Universidad de las Ciencias Informáticas

Facultad 6

Título: *"Módulo de Seguridad para el Sistema de Gestión de Datos Geológicos v2."*

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: *Yosbel Oscar Santiesteban Mesa*

Tutor: *Ing. Alberto Menendez Romero*

Ciudad de La Habana, junio de 2013

"Año 55 de la Revolución

Declaración de Autoría

Declaro ser el único autor de la presente tesis que tiene por título “Módulo de Seguridad para el Sistema de Gestión de Datos Geológicos v2” y se reconoce 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 _____.

_____	_____
Yosbel Oscar Santiesteban Mesa	Ing. Alberto Menendez Romero
Firma del Autor	Firma del Tutor

Declaración de Autoría

Datos de Contacto

Nombre y apellidos: Ing. Alberto Menéndez Romero

Correo electrónico: amenedez@uci.cu

Categoría docente: Especialista General.

Año de graduación: 2010.

Profesión: Ingeniero en Ciencias Informáticas.

Breve descripción: Graduado en la Universidad de las Ciencias Informáticas en el año 2010, de Ingeniero en Ciencias Informáticas. Cuenta con 2 años de experiencia en el tema. Se desempeña como Arquitecto de Software del proyecto Sistema de Gestión de Datos Geológico.

Resumen

Desde hace ya varios años, los bancos, las empresas y los organismos oficiales incorporaron el uso de sistemas informáticos para almacenar y procesar la información proveniente de sus actividades. Este acopio de información personal y el uso que se haga de ella implican problemas potenciales para la intimidad de los usuarios, lo que impone la necesidad de buscar alternativas que permitan garantizar el correcto acceso a la información de acuerdo al nivel de autorización de la persona que la gestiona, así como realizar un seguimiento de la gestión que tenga dicha información.

El presente trabajo de diploma contiene la investigación y desarrollo de un módulo para el Sistema de Gestión de Datos Geológicos en su versión 2.0, herramienta informática tipo web que automatiza los procesos de gestión más importantes que se desarrollan en la Oficina Nacional de Recursos Minerales (ONRM), dirigido a garantizar la seguridad de la información de dicho sistema. La aplicación desarrollada sobre el framework Symfony2 permite el control por niveles de acceso a los datos que se manipulan en el sistema. El proceso estuvo guiado por una metodología para lograr una mejor producción y organización, se identificaron salidas (o productos intermedios) en cada fase permitiendo planificar y controlar el proyecto y un proceso estándar en la organización.

PALABRAS CLAVE

Aplicación informática, autorización, control de acceso, framework, gestión, módulo, seguimiento, seguridad y web.

Índice de Tablas

Índice de Figuras

Índice

INTRODUCCIÓN

En el transcurso de la historia de la ciencia han existido determinados momentos clave en los que el ingenio de ciertas personas, unidos a toda una situación social e ideológica propicias, les han permitido percibir algo que nadie antes había sido capaz de ver. En muchas ocasiones esta revolución en los conceptos viene acompañada de innovaciones tecnológicas que constituyen grandes saltos en el desarrollo de nuevos aparatos destinados a revolucionar el progreso de la sociedad.

La propia necesidad humana de agilizar actividades, mejorar su forma de vida y transformar el entorno circundante, ha incitado la creación de artefactos con características capaces de humanizar los procesos; he aquí que el surgimiento de la computación haya marcado un hito significativo en el desarrollo de la especie humana. Con la aparición de las computadoras y luego el desarrollo de Internet, obtener información de interés es “algo tan sencillo como caminar”. Por ello hoy la mayor parte de la información que se consulta está disponible en formato digital.

Producto del avance tecnológico ha surgido un fenómeno propio del desarrollo de las Tecnologías de la Información y la Comunicación (TICs) y es que cualquier persona puede proporcionar información de interés y es esta la causa de que hoy no todo lo que se pueda encontrar en el amplio mundo del internet sea verídico. Para la empresa actual y más aún para las que tienen mayor prestigio internacional; es de vital importancia que la información con la que trabaje se encuentre en correcto estado y forma y que solo el personal autorizado pueda acceder a esta información; lo que supone un reto en el cual invertir para no dejar al descuido.

En Cuba existen varias instituciones que se dedican a la producción de aplicaciones informáticas para clientes nacionales e internacionales. Para estas estructuras productoras es fundamental que las soluciones informáticas que se desarrollen garanticen que sólo la información sea actualizada por quien se autorice, evitando que personas extrañas a la misma puedan alterar la integridad y consistencia de datos confidenciales.

Un ejemplo de institución en Cuba que se dedica a la producción de aplicaciones informáticas es la Universidad de las Ciencias Informáticas (UCI) que se ha convertido en uno de los centros de altos estudios más reconocidos del país, no sólo por la calidad de la enseñanza sino que desde sus inicios al calor de la

Batalla de Ideas, ha contemplado entre sus principales actividades la producción de software y por consiguiente la seguridad de estos y de la información que se maneja en los mismos.

La UCI está estructurada en facultades donde a su vez existen los Centros de Desarrollo que se encargan de investigar y producir aplicaciones informáticas en un área temática en particular. Ejemplo de ello es la facultad 6 la cual cuenta con el centro de desarrollo: Geoinformática y Señales Digitales (GEySED). En este se desarrolla el Sistema de Gestión de Datos Geológicos (SGDG) en su segunda versión el cual es una solución informática tipo web que automatiza los procesos de gestión más importantes que se desarrollan en cada área de la Oficina Nacional de Recursos Minerales (ONRM), sustentado en herramientas informáticas robustas, debidamente protegidas y aseguradas, que sigan estándares de calidad, legales y normativos de las Geociencias. Incluye además la representación geoespacial de los datos, influyendo de manera decisiva en la toma de decisiones de esta entidad.

La solución, conceptualizada con el nombre de SGDG está integrada por módulos, cada uno de ellos asociados a las principales áreas del negocio. Con el desarrollo de este producto se pretende lograr la preservación del alto grado del conocimiento geológico alcanzado en Cuba, la integración de todas las aplicaciones que hoy se utilizan en la ONRM en una arquitectura única, sustentada en herramientas informáticas libres, lo que supone un beneficio económico para dicha entidad, y la posibilidad de respuesta eficiente a las necesidades de información para la toma de decisiones.

La ONRM tiene una estructura de un solo nivel, su oficina central, que tiene las direcciones que a continuación se relacionan:

- Dirección General.
- Dirección de Documentación.
- Dirección de Informática.
- Dirección de Control Económico.
- Dirección de Registro y Control.
- Dirección de Técnica.
- Dirección de Hidrocarburos.

En esta estructura existe una cadena de responsabilidades debido a los múltiples cargos que hay, además se tiene la necesidad de intercambiar datos entre las direcciones y que el personal de una no pueda modificar la información de otra pero sí consultarla en determinadas ocasiones. El SGDG almacena y gestiona toda la información referente a la actividad minera en el país. Como parte principal de la información a proteger se encuentra el conocimiento adquirido sobre el níquel y el petróleo, el primero ya es parte importante de nuestra economía y el segundo puede llegar a convertirse también en un renglón importante de la misma. Dada la sensibilidad de la información anteriormente mencionada, por solo mencionar dos ejemplos, y su importancia para la economía se hace necesario implementar los mecanismos de seguridad que garanticen la integridad de la misma así como realizar un seguimiento de la gestión que tenga dicha información.

De acuerdo con la situación expuesta se puede definir como **problema a resolver** de la investigación actual:

¿Cómo garantizar el control y seguimiento por niveles de acceso de los datos que se manipulan en el Sistema de Gestión de Datos Geológicos v2?

Como resultado del análisis del propio problema se definió como **objeto de estudio** los procesos de autenticación, autorización y auditoría en la gestión de la seguridad de aplicaciones web; siendo el **campo de acción** la implementación de la seguridad de la información en el Sistema de Gestión de datos Geológicos v2.

Con el fin de resolver el problema descrito se hace necesario desarrollar el módulo de seguridad para el Sistema de Gestión de Datos Geológicos v2, siendo este el **objetivo general** de la investigación.

Por lo anteriormente expuesto, se propone como **idea a defender**: Con el desarrollo del módulo de seguridad para el Sistema de Gestión de datos Geológicos v2 se garantizará el control por niveles de acceso a los datos que se manipulen en él.

Para darle cumplimiento al objetivo se tiene como tareas de investigación:

1. Caracterizar le estructura interna de la Oficina Nacional de Recursos Minerales de acuerdo a organización, departamentos o áreas de trabajo.
2. Caracterizar el estado actual de las tecnologías existentes para el control por niveles de acceso (autenticación, autorización y auditoria).

3. Definir una estrategia de seguridad aplicable al Sistema de Gestión de datos Geológicos v2.
4. Modelar la solución en término de Diseño.
5. Implementar el módulo de seguridad propuesto.
6. Probar y validar la solución propuesta.

Con el correcto cumplimiento de las tareas se espera obtener los siguientes resultados:

1. Módulo de Seguridad para el Sistema de Gestión de Datos Geológicos v2.
2. Documentación técnica del proceso de desarrollo del módulo de Seguridad para el Sistema de Gestión de Datos Geológicos v2.

En el trascurso y desarrollo de esta investigación se tendrán en cuenta una serie de métodos científicos para la obtención, procesamiento y llegada a conclusiones, los cuales se exponen a continuación:

Métodos teóricos

Los métodos teóricos permiten estudiar las características del objeto de la investigación que no son observables directamente. Dentro de los métodos teóricos a utilizar se tienen los siguientes:

1. Histórico-Lógico: Este método ayuda a definir una sucesión lógica, necesaria para conocer la evolución y desarrollo de la temática de la aplicación de la seguridad en los sistemas informáticos, las principales etapas de su desenvolvimiento y las conexiones históricas fundamentales.
2. Modelación: Mediante este método es posible modelar la realidad futura de la investigación, los principales elementos que lo componen y su funcionamiento lo cual ayuda a descubrir y estudiar nuevas cualidades y relaciones del objeto de estudio.
3. Analítico – Sintético: Este método permite analizar y comprender la teoría y documentación relacionada con el tema de investigación, permitiendo así, extraer los elementos más relacionados e importantes con el objeto de estudio.

Métodos Empíricos

Los métodos empíricos permiten describir y explicar las características de los fenómenos analizados,

extrayendo la información que se necesita sobre ellos a través de observaciones. Dentro de los métodos empíricos a utilizar se tiene el siguiente:

1. Observación: Este método es de vital importancia ya que permite percibir a partir de la situación real que se investiga, cómo se desarrollan a groso modo los procesos que constituyen el objeto de estudio.

Para un mejor entendimiento de los avances de la investigación, se ha dividido el documento en varios capítulos:

En el capítulo 1 se especifican y explican los principales conceptos asociados al campo de acción definido, se realiza una descripción del objeto de estudio identificado, se detalla la situación problemática actual, además se expone el estudio y las principales conclusiones que se obtuvieron luego de que se valoraran las soluciones existentes relacionadas con la temática.

El capítulo 2 detalla las principales tecnologías, lenguajes de programación y herramientas que se utilizarán para la construcción de la solución.

En el capítulo 3 se describe propiamente la solución propuesta en términos del negocio, requisitos funcionales y no funcionales y los casos de uso del sistema, todo esto unido a los correspondientes diagramas que lo modelan.

El capítulo 4 presenta la construcción propuesta en función de diagramas de clases y estándares del diseño, generalidades y modelo de implementación, así como pruebas realizadas.

Se cuenta además con un glosario de términos, referencias bibliográficas, bibliografía y los anexos.

Capítulo 1: FUNDAMENTOS TEÓRICOS DE LA INVESTIGACIÓN.

1.1 Introducción

En el presente capítulo se realiza un análisis del objeto de estudio para caracterizar el control y seguimiento por niveles de acceso dentro de los procesos del Sistema de Gestión de Datos Geológicos. Se tratan los aspectos principales que conforman la fundamentación teórica de la investigación y que crean una visión de la misma. Se analizan algunas soluciones similares para obtener una perspectiva de la utilización de funcionalidades afines a las que se propone en la presente investigación.

1.2 Conceptos asociados al dominio del problema

Con el propósito de una mejor comprensión de los temas que serán tratados en este capítulo, directamente vinculados con el objeto de estudio de la investigación, se describen a continuación un grupo de conceptos relacionados al dominio del problema.

Control de Acceso: Es el proceso de conceder permisos a usuarios de acceder a objetos, información, datos como ficheros o impresoras en la red. Está basado en tres conceptos fundamentales: identificación, autenticación y autorización. Incluye autenticar la identidad de los usuarios y autorizar el acceso a datos o recursos. Son necesarios para proteger la confidencialidad, integridad y disponibilidad de los objetos, y por extensión, de la información que contienen, pues permiten que los usuarios autorizados accedan solo a los recursos que ellos requieren para realizar sus tareas. (Mondragón Sotelo, 2006)

Identificación: Es la acción por parte de un usuario de presentar su identidad a un sistema, usualmente se usa un identificador de usuario. Establece además, que el usuario es responsable de las acciones que lleve a cabo en el sistema. Esto está relacionado con los registros de auditorías que permiten guardar las acciones realizadas dentro del sistema y rastrearlas hasta el usuario autenticado. (Mondragón Sotelo, 2006)

Autenticación: Es el proceso de descubrir y comprobar la identidad de un usuario mediante el examen de sus credenciales y la validación de las mismas consultando a una autoridad determinada. Actualmente se utilizan una gran variedad de mecanismos de autenticación. (Mondragón Sotelo, 2006)

Autorización: Es el proceso que determina si el usuario o proceso previamente identificado y autenticado tiene permitido el acceso a los recursos. (Mondragón Sotelo, 2006)

Auditoría: Es un registro cronológico de los eventos relevantes a la seguridad de un sistema. Este registro puede luego examinarse para reconstruir un escenario en particular. (Mondragón Sotelo, 2006)

Confidencialidad: La información o los activos informáticos son accedidos solo por las personas autorizadas. Sería fácil mantener la confidencialidad de un sistema si nadie tuviera acceso a él, pero este tendría una disponibilidad nula. (Mondragón Sotelo, 2006)

Integridad: La información que está en correspondencia con la realidad solo puede ser modificada por las personas autorizadas y de la forma autorizada. (Mondragón Sotelo, 2006)

Disponibilidad: La información solo puede ser accedida por las personas autorizadas en el momento requerido. (Mondragón Sotelo, 2006)

Seguridad de la Información: Se entiende por seguridad de la información según ISO 27000 a todas aquellas medidas preventivas y reactivas del hombre, de las organizaciones y de los sistemas tecnológicos que permitan resguardar y proteger la información buscando mantener la confidencialidad, la autenticidad e integridad de la misma.

El concepto de seguridad de la información no debe ser confundido con el de seguridad informática, ya que este último sólo se encarga de la seguridad en el medio informático, pudiendo encontrar información en diferentes medios o formas. Para el hombre como individuo, la seguridad de la información tiene un efecto significativo respecto a su privacidad, la que puede cobrar distintas dimensiones dependiendo de la cultura del mismo.

Log: Un log es un registro oficial de eventos durante un rango de tiempo en particular. Para los profesionales en seguridad informática es usado para registrar datos o información sobre: quién, qué, cuándo, dónde y por qué: un evento ocurre. Por lo tanto se puede decir que un log es una evidencia digital. (VERONICA MACIAS, 2011)

1.3 Objeto de Estudio

1.3.1 Descripción General

Mecanismos de autenticación e identificación

Los métodos de autenticación se suelen dividir en tres grandes categorías en función de lo que utilizan para la verificación de identidad:

- Algo que el usuario conoce.
- Algo que el usuario posee.
- Una característica física del usuario o un acto involuntario del mismo.

Esta última categoría se conoce con el nombre de autenticación biométrica, es fácil ver ejemplos de cada uno de estos tipos de autenticación: una contraseña o frase es algo que el usuario conoce y el resto de personas no, una tarjeta de identidad es algo que el usuario lleva consigo, la huella dactilar es una característica física del usuario, y un acto involuntario podría considerarse que se produce al firmar. Por supuesto, un sistema de autenticación puede (y debe, para incrementar su confiabilidad) combinar mecanismos de diferentes tipos, como en el caso de una tarjeta de crédito junto al PIN (Personal Identification Number o Número de Identificación Personal en español) a la hora de utilizar un cajero automático.

Sistemas basados en algo conocido: contraseñas

El modelo de autenticación más básico consiste en decidir si un usuario es quien dice ser simplemente basándonos en una prueba de conocimiento que a priori sólo ese usuario puede superar; y desde Alí Babá y su “Ábrete Sésamo” hasta los más modernos sistemas, esa prueba de conocimiento no es más que una contraseña que en principio es secreta. Evidentemente, esta aproximación es la más vulnerable a todo tipo de ataques, pero también la más barata, por lo que se convierte en la técnica más utilizada en entornos que no precisan de una alta seguridad. En todos los esquemas de autenticación basados en contraseñas se cumple el mismo protocolo: (Villalón Huerta, 2002)

Las entidades (generalmente dos) que participan en la autenticación acuerdan una clave, clave que han de mantener en secreto si desean que la autenticación sea fiable. Cuando una de las partes desea autenticarse ante otra se limita a mostrarle su conocimiento de esa clave común, y si ésta es correcta se otorga el acceso a un recurso.

Sistemas basados en algo poseído: tarjetas inteligentes

Desde un punto de vista formal, una tarjeta inteligente (o smartcard) es un dispositivo de seguridad del tamaño de una tarjeta de crédito, resistente a la adulteración, que ofrece funciones para un almacenamiento seguro de información y también para el procesamiento de la misma en base a tecnología VLSI[1]. En la práctica, las tarjetas inteligentes poseen un chip empotrado en la propia tarjeta que puede implementar un sistema de ficheros cifrado y funciones criptográficas, y además puede detectar activamente intentos no válidos de acceso a la información almacenada; este chip inteligente, es el que las diferencia de las simples tarjetas de crédito, que solamente incorporan una banda magnética donde va almacenada cierta información del propietario de la tarjeta. (Villalón Huerta, 2002)

Las ventajas de utilizar tarjetas inteligentes como medio para autenticar usuarios son muchas frente a las desventajas; se trata de un modelo ampliamente aceptado entre los usuarios, rápido, y que incorpora hardware de alta seguridad tanto para almacenar datos como para realizar funciones de cifrado. Además, su uso es factible tanto para controles de acceso físico como para controles de acceso lógico a los hosts, y se integra fácilmente con otros mecanismos de autenticación como las contraseñas; y en caso de desear bloquear el acceso de un usuario, no se tiene más que retener la tarjeta cuando se introduzca en el lector o marcarla como inválida en una base de datos (por ejemplo, si se equivoca varias veces al teclear su PIN, igual que sucede con una tarjeta de crédito normal). Como principal inconveniente de las smartcards se puede citar el coste adicional que supone para una organización el comprar y configurar la infraestructura de dispositivos lectores y las propias tarjetas; aparte, que un usuario pierda su tarjeta es bastante fácil, y durante el tiempo que no disponga de ella o no puede acceder al sistema, o se ha de establecer reglas especiales que pueden

[1] **VLSI** es la sigla en inglés de **V**ery **L**arge **S**cale **I**ntegration, integración en escala muy grande, Tecnología de chips introducida en 1975 y que produce un nivel de integración de circuitos de entre 20.000 y 900.000 puertas lógicas por microprocesador.

comprometer nuestra seguridad (y por supuesto se ha de marcar como tarjeta inválida en una base de datos central, para que un potencial atacante no pueda utilizarla). También la distancia lógica entre la smartcard y su poseedor - simplemente se puede fijar en que la tarjeta no tiene una interfaz para el usuario - puede ser fuente de varios problemas de seguridad. (Villalón Huerta, 2002)

Sistemas de autenticación biométrica

A pesar de la importancia de la criptología en cualquiera de los sistemas de identificación de usuarios vistos, existen otra clase de sistemas en los que no se aplica esta ciencia, o al menos su aplicación es secundaria. Es más, parece que en un futuro no muy lejano estos serán los sistemas que se van a imponer en la mayoría de situaciones en las que se haga necesario autenticar un usuario: son más amigables para el usuario (no va a necesitar recordar contraseñas o números de identificación complejos, y, como se suele decir, el usuario puede olvidar una tarjeta de identificación en casa, pero nunca se olvidará de su mano o su ojo) y son mucho más difíciles de falsificar que una simple contraseña o una tarjeta magnética; las principales razones por la que no se han impuesto ya en nuestros días es su elevado precio, fuera del alcance de muchas organizaciones, y su dificultad de mantenimiento. (Villalón Huerta, 2002)

Estos sistemas son los denominados biométricos, basados en características físicas del usuario a identificar. El reconocimiento de formas, la inteligencia artificial y el aprendizaje son las ramas de la informática que desempeñan el papel más importante en los sistemas de identificación biométricos; la criptología se limita aquí a un uso secundario, como el cifrado de una base de datos de patrones retinales, o la transmisión de una huella dactilar entre un dispositivo analizador y una base de datos. La autenticación basada en características físicas existe desde que existe el hombre y, sin darnos cuenta, es la que más utilizan las personas en su vida cotidiana: a diario se identifican a personas por los rasgos de su cara o por su voz. Obviamente aquí el agente reconocedor lo tiene fácil porque es una persona, pero en el modelo aplicable a redes o sistemas de seguridad el agente ha de ser un dispositivo que, basándose en características del sujeto a identificar, le permita o deniegue acceso a un determinado recurso. (Villalón Huerta, 2002)

Los dispositivos biométricos tienen tres partes principales; por un lado, disponen de un mecanismo automático que lee y captura una imagen digital o analógica de la característica a analizar. Además disponen de una entidad para manejar aspectos como la compresión, almacenamiento o comparación de los datos capturados

con los guardados en una base de datos (que son considerados válidos), y también ofrecen una interfaz para las aplicaciones que los utilizan. Entre los más conocidos están los de verificación de voz, verificación de escritura, verificación de huellas, verificación de patrones oculares (Retina e Iris) y verificación de la geometría de la mano.

Auditoría

El control de la **auditoría** en aplicaciones en una entidad determinada es de vital importancia. Comúnmente, las aplicaciones presentan una gran vulnerabilidad lo que provoca un posible ataque y el mismo no necesariamente depende de la plataforma y tecnologías utilizadas. Estas vulnerabilidades pueden aparecer por un error en la codificación del sistema o simplemente un mal diseño de la aplicación. Para evitar la existencia de vulnerabilidades se implementa la auditoría donde para tener un buen monitoreo de lo que sucede en las aplicaciones de la empresa es recomendable hacer un resumen o reporte, ya sea diario, semanal, o según el tiempo que se estime, de las acciones realizadas sobre las mismas siempre recogiendo algunos conceptos que son de mucho interés como por ejemplo:

Logs: Son ficheros que almacenan información, generalmente esta información registra el acceso que realiza un usuario a una aplicación.

En estos ficheros se almacena:

- Páginas visitadas (Qué): Páginas de la aplicación que son visitadas por un usuario determinado o acción realizada.
- IP del visitante (Desde): Dirección de donde proviene la conexión, o sea, dirección de la PC donde está ubicado el usuario que accede a la aplicación.
- Fecha y hora de la conexión (Cuándo): Datos del momento de la conexión a la aplicación (fecha y hora, etc.).
- Sistemas (Dónde): Aplicación en la cual realizó acciones el usuario.
- Datos del usuario (Quién): Esto solo sería en el caso de que el usuario este previamente registrado en la aplicación, se registraría el nombre y otros datos.

1.3.2 Descripción actual del dominio del problema

El SGDG que se desarrolla actualmente estará integrado por 10 módulos, de los cuales 7 fueron concebidos durante la primera versión del sistema (Búsqueda Referativa, Registro Minero, Registro Petrolero, Inventario Nacional de Recursos y Reservas de Petróleo y Gas, Inventario Nacional de Recursos de Aguas Minerales, Nomencladores, Inventario de Minerales Sólidos) y a los cuáles se les agregará nuevas funcionalidades, los 3 restantes (Catálogo de Pozos, Estadística Minera e Identificación de Minas Abandonadas) constituyen áreas que aún no han sido informatizadas en la entidad.

En la actualidad la ONRM, utiliza como complemento fundamental de la ejecución de sus principales procesos el SGDG primera versión. Este sistema presenta algunas deficiencias que atentan contra la total satisfacción del cliente, y están relacionadas principalmente con la seguridad. Como problemas más sobresalientes se pueden mencionar la redundancia y duplicidad de la información, provocados por el manejo descentralizado de esta y la desvinculación entre las áreas de la entidad, a pesar de necesitar unas de otras; la gestión de roles, no se permite la creación de nuevos roles, ni la eliminación o actualización de roles establecidos en el sistema; la misma situación la poseen las áreas, donde tampoco se permite la creación de nuevas áreas o la eliminación y actualización de estas en caso de necesitarlo.

Producto del gran volumen de información que se gestiona en el día en la ONRM y el nivel de precisión que esta información requiere, se hace necesario el uso de herramientas informáticas que ayuden a monitorizar el manejo diario de toda esta información, al no realizarse una monitorización de lo que sucede en el sistema (implementación de logs), esto provoca la existencia de vulnerabilidades y por tanto posibles ataques a la aplicación.

1.3.3 Situación Problemática

Con la promoción de la Ley de Minas (Ley 76) en el año 1995, la ONRM se convierte en la entidad minero-petrolera responsable de salvaguardar el conocimiento geológico, así como administrar el aprovechamiento racional de los recursos minerales del país y la zona económica de Cuba.

Actualmente la entidad está estructurada en distintas áreas de dirección encargada de desarrollar importantes actividades concernientes a las aguas minerales, los minerales sólidos, el balance del petróleo, la gestión de

las actividades de explotación y exploración de los recursos en la isla, entre otras disímiles. Por tales motivos, la información que se gestiona en dicha entidad es de vital importancia para el país y requiere determinados niveles de clasificación y acceso.

La ONRM cuenta con el SGDG; una solución informática tipo web que automatiza los procesos de gestión más importantes que se desarrollan en cada área de la entidad, en el cual existe la necesidad de buscar alternativas que permitan garantizar el correcto acceso a la información de acuerdo al nivel de autorización de la persona que la gestiona, así como realizar un seguimiento de la gestión que tenga dicha información. Este sistema trabaja con información relevante y clasificada, por tal motivo se tiene la necesidad de un sistema de seguridad que establezca un control más riguroso en la gestión de la información de la que existe actualmente influyendo de manera decisiva en la integridad de los datos.

1.4 Análisis de soluciones existentes

El SGDG en su primera versión posee un módulo de seguridad, este no constituye un módulo del negocio propiamente, su función es proveer el acceso a la aplicación mediante la autenticación usuario/contraseña; adicionar o eliminar usuarios del sistema y controlar y establecer los determinados permisos de manera que cada cual puede acceder solo a la información que le es permitida. Este módulo está implementado en Symfony 1.1.4, la actual versión del SGDG es la 2.0 que se está construyendo en Symfony 2.2 el cual supone un cambio radical tanto en arquitectura interna como en filosofía de trabajo respecto a sus versiones anteriores lo que implica un cambio de tecnología.

1.4.1 Seguridad de los principales marcos de trabajo

En la actualidad al iniciar el desarrollo de un sistema se hace un estudio del estado de las principales soluciones que existen para solucionar la problemática existente. Los frameworks o marcos de trabajos integran un conjunto de componentes que resuelven muchos de los escenarios que pueden encontrarse en el transcurso de un desarrollo, esto permite agilizar el proceso y obtener resultados en breves períodos de tiempos. Como son soluciones muy genéricas, en la mayoría de los casos no abarcan todos los posibles contextos en el que se va a desempeñar una aplicación.

Symfony1

Symfony es un completo marco de trabajo diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. (Potencier, y otros, 2008)

Symfony cuenta con muchos plugin[2], un grupo de ellos es utilizado para gestionar la seguridad de las aplicaciones, dentro de este grupo el más usado, seguro y eficaz es: sfGuardPlugin.

SfGuardPlugin

En toda aplicación dinámica, siempre está presente la gestión de usuarios, permisos y roles. En Symfony esto puede resultar muy sencillo gracias a SfGuard. Este plugin gestiona de manera muy sencilla usuarios, roles, permisos y logins.

Básicamente, la autenticación y la seguridad en una aplicación es limitar el acceso a partes de la misma. Significa que los usuarios tendrán que iniciar una sesión (autenticación) para acceder a ciertas áreas (seguridad). Diferentes usuarios pueden tener diferentes privilegios (autorización). De esta manera se asegura una aplicación para diferentes tipos de usuarios, SfGuard Plugin brinda el modelo (usuario, grupo y objetos de permisos).

Zend Framework

En su nivel más simple, *Zend Framework* es una biblioteca de componentes escritos en *PHP5*, para facilitar el desarrollo de sitios web. Como está basada en *PHP5* (5.1.4 es la versión mínima necesaria), eso significa que es completamente *Orientada a Objetos*. (Rigazzi, 2008)

[2] Un **complemento** (o **plug-in** en inglés) Pequeño programa que añade alguna función a otro programa, habitualmente de mayor tamaño. Un programa puede tener uno o más conectores. Son muy utilizados en los programas navegadores para ampliar sus funcionalidades.

Zend está formado por muchos componentes, estos están divididos en grupos, donde uno de estos se encarga de gestionar la seguridad, los componentes del mismo son:

Zend_Auth permite comprobar y guardar credenciales de usuario de distintas maneras: utilizando la *Base de Datos,* el *método Diguest de Apache,* o *autenticación http simple.* Provee un adaptador de Interfaz para personalizar los mecanismos de autenticación, además almacenamiento automático de identidad para una fácil personalización. Es simple y extensible.

A su vez **Zend_Session** trabaja como un administrador de datos de sesión, al igual que en **PHP,** solo que ofrece algo de valor agregado.

El componente **Zend_ACL** es una implementación de Listas de Control de Acceso (ACL[3]) en PHP que permite asignar roles y permisos a usuarios o grupos de usuarios en la aplicación. Soporta herencia de roles y recursos. También soporta el control de acceso condicional basado en una interfaz de declaraciones.

Finalmente el componente **Zend_Log** que permite realizar un log de acciones en diferentes medios como la consola de *PHP,* archivos y base de datos mediante el anexo de varios redactores.

Code Igniter

CodeIgniter es un framework para desarrollo de aplicaciones - un conjunto de herramientas - para personas que construyen sitios web usando PHP. Su objetivo es permitirle desarrollar proyectos mucho más rápido que lo que podría hacer si escribiera el código desde cero, proporcionando un rico conjunto de bibliotecas para tareas comunes, así como una interfaz sencilla y una estructura lógica para acceder a esas bibliotecas. CodeIgniter le permite enfocarse creativamente en su proyecto al minimizar la cantidad de código necesaria para una tarea dada. (EllisLab, 2011)

[3] *La **Lista de Control de Acceso o ACL*** (del inglés, **Access Control List**) es un concepto de seguridad informática usado para fomentar la separación de privilegios. Es una forma de determinar los permisos de acceso apropiados a un determinado objeto, dependiendo de ciertos aspectos del proceso que hace el pedido.

CodeIgniter gestiona la seguridad de diferentes formas. Es justamente restrictivo sobre que caracteres en las cadenas URI para ayudar a minimizar la posibilidad de ataque utilizando esta vía. Las URL sólo pueden contener lo siguiente: Texto alfanumérico, "tilde"(~), punto (.), dos puntos (:), guión bajo (_), Guión (-).

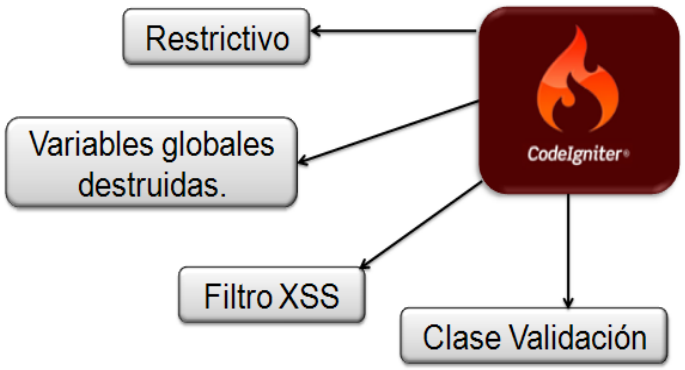


Figura 1: Seguridad en CodeIgniter

La Figura anterior muestra la arquitectura de seguridad que presenta este marco de trabajo. La misma solo implementa de forma muy sencilla un conjunto de restricciones, validaciones y filtros que no garantizan la seguridad en un entorno que requiera de una seguridad estricta. Las configuraciones que permite se realizan en XML, el esfuerzo para incorporar robustez es muy alto por lo que no es factible reutilizar este marco de trabajo.

Los datos GET son simplemente anulados por *CodeIgniter* ya que el sistema utiliza segmentos URI en vez de las tradicionales query strings de URL (a menos que la opción query string esté habilitada en su archivo config). El arreglo global GET es destruido por la clase de Entrada (input) durante la inicialización del sistema. (EllisLab, 2011)

Durante la inicialización del sistema, todas las variables globales son destruidas, excepto aquellas encontradas en los arreglos *$_POST* y *$_COOKIE*. La rutina de eliminación es efectivamente lo misma que register_globals = off. (EllisLab, 2011)

KumbiaPHP

KumbiaPHP es un framework para aplicaciones web libre escrito en *PHP5*. Basado en las prácticas de desarrollo web como ***DRY*** [4]y el principio ***KISS*** [5]para software comercial y educativo. Kumbia fomenta la velocidad y eficiencia en la creación y mantenimiento de aplicaciones web, reemplazando tareas de codificación repetitivas por poder, control y placer. (Tejeda, y otros, 2007)

La implementación de la seguridad usando *KumbiaPHP* se lleva a cabo con el uso de las listas de control de acceso *(ACL).* Cada lista ACL contiene una lista de Roles, unos recursos y unas acciones de acceso. Kumbia posee un componente para el manejo de autenticación de usuario (Auth) por diversos métodos, esto nace con la idea de seguir facilitando el desarrollo de aplicaciones en base al framework, solo con crear una instancia de la clase Auth y por el constructor de la clase se le pasan los parámetros necesarios para que entienda cual será el método de autenticación, la clase Auth, realiza el proceso de conectarse a la BD y verificar las condiciones que le pasamos por el constructor ayudando mucho al programador. (Tejeda, y otros, 2009)

Kumbia también tiene la clase Session que proporciona un acceso orientado a objetos a datos de sesión. Los datos de sesión son usualmente utilizados para mantener valores a través de toda la ejecución de una sesión de aplicación. El caso más común es mantener alguna variable para indicar si un usuario está logueado en el sistema o por ejemplo mantener la dirección de la cual viene un usuario antes de solicitarle autenticación y poderlo redireccionar después a ella de nuevo. Kumbia serializa automáticamente los datos de sesión. Esto significa que puedes guardar con seguridad objetos y en algunos casos valores de recurso (resource) como conexiones a bases de datos. (Tejeda, y otros, 2009)

Los Loggers permiten crear *logs* para almacenar información que pueda ser utilizada para registrar transacciones, revisar movimientos, hacer un debug, etc. Para la creación de estos se puede utilizar la clase Logger de Kumbia. El constructor de la clase Logger recibe el nombre del archivo donde se guardará el *log*,

[4] El principio **No te repitas** (en inglés **Don't Repeat Yourself** o **DRY**, también conocido como **Una vez y sólo una**) es una filosofía de definición de procesos que promueve la reducción de la duplicación especialmente en informática.

[5] El principio **Manténgalo Corto** y **Sencillo** (en inglés, **Keep It Short and Simple** o **KISS**) es aquel que recomienda el desarrollo empleando partes sencillas, comprensibles y con errores de fácil detección y corrección, rechazando lo enrevesado e innecesario en el desarrollo de sistemas complejos en ingeniería.

este se almacena en el directorio ***logs/***, si no se especifica el nombre del archivo creado es logYYYYMMDD.txt, esta clase contiene varios métodos que hacen más fácil el trabajo con él *log*. (Tejeda, y otros, 2009)

Sistema de gestión integral de seguridad Acaxia

Acaxia implementa cinco procesos para convertirse en una solución integral. La Figura siguiente refleja los conceptos identificados. (Gómez Baryolo, y otros, 2011)

Figura 2: Procesos de la Seguridad en ACAXIA.

El sistema de gestión integral de seguridad (Acaxia) es capaz de gestionar la seguridad de otros sistemas que se suscriban a él de forma centralizada. Para ello se controla la información de cada uno de ellos que serán provistos de seguridad o auditados en un momento determinado. (Gómez Baryolo, y otros, 2011)

De cada uno de estos sistemas se controlan las funcionalidades que brindan y de cada una de ellas las acciones que se realizan. Una vez registrada la estructura de los sistemas se procede a la creación de roles, donde se le asignarán los permisos a los diferentes niveles de la misma, estos roles serán asignados a los usuarios dentro de una o varias entidades, de esta forma si la seguridad se maneja de forma centralizada el usuario podrá acceder a dichas entidades desde cualquier punto del país. Los usuarios cuentan con un perfil configurable donde puede seleccionar el tema, el idioma, el tipo de escritorio para su entorno de trabajo, además de estos datos se recoge el dominio de entidades al cual tiene acceso, servidor de autenticación si se

cuenta con uno, rango de direcciones IP desde donde puede conectarse, usuario y contraseña. (Gómez Baryolo, y otros, 2011)

Valoración de las soluciones estudiadas

Los componentes de seguridad estudiados presentan soluciones particulares para escenarios muy específicos. La efectividad de las políticas de control de acceso están condicionadas por el correcto funcionamiento de sus tres procesos: identificación y autenticación, autorización y auditoría. Algunos de los modelos más avanzados según la bibliografía consultada, solo cubren los dos primeros procesos del control de acceso, aunque centran su fortaleza en uno de ellos. Si no se concibe de forma integrada los tres procesos del control de acceso y se especifican las entradas y salidas de cada uno de ellos, puede traer consigo problemas de integración entre los sistemas y pérdida de datos importantes para la auditoría, el módulo de seguridad es integrado a una aplicación general para la cual está hecho, esta aplicación está desarrollada en Symfony 2.2 por lo cual para lograr la integración de dicho módulo se hace necesario la construcción del mismo en esta misma versión conservando la compatibilidad.

La estructura interna de Acaxia redirecciona todas las peticiones a su sistema de ruta para determinar los permisos concedidos al usuario y si este puede acceder a determinada petición; luego de ello la petición se pasa al SGDG para su ejecución. Para este funcionamiento se debe registrar en Acaxia todas las rutas (URL) y patrones de rutas con las que cuenta el SGDG para luego asignarle los permisos. Esto supone un esfuerzo extra ya que el SGDG actualmente está en desarrollo y se piensa elaborar 10 módulos lo que acarrea a múltiples rutas.

Por otra parte en la interfaz de usuario el SGDG cuenta con una arquitectura de información, colores y fuente propia destinada a representar visualmente a la Oficina Nacional de Recursos Minerales. La integración con Acaxia exige que la gestión de usuario y autenticación en el sistema se realice en la parte de Acaxia, para lo cual se debe invertir tiempo en modificar su diseño para cumplir con los estándares de SGDG. Otra solución sería la Integración como módulo pero no es posible (al menos de forma trivial) hacer funcionar Acaxia como parte del SGDG. Es decir que, integrar Acaxia para que sea un módulo más dentro del SGDG, cumpliendo con los estándares de diseño y esquema de presentación, no se puede realizar de forma trivial. Al contrario, Acaxia

ha sido diseñado para ser utilizado desde los primeros momentos de desarrollo, donde el sistema en cuestión se debe integrar en Acaxia.

1.5 Conclusiones Parciales

Para la realización del presente trabajo se hizo un estudio de la lógica implementada en la primera versión, ya que no se puede utilizar dicha versión por lo descrito anteriormente se llegó a la conclusión de que esta lógica sirve de base para el desarrollo del nuevo módulo.

Se llega a la conclusión de utilizar los sistemas basados en contraseña pues las ventajas de utilizar este tipo de autenticación es que es sencillo, barato, y está altamente probado. Es sencillo porque el usuario no tiene que utilizar ningún dispositivo externo para autenticarse en el sistema. Es barato debido a que no se necesita de un coste adicional a la hora de entrar a un sistema y poner su contraseña, pero en cambio, los sistemas que utilizan objetos para poder autenticarse, necesitan de un gasto extra para utilizar un identificador de objetos. Los sistemas biométricos son mucho más novedosos y se están desarrollando a gran velocidad aunque tienen desventajas que retardan su desarrollo como el precio de los equipos de captación, conceptos éticos y poca costumbre de utilización. Está probado que los sistemas basados en contraseñas son los más utilizados en el mundo, porque con diferentes variantes son aplicados en casi todos los aspectos de la seguridad de la información.

Capítulo 2: TENDENCIAS Y TECNOLOGÍAS ACTUALES A UTILIZAR

2.1 Introducción

En este capítulo se realiza un análisis del estado del arte de los métodos, técnicas y herramientas empleadas para el modelado del subsistema así como de las tecnologías que se emplean en el desarrollo y documentación de la aplicación.

2.2 Metodologías de desarrollo

Una metodología es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información, las mismas están formadas por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo. (Metodologías ágiles y formales o robustas, 2009)

Una metodología tiene como objetivos: Obtener mejores aplicaciones, un mejor Proceso de Desarrollo que identifique salidas (o productos intermedios) de cada fase de forma que se pueda planificar y controlar los proyectos y un proceso estándar en la organización. (Ruiz, 2010)

La elección de una metodología de desarrollo de software, depende de varios factores tales como: tipo de proyecto o sistema a desarrollar, herramientas a utilizar en el mismo, las personas que lo van a desarrollar, los recursos disponibles y las especificaciones solicitadas por el cliente para el que se esté trabajando. Ya que estos factores no se muestran de la misma forma en todos los sistemas, se ha hecho muy difícil establecer modelos de desarrollo estándares a seguir, ya que el mismo debe ser un proceso adaptable y configurable, en correspondencia con el sistema a desarrollar.

Se selecciona la metodología RUP para la realización del presente módulo puesto que la misma dota de una amplia documentación a todo el proceso de desarrollo del software, teniendo en cuenta que el personal del proyecto cambia con regularidad se hace necesario que quede bien documentada cada fase de desarrollo por las cuales pasa el módulo.

2.2.1 El Proceso Unificado de Desarrollo de Software (RUP)

En primer lugar, el Proceso Unificado es un proceso de desarrollo de software. Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software. Sin embargo, el proceso unificado es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos. (Jacobson, y otros, 2000)

RUP es un proceso de desarrollo de software, una forma disciplinada de asignar tareas y responsabilidades en una empresa de desarrollo (quién hace qué, cuándo y cómo).Posee como objetivo principal asegurar la producción de software de calidad dentro de plazos y presupuestos predecibles. Sus características esenciales son: **Dirigido por casos de uso**; los casos de usos representan el hilo conductor que orienta las actividades de desarrollo. Se centra en la funcionalidad que el sistema debe poseer para satisfacer las necesidades del usuario. **Centrado en la arquitectura**; establecer una arquitectura candidata al inicio es un paso muy importante, pues será ella la que guíe el desarrollo del sistema. Esto permitirá el desarrollo en paralelo, minimizar la repetición de trabajos e incrementar la probabilidad de reutilización de componentes. **Iterativo (mini-proyectos) e incremental (versiones)**; el desarrollo iterativo brinda la posibilidad de que los elementos sean integrados progresivamente, facilita el rehúso y resulta un producto más robusto pues los errores se van corrigiendo en cada iteración. (Jacobson, y otros, 2000)

2.3 Lenguajes de modelado y desarrollo

2.3.1 El Lenguaje Unificado de Modelado (UML)

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un

acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos. (Jacobson, y otros, 2000)

Entre los rasgos principales que han contribuido a hacer de UML el estándar de la industria en la actualidad, se puede mencionar: (Rumbaugh, y otros, 2000)

- Permite modelar sistemas utilizando técnicas orientadas a objetos.
- Adecuado a las necesidades de conectividades actuales y futuras.
- Es un lenguaje muy expresivo que cubre las vistas necesarias para desarrollar y luego desplegar los sistemas.
- Ampliamente utilizado por la industria del software.
- Reemplaza a decenas de notaciones empleadas por otros lenguajes.
- Modela estructuras complejas.
- Comportamiento del sistema: casos de usos, diagramas de secuencia, de colaboración, que sirve para evaluar el estado de las máquinas.

2.3.2 PHP como lenguaje de Programación

PHP (acrónimo de Hypertext Preprocessor) es un lenguaje "del lado del servidor" (esto significa que PHP funciona en un servidor remoto que procesa la página Web antes de que sea abierta por el navegador del usuario) especialmente creado para el desarrollo de páginas Web dinámicas. Puede ser incluido con facilidad dentro del código HTML, y permite una serie de funcionalidades tan extraordinarias que se ha convertido en el favorito de millones de programadores en todo el mundo. Combinado con la base de datos MySQL, es el lenguaje estándar a la hora de crear sitios de comercio electrónico o páginas Web dinámicas. (Gallego Vázquez, 2003)

Entre sus características fundamentales están:

Alto rendimiento, mediante el uso de un único servidor puede servir millones de acceso al día. **Interfaces para diferentes sistemas de base de datos**, dispone de una conexión propia a todos los sistemas de bases de datos además de MySQL, puede conectarse directamente a las bases de datos de PostgreSQL, Oracle, dmb, FilePro, HyperWave, Informix, IterBase y Sybase, entre otras. PHP5 también cuanta con una interfaz SQL incorporada a un archivo plano, denominada SQLite. **Bibliotecas Incorporadas,** como se ha diseñado para su uso en la Web, PHP incorpora una gran cantidad de funciones integradas para realizar útiles tareas relacionadas con la Web. Puede generar imágenes GIF al instante, establecer conexiones a otros servicios de red, enviar correos electrónicos, trabajar con cookies y generar documentos PDF, todo con unas pocas líneas de código. **Bajo coste,** es gratuito, se puede descargar la última versión de http://www.php.net cuando se desee sin coste alguno. **Facilidad de aprendizaje y uso,** la sintaxis de PHP se basa en otros lenguajes de programación, principalmente en C y Perl. Si ya se conoce C o Perl, o un lenguaje de tipo C como C++ o java, no se tardará nada en utilizar PHP de manera productiva. (Welling, y otros, 2006)

Compatibilidad con el enfoque orientado a objetos, la versión 5 de PHP cuenta con completas funciones orientadas a objetos con la misma sintaxis de Java o C++ como por ejemplo herencia, atributos, métodos privados y protegidos, clases y métodos abstractos, interfaces, constructores y destructores, entre otras. **Portabilidad**, está disponible para una gran cantidad de sistemas operativos diferentes, se puede escribir código PHP en todos los sistemas operativos gratuitos del tipo Unix, como Linux y FreeBSD, versiones comerciales de Unix como Solaris e IRIX o en las diferentes versiones de Microsoft Windows. El código funcionará sin necesidad de aplicar ninguna modificación a los diferentes sistemas que ejecute PHP. **Disponibilidad de código abierto,** se dispone de acceso al código fuente de PHP. A diferencia de los productos comerciales y de código cerrado, si se desea modificar algo o agregar un elemento al programa se puede hacer con total libertad. No se necesitará esperar a que el fabricante publique parches, ni preocuparse porque cierre sus puertas o decida abandonar el producto. **Disponibilidad de asistencia técnica,** Zend Technologies (www.zend.com), la empresa responsable del motor e PHP, basa su desarrollo en la oferta de asistencia técnica y software de forma regular. (Welling, y otros, 2006)

2.4 Sistemas Gestores de Base de Datos

Sistema Gestor de Bases de Datos (SGBD) es un software que proporciona servicios para la creación, el almacenamiento, el procesamiento y la consulta de la información almacenada en base de datos de forma segura y eficiente. Un SGBD actúa como un intermediario entre las aplicaciones y los datos. (Sicilia, 2008)

El objetivo fundamental de un SGBD consiste en suministrar al usuario las herramientas que le permitan manipular, en términos abstractos, los datos, o sea, de forma que no le sea necesario conocer el modo de almacenamiento de los datos en la computadora, ni el método de acceso empleado. (Mato García, 1999)

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales. (rafaelma, 2010)

PostgreSQL utiliza un modelo cliente/servidor y usa *multiprocesos* en vez de *multihilos* para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Algunas de las características más importantes y soportadas por PostgreSQL: (rafaelma, 2010)

- Funciones/procedimientos almacenados en numerosos lenguajes de programación, entre otros PL/pgSQL (similar al PL/SQL de Oracle), PL/Perl, PL/Python y PL/Tcl
- Replicación asincrónica/sincrónica
- Unicode[6]
- Multi-Version Concurrency Control (MVCC en español Acceso concurrente multiversión)
- Múltiples métodos de autentificación
- Acceso encriptado vía SSL[7]

[6] **Unicode** es un estándar de codificación de caracteres diseñado para facilitar el tratamiento informático, transmisión y visualización de textos de múltiples lenguajes y disciplinas técnicas, además de textos clásicos de lenguas muertas.

[7] ***Secure Sockets Layer*** (***SSL***; en español capa de conexión segura) es un protocolo criptográfico que proporciona comunicaciones

- Completa documentación
- Licencia BSD
- Disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows 32/64bit.
- Numerosos tipos de datos y posibilidad de definir nuevos tipos. Además de los tipos estándares en cualquier base de datos, tiene disponibles, entre otros, tipos geométricos, de direcciones de red, de cadenas binarias, UUID, XML, matrices, etc.

- APIs[8] para programar en C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, PHP, Lisp, Scheme, Qt y muchos otros.

2.5 Framework de desarrollo

Un framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, un framework proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, un framework facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas. (Potencier, y otros, 2008)

2.5.1 Symfony2

Symfony es un completo framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador

---

seguras por una red, comúnmente Internet.

[8] **API** (Interfaz de Programación de Aplicaciones)

dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web. (Potencier, y otros, 2008)

Symfony está desarrollado completamente con 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 SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows. A continuación se muestran algunas de sus características. (Potencier, y otros, 2008)

Symfony2 es la versión más reciente de Symfony, el popular framework para desarrollar aplicaciones PHP. Se anunció por primera vez a principios de 2009 (http://www.symfony.es/2009/03/06/asi-seran-las-novedades-de-symfony-20/) y supone un cambio radical tanto en arquitectura interna como en filosofía de trabajo respecto a sus versiones anteriores. (Eguiluz, 2011)

Symfony2 ha sido ideado para exprimir al límite todas las nuevas características de PHP 5.3 y por eso es uno de los frameworks PHP con mejor rendimiento. Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en tu proyecto. (Eguiluz, 2011)

Características: (Belmonte Ruiz, 2011)

Alto rendimiento: Symfony2 ha sido desarrollado teniendo en cuenta el rendimiento como mayor prioridad, por lo que es uno de los frameworks más rápidos. Hasta 3 veces más rápido que Symfony 1.4 o Zend Framework 1.10 y consume la mitad de la memoria.

Usabilidad avanzada: Es un framework fácil de utilizar gracias a que cuenta con una API de desarrollo muy intuitiva.

Extensible: Symfony2 se construye a base de bundles. Un bundle es un conjunto estructurado de archivos que implementan una característica única y que puede ser fácilmente compartido con otros desarrolladores. En Symfony2 todo es un bundle. Los bundles permiten una forma limpia de configurar y personalizar el sistema.

Flexible: Gracias a que Symfony2 cuenta con un micro-kernel basado en un contenedor de inyección de dependencia y un manejador de eventos es muy fácil configurarlo a voluntad.

Construido para desarrolladores: Symfony2 proporciona las herramientas que en gran medida mejoran la productividad de los desarrolladores, como la famosa barra de depuración web, soporte nativo de entornos, páginas detalladas de errores y mucho más.

Construido en base a otros grandes frameworks: Symfony2 tomo lo mejor de los conceptos de otros frameworks de desarrollo como Django, Spring y Ruby on Rails. También aprovecha componentes de Zend Framework y de Doctrine.

Listo para usar: Symfony2 cuenta con todas las características que el desarrollador de aplicaciones web necesita. También proporciona seguridad integrada y promueve el desarrollo web utilizando buenas prácticas.

2.5.2 Extjs

Extjs es una biblioteca Javascript que permite construir aplicaciones complejas en internet además de flexibilizar el manejo de componentes de la página como el DOM, peticiones AJAX, DHTML, tiene la gran funcionalidad de crear interfaces de usuario bastante funcionales. (Slocum, y otros, 2007)

Esta biblioteca incluye:

- Componentes UI[9] de alto desarrollo y personalizables.
- Modelo de componentes extensibles.
- Un API fácil de usar.
- Licencias Open Source (GPL) y comerciales

Ventajas (Sánchez Rosas, 2008)

[9] **UI** (user interface o interfaz de usuario) medio que permite a una persona comunicarse con una máquina. La interfaz, en este caso, está compuesta por los puntos de contacto entre un usuario y el equipo.

Permite crear aplicaciones complejas utilizando componentes predefinidos así como un manejador de layouts[10] similar al que provee Java Swing, gracias a esto provee una experiencia consistente sobre cualquier navegador, evitando el tedioso problema de validar que el código escrito funcione bien en cada uno (Firefox, IE, Safari, etc.). Además la ventana flotante que provee ExtJS es excelente por la forma en la que funciona. Al moverla o redimensionarla solo se dibujan los bordes haciendo que el movimiento sea fluido lo cual le da una ventaja tremenda frente a otros.

Usar un motor de *render* como Extjs nos permite tener además estos beneficios:

- Existe un balance entre Cliente – Servidor. La carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo.
- Comunicación asíncrona. En este tipo de aplicación el motor de *render* puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente se de cuenta.

- Eficiencia de la red. El tráfico de red puede disminuir al permitir que la aplicación elija que información desea transmitir al servidor y viceversa, sin embargo la aplicación que haga uso de la pre-carga de datos puede que revierta este beneficio por el incremento del tráfico.

Desventajas (Sánchez Rosas, 2008)

1. Necesidad de una plataforma: Pues se tiene dependencia del paquete ExtJS para obtener los resultados deseados.
2. Su sistema de licenciamiento no contempla la licencia LGPL, lo que obliga al programador a realizar el código 100% GPL o a pagar una licencia de desarrollo.

[10] **layouts** son maneras predefinidas de acomodar controles o componentes visuales en formas o ventanas visuales.

2.6 Entorno de Desarrollo Integrado (IDE)

Un entorno de desarrollo integrado (IDE) (también conocido como entorno de diseño integrado, entorno integrado de depuración o entorno de desarrollo interactivo) es una aplicación de software que proporciona servicios integrales a los programadores de computadoras para el desarrollo de software. Un IDE normalmente se compone de: (Blanco, 2012)

1. Un editor de código fuente
2. Un compilador y / o un intérprete
3. Automatización de generación de herramientas
4. Un depurador

El límite entre un entorno de desarrollo integrado y otras partes del entorno de desarrollo de software más amplio, no está bien definido. A veces, un sistema de control de versiones y varias herramientas integradas para simplificar la construcción de una interfaz gráfica de usuario. Muchos entornos de desarrollo modernos también tienen un navegador de clases, un inspector de objetos, y una jerarquía de clases diagrama, normalmente todo ello para su uso con el desarrollo de software orientado a objetos. (Blanco, 2012)

NetBeans IDE es un entorno de desarrollo - una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java - pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso. (Oracle, 2013)

Permite crear aplicaciones Web con PHP 5, un potente debugger integrado, hecho a la medida para el desarrollo de sitios web PHP que comprende una gran variedad de lenguajes de scripting y mark-up. El editor de PHP es dinámicamente integrado con HTML, JavaScript y funciones de edición CSS, contiene depurador de código PHP mediante el cual se puede inspeccionar variables locales, relojes de ajuste, establecer puntos de interrupción y evaluar el código. Además viene con soporte para Symfony2, soporte para JavaFX para la creación de aplicaciones web que tienen las características y capacidades de aplicaciones de escritorio, incluyendo aplicaciones multimedia interactivas, soporte para WebLogic (un servidor de aplicaciones Java EE y también un servidor web HTTP).

2.7 Visual Paradigm como herramienta CASE

Las herramientas CASE (del inglés Computer Assisted Software Engineering o Ingeniería de Software Asistida por Computación en español) son utilizadas para automatizar o apoyar una o más fases del proceso de desarrollo de software. Entre las más utilizadas se encuentran Rational Rose y Visual Paradigm.

Visual Paradigm soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Facilita el modelado de artefactos en un Proceso de Desarrollo de Software, mediante el Lenguaje de Modelado UML (Unified Modeling Language) el software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite representar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Esta es la herramienta de modelado que se utilizará para el desarrollo del módulo, pues a pesar de ser una herramienta propietaria la Universidad posee una licencia para el trabajo con la misma (Academic Partner Program), Esta licencia para el software Visual Paradigm está diseñada para garantizar la libertad de utilizar el software para fines académicos. Visual Paradigm otorga al destinatario una licencia libre de regalías, no exclusiva y no transferible para utilizar el Software bajo los términos y condiciones establecidos a continuación: *Esta licencia se aplica a usted, el beneficiario, sólo si usted es un miembro de una institución sin fines comerciales y académicos, por ejemplo, una universidad. La licencia se vence tan pronto como usted ya no es un miembro de esta institución*. Esta es una de las herramientas más potentes que existe en la actualidad para asistir el proceso de construcción del software. El hecho de ser una herramienta que soporta el trabajo concurrente de varios usuarios es una característica favorable para el trabajo en equipos grandes y proyectos complejos.

2.8 Servidor Web

Un **servidor web** es un programa que se ejecuta continuamente en un computador, manteniéndose a la espera de peticiones de ejecución que le hará un cliente o un usuario de Internet. (CAVSI, 2013)

El **servidor web** se encarga de contestar a estas peticiones de forma adecuada, entregando como resultado una página web o información de todo tipo de acuerdo a los comandos solicitados. Este intercambio es

mediado por el navegador y el servidor que hablan el uno con el otro mediante HTTP. Se pueden utilizar varias tecnologías en el servidor para aumentar su potencia más allá de su capacidad de entregar páginas HTML; éstas incluyen scripts, seguridad SSL y páginas activas del servidor (ASP[11]). (CAVSI, 2013)

La principal función de un servidor Web es almacenar los archivos de un sitio y emitirlos por Internet para poder ser visitado por los usuarios. Básicamente, un servidor Web es una gran computadora que guarda y transmite datos vía Internet. Cuando un usuario entra en una página de Internet su navegador se comunica con el servidor, enviando y recibiendo datos que determinan qué es lo que ve en la pantalla. (Duplika, 2010)

2.8.1 Apache

Apache es un servidor web flexible, rápido y eficiente, consecutivamente actualizado y adaptado a los nuevos protocolos (HTTP). Apache HTTP es de tecnología Open Source sólido y para uso comercial creado por la Apache Software Foundation. Se propone el uso de este servidor web porque es personalizable.

Los archivos en cuanto a la administración de configuración de Apache están en ASCII[12], por lo que posee un formato simple, y logran ser editados tan solo con un editor de texto. Estos son transferibles, lo que admite la clonación positiva de un servidor. Puede ser el servidor administrado vía línea de instrucciones, lo que hace la administración remota muy útil.

Además se trata de un servidor muy eficiente. Mucho esfuerzo se ha puesto en optimizar el rendimiento del código “C” de Apache. Como resultado, este se ejecuta rápido y consume pocos recursos de sistema en comparación a otros servidores. Además, Apache corre en una amplia variedad de sistemas operativos, incluyendo varias versiones de UNIX, Windows9x/NT, MacOS (Sobre Power PC), y varios otros.

[11] Una página ASP es una página html que incluye uno o más scripts (pequeños programas) esto permite crear dinámicamente páginas Web mediante HTML, scripts, y componentes de servidor ActiveX reutilizables.

[12] ASCII Sistema de codificación de siete bits que asigna un número del 0 al 127 a cada letra, número, carácter especial y carácter de control que recoge: los ficheros de texto contienen información escrita en código ASCII, lo que les permite ser entendidos por la mayoría de los programas del mercado.

Apache Web Server tiene un gran conjunto de funcionalidades de gran alcance. Estas características principales, junto con las extensiones creadas por programadores de todo el mundo, ayudan a que la plataforma Apache sea competitiva incluso frente a rivales de alto precio. Apache ha incorporado en su soporte a una amplia gama de lenguajes de programación web, como Perl, PHP y Python. Estos lenguajes son fáciles de aprender y se pueden utilizar para crear potentes aplicaciones en línea. Apache también incluye soporte "SSL" y "TLS". Estos son los protocolos para enviar datos encriptados a través de Internet, y son importantes en el desarrollo de tiendas seguras en línea y otras aplicaciones que requieren privacidad. (Aries, 2012)

2.9 Conclusiones Parciales

Después de culminar con el enfoque general de lo que ha tratado el capítulo, se llega a la conclusión de que la metodología a utilizar en las actividades de construcción del software será RUP, el Visual Paradigm será la herramienta Case y UML como lenguaje de modelado. El lenguaje de programación a utilizar del lado del servidor será PHP y del lado del cliente serán utilizados HTML y JavaScript con su biblioteca Ext utilizando el IDE NetBeans. Por las ventajas que brinda el uso de un framework y un servidor web en la implementación de soluciones para la web es que se selecciona Symfony2 y Apache respectivamente, para la construcción y despliegue de los subsistemas.

Capítulo 3: PRESENTACIÓN DE LA SOLUCIÓN PROPUESTA

3.1 Introducción

En este capítulo se comienza la descripción de la aplicación a través de la definición de las funcionalidades y características que la misma debe cumplir. Además se presentan para una mejor comprensión de la aplicación una serie de conceptos asociados al entorno que fue estudiado para llevar a cabo una modelación de la solución.

3.2 Dominio del sistema

El modelo del dominio muestra (a los modeladores) clases conceptuales significativas en un dominio del problema; es el artefacto más importante que se crea durante el análisis orientado a objetos. (Larman, y otros, 2003)

Un modelo del dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. También se les denomina modelos conceptuales (término utilizado en la primera edición del libro de Larman), modelo de objetos del dominio y modelos de objetos de análisis. (Larman, y otros, 2003)

El modelo de dominio, es un subconjunto del modelo de negocio, se utiliza cuando los procesos del negocio no pueden ser definidos con claridad, recoge los tipos de objetos más importantes que existen en el sistema, y tiene por objetivo ayudar a comprender mejor los conceptos que utilizan los usuarios, con los que se trabaja. Se decidió elaborar un Modelo de dominio, teniendo en cuenta que los procesos del módulo no dependen directamente del personal de la ONRM, ya que se tiene el conocimiento de las funcionalidades que debe realizar el mismo.

3.2.1 Descripción del dominio

El sistema a implementar constituye un módulo para el SGDGv2, el cual es una aplicación web que automatiza los procesos de gestión más importantes que se desarrollan en cada área de la ONRM. El objetivo del módulo a desarrollar es garantizar el correcto acceso a la información de acuerdo al nivel de autorización de la persona

que la gestiona, así como realizar un seguimiento de la gestión que tenga dicha información. El administrador será el encargado de administrar los permisos en cada área de los usuarios registrados en el sistema.

3.2.2 Descripción de las clases del modelo de dominio

Usuario: Persona autorizada a acceder a determinadas áreas y realizar acciones específicas según sus permisos.

Permiso: Derechos que les son otorgados a los usuarios para acceder a determinadas áreas y realizar acciones específicas.

Área: Módulo del SGDG al que solo pueden acceder usuarios específicos según sus permisos.

Sistema: Aplicación de software en funcionamiento, accesible al personal de la entidad.

Acción: Acciones que realizan los usuarios sobre el sistema, ya sean de acceso al sistema y sus módulos, de acceso a datos, consultas, etc.

Evento: Representa una acción realizada por el usuario en el sistema, donde se guarda información de interés que ayudan a identificar el tipo de acción realizada, la fecha, hora y el usuario que la realizó entre otros datos.

Información: Un conjunto organizado de datos, que puede ser que se encuentre en la aplicación o que se genere después del suceso de un evento.

Reporte: Conjunto de información que se utiliza para controlar el trabajo que se realiza sobre las aplicaciones.

3.2.3 Diagrama de clases del Modelo del Dominio

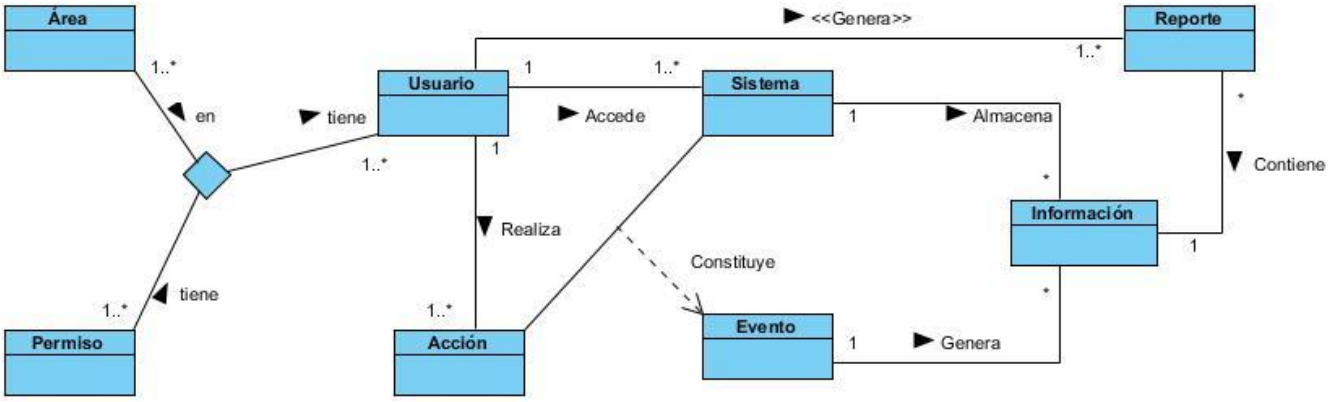


Figura 3: Diagrama de clases del Modelo del Dominio.

3.3 Especificación de Requisitos

3.3.1 Requisitos funcionales

Los requisitos funcionales son las funcionalidades que debe cumplir el sistema final, las mismas representan acciones específicas requeridas por el cliente.

RF1: Autenticar usuario: El requisito permitirá verificar el usuario y la contraseña de determinado usuario para identificarse y poder entrar al sistema con los permisos otorgados al usuario que le permiten realizar determinada acción. El usuario puede entrar al sistema sin antes autenticarse, pero solamente obtendrá los permisos que se le han otorgado en determinada área luego de autenticarse correctamente. Dicha acción puede ser llevada a cabo por cualquier usuario del sistema. Los datos que deben ser llenados para autenticarse son ***Usuario y Contraseña***.

RF2: Adicionar Usuario: El sistema debe ser capaz de permitir que se adicionen usuarios. Para adicionar un usuario se hace necesario introducir los campos: ***Nombre, Usuario, Contraseña, Verificar Contraseña, Teléfono, Correo.*** Esta acción solo la podrá realizar el administrador del sistema.

RF3: Modificar Usuario: El sistema debe permitir que se modifiquen los datos de los usuarios registrados en el sistema. Se debe poder cambiar en cualquier momento los siguientes datos de un usuario registrado en el sistema: ***Nombre, Usuario, Teléfono y Correo.*** Esta acción solo la podrá realizar el administrador del sistema.

RF4: Eliminar Usuario: El sistema debe ser capaz de permitir que se eliminen usuarios del sistema. Esta acción solo la podrá realizar el administrador del sistema.

RF5: Adicionar Permiso: El sistema debe ser capaz de permitir que se adicionen permisos, para adicionar un permiso se hace necesario introducir los campos: ***Nombre, Descripción.*** Esta acción solo la podrá realizar el administrador del sistema.

RF6: Modificar Permiso: El sistema debe ser capaz de permitir que se modifiquen los datos de los permisos registrados en el sistema, para modificar un permiso se hace necesario actualizar los campos: ***Nombre y Descripción.*** Esta acción solo la podrá realizar el administrador del sistema.

RF7: Eliminar Permiso: El sistema debe ser capaz de permitir que se eliminen permisos del sistema. Esta acción solo la podrá realizar el administrador del sistema.

RF8: Asignar permisos a un usuario en un área determinada: El sistema debe ser capaz de permitir que se asignen permisos a un usuario en un área determinada, controlando que cada usuario pueda realizar solo las acciones que el permiso le autoriza en el área. Un mismo usuario podrá tener diferentes permisos en cada área. Las áreas son los módulos que conforman el SGDG. Dicha acción solo puede ser llevada a cabo por el administrador del sistema.

RF9: Cambiar contraseña: El sistema debe ser capaz de permitir el cambio de contraseña para los usuarios registrados cuando lo estimen conveniente. La acción puede ser llevada a cabo por cualquier usuario del SGDG registrado en la base de datos.

RF10: Registrar eventos: El sistema debe permitir registrar los eventos auditables de los usuarios.

RF11: Generar Reportes: El sistema debe permitir generar reportes con información de las acciones realizadas por los usuarios del sistema. Esta acción solo la podrá realizar el administrador del sistema.

3.3.2 Requisitos no funcionales

RNF1: Usabilidad: El sistema debe poder ser usado por cualquier persona que tenga conocimientos básicos de computación. También debe poseer una interfaz que contenga los colores característicos de la ONRM. La información deberá estar disponible en todo momento, limitada solamente por las restricciones de acuerdo a las políticas de seguridad definidas.

RNF2: Fiabilidad: Al sistema se accederá a través de la autenticación convencional: usuario y contraseña. Cada usuario debe tener solo los permisos necesarios para realizar las operaciones que le sean permitidas en el módulo y debe mantenerse la consistencia de los datos en correspondencia con la realidad.

RNF3: Eficiencia: El tiempo de respuesta promedio por transacción de las peticiones realizadas estará en el rango de 2 a 5 segundos, en dependencia de la cantidad de información a procesar.

RNF4: Soporte: El período de soporte así como las restricciones asociadas se manejarán entre el equipo de desarrollo y los clientes.

RNF5: Restricciones de diseño: La aplicación debe de ser compatible con los siguientes navegadores: Internet Explorer versión 9.0 o superior y Mozilla 7.0 o superior.

La aplicación debe ser compatible con los Sistemas Operativos: Windows XP o superior y cualquier distribución de Linux.

RNF6: Requisitos para la documentación de usuarios en línea y ayuda del sistema: El módulo contará con un Manual de Usuario para la documentación del personal que interactúe con él, especialmente aquellos que se encarguen de su mantenimiento.

RNF7: Interfaz

Interfaces de usuario: El sistema debe tener una apariencia profesional y un diseño gráfico sencillo, con la utilización de las tonalidades de los colores carmelita, verde, anaranjado y gris fundamentalmente pues son los colores representativos de la entidad.

Interfaces Hardware: Las computadoras que utilizarán el software a desarrollar deberán tener 512 MB de Memoria tipo RAM como mínimo, recomendable 1024 MB. Velocidad de procesamiento del microprocesador 1GHz o superior.

Interfaces Software: Las computadoras que utilizarán el software deben tener instalado:

- Windows 2000 NT, Windows XP Profesional ó GNU/Linux en cualquier distribución.
- Navegador Web compatible con IE 9.0 o superior, Mozilla con versión 7.0 o superior.

El nodo (PC) que alojará la aplicación y la Base de Datos deberá tener instalado un servidor Apache con versión 5.3.10 de PHP, framework Symfony y PostgreSQL versión 9.1.

Interfaces de Comunicación: Se tendrá acceso a los demás módulos del sistema como son: Registros Mineros, Registros Petroleros, Inventario de Agua, Inventario de Minerales, Inventario de Petróleo y Gas, Metadatos y Nomencladores incluido el Portal de la ONRM.

RNF9: Requisitos legales, de copyright y otros: El sistema debe ajustarse y regirse por la ley, decretos leyes, decretos, resoluciones y manuales (órdenes) establecidos, que norman los procesos que serán automatizados.

3.4 Descripción del sistema propuesto

El sistema propuesto es un módulo para el SGDG destinado a administrar el acceso a la aplicación mediante la autenticación usuario/contraseña; gestionar usuarios del sistema, controlar y establecer los determinados permisos de manera que cada cual puede acceder solo a la información que le es permitida en cada área y almacenar información referente a las acciones realizadas por los usuarios. El mismo debe regular el acceso a los módulos en cuanto a: usuarios con permiso de administrador, consultor y probador, y además permitir la auditoría de la aplicación.

3.4.2 Descripción de los actores

Una vez visto los requisitos funcionales como no funcionales se muestran los actores identificados que realizan estas acciones.

Tabla 1 Descripción de Actores del Sistema

Actor	Justificación
Usuario	Persona autorizada a acceder a determinadas áreas y realizar acciones específicas según sus permisos.
Administrador	Usuario autorizado para gestionar todo lo referente a la seguridad del sistema y asignar permisos a un usuario en un área determinada.
Sistema	Aplicación que define el control de acceso sobre los archivos, permitiendo o no el acceso a las áreas a usuarios según los privilegios dados por el administrador y las restricciones establecidas. También es el encargado de registrar los eventos realizados por los usuarios.

3.4.3 Casos de Uso del Sistema (CUS)

Los requisitos funcionales descritos anteriormente fueron agrupados en 7 casos de uso y se observa su interacción con los actores descritos a través del diagrama de casos de uso del sistema.

3.4.3.1 Diagrama de Casos de Uso del Sistema

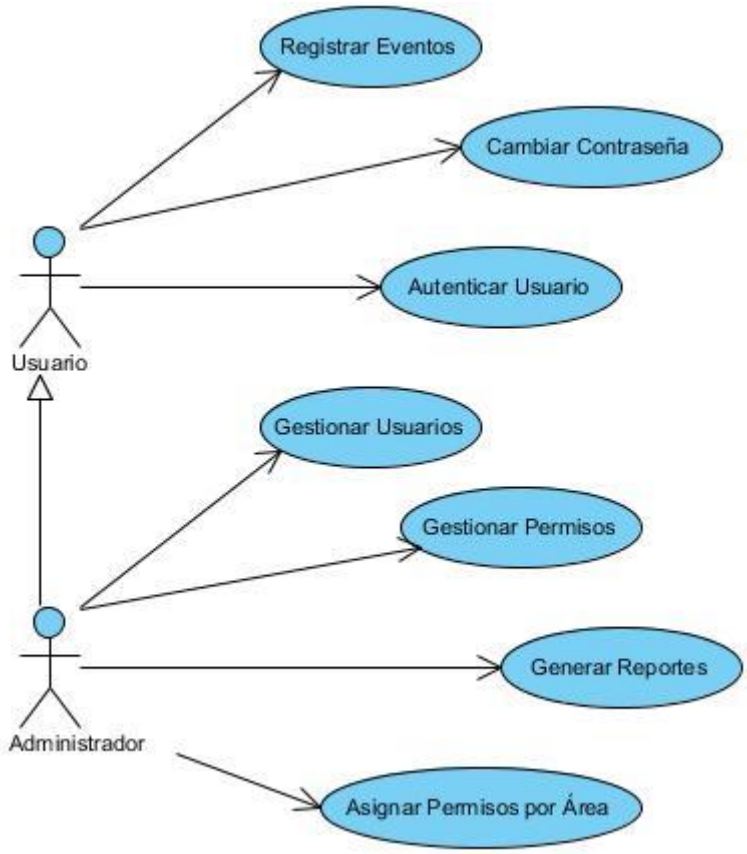


Figura 4: Diagrama de casos de uso del sistema.

3.4.3.2 Descripción de Casos de Uso del Sistema

Una vez identificados los casos de uso del sistema, se realiza una descripción de lo que el sistema debe hacer cuando interactúa con sus actores. En ella se detalla paso a paso el flujo de eventos que ocurren en dicha interacción y se especifican las acciones alternas de cada uno de los escenarios. A continuación se muestra la descripción textual correspondiente a uno de los casos de uso críticos del módulo.

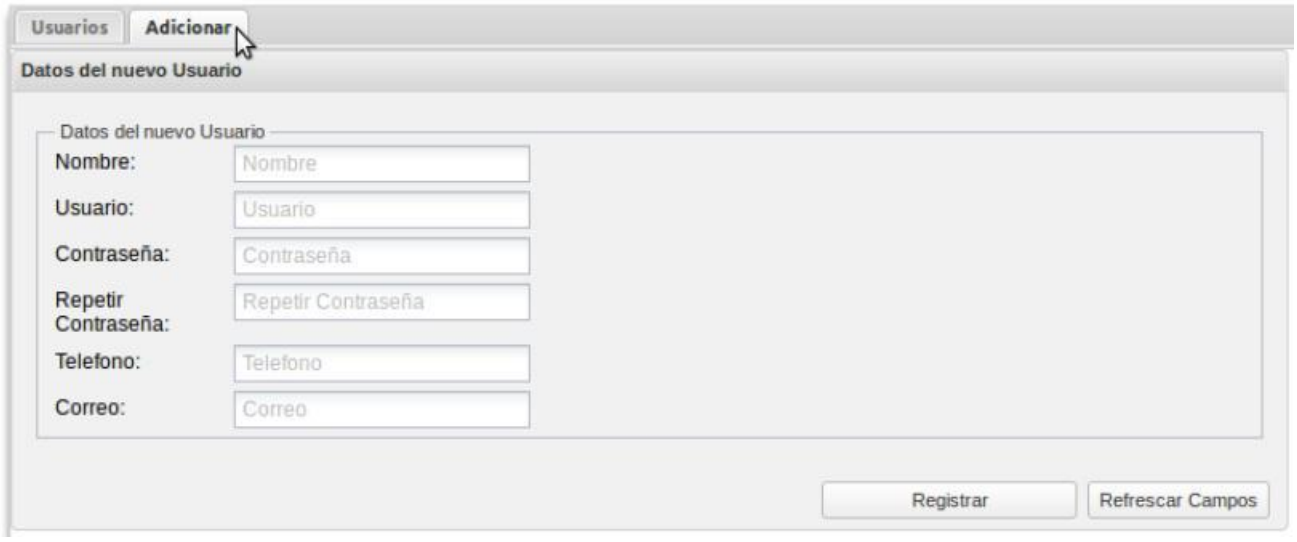
Tabla 2 Descripción Caso de Uso "Gestionar Usuario"

Caso de Uso:	Gestionar Usuario
Actores:	Administrador
Resumen:	El CU se inicia cuando el administrador selecciona la opción “Usuarios” en el menú “Gestión” para realizar las acciones de adicionar un usuario, modificar los datos de este o eliminarlo. El CU termina una vez realizada la acción seleccionada.
Precondiciones:	El usuario debe estar autenticado como administrador.
Referencias	RF2, RF3, RF4
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El administrador selecciona la opción “Usuarios” en el menú “Gestión”. Ver prototipo de interfaz #1. 3. El administrador selecciona la opción deseada.	2. El sistema muestra una página con las opciones de “Buscar”, “Adicionar”, “Modificar” y “Eliminar”. 4. Si selecciona la opción: • Adicionar (Ir a la sección Adicionar usuario) • Buscar (Ir a la sección Buscar usuario) • Modificar (Ir a la sección Modificar) • Eliminar (ir a la sección Eliminar)
Sección “Adicionar usuario”	
Acción del Actor	**Respuesta del Sistema**
	1. El sistema muestra una página que contiene el formulario para adicionar un nuevo usuario. Los campos a llenar son los siguientes: Nombre, Usuario, contraseña, verificar contraseña, teléfono, correo. Ver

2. El administrador introduce datos en los campos. 3. El administrador da clic en el botón “Registrar Usuario”.	prototipo de interfaz #1. 4. El sistema verifica que los datos insertados sean correctos. 5. El sistema adiciona un nuevo usuario y lo muestra en la lista de los usuarios existentes que contiene la página de la opción “Buscar” y termina el caso de uso.

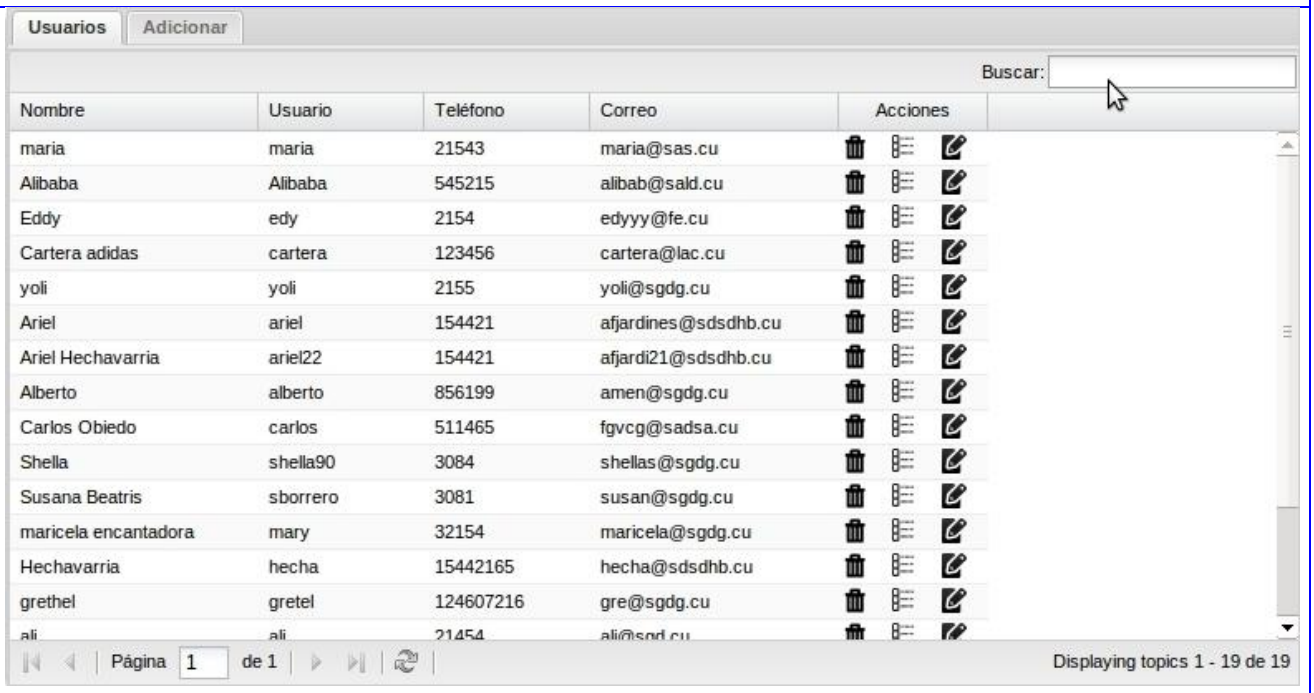
Prototipo de interfaz #1



Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
2. El administrador no introduce datos o introduce datos incorrectos en los campos. 3. El administrador da clic en el botón “Registrar Usuario”.	4. El sistema señala en rojo los campos que tienen datos incorrectos o están vacíos, mostrando un mensaje “Existen datos incorrectos en el formulario”.
Sección “Buscar usuario”	
Acción del Actor	**Respuesta del Sistema**

1. El administrador introduce en el campo Buscar, cualquier dato que contengan los campos Nombre, usuario, correo, teléfono.	2. El sistema busca el usuario que cumpla con las especificaciones de los datos insertados y lo muestra en una tabla con la siguiente estructura: Nombre, usuario, correo, teléfono, Acciones (Modificar, Eliminar). Ver prototipo de interfaz #2.

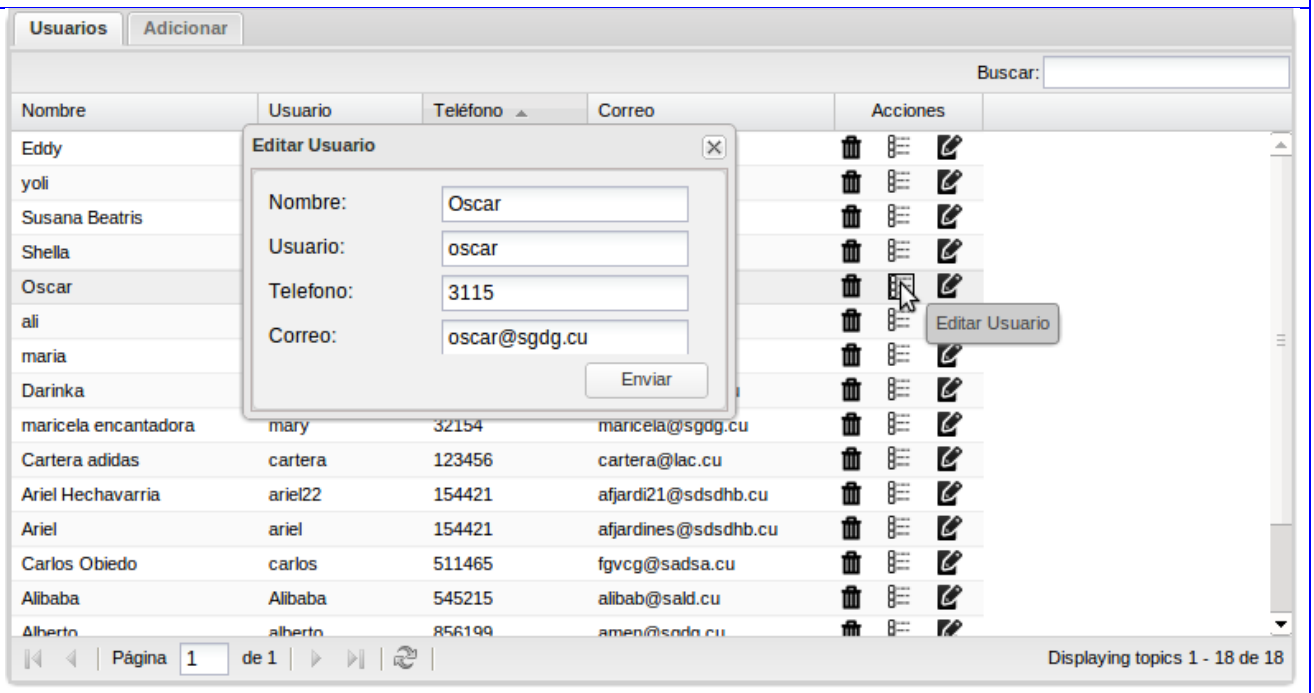
Prototipo de interfaz #2



Sección “Modificar datos”	
Acción del Actor	**Respuesta del Sistema**
2. El administrador modifica en los campos los datos que desee y presiona el botón “Enviar”.	1. El sistema muestra un formulario con los campos (nombre, usuario, contraseña, teléfono y correo) que contienen los datos que inicialmente habían sido insertados. Ver prototipo de interfaz #3.

	3. El sistema actualiza los datos del usuario y finaliza el caso de uso.

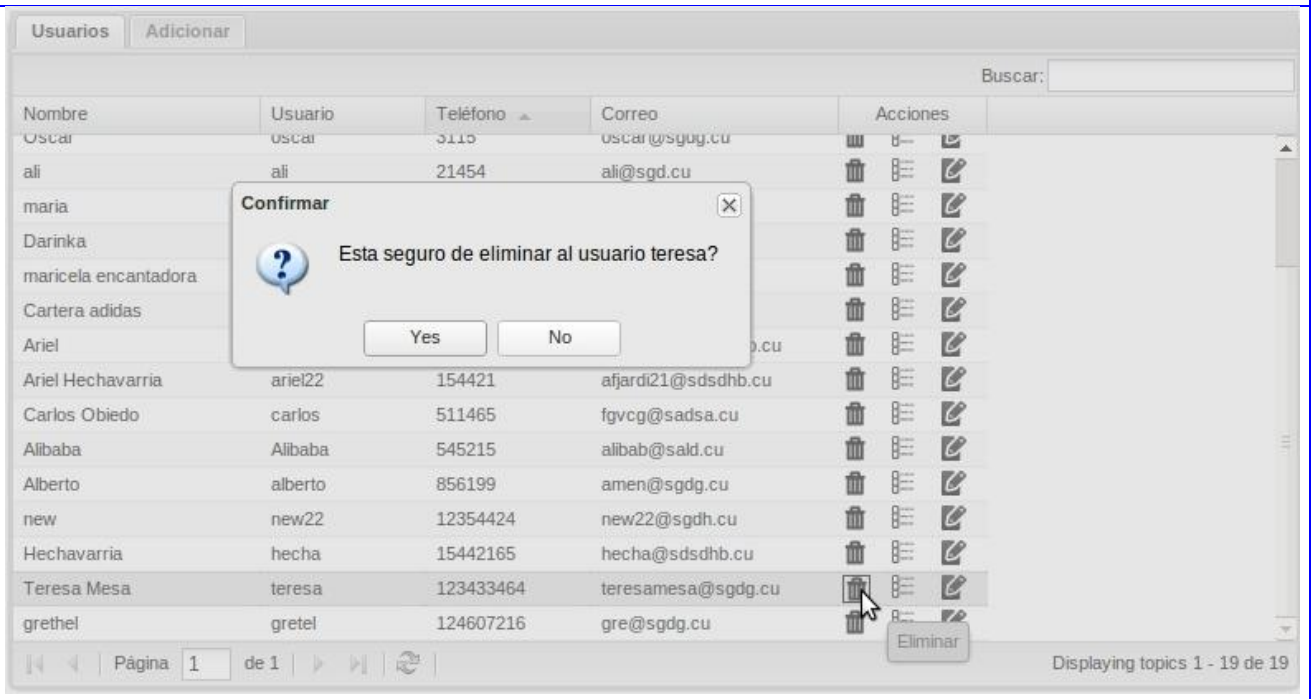
Prototipo de interfaz #3



Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
2. El administrador da clic en el botón “Cerrar”.	3. El sistema cancela la acción realizada y redirecciona la aplicación a la página de la opción “Buscar” y termina el caso de uso.
Sección “Eliminar datos”	
Acción del Actor	**Respuesta del Sistema**
2. El administrador da clic en el botón “Aceptar”.	1. El sistema muestra un mensaje de confirmación “¿Está seguro que desea eliminar el usuario seleccionado?”. Ver

	prototipo de interfaz #4. 3. El sistema elimina el usuario y actualiza la lista de estos y termina el caso de uso.

Prototipo de interfaz #4



Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
2. El administrador da clic en el botón “No”.	3. El sistema no elimina el usuario.

Poscondiciones	El sistema ha insertado, modificado o eliminado los datos, quedando actualizado luego de haber realizado cualquiera de las acciones descritas anteriormente.

3.5 Conclusiones parciales

En este capítulo se hizo la realización del modelo de dominio que permitió identificar y detallar los principales eventos, términos y conceptos presentes en el entorno que se investiga garantizando así el establecimiento de las bases para identificar las funcionalidades del sistema que se desea construir.

La realización del proceso de captura de requisitos concluyó con la obtención de las condiciones y las cualidades que el sistema debe cumplir para poder funcionar correctamente, unido a esto se debe especificar que el levantamiento de los requisitos no funcionales, le dan al software el nivel de calidad que debe cumplir. También se muestra un Modelo de Casos de Uso que permitió tener una vista global del sistema a desarrollar. Además este capítulo, garantizó que a partir de las descripciones textuales de los casos de uso identificados se tenga la idea exacta de qué se quiere llevar a cabo con cada funcionalidad, a partir de la acción del actor y la respuesta que debe dar el sistema.

Capítulo 4: CONSTRUCCIÓN DE LA SOLUCIÓN PROPUESTA

4.1 Introducción

En este capítulo se profundizará en el desarrollo del sistema propuesto, además se representará el modelo de diseño, sus atributos y responsabilidades. Se analizará la arquitectura que tiene el sistema para el cumplimiento de sus requerimientos y se expondrán los distintos diagramas de clases de diseño y secuencia con el objetivo de un mejor entendimiento del sistema desde su núcleo, también se mostrarán los diagramas de despliegue para abordar el recorrido por los flujos de trabajo implementación y prueba, se identificarán los componentes del sistema para establecer sus relaciones mediante la elaboración del diagrama de componentes y por último los diseños de los diferentes casos de prueba que permitirán demostrar la funcionalidad del sistema.

4.2 Modelo de análisis.

El modelo de análisis es un modelo de objetos conceptual que además de refinar los requisitos proporciona una estructura centrada en el mantenimiento de aspectos tales como la flexibilidad ante los cambios y la reutilización. Puede considerarse una primera aproximación al modelo de diseño teniendo en cuenta que mediante la observación de la estructura del modelo de análisis durante el diseño, se obtendrá un sistema flexible ante los cambios en los requisitos e incluirá elementos que podrán ser reutilizados cuando se construyan sistemas similares. (Jacobson, y otros, 2000)

3.5.1 Diagrama de clases del análisis

Los diagramas de clases permiten mostrar las clases participantes en la realización de los casos de uso y sus relaciones. Una clase del análisis se centra en el tratamiento de los requisitos funcionales y define atributos que normalmente son conceptuales y reconocibles en el dominio del problema. Las clases del análisis encajan siempre en uno de los tres estereotipos básicos:

Interfaz: Las clases de interfaz modelan las partes del sistema que dependen de sus actores, lo cual implica que clarifican y reúnen los requisitos en los límites del sistema.

Controladora: Estas clases representan coordinación, secuencia, transacciones y control de otros objetos y se usan con frecuencia para encapsular las funcionalidades de un caso de uso.

Entidad: Las clases entidad se utilizan para modelar los datos que posean una vida larga y que son a menudo persistentes. Describen la información y el comportamiento asociado a algún fenómeno o concepto del mundo real.

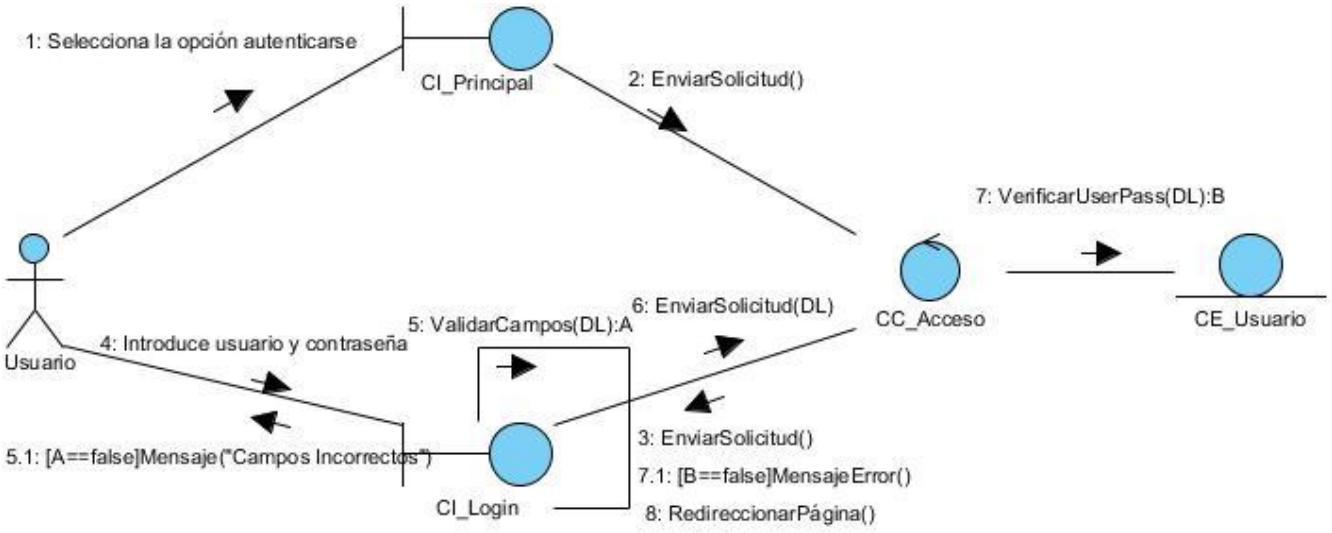
3.5.2 Diagramas de interacción

La secuencia de acciones en un caso de uso comienza cuando el actor envía algún tipo de mensaje al sistema. Cuando esto sucede un objeto de interfaz recibe este mensaje y envía a su vez un mensaje a algún otro objeto. De esta forma los objetos implicados interactúan para llevar a cabo el caso de uso. En el análisis este proceso se describe mediante diagramas de colaboración cuando el objetivo fundamental es identificar requisitos y responsabilidades sobre los objetos y mediante diagramas de secuencia cuando se persigue identificar secuencias de interacción detalladas y ordenadas cronológicamente. (Jacobson, y otros, 2000)

3.5.2.1 Diagramas de colaboración

A continuación se muestra el diagrama de colaboración correspondiente a uno de los casos de uso críticos del módulo.

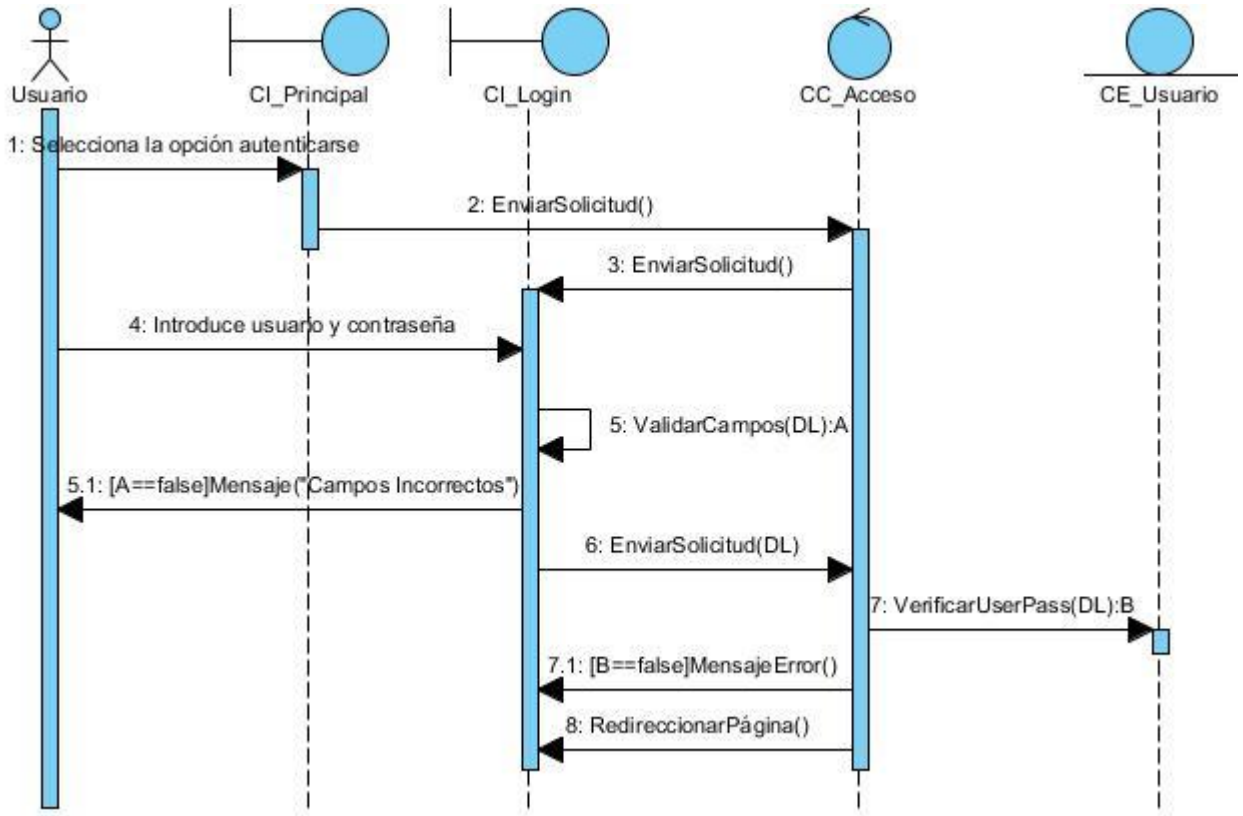
Figura 5: DCO Caso de Uso "Autenticar Usuario"



3.5.2.2 Diagramas de secuencia

A continuación se muestra el diagrama de secuencia correspondiente a uno de los casos de uso críticos del módulo.

Figura 6: DS Caso de Uso "Autenticar Usuario"



4.3 Patrones utilizados

4.3.1 Patrón arquitectónico Modelo-Vista-Controlador

Para el desarrollo del módulo propuesto se seleccionó el patrón MVC o Modelo-Vista-Controlador, el cual basa su robustez en que la lógica de la interfaz de usuario varía mucho más rápido que la lógica de negocio y almacenamiento, con lo que pretende separar las capas, de tal forma que un cambio en una de ellas no tenga

un impacto elevado en otra. Está dividida en tres capas o niveles:
Modelo: es la representación específica de la información con la cual el sistema opera.

Vista: presenta la información del sistema al usuario y genera los eventos de la interacción con éste.

- Captura eventos del usuario y se los envía al sistema a través del controlador.
- Recibe mandatos del controlador y muestra información al usuario.

Controlador: recibe eventos del usuario, invoca servicios ofrecidos por el modelo y selecciona la vista adecuada para presentar los resultados.

4.3.2 Patrones de diseño

Los patrones de diseño son un conjunto de estrategias, o buenas prácticas, que pueden facilitar el trabajo en muchas situaciones a la hora de realizar una aplicación orientada a objetos. Son soluciones reutilizables de problemas comunes del diseño orientado a objetos. Constituyen soluciones basadas en la experiencia que permiten flexibilizar el código haciéndolo satisfacer ciertos criterios y describir ciertos aspectos de la organización de un programa.

Patrones GOF

- **Observador (Observer):** Patrón de comportamiento a nivel de objetos. Es un patrón de diseño que define una dependencia del tipo *uno-a-muchos* entre objetos, de manera que cuando uno de los objetos cambia su estado, notifica este cambio a todos los dependientes. Se trata de un patrón de comportamiento, es decir, está relacionado con algoritmos de funcionamiento y asignación de *responsabilidades* a clases y objetos. Los patrones de comportamiento describen no solamente estructuras de relación entre objetos o clases sino también *esquemas de comunicación* entre ellos.

  Este patrón también se conoce como el patrón de publicación-inscripción o modelo-patrón. Estos nombres sugieren las ideas básicas del patrón, que son: el objeto de datos, que se le puede llamar Sujeto a partir de ahora, contiene atributos mediante los cuales cualquier objeto Observador o vista se puede suscribir a él pasándole una referencia a sí mismo.

Este patrón se utiliza en el módulo para registrar un evento específico del sistema, para esto se crea una clase SeguridadListener.php (la clase observador) a la cual el despachador de eventos de Symfony2 le pasa el evento al cual está subscrito cada vez que este ocurra, esta clase obtiene información de cada evento y la almacena en la base de datos.

- El patrón **Adaptador** (**Adapter**) se utiliza para transformar una interfaz en otra, de tal modo que una clase que no pudiera utilizar la primera, haga uso de ella a través de la segunda. Estructura, tanto a nivel de clases como a nivel de objetos. Su propósito es convertir la interfaz de una clase para que se adapte a lo que el cliente que la usa necesita, permitiendo así que trabajen juntas clases cuyas interfaces son incompatibles.

  Un ejemplo de su uso en el módulo es en la clase Usuario.php que implementa la clase UserInterface de Symfony2 para comportarse como una clase proveedora de usuarios que utiliza el sistema de seguridad para autenticar a los usuarios en el sistema.

Patrones GRASP

Los patrones GRASP[13] describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. (Saavedra Gutierrez, 2007)

- **Patrón Experto:** La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase contiene toda la información necesaria para realizar la labor que tiene encomendada. Este patrón tiene como objetivo principal asignar una responsabilidad al experto en información.

  Su uso se evidencia en el uso del patrón arquitectónico MVC (Modelo-Vista-Controlador) para implementar las funcionalidades relacionadas con los usuarios. Este modelo separa las funcionalidades

[13] GRASP es un acrónimo que significa General Responsibility Asignment Software Patterns (Patrones Generales de Software para Asignar Responsabilidades).

de las clases asignando una tarea determinada según la responsabilidad que esta posee en el funcionamiento del módulo, ya sea, accediendo a los datos que se encuentran almacenados en la base de datos del sistema, mostrando al usuario la información correspondiente o gestionando las diversas operaciones que se deben realizar. Ejemplo de ello es que el módulo contiene un fichero UsuarioController.php que contiene las funciones necesarias para gestionar un usuario, el fichero UsuarioRepository.php se encarga de obtener los datos necesarios mediante consultas SQL y en el caso del fichero Usuarios.html.twig que visualiza la imformación.

- **Alta cohesión:** La cohesión es la medida en la que un componente se dedica a realizar solo la tarea para la cual fue creado, delegando las tareas complementarias a otros componentes. (Una clase debe de hacer lo que respecta a su entidad, y no hacer acciones que involucren a otra clase ó entidad). Su uso se evidencia en las clases entidades del módulo por ejemplo en el fichero Usuario.php que es la única clase encargada de almacenar la información de los usuarios almacenados en la base de datos.

- **Bajo acoplamiento:** El acoplamiento es la medida de cuánto una clase está conectada (tiene conocimiento) a otras clases. Es un patrón evaluativo: un bajo acoplamiento permite que el diseño de clases sea más independiente, reduce el impacto de los cambios y aumenta la reutilización. Este patrón puede no ser importante si la reutilización no es un objetivo. Tiene como principal objetivo asignar una responsabilidad para mantener el bajo acoplamiento, o sea, mantener las clases lo menos ligadas posible. Soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, acrecentando la oportunidad de una mayor productividad.

  Este patrón se evidencia en la creación de la clase tr_Perm_User_Area.php, la misma está creada para representar la relación entre las clases Usuario.php, Permiso.php y Area.php, manteniendo a estas clases lo menos ligadas posible y más independientes.

4.4 Modelo de diseño

En el diseño se modela el sistema y se encuentra su forma para que soporte todos los requisitos. La entrada principal para esta etapa es el resultado del análisis, es decir, el modelo de análisis, que proporciona una comprensión detallada de los requisitos e impone una estructura del sistema que debe tratarse de conservar lo más fielmente posible. Los propósitos elementales del diseño son: (Jacobson, y otros, 2000)

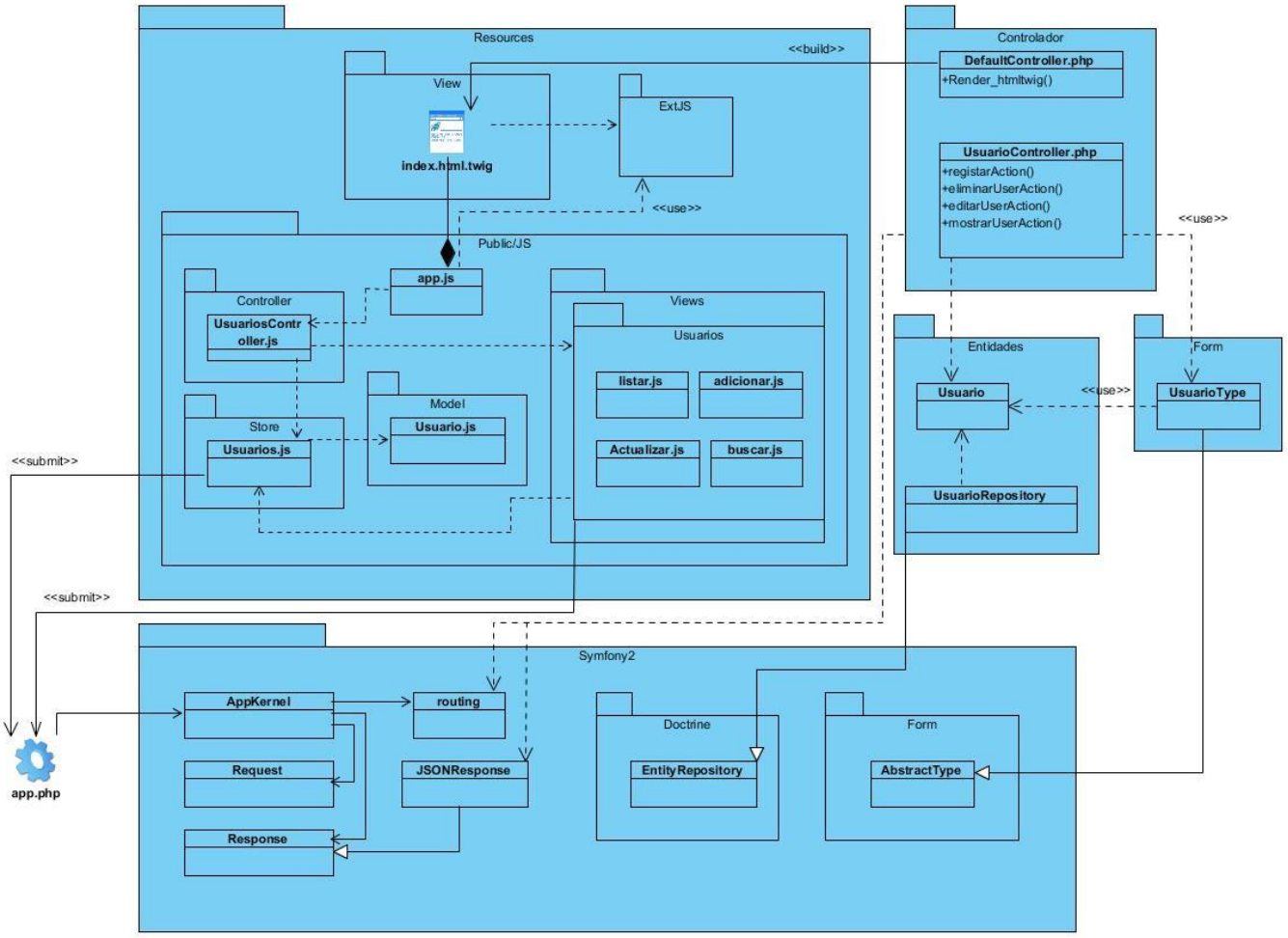
- Adquirir una comprensión profunda de todo lo relacionado con los requisitos no funcionales y las restricciones asociadas al lenguaje de programación que se utilizará para el desarrollo del sistema.
- Crear un punto de partida para las actividades de implementación subsiguientes.

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso, centrándose en cómo los requisitos funcionales y no funcionales, junto a restricciones de implementación, tienen impacto en el sistema a desarrollar.

4.4.1 Diagramas de clases del diseño

El diagrama de clases del diseño presentado a continuación está basado en la implementación que realiza el framework Symfony utilizado, de la arquitectura MVC. Debido a la complejidad y la extensión de la solución, incluyendo específicamente todas las clases contenidas en el framework, y siguiendo la premisa de ajustarse al flujo propio de la solución y no a los detalles transparentes al equipo de desarrollo, se decidió representar únicamente las clases y extensiones web que tuvieran que ver directamente con la construcción de la solución, sin dejar de incluir las partes fundamentales del framework que ayuden a su comprensión. Para ver el resto de los diagramas de clases del diseño dirigirse al Anexo2.

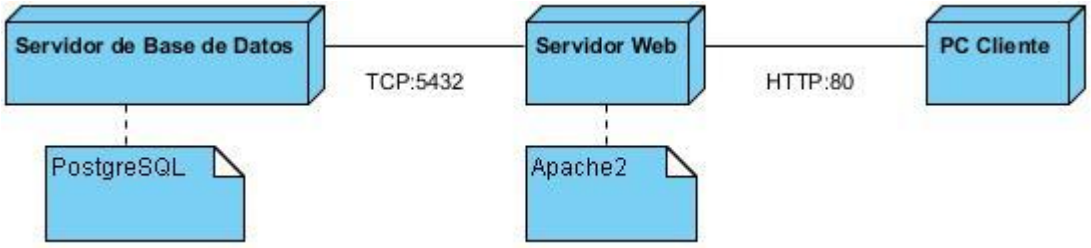
Figura 7: DCD Caso de Uso “Gestionar Usuarios”



4.5 Diagrama de despliegue

El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos del cómputo. Este se utiliza como entrada principal en las actividades de diseño e implementación.

Figura 8: Diagrama de Despliegue.



4.6 Modelo de datos

El Modelo de datos describe las tablas que constituyen las entidades asociadas al dominio del problema y que serán almacenadas en la base de datos. En el módulo propuesto se definen siete tablas: tabla **tusuario_seg**, tabla **tpermiso_seg**, tabla **tarea_seg,** tabla **tlog_seg**, tabla **tacciones_seg**, tabla **tr_Perm_User_Area** y la tabla **tr_Perm_Area_Acc**.

La tabla tusuario_seg almacena la información referente a los usuarios registrados en el sistema. Contiene los campos que se muestran en la tabla a continuación:

Tabla 3 Descripción de campos de la tabla tusuario_seg.

Campo	Tipo de Dato	Descripción
ID del usuario	Entero	Valor único que identifica al usuario en la base de datos.
Nombre del usuario	Cadena de caracteres alfa-numérico.	Cualquier combinación de letras y números.
Usuario	Cadena de caracteres alfa-numérico.	Valor único con el cual se autentica el usuario en el sistema.
Clave del usuario	Cadena de caracteres alfa-numérico.	Contraseña para la autenticación del usuario en el sistema.
Correo del usuario	Cadena de caracteres.	Dirección de correo electrónico del usuario.
Teléfono del usuario	Entero	Almacena el teléfono del usuario.

La tabla tpermiso_seg almacena la información referente a los permisos registrados en el sistema. Contiene los campos que se muestran en la tabla a continuación:

Tabla 4 Descripción de campos de la tabla tpermiso_seg.

Campo	Tipo de Dato	Descripción
ID del permiso	Entero	Valor único que identifica al permiso en la base de datos.
Nombre del permiso	Cadena de caracteres alfa-numérico.	Cualquier combinación de letras y números. Identifica el nombre del permiso.
Descripción del permiso	Cadena de caracteres alfa-numérico.	Almacena una cadena con la cual se describe al permiso.

La tabla tarea_seg almacena la información referente a las Áreas registradas en el sistema. Contiene los campos que se muestran en la tabla a continuación:

Tabla 5 Descripción de campos de la tabla tarea_seg.

Campo	Tipo de Dato	Descripción
ID del área	Entero	Valor único que identifica al área en la base de datos.
Nombre del área	Cadena de caracteres alfa-numérico.	Cualquier combinación de letras y números. Identifica el nombre del área.
Descripción del área	Cadena de caracteres alfa-numérico.	Almacena una cadena con la cual se describe al área.
Etiqueta del área	Cadena de caracteres alfa-numérico.	Almacena una etiqueta que identifica al área.

La tabla taccion_seg almacena la información referente a las acciones que se realizan en cada área de la aplicación. Contiene los campos que se muestran en la tabla a continuación:

Tabla 6 Descripción de campos de la tabla taccion_seg.

Campo	Tipo de Dato	Descripción
ID de la acción	Entero	Valor único que identifica la acción en la base de datos.
Nombre de la acción	Cadena de caracteres alfa-numérico.	Cualquier combinación de letras y números. Identifica el nombre de la acción.
Descripción de la acción	Cadena de caracteres alfa-numérico.	Almacena una cadena con la cual se describe la acción.
área	numérico	Almacena el ID del área a la cual pertenece esta acción.

La tabla tlog_seg almacena la información referente a los eventos del sistema: Contiene los campos que se muestran en la tabla a continuación:

Tabla 7 Descripción de campos de la tabla tlog_seg.

Campo	Tipo de Dato	Descripción
ID del log	Entero	Valor único que identifica el log en la base de datos.
Fecha	Date	Almacena la fecha en que se registran las acciones de los usuarios.
Usuario	Cadena de caracteres alfa-numérico.	Valor único con el cual se autentica el usuario en el sistema.
Dirección IP	Numérico.	Dirección “IP” desde la cual el usuario accede al sistema.
Detalles	Cadena de caracteres.	Almacena detalles de los eventos tales como errores y acciones ejecutas por el usuario.

La tabla tr_Perm_User_Area almacena la información referente a la relación entre las tablas tusuario_seg, tpermiso_seg y tarea_seg. Contiene los campos que se muestran en la tabla a continuación:

Tabla 8 Descripción de campos de la tabla tr_Perm_User_Area.

Campo	Tipo de Dato	Descripción

ID del usuario	Entero	Valor único, es una llave primaria y a la vez llave foránea obtenida de la relación con la tabla tusuario_seg.
ID del permiso	Entero	Valor único, es una llave primaria y a la vez llave foránea obtenida de la relación con la tabla tpermiso_seg.
ID del área	Entero	Valor único, es una llave primaria y a la vez llave foránea obtenida de la relación con la tabla tarea_seg.

La tabla tr_Perm_Area_Acc almacena la información referente a la relación entre las tablas taccion_seg, tpermiso_seg y tarea_seg. Contiene los campos que se muestran en la tabla a continuación:

Tabla 9 Descripción de campos de la tabla tr_Perm_Area_Acc.

Campo	Tipo de Dato	Descripción
ID de la acción	Entero	Valor único, es una llave primaria y a la vez llave foránea obtenida de la relación con la tabla taccion_seg.
ID del permiso	Entero	Valor único, es una llave primaria y a la vez llave foránea obtenida de la relación con la tabla tpermiso_seg.
ID del área	Entero	Valor único, es una llave primaria y a la vez llave foránea obtenida de la relación con la tabla tarea_seg.

Figura 9: Diagrama Entidad - Relación.

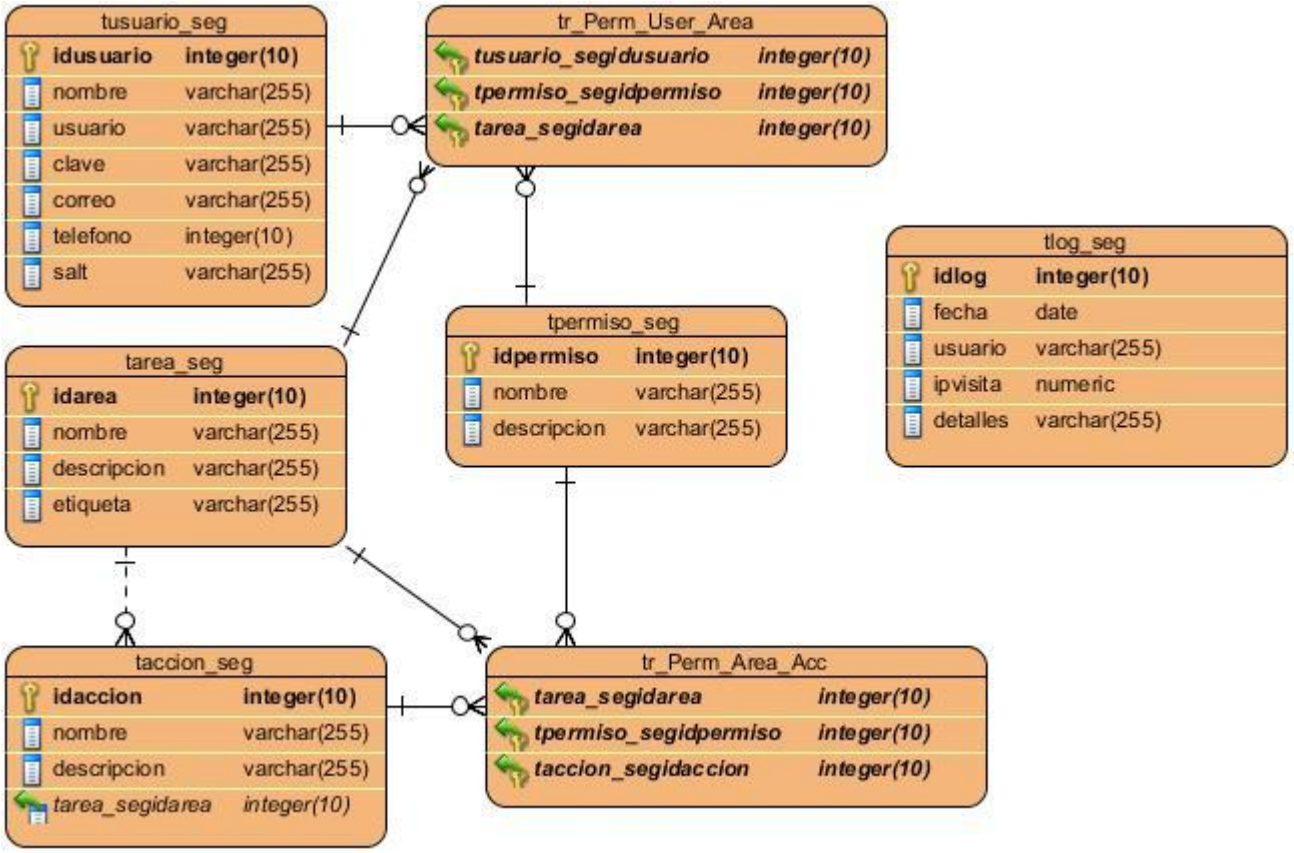
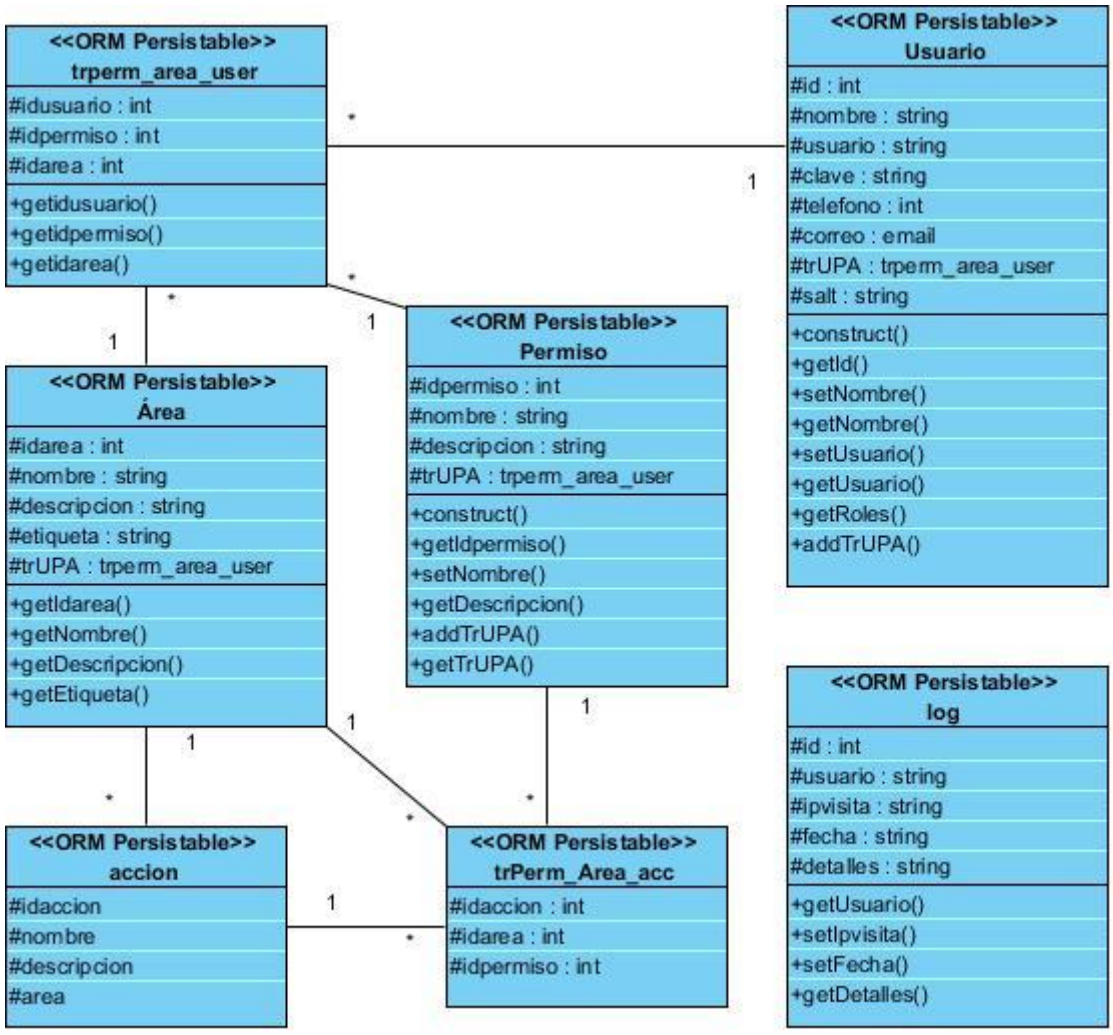


Figura 10: Diagrama Clases Persistentes.



4.7 Modelo de implementación

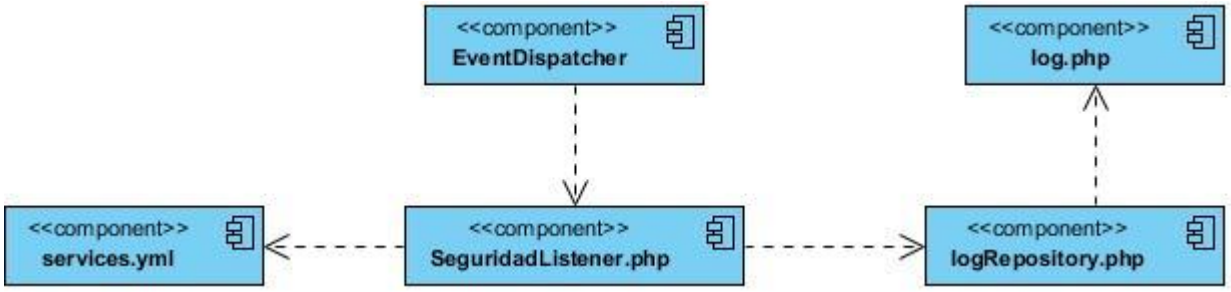
El modelo de implementación es una correspondencia directa entre los modelos de diseño y de despliegue. Describe cómo los elementos del diseño como las clases se implementan en términos de componentes como ficheros de código fuente, ejecutables, entre otros. Representa además cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados y cómo dependen los componentes unos de otros. (Jacobson, y otros, 2000)

4.7.1 Diagrama de componentes

Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño. Los componentes representan todos los tipos de elementos software incluidos en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, bibliotecas cargadas dinámicamente, entre otros.

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Un diagrama de componentes permite visualizar con más facilidad la estructura general del sistema y el comportamiento del servicio que los componentes proporcionan y utilizan a través de las interfaces. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro. A continuación se muestra el diagrama de componente para el CU registrar eventos.

Figura 11: DCOMP – Caso de Uso "Registrar Eventos"



4.8 Modelo de prueba

Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente. La prueba de software es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación.

4.8.1 Pruebas de caja negra

La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta

de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene.

Se centran principalmente en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

Casos de prueba de caja negra

La estrategia de prueba concebida para el módulo propuesto incluye la confección de siete casos de prueba de caja negra, es decir, uno por cada CU. A continuación se describe el caso de prueba autenticar usuario, incluyendo los resultados obtenidos en su aplicación durante la última iteración de pruebas realizadas.

Tabla 10 Caso de prueba diseñado para el CUS "Autenticar Usuario".

Nombre la sección	Escenario	Descripción de la funcionalidad	Flujo central
SC 1 Autenticar usuario.	EC 1.1: Autenticar usuario satisfactoriamente.	El sistema autentica al usuario y lo direcciona a la portada.	• Seleccionar la opción "Iniciar sesión" en la portada de la aplicación. • Introducir usuario y contraseña en el formulario. • Presionar el botón "Acceder Ahora".
	EC 1.2: Se introducen datos inválidos.	El sistema muestra los campos incorrectos con el borde en color rojo.	• Seleccionar la opción "Iniciar sesión" en la portada de la aplicación. • Introducir usuario y contraseña en el formulario. • Presionar el botón

			"Acceder Ahora".
	EC 1.3: Se dejan campos en blanco.	El sistema muestra un mensaje "Por favor rellena este campo"	• Seleccionar la opción "Iniciar sesión" en la portada de la aplicación. • Presionar el botón "Acceder Ahora".

Tabla 11 Descripción de las variables para el CUS "Autenticar Usuario".

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Nombre de usuario	Campo de texto	No	Este campo no puede estar en blanco. Admite tanto letras como números.
2	Contraseña	Campo de texto	No	Este campo no puede estar en blanco. Admite tanto letras como números y caracteres extraños y no admite una cantidad de caracteres menores a seis.

Tabla 12 Matriz de Datos para el CUS "Autenticar Usuario".

ID del escenario	Escenario	Nombre de la sección	**V1**	V2	Respuesta del Sistema	Resultado de la Prueba
EC 1.1	EC 1.1: Autenticar usuario satisfactoriamente.	Autenticar usuario.	V (Oscar)	V (Oscar2)	El sistema autentica al usuario y lo direcciona a la portada.	Satisfactoria

EC 1.2	Se introducen datos inválidos.	Autenticar usuario.	I (Osc@r)	V (Oscar2)	El sistema muestra los campos incorrectos con el borde en color rojo.	Satisfactoria
EC 1.3	Se dejan campos en blanco.	Autenticar usuario.	I ()	I ()	El sistema muestra un mensaje “Por favor rellena este campo”	Satisfactoria

En la primera iteración de pruebas al sistema se encontraron una serie de No Conformidades que fueron corregidas, luego se procedió a realizar una segunda iteración, la cual arrojó resultados satisfactorios, por lo que no se necesitó la realización de una tercera iteración.

4.9 Conclusiones parciales

Durante el desarrollo de este capítulo se llevaron a cabo los flujos de trabajo: Análisis y Diseño, Implementación y Prueba por lo que se generaron los artefactos correspondientes a cada uno de ellos. Durante el análisis se realizó un estudio de los requisitos capturados, refinándolos y estructurándolos, alcanzando una comprensión más precisa de los mismos y una descripción fácil de mantener.

Con el modelo de datos se logró describir los elementos que intervienen en el problema y la forma en que se relacionan estos elementos entre sí. La realización del modelo de diseño permitió representar la estructura del módulo a desarrollar. Se expuso el modelo de Implementación, donde se especificaron los distintos componentes utilizados para desarrollar la aplicación y la relación entre ellos, lográndose traducir el diseño en términos de componentes ejecutables. También se elaboró el modelo de despliegue con el que se muestra la distribución física del sistema. Finalmente se realizaron las pruebas pertinentes para verificar el cumplimiento de los requisitos que debe tener el sistema.

Conclusiones Generales

Una vez concluida la investigación y luego de obtenidos los resultados, es posible resaltar una serie de conclusiones que se enumeran a continuación:

- El estudio realizado de los principales conceptos relacionados con el objeto de estudio de la investigación permitió elaborar el marco teórico de la misma, teniendo como resultado la propuesta de solución de la problemática planteada.
- El proceso de desarrollo de la solución propuesta fue guiado por la metodología de desarrollo seleccionada, quedando documentada cada etapa del ciclo de vida. Los artefactos generados servirán como base de futuras actualizaciones del módulo.
- Se realizó un Modelo de Dominio, debido a que los procesos del módulo no dependen directamente del personal de la ONRM, el cual abarca las definiciones y relaciones entre los conceptos relacionados con el entorno que enmarca el sistema.
- Las tecnologías, herramientas, lenguajes y framework seleccionados para el desarrollo de la aplicación que centra la investigación obedecen a criterios de selección de tecnologías libres y multiplataforma, de acuerdo con las políticas que impulsa la universidad.
- Los requisitos funcionales y no funcionales identificados describen las características del sistema y sirvieron de punto de partida para las etapas de desarrollo posteriores.
- El Análisis y Diseño del sistema fue realizado mediante diagramas, modelos y descripciones, que proveen una abstracción de las funcionalidades del mismo.
- La vista arquitectónica del sistema se basa en la utilización del patrón MVC, el cual separa la implementación del sistema en tres capas, permitiendo que los cambios realizados en cada una de ellas no tengan gran trascendencia en las demás.

Recomendaciones

Luego de haber concluido el sistema y cumplido los objetivos trazados, se plantean la siguiente recomendación:

- Implementarle a la aplicación la funcionalidad de exportar como “Word” y “PDF” para cuando se realice una auditoría tener los reportes en formato duro.

Referencias Bibliográficas

Aries, Benjamin. 2012. **eHow en Español. [En línea] 2012. http://www.ehowenespanol.com/ventajas-apache-web-server-lista_109947/.**

Belmonte Ruiz, Francisco. 2011. **parasitovirtual.wordpress. [En línea] 3 de febrero de 2011. http://parasitovirtual.wordpress.com/category/cursos-y-articulos/desarrollo-de-webs/php/symfony/.**

Blanco, Carlos. 2012. **CarlosBlanco.pro. [En línea] 8 de abril de 2012. http://carlosblanco.pro/2012/04/entornos-desarrollo-integrado-introduccion/.**

cavsi. 2012. **cavsi. [En línea] 2012. http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.**

CAVSI. 2013. **CAVSI. [En línea] 2013. http://www.cavsi.com/preguntasrespuestas/que-es-un-servidor-web/.**

Duplika. 2010. **Duplika Web Hosting. [En línea] 20 de septiembre de 2010. http://www.duplika.com/blog/que-son-los-servidores-web-y-por-que-son-necesarios.**

Eguiluz, Javier. 2011. ***Desarrollo web ágil con Symfony2 - Primera edición.*** **2011.**

EllisLab, Inc. 2011. **CodeIgniter Guía del Usuario en Español. [En línea] 03 de 09 de 2011. http://www.codeigniterespanol.com/manual-codeigniter2.0.3-espanol.pdf.**

Foundation, Cake Software. 2012. **Cake PHP. [En línea] 2012. http://book.cakephp.org/2.0/en/core-libraries/components/security-component.html.**

—. 2013. **CakePHP Cookbook Documentation. [En línea] 20 de Febreo de 2013. http://book.cakephp.org/2.0/_downloads/es/CakePHPCookbook.pdf.**

Gallego Vázquez, Jose Antonio. 2003. ***Desarrollo Web con PHP y MySQL.*** **Madrid : s.n., 2003. ISBN:84-415-1525-5.**

Gallo, Elisa y Vergara, Mikel. **European Software Institute. [En línea] http://www.esi.es/Berrikuntza.**

Gómez Baryolo, Oiner, Rivero Pino, Noel Jesús y López Méndez, Daniel E. 2011. **Serie Científica de la Universidad de las Ciencias Informáticas. [En línea] 15 de Julio de 2011. https://publicaciones.uci.cu/index.php/SC/article/view/702/432. ISSN/RNPS.**

González Blanchard, David Antonio. 2010. **Scribd Inc. [En línea] 28 de febrero de 2010. http://es.scribd.com/doc/27613468/FrameWork-Cake-PHP.**

IRIS-CERT. 2008. **Rediris. [En línea] 12 de Noviembre de 2008. http://www.rediris.es/cert/doc/unixsec/node14.html.**

Jacobson, I, Booch, G y Rumbaugh, J. 2000. ***El Proceso Unificado De Desarrollo De Software.*** **2000.**

Larman, Craig y Hall, Prentice. 2003. ***UML y Patrones 2ª Edición.*** **2003.**

Mato García, LIC. Rosa María. 1999. ***DISEÑO de BASES DE DATOS.*** **1999.**

Metodologías ágiles y formales o robustas. Piñero Pérez, Dr. Pedro Y. y Leyva Vázquez, MsC Maikel Yelandi. 2009. **La Habana (UCI) : s.n., 2009.**

Mondragón Sotelo, Martin. 2006. **My Group Net - La comunidad de programación . [En línea] 31 de Julio de 2006. http://mygnet.net/articulos/seguridad/763/index.php.**

Oracle, Corporation. 2013. **NetBeans IDE. [En línea] 2013. http://netbeans.org/index_es.html.**

Potencier, Fabien y Zaninotto, François. 2008. ***Symfony la guía definitiva.*** **2008.**

rafaelma. 2010. **postgresql.org.es. [En línea] 02 de Octubre de 2010. http://www.postgresql.org.es/sobre_postgresql.**

reakguz. 2012. **[En línea] marzo de 2012. http://www.buenastareas.com/ensayos/Principales-Sistemas-Gestores-De-Base-De/3602722.html.**

Rigazzi, Pablo. 2008. **Zend Framework. [En línea] 2008. http://spanish.zendfw.com/que-es-zend-framework/.**

Rojas, Nohemi. 2010. **Blog de WordPress. [En línea] 25 de marzo de 2010. http://nohemirojas.wordpress.com/2010/03/25/cms-y-framaework-dos-conceptos-distintos/.**

Ruiz, Francisco. 2010. **[En línea] 2010. http://www.ctr.unican.es/asignaturas/is1/is1-t02-trans.pdf.**

Saavedra Gutierrez, Jorge A. 2007. **El Mundo Informático. [En línea] 7 de v de 2007. http://jorgesaavedra.wordpress.com/category/patrones-grasp/.**

Sánchez Rosas, Juan Eladio. 2008. **Desarrollo en Web. [En línea] 22 de Octubre de 2008. http://blogs.antartec.com/desarrolloweb/2008/10/extjs-lo-bueno-lo-malo-y-lo-feo/.**

Sicilia, Miguel Angel. 2008. **Conexions. [En línea] 23 de Septiembre de 2008. http://cnx.org/content/m17543/latest/.**

Slocum, Jack, y otros. 2007. **EcuRed. [En línea] 1 de abril de 2007. http://www.ecured.cu/index.php/Sencha_Ext_JS.**

Tejeda, Deivinson, Silveira, Emilio y Gutierrez, Andres Felipe. 2007. **KumbiaPHP Framework. [En línea] 2007. http://www.kumbiaphp.com/blog/about/.**

Tejeda, Deivinson, y otros. 2009. ***Kumbia PHP Framework. Porque Programar debería ser más fácil.*** **Colombia : s.n., 2009.**

VERONICA MACIAS, SANDRA. 2011. **teossvero.blogspot.com. [En línea] 23 de mayo de 2011. http://teossvero.blogspot.com/2011/05/concepto-de-logs-registros.html.**

Villalón Huerta, Antonio. 2002. ***SEGURIDAD EN UNIX Y REDES.*** **2002.**

Welling, Luke y Thomson, Laura. 2006. ***Desarrollo Web con PHP y MySQL.*** **2006.**