable: Joyce Suárez Fabre	
Descripción: Se crea el código que permite autenticar los usuarios.	

Tabla 15 Implementar la funcionalidad para autenticar los usuarios

Tarea de Ingeniería

Número Tarea:12	**Número Historia de Usuario: HU 4**
Nombre Tarea: Desarrollar el formulario para cambiar la contraseña.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 17/05/2013	**Fecha Fin: 23/05/2013**
Programador Responsable: Joyce Suárez Fabre	
Descripción: Se crea el código que permite desarrollar el formulario para cambiar la contraseña de un usuario determinado.	

Tabla 16 Desarrollar formulario para cambiar contraseña

Nota: Un día se tomó como 0.14 unidades

2.8 Diseño

Este momento del proceso de desarrollo de software se propone comprender perfectamente los requisitos del software y transformarlos a un diseño que indique cómo debe ser implementado el sistema. El diseño tiene el propósito de formular los modelos que se centran en los requisitos no funcionales y en el dominio de la solución y prepara el ambiente para la implementación y la prueba del sistema.

2.8.1 Modelo de datos

El modelo de datos es el conjunto de reglas y conceptos que permiten describir y manipular los datos que se desean almacenar en la base de datos. Es una colección de conceptos bien definidos matemáticamente que ayudan a expresar las propiedades estáticas y dinámicas de una aplicación con un uso de datos intensivo. La base de datos de la investigación contiene 14 tablas en las que son recogidos los módulos creados, los materiales para la creación de los mismos, las asignaciones realizadas así como las informaciones publicadas. (Ver Anexo 1).

2.8.1.1 Descripción de las tablas de la Base de Datos

A continuación se muestran las tablas de la base de datos del SIPP que se utilizan en la implementación del módulo de seguridad del sistema y la descripción de los datos almacenados por las mismas:

Nombre entidad:	usuario		**Nombre de la BD:**	sipp
Descripción de la entidad:	En esta entidad se almacenan los datos de los usuarios del sistema.			
Atributo / Campo	**Tipo**	**Nulo**	**Comentarios**	
id	integer	No	El id es el identificador de cada usuario.	
nombre	Character(50)	No	Guarda el nombre de cada usuario.	

password	Character(50)	No	Guarda la contraseña de cada usuario.
usuario	Character(20)	No	Guarda el usuario de cada persona en el sistema.
sexo	Character(1)	No	Guarda el sexo de cada usuario.
id_pozo	Integer	Sí	Guarda el id del pozo al que está asignado cada usuario.
numero_identidad	Character(11)	No	Guarda el número del carnet de identidad de cada usuario.

Tabla 17 Entidad: Usuarios del sistema

Nombre entidad:	rol		**Nombre de la BD:**	sipp
Descripción de la entidad:	En esta entidad se almacenan los datos de los roles del sistema.			
Atributo / Campo	**Tipo**	**Nulo**	**Comentarios**	
id	integer	No	El id es el identificador de cada rol.	
rol	Character(25)	No	Guarda el rol.	
nombre_rol	Character(30)	No	Guarda el nombre completo del rol.	
lugar	Character(30)	No	Guarda el lugar a donde está asignado el rol.	

Tabla 18 Entidad: Roles del sistema

Nombre entidad:	usuario_rol		**Nombre de la BD:**	sipp
Descripción de la entidad:	En esta entidad se almacenan la relación entre los usuarios con los roles existentes.			
Atributo / Campo	**Tipo**	**Nulo**	**Comentarios**	
id	serial	No	El id es el identificador de cada relación entre usuario y rol.	
id_pozo	integer	No	Guarda el id del pozo al que pertenece el usuario.	
id_usuario	integer	No	Guarda el id del usuario.	
idrol	integer	No	Guarda el id del rol al que está asignado el usuario.	

Tabla 19 Entidad: Relación entre usuarios y roles

Nombre entidad:	pozo		**Nombre de la BD:**	sipp
Descripción de la entidad:	En esta entidad se almacenan los datos de cada pozo existente en el sistema.			
Atributo / Campo	**Tipo**	**Nulo**	**Comentarios**	
id	serial	No	El id es el identificador de cada pozo.	
nombre_pozo	name	No	Guarda el nombre del pozo.	
profundidad	double(10)	No	Guarda la profundidad del pozo.	
duración	integer	No	Guarda la duración de la perforación.	
presupuesto_pozo	double(10)	No	Guarda el presupuesto con que se cuenta en el pozo.	
altura_nivel_mar	double(10)	No	Guarda la altura sobre el nivel del mar del	

			pozo.
coord._x	double(10)	No	Guarda el valor de la coordenada x del pozo.
coord._y	double(10)	No	Guarda el valor de la coordenada y del pozo.
fecha_inicio_estimada	char(1)	No	Guarda la fecha de inicio de perforación estimada.

Tabla 20 Entidad: Datos de pozo

2.8.2 Diagramas de clases del diseño

Los diagramas de clases muestran cómo se lleva a cabo la colaboración entre las clases para dar cumplimiento a un requisito determinado, para la elaboración de estos diagramas se hace uso de estereotipos que ayudan a representar de manera más fácil la función y el carácter de las clases dentro de la realización del requisito. Seguidamente se muestran los diagramas de clases del diseño realizados durante la implementación del módulo de seguridad:

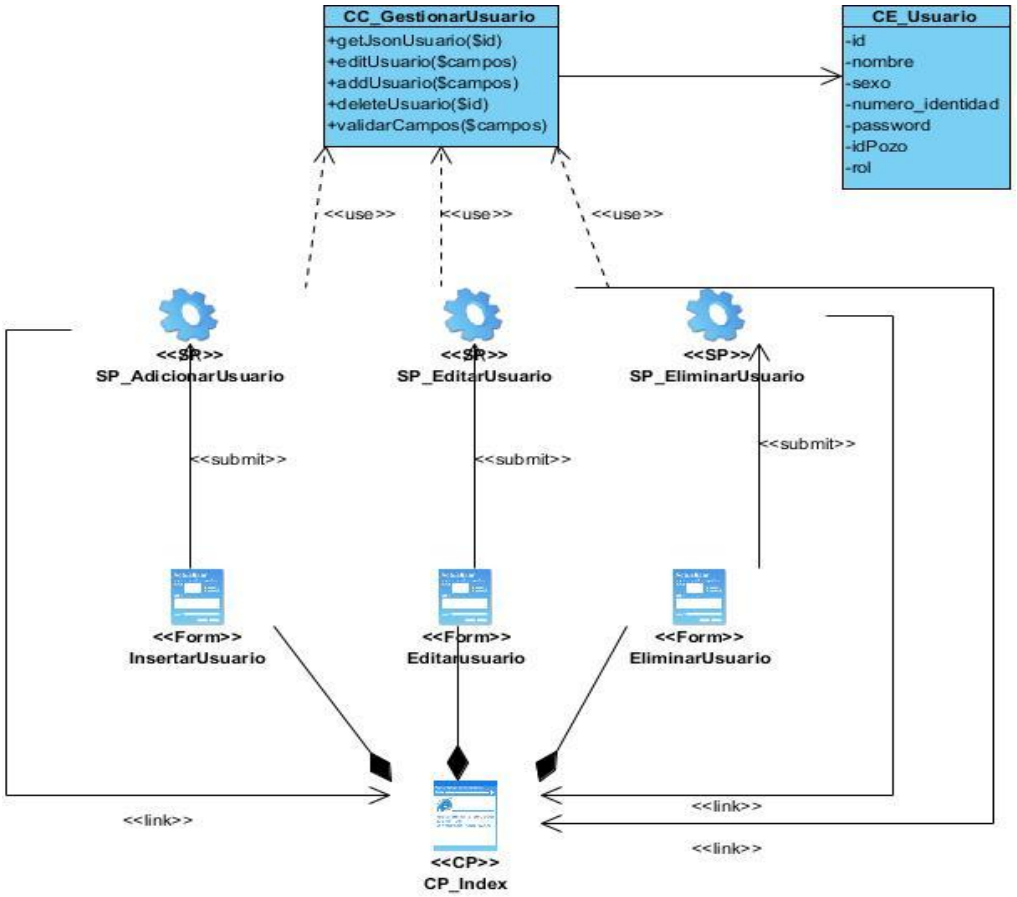

Figura 2 Diagrama de clases del diseño: Gestionar usuario

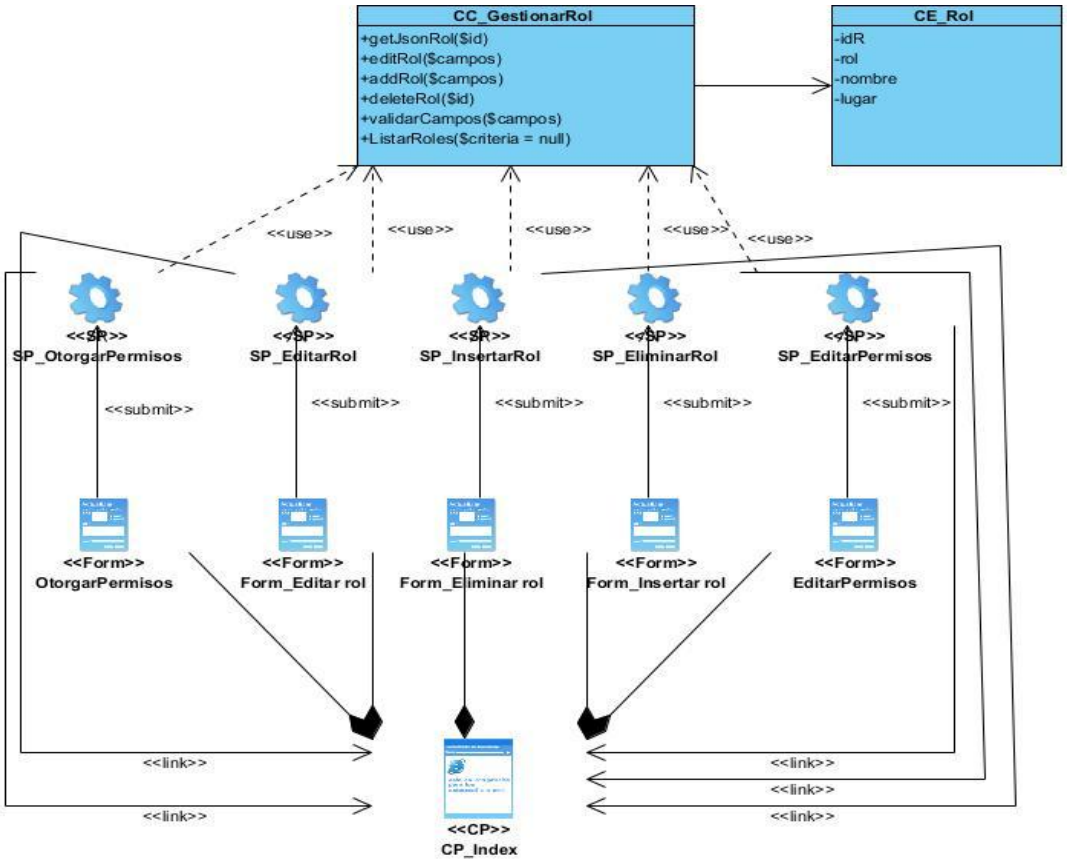

Figura 3 Diagrama de clases del diseño: Gestionar rol

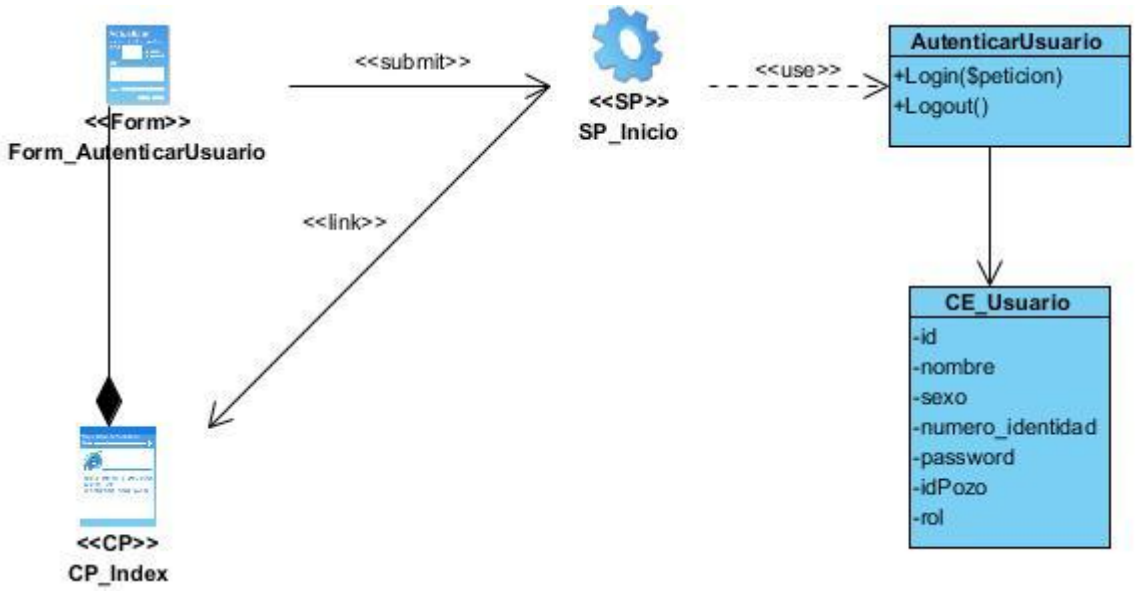

Figura 4 Diagrama de clases del diseño: Autenticar usuario

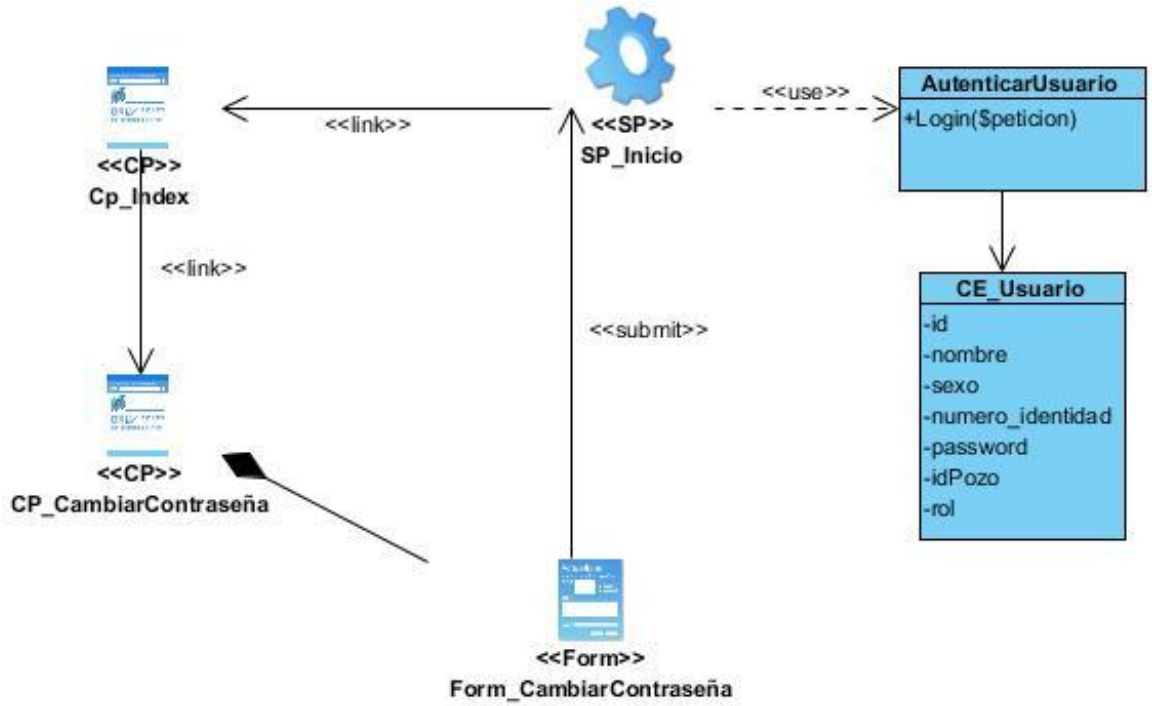

Figura 5 Diagrama de clases del diseño: Cambiar contraseña

2.8.3 Arquitectura del sistema

El proceso en el cual se define una solución para los requisitos técnicos y operacionales de un sistema es conocido como diseño de arquitectura. Donde se definen qué componentes conforman el sistema, cómo se relacionan entre ellos, y cómo mediante su interacción llevan a cabo la funcionalidad especificada.

Symfony está basado en un patrón básico de diseño conocido como arquitectura MVC (Modelo Vista Controlador), el cual como su nombre lo indica está formado por 3 niveles. Este patrón permite la separación de los datos de la aplicación, la interfaz de usuario y la lógica del control entre componentes distintos.

A continuación se hace referencia a las tres capas que componen a la arquitectura definida:

El **Modelo** representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.

La **Vista** transforma el modelo en una página web que permite al usuario interactuar con ella.

El **Controlador** se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

La arquitectura MVC separa la lógica del negocio de la presentación pero en el trabajo con Symfony se separan capas más allá del modelo MVC. En el caso del modelo se puede separar en: acceso a datos y abstracción de datos de modo que si se cambia el sistema gestor de base de datos sería suficiente con actualizar la capa de abstracción de datos para que el modelo volviera a la normalidad.

Las páginas web suelen contener elementos que son comunes y que se muestran de forma idéntica para toda la aplicación, de ahí que la Vista en Symfony se separe en *layouts* y plantillas donde los *layouts* son todos esos elementos comunes y las plantillas son aquellas que se encargan de visualizar las variables definidas en el controlador.(13)

2.8.4 Patrones de diseño

La experiencia de varios años de gestión de software, los errores, ventajas y aprendizaje obtenidos de los mismos ha favorecido la aparición de los patrones. Un patrón es una solución planteada a un problema que ha sido probada y que puede ser reutilizada para evitar que el mismo problema vuelva a ocurrir.(27)

La utilización de patrones en la construcción de software es de gran ventaja para los desarrolladores de la misma ya que permite:

- Ahorrar tiempo en la búsqueda de soluciones que ya existen a problemas conocidos y solucionados anteriormente.
- Permite además lograr un lenguaje común entre los diseñadores a la hora de modelar los proyectos, estandarizando el modo en el que se realizan los diseños
- Facilita el aprendizaje de las nuevas generaciones a partir de conocimientos ya existentes de otros diseñadores.

Entre los principales patrones aplicados para el desarrollo del módulo de seguridad de SIPP se encuentran los patrones de diseño GRASP (Patrones Generales de Software para Asignar Responsabilidades). Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.(28)

Experto*:* Define que la responsabilidad de realizar una labor está en la clase que tiene las condiciones para realizarla (atributos). O sea que una clase tenga toda la información necesaria para realizar la labor que tiene encomendada. (28)Es además uno de los patrones más utilizados en el trabajo con Symfony y se evidencia, con el uso de Propel para el mapeo de las bases de datos donde se crean las clases del modelo que contienen toda la lógica del negocio (Peer del Modelo), estas clases poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de cada tabla de la base de datos.(13) Ejemplos de estas clases utilizadas en el módulo de seguridad son *UsuarioPeer* y *RolPeer*.

Controlador*:* Este patrón está diseñado para asignar la responsabilidad de controlar el flujo de eventos del sistema en una clase en específico, esto facilita la centralización de las actividades y permite a su vez una mejor validación y seguridad de las mismas. Las clases controladoras no realizan todas las actividades sino que delegan las mismas a otras clases con las que mantiene un modelo de alta cohesión (28)Con la utilización del *framework* Symfony se pone de manifiesto este patrón ya que todas las peticiones web son controladas por un solo controlador frontal que es el punto de entrada único de toda la aplicación en un entorno determinado.(13)

Alta Cohesión*:* Se utiliza para dar una medida de cuan relacionadas y enfocadas están las responsabilidades de una clase. Define que la información que es almacenada en una clase debe de ser coherente y estar relacionada con dicha clase. La utilización de este patrón hace posible que las clases sean más fáciles de comprender, de reutilizar y almacenar .(28) Symfony permite una mejor organización del trabajo en cuanto a la estructura del proyecto y en cuanto a la asignación de responsabilidades con una alta cohesión. Las clases action.php están formadas por varias funcionalidades que están estrechamente relacionadas entre sí, es la encargada de definir las acciones para las plantillas y colaborar con otras para la realización de las diferentes operaciones (13)

Bajo Acoplamiento*:* Define que las clases deben estar ligadas entre sí lo menos posible, haciendo así que las clases sean más fáciles de entender cuando estén aisladas y más fáciles de reutilizar, por lo que si ocurre alguna modificación la repercusión de la misma en el resto de las clases sea leve.(28)Con la utilización de Symfony se puede apreciar claramente como la propia estructura del mismo proporciona la utilización de este patrón, las clases action.php solamente heredan de sfAction de esta misma forma las clases que implementan la lógica del negocio y de acceso a datos se encuentran en el modelo, las cuales no tienen asociaciones con las de la vista o el controlador, lo que proporciona que la dependencia en este caso sea baja.(13) Algunas clases desarrolladas en la aplicación y que cumplen este patrón son *newAction* y *editAction.*

Controlador Frontal*:* Es un patrón de vital importancia para las aplicaciones web ya que posibilita que toda la información fluya a través de una página controladora. Proporciona un punto de partida único que controla y gestiona las peticiones hechas a nuestra aplicación web.(13)

Patrones de la banda de los cuatro

Los patrones de la banda de los cuatro, conocidos como patrones GoF por sus siglas en inglés, son soluciones concretas y técnicas. Se basan en la experiencia acumulada al resolver problemas iterativos y favorecen la reutilización de código.

Fachada: Este patrón sirve para proveer una interfaz unificada que haga de intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas. Es utilizado para reducir la dependencia entre clases, ya que ofrece un punto de acceso al resto de las clases, si éstas cambian o se sustituyen por otras solo hay que actualizar la clase Fachada sin que el cambio afecte a las aplicaciones clientes. Fachada no oculta las clases sino que ofrece una forma más sencilla de acceder a ellas y se pone de manifiesto con la clase interfaz del dominio en la capa del dominio de la arquitectura propuesta.(27) Se evidencia en la clase *sfConfig*, la cual crea un objeto que proporciona métodos estáticos para poder acceder a los parámetros de configuración desde cualquier punto de la aplicación.(13)

Decorador*:* El patrón Decorador añade responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la especialización mediante herencia, cuando se trata de añadir funcionalidades. Es aplicado a las vistas donde el

contenido de la plantilla se integra en el *layout*, o si se mira desde el otro punto de vista, el *layout* decora la plantilla. De manera que el *layout* contiene el código HTML y los elementos comunes a todas las páginas, mientras las plantillas se encargan de mostrar el resultado de las acciones. (27)Las clases *sfView*, *sfFilter*, y *sfWidgetFormSchemaDecorator* implementan dicho patrón. (13)

Acción*:* El propósito es encapsular una petición en un objeto, permitiendo así parametrizar a los clientes con diferentes peticiones, hacer cola o llevar un registro de las peticiones y poder deshacer las operaciones. (27)Su aplicación se evidencia en la clase *sfFrontWebController*. Esta clase es la encargada de determinar cuál módulo y acción deben responder a las solicitudes de los usuarios. Encapsula las peticiones en forma de objetos permitiendo así parametrizar los clientes utilizando distintas peticiones, encolar las peticiones y ofrecer la posibilidad de deshacer las operaciones. (13)

Cadena de Responsabilidades*:* Permite establecer una cadena de objetos receptores a través de los cuales se pasa una petición formulada por un objeto emisor. Cualquiera de los objetos receptores puede responder a la petición en función de un criterio establecido.(27) Su aplicación se encuentra reflejada en la clase *sfEventDispatcher*, la cual rige el funcionamiento casi completo del *framework*.(13)

2.8.5 Diagrama de paquetes

En el diagrama de paquetes se muestra el sistema dividido en agrupaciones lógicas y las dependencias existentes entre esas agrupaciones. Los diagramas de paquetes suministran una descomposición de la jerarquía lógica de un sistema. En la figura 1 se muestra el diagrama de paquetes del módulo de seguridad de SIPP v2.0.

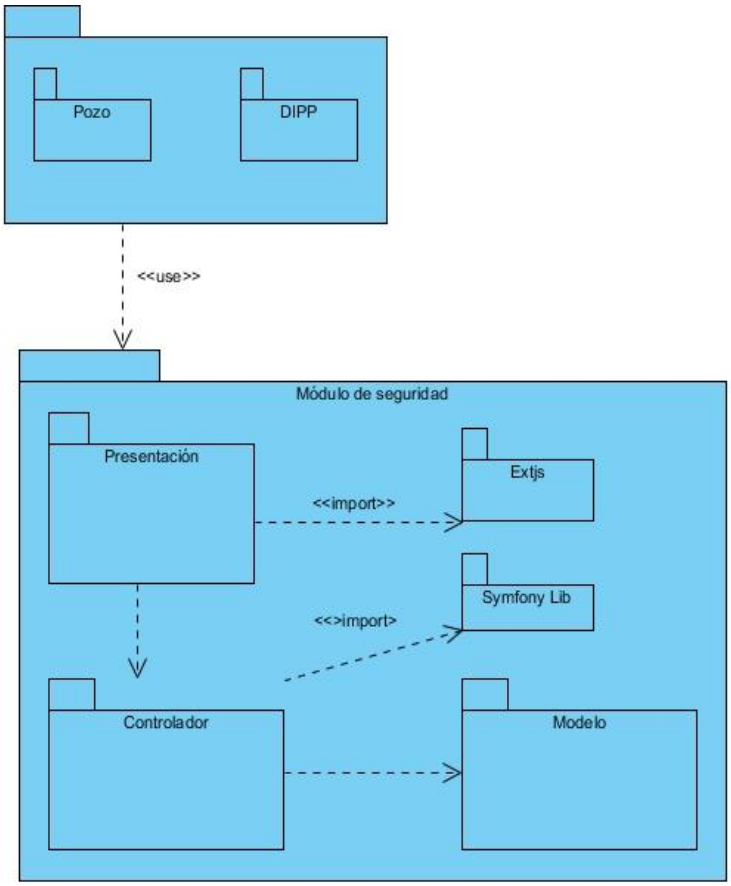

Figura 6 Diagrama de paquetes

2.9 Conclusiones del capítulo

En este capítulo fueron descritos un conjunto de requerimientos que debe cumplir el módulo describiendo las HU para una mejor comprensión de lo que se deberá implementar y dividiendo cada una en tareas de ingenierías. Con la definición del patrón arquitectónico a utilizar así como los patrones de diseño se llegó a una comprensión más exacta de las características del módulo a desarrollar y se realizó el modelo de datos del mismo, aspectos todos que dan paso a la implementación.

Capítulo 3: Implementación y Prueba

3.1 Introducción

En este capítulo se muestran los elementos correspondientes a las etapas de implementación y prueba del sistema. Destacándose los casos de prueba de aceptación como los artefactos fundamentales.

3.2 Estándares de codificación utilizados

Un estándar de codificación permite tener un código entendible y organizado. Son muy empleados para asegurar la unificación en el código y tratar que todos sigan reglas a la hora de programar. Son pautas que no están enfocadas a la lógica del programa sino a la apariencia y estructura del código.
Symfony tiene definido sus propios estándares de codificación y para el desarrollo del módulo en cuestión el proyecto SIPP ha decidido utilizarlos como se muestra a continuación.

- No usar las tabulaciones en el código. La sangría se hace por pasos de 2 espacios.
- No poner espacios después de un paréntesis de apertura y antes de un cierre.
- Las llaves van siempre en su propia línea.
- Utilice llaves para indicar la estructura de control del cuerpo, independientemente del número de declaraciones que contiene.
- Se debe declarar explícitamente la visibilidad de los métodos utilizando las palabras clave: *private*, *protected* o *public*.

- En el cuerpo de una función, las declaraciones de retorno deben tener una línea en blanco delante para aumentarla legibilidad.
- Todos los comentarios en una línea deben estar en este formato.
- Evite la evaluación de las variables dentro de cadenas, en lugar de optar por la concatenación.
- Utilice minúsculas en las constantes de PHP: falso, verdadero y nulo. Lo mismo ocurre con *array* (). Al contrario, siempre use cadenas en mayúsculas con las constantes definidas por el usuario, como la define (*MY_CONSTANT'*, *'foo/ bar*"). Mejor trate de utilizar siempre las constantes de clase.
- Para los archivos que contienen sólo código PHP los *tags* de demarcación (“<?“) no estarán permitidos, además no es requerido por PHP y omitirlos nos previene de algún accidente ocasionado por un espacio en blanco.
- Los parámetros van siempre en minúsculas.
- El nombre de las variables debe estar compuesto de caracteres alfa numéricos, el carácter *Underscore (_)* está permitido. Siempre tiene que comenzar con letra minúscula. Además siempre debe inicializarse y sobretodo deben tener nombres significativos.
- Cuando se le asigna un texto literal (sin contenido de variables) se utilizarán comillas dobles.

- Para concatenar *Strings* se utilizará el operador "." (punto), con un espacio por medio para mejorar la lectura.
- En las declaraciones *if/then/else* deberá existir un espacio antes y después del paréntesis condicional, lo mismo se aplica al *elseif*.
- Las funciones y métodos complicados deberán tener un bloque de documentación. El mismo será entre /**/ cuando sean de 2 líneas en adelante y // cuando sea una sola línea.

3.3 Diagrama de despliegue

Los diagramas propician un mejor entendimiento de la implementación del software. Estos permiten comprender con claridad lo que se ha implementado. Un diagrama de despliegue modela la arquitectura en tiempo de ejecución de un sistema. La figura 2 muestra el despliegue de los diferentes nodos físicos y la forma de conexión entre ellos.

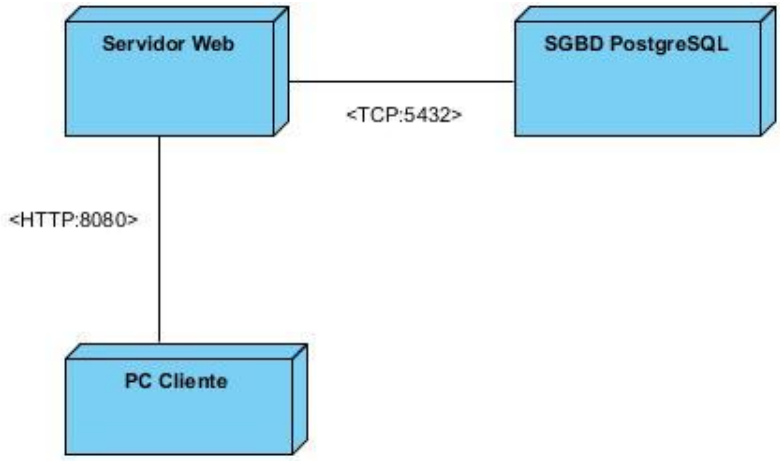

Figura 7 Diagrama de despliegue

A continuación se expone una breve descripción de los nodos y sus conexiones:

PC Cliente: Este nodo representa todas las PC clientes que se pueden conectar para acceder a los servicios que brinda la aplicación. Accede al *Servidor Web* mediante el protocolo HTTP y el puerto 8080.

Servidor web: Nodo que funcionará como servidor donde estará instalada la aplicación *web*. Se conecta al SGBD PostgreSQL mediante el protocolo TCP por el puerto 5432 con el objetivo de obtener la información referente a los usuarios y roles.

SGBD PostgreSQL: PC donde estará la base de datos, se encargará de permitir gestionar los datos con que trabajará la aplicación.

3.4 Prueba

Una de las últimas fases del ciclo de vida antes de entregar un programa para su explotación, es la fase de pruebas. El desarrollo del software implica una serie de actividades de producción donde es común la posibilidad de que aparezca una falla humana. Los errores pueden presentarse desde el inicio del proceso con la especificación

errónea de los requisitos; de igual forma en los posteriores pasos del diseño y desarrollo. Debido a la imposibilidad humana de trabajar y comunicarse de forma perfecta, el desarrollo del software ha de ir acompañado de una actividad que garantice la calidad.

3.4.1 Métodos de prueba

Las pruebas son acciones en las cuales el sistema es ejecutado bajo condiciones o requerimientos determinados. Los resultados son chequeados y registrados. Las pruebas verifican los resultados de la implementación del sistema.

Pruebas de aceptación en la metodología SXP

Las pruebas de aceptación se crean a partir de las HU. El cliente define los escenarios para verificar si la HU ha sido correctamente implementada. Una HU puede tener una o varias pruebas de aceptación. El cliente es responsable de verificar el pasaje de las pruebas de aceptación y priorizar la corrección de los resultados fallidos. Las pruebas de aceptación son pruebas tipo caja negra a nivel del sistema: cada una corresponde a un resultado producido por la aplicación. Deben crearse en cada iteración, ser automáticas, correrse frecuentemente, publicarse sus resultados y programarse su corrección para la próxima iteración. El principal objetivo de estas pruebas es comprobar que las funcionalidades de un sistema requeridas por el cliente se cumplan correctamente.

De los casos de prueba realizados al módulo de seguridad de SIPP v 2.0 en la primera iteración resultaron insatisfactorios 3, en la segunda iteración 2 y en la tercera iteración 1 lo que representa un 33,3 % de las pruebas efectuadas.

A continuación se muestran los casos de prueba de aceptación con resultados satisfactorios:

Caso de Prueba de Aceptación	
Código Caso de Prueba: S-HU1-P1	**Nombre Historia de Usuario:** Gestionar Usuario
Nombre de la persona que realiza la prueba: Joyce Suárez Fabre	
Descripción de la Prueba: Prueba para la funcionalidad Insertar usuario.	
Condiciones de Ejecución: El usuario debe estar autenticado. El usuario debe tener los permisos necesarios para realizar esta acción.	
Entrada / Pasos de ejecución: El usuario inserta los datos necesarios y se ejecuta la acción Aceptar.	
Resultado Esperado: Los usuarios son insertados correctamente y se muestra un mensaje de aviso.	
Evaluación de la Prueba: Satisfactoria	

Tabla 21 Caso de Prueba de Aceptación: Adicionar usuario

Caso de Prueba de Aceptación	
Código Caso de Prueba: S-HU1-P2	**Nombre Historia de Usuario:** Gestionar Usuario
Nombre de la persona que realiza la prueba: Joyce Suárez Fabre	
Descripción de la Prueba: Prueba para la funcionalidad Modificar usuario.	
Condiciones de Ejecución: El usuario debe estar autenticado. El usuario debe tener los permisos necesarios para realizar esta acción. El usuario debe existir en la base de datos.	
Entrada / Pasos de ejecución: El usuario modifica los datos necesarios y se ejecuta la acción Aceptar.	
Resultado Esperado: Los datos del usuario son modificados correctamente y se muestra un mensaje de aviso.	
Evaluación de la Prueba: Satisfactoria	

Tabla 22 Caso de Prueba de Aceptación: Modificar usuario

Caso de Prueba de Aceptación	
Código Caso de Prueba: S-HU1-P3	**Nombre Historia de Usuario:** Gestionar Usuario
Nombre de la persona que realiza la prueba: Joyce Suárez Fabre	
Descripción de la Prueba: Prueba para la funcionalidad Eliminar usuario.	
Condiciones de Ejecución: El usuario debe estar autenticado. El usuario debe tener los permisos necesarios para realizar esta acción. El usuario a eliminar debe existir en la base de datos.	
Entrada / Pasos de ejecución: Es seleccionado el usuario a eliminar y se ejecuta la acción Aceptar.	
Resultado Esperado: El usuario es eliminado correctamente y se muestra un mensaje de aviso.	
Evaluación de la Prueba: Satisfactoria	

Tabla 23 Caso de Prueba de Aceptación: Eliminar usuario

Caso de Prueba de Aceptación

Código Caso de Prueba: S-HU1-P4	**Nombre Historia de Usuario:** Gestionar Usuario
Nombre de la persona que realiza la prueba: Joyce Suárez Fabre	
Descripción de la Prueba: Prueba para la funcionalidad Buscar usuario.	
Condiciones de Ejecución: El usuario debe estar autenticado. El usuario debe tener los permisos necesarios para realizar esta acción.	
Entrada / Pasos de ejecución: Se introduce el usuario o el nombre de la persona que se quiere buscar.	
Resultado Esperado: Se muestra el usuario indicado si se encuentra en la Base de Datos.	
Evaluación de la Prueba: Satisfactoria	

Tabla 24 Caso de Prueba de Aceptación Buscar usuario

Caso de Prueba de Aceptación	
Código Caso de Prueba: S-HU2-P5	**Nombre Historia de Usuario:** Autenticar Usuario
Nombre de la persona que realiza la prueba: Joyce Suárez Fabre	
Descripción de la Prueba: Prueba para la funcionalidad Autenticar usuario.	
Condiciones de Ejecución: El usuario debe existir en la base de datos.	
Entrada / Pasos de ejecución: Se introduce el usuario y contraseña y se ejecuta la acción Aceptar.	
Resultado Esperado: El usuario accede a la información según los permisos del rol que tiene asignado.	
Evaluación de la Prueba: Satisfactoria	

Tabla 25 Caso de Prueba de Aceptación: Autenticar usuario

Caso de Prueba de Aceptación	
Código Caso de Prueba: S-HU3-P6	**Nombre Historia de Usuario:** Cambiar Contraseña
Nombre de la persona que realiza la prueba: Joyce Suárez Fabre	
Descripción de la Prueba: Prueba para la funcionalidad Cambiar contraseña	
Condiciones de Ejecución: El usuario debe existir en la base de datos.	
Entrada / Pasos de ejecución: Se introduce el usuario, la contraseña actual y la nueva	

contraseña y se ejecuta la acción Aceptar.
Resultado Esperado: Se guarda la nueva contraseña y se muestra un mensaje de aviso.
Evaluación de la Prueba: Satisfactoria

Tabla 26 Caso de Prueba de Aceptación: Cambiar contraseña

Caso de Prueba de Aceptación	
Código Caso de Prueba: S-HU4-P7	**Nombre Historia de Usuario:** Gestionar Rol
Nombre de la persona que realiza la prueba: Joyce Suárez Fabre	
Descripción de la Prueba: Prueba para la funcionalidad insertar rol.	
Condiciones de Ejecución: El usuario debe estar autenticado. El usuario debe tener los permisos necesarios para realizar esta acción.	
Entrada / Pasos de ejecución: El usuario inserta los datos necesarios y se ejecuta la acción Aceptar.	
Resultado Esperado: El rol es insertado correctamente y se muestra un mensaje de aviso.	
Evaluación de la Prueba: Satisfactoria	

Tabla 27 Caso de Prueba de Aceptación: Adicionar rol

Caso de Prueba de Aceptación	
Código Caso de Prueba: S-HU4-P8	**Nombre Historia de Usuario:** Gestionar Rol
Nombre de la persona que realiza la prueba: Joyce Suárez Fabre	
Descripción de la Prueba: Prueba para la funcionalidad Modificar rol.	
Condiciones de Ejecución: El usuario debe estar autenticado. El usuario debe tener los permisos necesarios para realizar esta acción. El rol debe existir en la base de datos.	
Entrada / Pasos de ejecución: El usuario modifica los datos necesarios y se ejecuta la acción Aceptar.	
Resultado Esperado: Los datos del rol son modificados correctamente y se muestra un mensaje de aviso.	
Evaluación de la Prueba: Satisfactoria	

Tabla 28 Caso de Prueba de Aceptación: Modificar rol

Caso de Prueba de Aceptación

Código Caso de Prueba: S-HU1-P3	**Nombre Historia de Usuario:** Gestionar Rol
Nombre de la persona que realiza la prueba: Joyce Suárez Fabre	
Descripción de la Prueba: Prueba para la funcionalidad Eliminar rol.	
Condiciones de Ejecución: El usuario debe estar autenticado. El usuario debe tener los permisos necesarios para realizar esta acción. El rol debe existir en la base de datos.	
Entrada / Pasos de ejecución: Es seleccionado el rol a eliminar y se ejecuta la acción Aceptar.	
Resultado Esperado: El rol es eliminado correctamente y se muestra un mensaje de aviso.	
Evaluación de la Prueba: Satisfactoria	

Tabla 29 Caso de Prueba de Aceptación: Eliminar rol

3.4 Conclusiones del capítulo

En este capítulo se presentaron los estándares de códigos utilizados en el desarrollo del módulo así como el diagrama de despliegue que permite un mejor entendimiento de la implementación del sistema. Se realizaron además las pruebas de aceptación que validan la solución propuesta.

Conclusiones generales:

Una vez desarrollada la presente investigación, se puede arribar a las siguientes conclusiones generales:

- ✓ Con el desarrollo del módulo de seguridad de SIPP v2.0 se eliminan los conflictos de acceso a la información permitiendo la gestión de usuarios y roles a través de las funcionalidades del sistema.
- ✓ A partir de la elaboración del marco teórico de la investigación se realizó un estudio de la gestión de la seguridad en aplicaciones web lo que aportó los conocimientos necesarios para la implementación del módulo en cuanto a herramientas, tecnologías y metodología a utilizar así como los aspectos básicos para garantizar la seguridad en sistemas de este tipo.
- ✓ Mediante el diseño se logró establecer una base sólida, obteniendo una visión del sistema que encaminó a qué hacer mediante las HU y cómo hacerlo mediante los requisitos no funcionales.
- ✓ Se realizaron pruebas de aceptación que permitieron validar el módulo garantizando consistencia y firmeza al producto desarrollado.

Con el desarrollo del Módulo de Seguridad para SIPP v2.0 se le da cumplimiento al objetivo general trazado desde el inicio del trabajo.

Recomendaciones:

Una vez concluido este trabajo de diploma y con el objetivo de seguir perfeccionando el funcionamiento del Módulo de Seguridad se recomienda:

- Profundizar en el estudio de las actividades que realizan los usuarios durante el proceso de perforación y fuera del mismo, a fin de encontrar nuevas funcionalidades que aumenten la seguridad del sistema. Ejemplo: Registros de Dominio, manejo de usuarios por cuentas de correo.
- Investigar sobre nuevas funciones o complementos del *framework* de desarrollo con respecto a seguridad para posibles inclusiones en el módulo.

Bibliografía citada:

1. INFORMÁTICAS, P. D. L. U. D. L. C. Misión. 20 de noviembre 2013 nº Disponible en: http://www.uci.cu/mision.

2. ANDALUCÍA, M. D. D. D. L. J. D. Seguridad en aplicaciones web. 1 de junio de 2013 nº Disponible en: http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/212.

3. MSDN. Administrar los usuarios de sitios web con roles. 30 de noviembre de 2013 nº Disponible en: http://msdn.microsoft.com/es-es/library/t32yf0a9%28v=vs.100%29.aspx.

4. DONTNETNUKE. DotNetNuke Professional Edition. 14 de enero de 2013 nº Disponible en: http://www.dotnetnuke.com/Products/Professional-Edition-Trial/Professional-Edition-Trial-Download.aspx.

5. RESOURCES, A. Introducción a ArcGIS 16 de febrero de 2013 nº Disponible en: http://resources.arcgis.com/es/help/getting-started/articles/026n00000014000000.htm.

6. SOLÍS, J. Seguridad en sistemas de control de acceso basados en Roles. 1 de junio de 2013 nº Disponible en: http://www.equipom45.es/m45blog/25-seguridad/135-control-acceso-basado-roles.html.

7. CUEVAS, D. T. Análisis y diseño del Sistema de Manejo Integral de Perforación de Pozos. 2009, nº

8. PENADÉS, P. L. Y. M. D. C. Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). 18 de enero de 2013 nº

9. KNIBERG, H. Scrum y XP desde las trincheras. 7 de marzo de 2013 2007, nº

10. ROMERO, G. M. P. Trabajo de diploma: Metodología ágil para proyectos de software libre. 5 de marzo de 2013 2008, nº

11. KIOSKEA.NET. Lenguajes de Programación 10 de febrero de 2013 nº Disponible en: http://es.kioskea.net/contents/langages/langages.php3.

12. PHP.NET. 17 de enero de 2013 nº Disponible en: http://php.net.

13. FABIEN POTENCIER, F. Z. Symfony,la guía definitiva. 17 de diciembre 2012 nº Disponible en: http://www.librosweb.es/symfony/.

14. FRAMEWORK, Z. Framework 2--Introduccion. 20 de enero de 2013 nº Disponible en: http://framework.zend.com/manual/1.12/en/zend.acl.introduction.html--Zend

15. HTTP://DOJOTOOLKIT.ORG/. Dojo Toolkit. 13 de febrerod e 2013 nº Disponible en: http://dojotoolkit.org/.

16. EGUILUZ, J. Introducción a AJAX. 20 de enero de 2013 nº Disponible en: http://librosweb.es/ajax/capitulo_10/la_libreria_jquery.html.

17. BULLOCK, C. Ext JS. Comunidad en Español. . 10 de febrero de 2013 nº Disponible en: http://extjses.com/.

18. ALVAREZ, S. Introducción a este concepto y características especiales. 29 de noviembre de 2012 nº Disponible en: http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.

19. Sistema de Gestión de Base de Datos. 20 de enero de 2013 nº Disponible en: http://unecomputacion.wordpress.com/2008/07/08/postgresql/.

20. CASTELLANO, A. P. Y. T. E. Servidor Apache. 15 de febrero de 2013 nº Disponible en: http://www.abcdatos.com/webmasters/programa/servidor-apache.html.

21. BUSTILLOS, R. O. Entorno de desarrollo Integrado. 20 de enero de 2013 nº

22. NETBEANS.ORG. Netbeans. 16 de febrero de 2013 nº Disponible en: http://netbeans.org.

23. ERNEST TENIENTE LÓPEZ, A. O. R., ENRIC MAYOL SARROCA,CRISTINA GÓMEZ SEONE. Diseño de sistemas software en UML. 3 de marzo de 2013 nº Disponible en: http://books.google.com.ar/books?id=p7nD8_g77_MC&lpg=PT38&dq=uml%20paso%20a%20paso&hl=es&pg=PP1#v=onepage&q=uml%20paso%20a%20paso&f=false.

24. FREEDOWNLOADMANAGER. Visual Paradigm para UML. 18 de diciembre de 2012 nº Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(Iglesia_Anglicana)_%5BMac_OS_X_cuenta_14717_p/.

25. PRESSMAN, R. Ingenieria del Software, un enfoque práctico. 10 de abril de 2013 2002, nº

26. SEVILLA, D. D. L. Y. S. I. E. T. S. D. I. I. U. D. Ingeniería de Requisitos en Aplicaciones para la Web –Un estudio comparativo. 30 de marzo de 2013 nº

27. ERICH GAMMA, R. H., RALPH JOHNSON, JOHN VLISSIDES - ADDISON WESLEY Design Patterns. Elements of Reusable Object-Oriented Software 15 de abril de 2013 nº

28. MARCELLO VISCONTI, H. A. Patrones GRASP : Fundamentos de Ingeniería de Software. 18 de abril de 2013 nº

Bibliografía

- INFORMÁTICAS, P. D. L. U. D. L. C. Misión. 20 de noviembre 2013 nº Disponible en: http://www.uci.cu/mision.
- ANDALUCÍA, M. D. D. D. L. J. D. Seguridad en aplicaciones web. 1 de junio de 2013 nº Disponible en: http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/212.
- MSDN. Administrar los usuarios de sitios web con roles. 30 de noviembre de 2013 nº Disponible en: http://msdn.microsoft.com/es-es/library/t32yf0a9%28v=vs.100%29.aspx.
- DONTNETNUKE. DotNetNuke Professional Edition. 14 de enero de 2013 nº Disponible en: http://www.dotnetnuke.com/Products/Professional-Edition-Trial/Professional-Edition-Trial-Download.aspx.
- RESOURCES, A. Introducción a ArcGIS 16 de febrero de 2013 nº Disponible en: http://resources.arcgis.com/es/help/getting-started/articles/026n00000014000000.htm.
- SOLÍS, J. Seguridad en sistemas de control de acceso basados en Roles. 1 de junio de 2013 nº Disponible en: http://www.equipom45.es/m45blog/25-seguridad/135-control-acceso-basado-roles.html.
- CUEVAS, D. T. Análisis y diseño del Sistema de Manejo Integral de Perforación de Pozos. 2009, nº
- PENADÉS, P. L. Y. M. D. C. Métodologías ágiles para el desarrollo de software:
- eXtreme Programming (XP). 18 de enero de 2013 nº
- KNIBERG, H. Scrum y XP desde las trincheras. 7 de marzo de 2013 2007, nº
- ROMERO, G. M. P. Trabajo de diploma: Metodología ágil para proyectos de software libre. 5 de marzo de 2013 2008, nº
- KIOSKEA.NET. Lenguajes de Programación 10 de febrero de 2013 nº Disponible en: http://es.kioskea.net/contents/langages/langages.php3.
- PHP.NET. 17 de enero de 2013 nº Disponible en: http://php.net.
- FABIEN POTENCIER, F. Z. Symfony,la guía definitiva. 17 de diciembre 2012 nº Disponible en: http://www.librosweb.es/symfony/.
- FRAMEWORK, Z. Framework 2--Introduccion. 20 de enero de 2013 nº Disponible en: http://framework.zend.com/manual/1.12/en/zend.acl.introduction.html--Zend
- HTTP://DOJOTOOLKIT.ORG/. Dojo Toolkit. 13 de febrerod e 2013 nº Disponible en: http://dojotoolkit.org/.

Bibliografía

- EGUILUZ, J. Introducción a AJAX. 20 de enero de 2013 nº Disponible en: http://librosweb.es/ajax/capitulo_10/la_libreria_jquery.html.
- BULLOCK, C. Ext JS. Comunidad en Español. . 10 de febrero de 2013 nº Disponible en: http://extjses.com/.
- ALVAREZ, S. Introducción a este concepto y características especiales. 29 de noviembre de 2012 nº Disponible en: http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.
- Sistema de Gestión de Base de Datos. 20 de enero de 2013 nº Disponible en: http://unecomputacion.wordpress.com/2008/07/08/postgresql/.
- CASTELLANO, A. P. Y. T. E. Servidor Apache. 15 de febrero de 2013 nº Disponible en: http://www.abcdatos.com/webmasters/programa/servidor-apache.html.
- BUSTILLOS, R. O. Entorno de desarrollo Integrado. 20 de enero de 2013 nº
- NETBEANS.ORG. Netbeans. 16 de febrero de 2013 nº Disponible en: http://netbeans.org.
- ERNEST TENIENTE LÓPEZ, A. O. R., ENRIC MAYOL SARROCA,CRISTINA GÓMEZ SEONE. Diseño de sistemas software en UML. 3 de marzo de 2013 nº Disponible en: http://books.google.com.ar/books?id=p7nD8_g77_MC&lpg=PT38&dq=uml%20paso%20a%20paso&hl=es&pg=PP1#v=onepage&q=uml%20paso%20a%20paso&f=false.
- FREEDOWNLOADMANAGER. Visual Paradigm para UML. 18 de diciembre de 2012 nº Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(Iglesia_Anglicana)_%5BMac_OS_X_cuenta_14717_p/.
- PRESSMAN, R. Ingenieria del Software, un enfoque práctico. 10 de abril de 2013 2002, nº
- SEVILLA, D. D. L. Y. S. I. E. T. S. D. I. I. U. D. Ingeniería de Requisitos en Aplicaciones para la Web –Un estudio comparativo. 30 de marzo de 2013 nº
- ERICH GAMMA, R. H., RALPH JOHNSON, JOHN VLISSIDES - ADDISON WESLEY Design Patterns. Elements of Reusable Object-Oriented Software 15 de abril de 2013 nº
- MARCELLO VISCONTI, H. A. Patrones GRASP : Fundamentos de Ingeniería de Software. 18 de abril de 2013 nº
- AGUERO, P. M. Z. Biblioteca Virtual de Derecho,Economía y Ciencias Sociales. 6 de noviembre de 2012 nº Disponible en: http://www.eumed.net/librosgratis/2010e/822/Metodos%20del%20conocimiento%20teorico.htm
- HORNA, M. G. ExtJSIntroducción. 2009, nº

- NETBEANS.ORG. Netbeans. 16 de febrero de 2013 nº Disponible en: http://netbeans.org.

- PHP.NET. 17 de enero de 2013 nº Disponible en: http://php.net.

Anexos:

Anexo1:

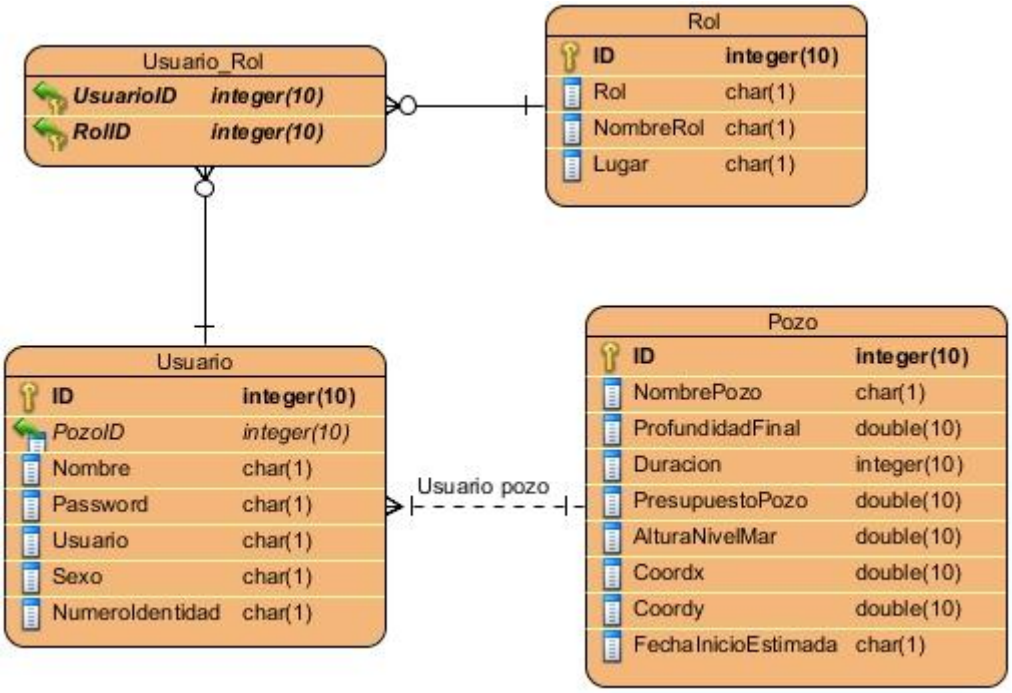

Figura 8 Modelo de datos

Anexo 2:

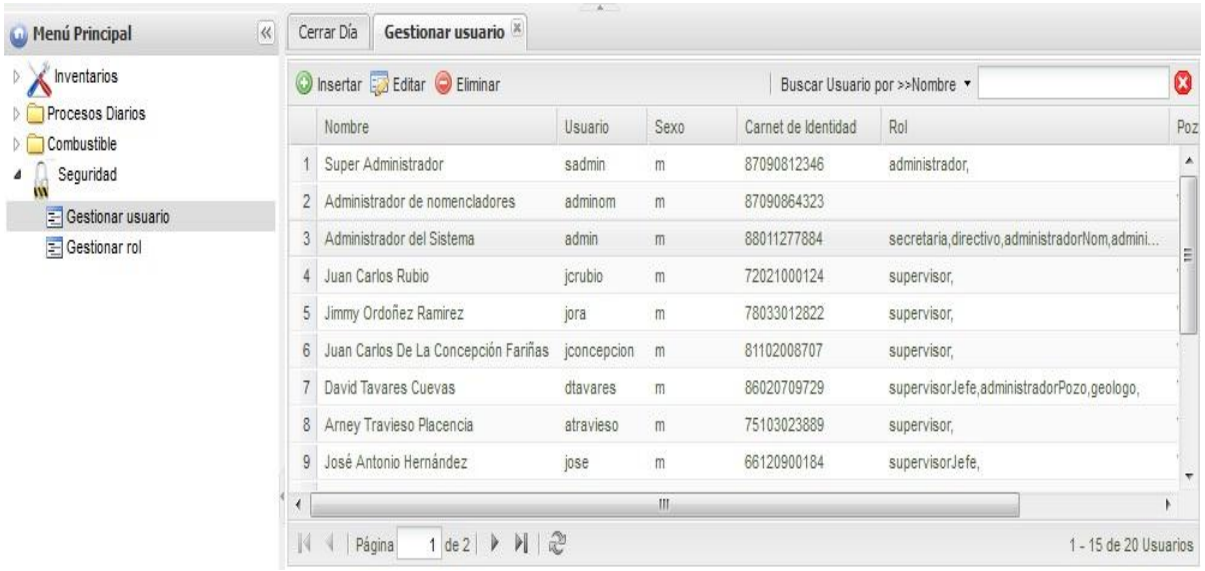

Figura 9 Gestionar usuario

Anexo 3:

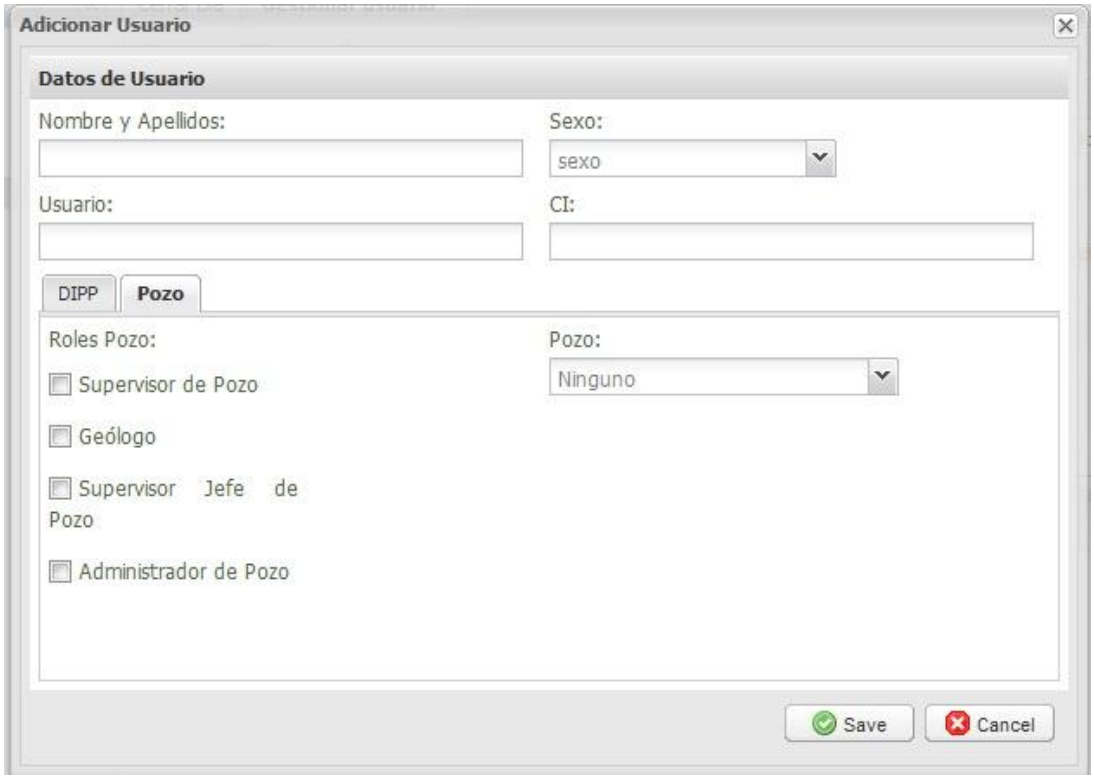

Figura 10 Adicionar usuario Pozo

Anexo 4:

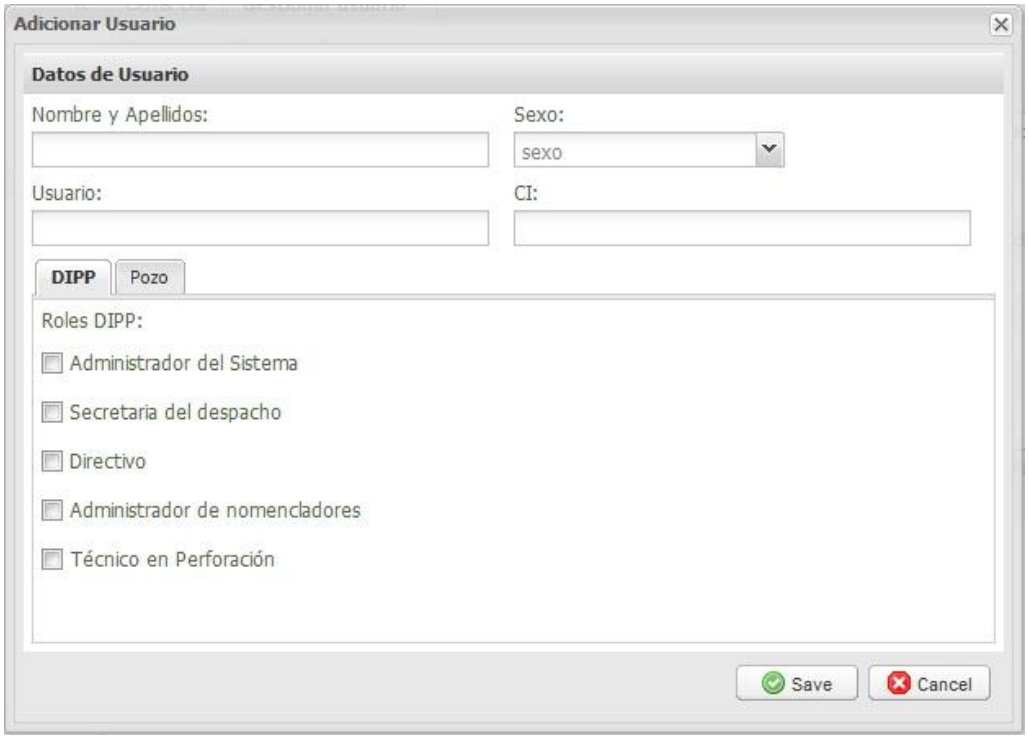

Figura 11 Adicionar usuario DIPP

Anexo 5:

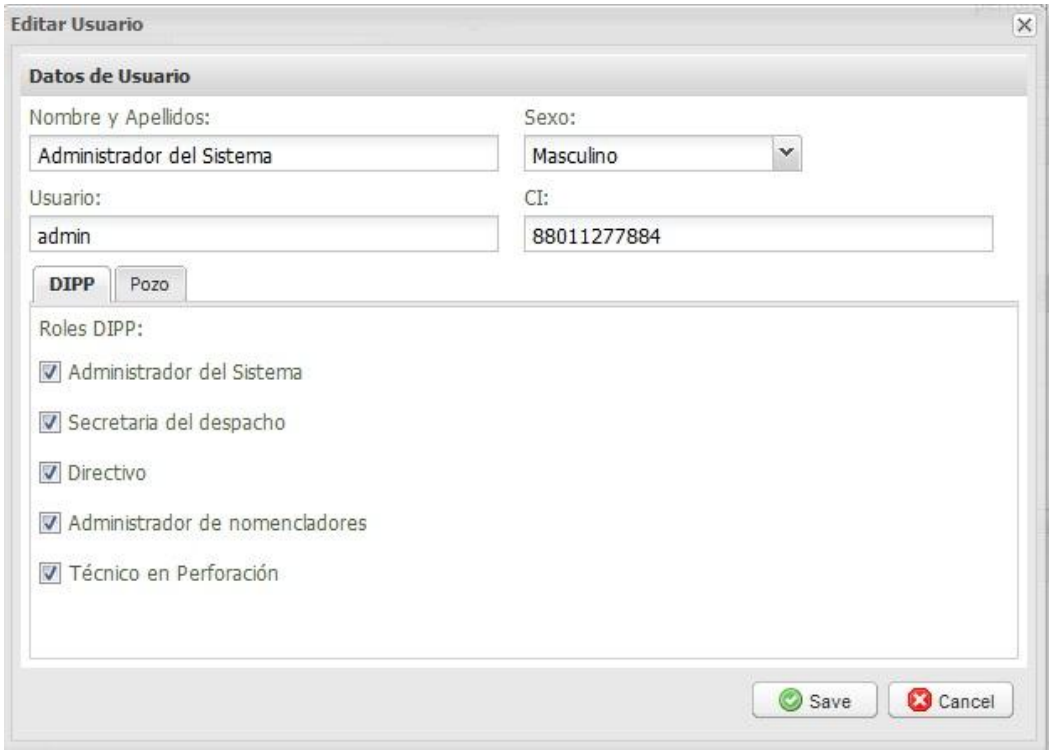

Figura 12 Editar usuario

Anexo 6:

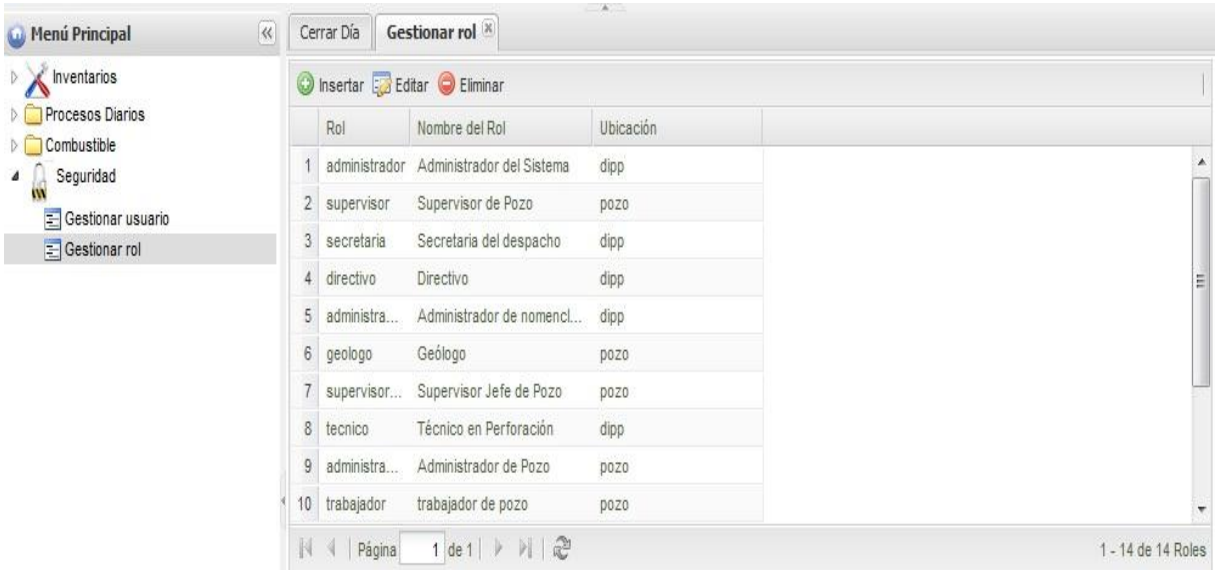

Figura 13 Gestionar rol

Anexo 7:

Figura 14 Adicionar rol

Anexo 8:

Figura 15 Editar rol

Anexo 9:

Figura 16 Otorgar permisos