Universidad de las Ciencias Informáticas
Facultad 7

Título: *Componente de Seguridad para aplicaciones del Área Temática Sistemas de Apoyo a la Salud*

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO

DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autores: Karina Centeno Díaz

Danisbel Rojas Ríos

Héctor Manuel Solis Mulet

Tutores: Ing. Annia Arencibia Morales

Ing. Karel Gómez Velázquez

Asesora: Lic. Yaima Margarita Riveri Ruiz

Ciudad de La Habana, junio de 2008

"Año 50 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los 20 días del mes de junio del año 2008:

Karina Centeno Díaz	Danisbel Rojas Ríos	Héctor Manuel Solis Mulet
Autora	**Autor**	**Autor**

Ing. Annia Arencibia Morales	Ing. Karel Gómez Velázquez
Tutora	**Tutor**

DATOS DE CONTACTO

Ing. Karel Gómez Velázquez *(kgomez@uci.cu): Ingeniero en Ciencias Informáticas, graduado en la UCI, en el curso 2006-07. Posee la Categoría Docente de Profesor Adiestrado. Ha participado en proyectos de desarrollo de Sistemas Informáticos para la Salud desde el año 2005. Actualmente imparte asignaturas de Técnicas de Programación. Además, se desempeña como Asesor de Tecnologías y Arquitectura en la Facultad 7.*

Ing. Annia Arencibia Morales *(aarencibia@uci.cu): Graduada de Ingeniero en Ciencias Informáticas en la UCI, en el curso 2006-07. Posee Categoría Docente de Profesor Adiestrado. Ha participado en proyectos de desarrollo de Sistemas Informáticos para la Salud desde el año 2005. Imparte la asignatura de Investigación de Operaciones y Probabilidad y Estadística. Actualmente se desempeña como Líder de Proyecto del Área Temática Sistemas de Apoyo a la Salud.*

Lic. Yaima Margarita Riveri Ruiz *(maggie@uci.cu): Licenciada en Educación, especialidad Lengua Inglesa. Posee la Categoría Docente de Profesor Instructor. En cinco años de labor docente, ha impartido las asignaturas de la disciplina Idiomas Extranjeros. Actualmente se desempeña como Jefa de la asignatura Idioma Extranjero IV.*

"...lo fundamental es hacer algo nuevo cada día y luego perfeccionar lo que se ha hecho el día anterior..."

Ernesto Guevara de la Serna

AGRADECIMIENTOS

A la Revolución y a nuestro Comandante Fidel Castro, por habernos dado la oportunidad de estudiar en esta excelente universidad.

A Karel y Annia, por todo el apoyo y la confianza depositada. Por ser ejemplos a imitar por cada uno de nosotros y por contribuir de forma excelente a nuestra formación como profesionales.

A nuestro tribunal de tesis por ayudarnos a desarrollar de forma exitosa nuestro Trabajo de Diploma.

A todos nuestros profesores de la UCI, en especial a Yoe, Yasel, Yenisel, Tiuska y Yaima que depositaron su granito de arena para que este momento fuera posible.

A todos nuestros compañeros y amigos de siempre que estuvieron en las buenas y en las malas, especialmente: Alexis (El tigre), Ramos, Alfredo, Tula, Rotceh y Bolmey.

A nuestros padres de forma muy especial por guiarnos por el camino correcto y ayudarnos a alcanzar este sueño tan deseado.

A todos aquellos que de una forma u otra nos ayudaron.

Muchas Gracias

DEDICATORIA

Karina Centeno Díaz:

A las personas que más amo y quiero en la vida, "mi mamá y mi papá": Marita y Sergio, por apoyarme en todo momento y enseñarme el camino correcto para llegar a ser alguien en la vida. A mi familia, que quiero mucho, mis tías: Marta, Mirta y Mercy, mis primas: Yeisis, Carmen Rosa, Daniela y Dayamí. A Eslavy "Mi tata", el gran amor de mi vida, por estar siempre a mi lado y apoyarme siempre en todo, sin dudas conocerte fue lo mejor que me ha pasado, te amo mucho mi amor. A Pepe "El amore", por ser para mí el gran hermano que nunca tuve y ayudarme a alcanzar mi sueño más deseado: graduarme. A mis compañeros de tesis: Pepe y Hectico por su incondicionalidad y responsabilidad, por haberme dado la oportunidad de poder compartir con personas tan especiales, siempre los recordaré. A mis amigos: Sandra, Lisandra e Isnel, por estar siempre en las buenas y en las malas. A mis tutores: Karel y Annia, por su esfuerzo, dedicación, por nunca decir un "no", por ser ejemplo a imitar para mí y por ayudarme a mi formación como profesional.

Danisbel Rojas Ríos:

A la personita más importante en mi vida, "mami", por enseñarme un poquito todos los días para llegar hasta aquí, por apoyarme siempre y darme todo el cariño del mundo. A mi familia, que es la más linda del mundo, especialmente a mis hemanos: Pepe, Yanay, Nené y Raciel, a mis tías: Merci, Norma y Juanita, a mis papas: Radibel y Domingo, a la piquitica de papito "Rocio" y a mis vecinos que siempre me han apoyado. A Karina "El amore", por ser para mí la hermana que nunca tuve y ayudarme a alcanzar mi sueño más deseado: graduarme. A "Yadiurvis", mi novia linda por quererme tanto y darme tanto apoyo. A mis compañeros de tesis: Karina y Hectico por su incondicionalidad y responsabilidad, por haberme dado la oportunidad de poder compartir con personas tan especiales. A mis tutores: Karel y Annia, que me ayudaron a realizar este sueño.

Héctor Manuel Solis Mulet:

A toda mi familia y de forma especial a las personas que más amo en esta vida mis hermanos Jesús y Gretel y mis padres Héctor y Dania, por entregarme de forma incondicional su amor y dedicación, su apoyo en todo momento y haberme encaminado de forma correcta en la vida.
A mis compañeros de tesis: Karina y Pepe.
A mis tutores: Karel y Annia.

RESUMEN

La investigación está encaminada a obtener un producto de software que proporcione eficaces procesos de Autenticación, Autorización y Auditoría (AAA) para aplicaciones basadas en Servicios Web XML. El mismo permite una gestión eficiente de usuarios, asignación de roles y privilegios de acceso a todos los sistemas externos que consuman los servicios proporcionados por este. Por otra parte brinda un eficiente y óptimo proceso de trazabilidad y auditoría, de manera que se lleve un control estricto de las operaciones en que se involucran los usuarios de los sistemas.

Este sistema permite, mediante los procesos de administración proporcionados, una gestión eficiente de todos los requerimientos de seguridad para aquellos sistemas externos que consuman sus servicios, lográndose de esta forma la reutilización del código y evitándose que se realicen acciones innecesarias fuera de cada negocio. Esto se logra mediante la publicación de Servicios Web XML debidamente descritos utilizando el Lenguaje de Descripción de Servicios Web (WSDL).

Su desarrollo está basado en tecnologías libres, multiplataformas y sobre una arquitectura en capas, utilizando PHP 5 como lenguaje de programación, se implementa el patrón de arquitectura Modelo Vista Controlador, PostgreSQL 8.2 como Sistema de Gestión de Bases de Datos, metodología AJAX para realizar más eficientemente las peticiones al servidor y la librería YUI para obtener una interfaz visual moderna. Utiliza estándares abiertos como XML lo que permite la interoperabilidad entre aplicaciones desarrolladas sobre diferentes plataformas. Asimismo utiliza el protocolo de transporte HTTPS que garantiza la confidencialidad de los datos.

Palabras Claves: *Seguridad, Autenticación, Autorización, Auditoría, Servicios.*

TABLA DE CONTENIDOS

INTRODUCCIÓN

A través del Programa para Informatizar la Sociedad, el Estado Socialista Cubano, ha experimentado un incremento en la utilización de las tecnologías de la información en los últimos años. Con el fin de alcanzar un mejoramiento en la infraestructura tecnológica para satisfacer las necesidades de almacenamiento y acceso; la utilización de la información tanto en la esfera socioeconómica como política. Lo que permitirá elevar la calidad de vida, la eficiencia y eficacia de los servicios prestados, la preparación del capital humano y la competitividad del país. Esta constituye una vía para alcanzar una posición de vanguardia a nivel mundial en el empleo de estas tecnologías.

Existen varias empresas e instituciones pertenecientes al Ministerio de Informática y Comunicaciones (MIC) especializadas en el desarrollo y mantenimiento de software. Entre ellas: la Empresa Nacional de Software DESOFT y la Empresa de Soluciones Informáticas (SOFTEL). Esta última orienta su trabajo hacia el desarrollo de productos y servicios informáticos para la salud. Del mismo modo, la Universidad de las Ciencias Informáticas (UCI), adscrita al Consejo de Estado también se alinea a este objetivo. Como parte de ella, la Facultad 7, implementa en el Plan de Estudio de sus estudiantes, un Segundo Perfil relacionado con la Informática para la Salud.

La estructura organizativa aprobada para los procesos productivos de esta facultad está compuesta por varias Áreas Temáticas; una de ellas es el Área Temática Sistemas de Apoyo a la Salud (SAS). Esta es una estructura que agrupa un conjunto de proyectos, que coinciden en el tema de desarrollo e investigación. El Área Temática SAS tiene como propósito el desarrollo y soporte de aquellos productos de software que informaticen los procesos no asistenciales del Ministerio de Salud Pública (MINSAP). Los proyectos asociados a esta área, actualmente son: Sistema para la Planificación, Balance y Distribución de Suministros Médicos, Docencia Médica, Colaboración Médica, Administración Remota de Servidores, Informatización de la Facultad y Sistema de Información Estadístico Complementario de Salud.

En el proceso de diseño de un sistema informático debe considerarse la implementación de mecanismos que permitan la gestión de requerimientos de seguridad. La fortaleza de estos, depende de la sensibilidad e importancia de la información gestionada por el sistema, para evitar la corrupción e inconsistencia de los datos. Asimismo, es importante restringir las operaciones que puedan realizar los

usuarios en los sistemas y llevar un control estricto de estas. Lo que permitirá posteriores procesos de auditoría; garantizando así, el uso correcto de los privilegios asignados a los mismos.

Entre los antecedentes de sistemas cubanos para el control de la seguridad en productos de software se encuentra el Sistema de Autenticación, Autorización y Auditoría (SAAA). Este fue desplegado en diciembre de 2003, como componente de seguridad, para las aplicaciones desarrolladas con el fin de informatizar el Sistema Nacional de Salud (SNS). En su versión actual, no gestiona los requerimientos de seguridad para proyectos como los desarrollados en el Área Temática SAS que está integrada por componentes heterogéneos en cuanto a: negocio, información manipulada y distribuciones físicas.

El SAAA presenta un conjunto de limitaciones: en primer lugar, limitado nivel para la administración de usuarios, puesto que los tipos de usuario están predefinidos (administrador, editor y visualizador); y no pueden ser creados nuevas clasificaciones. En segundo lugar, los roles actuales representan el privilegio que puede ser asignado al usuario sobre las funcionalidades del sistema y no la ocupación que desempeña en el SNS. Por último, varios usuarios pueden acceder a la aplicación simultáneamente, usando la misma credencial de acceso y se otorga la prioridad al último autenticado, trayendo como consecuencia, la privación del acceso al primer usuario.

Ante las limitaciones mencionadas, en el año 2007 se concibió el desarrollo del Centro de Control para el Sistema de Información para la Salud. Donde se desarrolló una versión renovada del componente de seguridad SAAA, este hereda los requerimientos de su antecesor y soluciona algunas de las dificultades ya identificadas, agregando además nuevos requerimientos funcionales.

El Centro de Control aportó los siguientes beneficios:

- Homologación de datos de los usuarios con el Registro de Ciudadanos debido a necesidades de los nuevos módulos en desarrollo.

- El perfeccionamiento de los procesos de gestión de usuarios y asignación de privilegios.

- La eliminación de los usuarios y sus respectivas trazas por el sistema, es lógica y no física, fortaleciéndose el proceso de auditoría.

- Generación automática de ficheros de configuración y despliegue.

A pesar de que este nuevo componente logra mejorar las prestaciones del SAAA, aún existen algunas limitaciones, ellas son:

- Un usuario no puede jugar diferentes roles de gestión a diferentes niveles jerárquicos con un mismo conjunto de credenciales de acceso.

- Un usuario no puede jugar diferentes roles de gestión en un mismo nivel para un módulo determinado.

- Las bases de datos de trazas tienden a crecer con mucha rapidez y actualmente no se tiene en cuenta el tiempo por el cual serán almacenadas las mismas físicamente o una cantidad máxima de almacenamiento, provocando esto que haya que mantener manualmente la acumulación de la información en la base de datos.

- No se pueden hacer reportes sobre las trazas históricas de la aplicación.

- No existe un codificador de roles de usuarios.

- No permite organizar los módulos en grupos de acuerdo al objetivo por el cual fueron creados.

En los proyectos del Área Temática SAS, la gestión de la seguridad resulta de vital importancia para su correcto funcionamiento y para el manejo de sus negocios individuales. El control de la seguridad se realiza de forma independiente en cada proyecto, no existe una estandarización en cuanto a la implementación y utilización de los procesos de seguridad. Los privilegios de los distintos niveles de visibilidad, se asignan individualmente a cada usuario. Los roles son estáticos debido a la no existencia de un codificador de roles y además no se pueden crear grupos de usuarios.

El registro de las trazas históricas consta de una implementación muy básica. No es lo suficientemente profundo, como para llegar a la raíz de todas las acciones que realizan los usuarios dentro del sistema, debido a que no identifica individualmente los accesos y peticiones establecidos de los que pudieran ser no autorizados. No admite que se realicen búsquedas exactas o por parámetros definidos que

permitan identificar rápidamente alguna violación o problema que se haya presentado. No permite hacer reportes personalizados de acuerdo a criterios previamente seleccionados.

Por estas razones, se hace necesario realizar el Componente de Seguridad para las nuevas aplicaciones del Área Temática SAS, para sustituir los mecanismos de seguridad actuales de cada proyecto. Entre las funcionalidades previstas se encuentran: la administración eficiente de los usuarios y la asignación de sus privilegios en dependencia de su nivel. Así como, la optimización de la auditoría, de tal manera que se tenga un control estricto de las operaciones en las que se involucran los usuarios del sistema.

En este sentido, el ***Problema a resolver*** por el Componente de Seguridad es: ¿Cómo fortalecer los procesos de Autenticación, Autorización y Auditoría en los productos desarrollados por el Área Temática Sistemas de Apoyo a la Salud?

Objeto de estudio: Proceso de desarrollo de sistemas informáticos en el Área Temática Sistemas de Apoyo a la Salud.

Campo de acción: Proceso de gestión de requerimientos de seguridad para los productos del Área Temática Sistemas de Apoyo a la Salud.

Para resolver Para resolver el problema identificado se propone el siguiente ***Objetivo general***: Desarrollar un Componente de Seguridad que estandarice los procesos de Autenticación, Autorización y Auditoría en los productos que se desarrollan en el Área Temática Sistemas de Apoyo a la Salud.

Para dar cumplimiento al objetivo anteriormente planteado se definen las siguientes ***Tareas de la investigación:***

1. Asimilación de la Arquitectura definida por el Ministerio de Salud Pública (MINSAP) para el desarrollo de sus aplicaciones (Orientada a Servicios y Basada en Componentes (SOA-CBA)).

2. Realización de un estudio de las tendencias y tecnologías actuales para realizar el proceso de implementación de dicha solución automatizada como por ejemplo PHP, XML, Servicios Web XML, AJAX, Smarty, PostgreSQL, XHTML y JavaScript.

3. Análisis de los procesos de negocio actuales de Autenticación, Autorización y Auditoría (AAA) en los componentes desarrollados en el Área Temática.

4. Estudio de los servicios públicos de los Componentes SAAA, Registro de Unidades de Salud (RUS), Registro del Personal de la Salud (RPS) y Registro de Ciudadanos (RC) del Registro Informatizado de Salud (RIS) para establecer la integración requerida con los mismos.

5. Desarrollo de los Flujos de Trabajo del Proceso Unificado de Desarrollo “Modelo de Negocio”, ”Gestión de Requerimientos”, “Análisis y Diseño” e “Implementación”, para este Componente de Seguridad.

6. Desarrollo de Servicios Web XML públicos para los componentes desarrollados por el Área Temática con sus respectivas descripciones (WSDL).

En sentido general se puede destacar que el desarrollo del Componente de Seguridad proporcionará un grupo de beneficios para las aplicaciones del Área Temática Sistemas de Apoyo a la Salud. Entre ellos se pueden mencionar los siguientes:

- Gestión eficiente de requerimientos de seguridad para todos los sistemas informáticos que pertenecen al Área Temática SAS. Proporciona facilidades de mantenimiento a estos sistemas y permite la agilización del proceso de construcción de las nuevas aplicaciones que se integren al Área. Los desarrolladores podrán obviar los mecanismos de seguridad que serán proporcionados de manera automática por el Componente de Seguridad en el momento de la integración.

- Aumento de los niveles de integración entre los diferentes sistemas del Área Temática, evitando que cada uno posea de manera aislada la administración de sus usuarios, esta información será almacenada y gestionada centralizadamente. De igual manera, permitirá organizar los módulos por grupos permitiendo la unificación de componentes heterogéneos de acuerdo a su negocio.

- Perfeccionamiento de los procesos de gestión de usuarios y asignación de privilegios.

- Gestión eficiente del proceso de Auditoria de los proyectos que conforman el Área Temática, permitiendo hacer reportes de las trazas históricas de acuerdo a diferentes parámetros y controlando la acumulación de información de esta naturaleza en las bases de datos.

- Permite la integración con servidores LDAP, posibilitando la reutilización de los datos de usuarios de un organismo determinado, evitando de esta forma la duplicación de la información en la base de datos.

- Está desarrollado en forma de producto. Su funcionamiento no está limitado sólo a los productos desarrollados por el Área Temática, es capaz de gestionar los requerimientos de seguridad de otros sistemas con características similares.

- Constituye un valor agregado a los productos desarrollados por el Área Temática SAS, debido a que además de resolver su negocio tendrán integrado su propio Sistema de Seguridad.

El presente documento se encuentra estructurado en cuatro capítulos, el primero de ellos, "**FUNDAMENTACIÓN TEÓRICA**", ubica al lector en el Ambiente de Desarrollo del Componente de Seguridad, justificándose las tendencias, tecnologías, metodologías y herramientas que fueron utilizadas para el desarrollo del mismo. Seguidamente el capítulo, "**CARACTERÍSTICAS DEL SISTEMA**", contiene un marco conceptual asociado a la información que será manipulada por el sistema, llegándose a un acuerdo sobre las funcionalidades, requerimientos deseados y el objeto de automatización.

El tercer capítulo **"DISEÑO DEL SISTEMA"** se centra en la modelación detallada y la construcción de la estructura de la aplicación. En el cuarto y último, **"IMPLEMENTACIÓN",** se implementan las clases y subsistemas en términos de componentes. Se presenta la propuesta de solución para lograr una gestión más eficiente de los requerimientos de seguridad de los proyectos que pertenecen al Área Temática SAS.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

El presente capítulo tiene como objetivo abordar los diferentes elementos que brindan la base teórica y conceptual para el desarrollo del Componente de Seguridad. Se exponen elementos de seguridad tomándolos como punto de partida para la concepción de la solución propuesta. Además, se realiza un análisis de las técnicas, tecnologías, metodologías y herramientas de software con las que se realiza el proceso de desarrollo.

1.2 Objeto de automatización e información manipulada

Un sistema o componente con el objetivo de gestionar los requerimientos de seguridad de aplicaciones Web, debe apoyarse en algunos elementos esenciales con el fin de lograr el propósito de su implementación. Estos elementos consisten básicamente en:

- El control de acceso de los diferentes usuarios a las aplicaciones, el cual constituye una poderosa herramienta para proteger la entrada a un sistema completo o sólo a ciertos directorios concretos e incluso a ficheros o programas individuales; este control consta generalmente de dos pasos:

    - **Autenticación:** es el proceso de verificación de la identidad digital de un remitente de una comunicación que hace una petición para conectarse a un sistema. El remitente puede ser una persona que usa un ordenador u otro medio electrónico, un ordenador por sí mismo o un programa. En otras palabras es un modo de asegurar que los usuarios son realmente quienes dicen ser y que tienen la autorización para realizar funciones en el sistema.

    - **Autorización:** proceso por el cual se autoriza al usuario identificado a acceder a determinados recursos del sistema, es decir se comprueba que los usuarios con identidad válida solo tengan acceso a aquellos recursos sobre los cuales tienen privilegios.

- Precisamente teniendo en cuenta estas razones el Componente de Seguridad implementa una fuerte política de Auditoría gracias a la cual quedan registrados todos los accesos y peticiones realizadas por los usuarios, quedando siempre almacenados un conjunto de datos como: usuario, servicio que consume, componente y dirección IP desde el cual accede, fecha, hora, tipo de traza que genera y una descripción que permite aumentar el nivel de detalles acerca de las acciones de los usuarios, contribuyendo de esta forma a facilitar el proceso de análisis de las trazas.

También es posible realizar búsquedas avanzadas de las mismas. Para ello el sistema brinda la posibilidad de utilizar varios parámetros para ir filtrando la información, estos son: nombre del organismo al cual pertenece el usuario, nombre del componente, usuario, tipos de traza, periodo de tiempo y rangos de direcciones IP, con el objetivo de hacer más flexible y eficiente la búsqueda. Otra funcionalidad es la creación de reportes en formato PDF que permite imprimir la información y facilita el proceso de auditoría del sistema, aún cuando no se dispone de un ordenador.

Las bases de datos de trazas tienden a crecer con mucha rapidez. Como una vía alternativa para solventar esta realidad, el sistema brinda la posibilidad de eliminar las trazas de los usuarios. Cuando se elimina una, pasa a formar parte de otra base de datos donde se registra con el mismo formato pero bajo la categoría de traza histórica. En caso de ser eliminada, nunca se pierde el control sobre la misma, debido a que es guardada en un fichero en el servidor, posibilitando su persistencia futura mediante el uso de dispositivos de almacenamiento externo (CD, DVD o Discos extraíbles). Para complementar este proceso, el Componente de Seguridad permite recuperar las trazas que una vez fueron eliminadas y que constituyen una porción de información importante en un momento determinado.

1.3 Seguridad en Aplicaciones Web

En el mundo de hoy, debido al alto grado de interconectividad existente entre ordenadores y otros dispositivos electrónicos; la Web se ha convertido en un importante medio de comunicación e intercambio de información, que ofrece un acceso abierto a un conjunto de datos que explícitamente se hace pública. Por tanto, es necesario limitar el acceso a informaciones reservadas o útiles para un conjunto restringido de personas. Otro aspecto que cobra especial importancia es la seguridad de la

información que se intercambia en la Web, que cada vez está más expuesta a personas que incurren en delitos informáticos.

Según Simson Garfinkel en “Web security, privacy & commerce” la seguridad en la Web está formada por tres etapas fundamentales: [1]

- **Seguridad de la computadora del usuario:** Los usuarios deben contar con navegadores y plataformas seguras, libres de virus y vulnerabilidades. También debe garantizarse la privacidad de los datos del usuario.

- **Seguridad del servidor Web y de los datos almacenados**: Se debe garantizar la operación continua del servidor que los datos no sean modificados sin autorización (integridad) y que la información sólo sea distribuida a las personas autorizadas (control de acceso).

- **Seguridad de la información que viaja entre el servidor Web y el usuario**: Garantizar que la información en tránsito no sea leída (confidencialidad), modificada o destruida por terceros. También es importante asegurar que el enlace entre cliente y servidor no pueda interrumpirse fácilmente (disponibilidad).

1.4 Ataques sobre aplicaciones Web

La evolución de las tecnologías y el progresivo avance de las aplicaciones Web han servido de apoyo para muchas empresas en el mundo, para vender sus productos u ofrecer sus servicios mediante el uso del comercio electrónico. Por otra parte los atacantes han implementado mecanismos con mayor grado de complejidad y eficiencia con el fin de materializar sus amenazas cometiendo diferentes tipos de delitos informáticos como robo y fraude a diferentes niveles. A continuación se exponen los cinco ataques más comunes sobre aplicaciones Web según un estudio realizado por *www.securityfocus.com*: [2]

- Ejecución de código remotamente: Como su nombre lo indica, esta vulnerabilidad permite al atacante ejecutar código en el servidor vulnerable y obtener información almacenada en él.

- Inyección de código SQL: Es una técnica de ataque usada para explotar sitios Web que construyen sentencias SQL a partir de entradas facilitadas por el usuario.

- Ataques de Formato de Cadena: Estos alteran el flujo de una aplicación utilizando las capacidades proporcionadas por las librerías de formato de cadenas para acceder a otro espacio de memoria.

- Cross-site Scripting (XSS): Es una técnica de ataque que fuerza a un sitio Web a repetir un código ejecutable facilitado por el atacante, y que se cargará en el navegador del usuario.

- Enumeración de nombres de usuario: Tipo de ataque donde los mensajes de validaciones de autenticación le dicen al atacante si el nombre de usuario proporcionado es correcto o no. Explotando esta vulnerabilidad el atacante puede experimentar con diferentes nombres de usuario y determinar cuales son válidos.

1.5 Protocolo LDAP

LDAP (*Lightweight Directory Access Protocol*) es un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red. También es considerado una base de datos -aunque su sistema de almacenamiento puede ser diferente- a la que pueden realizarse consultas. Está basado en el estándar X.500. Habitualmente, almacena la información de usuario y contraseña siendo utilizado para autenticarse, aunque es posible almacenar otra información (datos de contacto del usuario, ubicación de diversos recursos de la red, permisos, certificados, entre otros). En sentido general es un protocolo de acceso unificado a un conjunto de información sobre una red. [3]

Un directorio LDAP se destaca entre los demás tipos de bases de datos por las siguientes características: [4]

- Logra mayor rendimiento en la lectura de registros.
- Permite replicar el servidor de forma muy sencilla y económica.
- Muchas aplicaciones de todo tipo tienen interfaces de conexión a LDAP y se pueden integrar fácilmente.

- Dispone de un modelo de nombres globales que asegura que todas las entradas son únicas.
- Usa un sistema jerárquico de almacenamiento de información.
- Permite múltiples directorios independientes.
- Funciona sobre TCP/IP necesario para el acceso a Internet y SSL.
- La mayoría de aplicaciones disponen de soporte para LDAP.
- La mayoría de servidores LDAP son fáciles de instalar, mantener y optimizar.

Dadas las características de LDAP sus usos más comunes son: [5]

- Servidores de certificados públicos y llaves de seguridad.
- Autenticación única o Single Sign On (SSO) para la personalización de aplicaciones.
- Perfiles de usuarios centralizados.
- Libretas de direcciones compartidas.
- Directorios de información.
- Sistemas de autenticación/autorización centralizada.
- Sistemas de correo electrónico.
- Sistemas de alojamiento de páginas Web y FTP.

1.6 Tendencias y Tecnologías Actuales

En este epígrafe se tratan los conceptos fundamentales relacionados con la arquitectura, tecnologías, herramientas y metodología que pueden ser utilizadas para el proceso de desarrollo del Componente de Seguridad.

Es preciso definir qué se entiende por Arquitectura de Software, según Roger Pressman esta representa, ***"(…) la descripción de los subsistemas o componentes de un sistema informático y las relaciones entre ellos (…)".*** Es decir, el conjunto de decisiones significativas sobre la organización del sistema, la selección de los elementos estructurales y las interfaces por las que está compuesto. Describe los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. [6]

Los patrones arquitectónicos y de diseño utilizados para el desarrollo del Componente de Seguridad son: Modelo-Vista-Controlador (MVC), Arquitectura en tres capas, Arquitectura Orientada a Servicios y

Basada en Componentes. Estos patrones no serán analizados según su clasificación natural de diseño o arquitectura, sino como una secuencia lógica que permita justificar su utilización.

1.6.1 Sistemas Distribuidos. Modelo Cliente – Servidor

Un sistema de cómputo distribuido, es una colección de sistemas de cómputo autónomos, llamados nodos. Estos están interconectados a través de una red y de software de comunicaciones, capaces de cooperar para la realización de una tarea común; mientras que la computación distribuida es el conjunto de modelos, técnicas y herramientas computacionales. Las mismas ayudan a resolver los problemas planteados por los sistemas distribuidos en cuanto a la generación, el procesamiento y la utilización de la información requerida para su funcionamiento. [7]

El contexto tecnológico moderno exige la distribución de los sistemas. Se debe tener en cuenta para este proceso, la utilización de una fuerte política en cuanto a compatibilidad, estándares, escalabilidad y tecnologías. La amplia difusión y aceptación de los sistemas distribuidos se debe a un grupo de logros, entre los que se destacan:

- El desarrollo de los microprocesadores, que permitió un descenso en el costo y tamaño de los ordenadores.
- Aumento de las capacidades de los mismos y mayor acceso por parte de los usuarios.
- Desarrollo de las redes de área local y de las comunicaciones que permitieron la conexión de ordenadores con posibilidad de transferencia de datos a alta velocidad.

Entre las principales ventajas de los sistemas distribuidos se encuentran: [8]

- **Rapidez de respuesta y rendimiento:** Los recursos se encuentran compartidos, de manera que una petición determinada pueda ser procesada con mayor velocidad.
- **Fiabilidad:** En el caso de los sistemas no distribuidos, si el servidor principal falla, el sistema colapsa en su totalidad, no ofreciéndose ningún servicio hasta que se repare. Sin embargo si se

realiza una distribución del sistema, la salida de servicio de un servidor no implica que se dejen de brindar los servicios que no hayan sido afectados.

- **Escalabilidad:** Este tipo de sistema puede crecer más fácilmente que uno centralizado. Pues si se decide mejorar su productividad es necesario adquirir un nuevo servidor principal, mientras que si desea mejorar uno distribuido basta con incrementar el número de servidores que conforman el sistema.

- **Modularidad y reutilización en el desarrollo:** Esto permite la agilización de los procesos de desarrollo pues se tendrá en cuenta para implementar un nuevo componente las funcionalidades que podrá obtener de otro ya desplegado.

Todos estos elementos, no descartan la existencia de desventajas en los sistemas distribuidos, entre las más significativas están: el uso ineficiente de los recursos distribuidos, la capacidad reducida para administrar apropiadamente grupos de procesadores y memoria localizada en distintos sitios. Además, se complejiza el proceso de administración y mantenimiento y existe una gran dependencia del desempeño de la red y de la confiabilidad de la misma.

El modelo de arquitectura Cliente-Servidor de un sistema distribuido, es el más conocido y difundido en la actualidad. Hay un conjunto de procesos servidores, cada uno actuando como un administrador para una colección de recursos de un tipo; y una colección de procesos clientes, cada uno llevando a cabo una tarea que requiere acceso a algunos recursos hardware y software compartidos. Es decir, el proceso cliente es el encargado de permitir al usuario formular requerimientos y enviarlos al servidor, quien es el autorizado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. El servidor gestiona las funciones relacionadas con las reglas del negocio y los accesos a datos.

Algunas ventajas del modelo Cliente - Servidor son:

- La estructura inherentemente modular facilita la integración a nuevas tecnologías con sistemas heterogéneos y crecimiento de la infraestructura computacional, favoreciendo así la escalabilidad de las soluciones.

- Permite una agilización en el desarrollo y mantenimiento de aplicaciones debido a que se pueden utilizar herramientas existentes (por ejemplo servidores de SQL, o herramientas de bajo nivel como sockets o RPC).
- Considerable reducción del tráfico en la red, debido a que el cliente se conecta al servidor sólo cuando necesita un servicio; una vez que se obtiene la respuesta, la conexión se cierra dejándose libre la red.

Por otra parte deben tenerse en cuenta las desventajas siguientes:

- Es necesario crear estrategias para el manejo de errores, para mantener la consistencia de los datos y gestionar la seguridad de un esquema Cliente - Servidor.

- Pobre desempeño del esquema Cliente - Servidor cuando existen elevados niveles de congestión en la red provocando dificultad de tráfico de datos.

1.6.2 Patrones de arquitectura y diseño

Un patrón en sentido general define una posible solución correcta para un problema de diseño dentro de un contexto dado, describiendo las cualidades invariantes de todas las soluciones. Estos pretenden: proporcionar catálogos de elementos reusables en el diseño de sistemas software, evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente y estandarizar el modo en que se realiza el diseño. Existen varios tipos de patrones, dependiendo del contexto particular en el cual se aplican o de la etapa en el proceso de desarrollo, algunos de estos tipos son: Patrones de Diseño, de Arquitectura, para Ambientes Distribuidos, entre otros.

1.6.2.1 Arquitectura Orientada a Servicios y Basada en Componentes (SOA-CBA)

Las instituciones exigen aplicaciones más complejas cada vez, con menos tiempo y presupuesto. Para crear estas aplicaciones, se requiere en muchos casos de funcionalidades ya implementadas como parte de otros sistemas. La Arquitectura Orientada a Servicios (SOA) nace como una estrategia de integración y constituye una tendencia creciente que intenta reconciliar la visión técnica y de negocios,

basándose en estándares abiertos y promoviendo la interoperabilidad entre diversas organizaciones y plataformas de manera eficiente y flexible a los cambios.

Actualmente todos los proveedores de tecnología están abocados a soportar este tipo de arquitecturas tanto en empresas pequeñas en crecimiento como en grandes corporaciones debido a que SOA no es una moda pasajera, sino que es una alternativa que permite enfrentarse a los retos de las organizaciones, que necesitan una respuesta flexible y rápida en entornos competitivos y de alta complejidad.

SOA define diferentes capas de software, estas son: aplicativa básica, que son sistemas desarrollados bajo cualquier arquitectura o tecnología, geográficamente dispersos y bajo cualquier figura de propiedad. La segunda es la de exposición de funcionalidades, donde las mismas son expuestas en forma de servicios. Otra es, la de integración de servicios, esta facilita el intercambio de datos entre elementos de la capa aplicativa orientada a procesos empresariales internos o en colaboración. Además, la de composición de procesos, que los definen en términos del negocio y sus necesidades, y varia en función del negocio. Por último, la de entrega, donde los servicios son desplegados a los usuarios finales.

La Arquitectura Orientada a Servicios nos brinda un conjunto de beneficios. Algunos de ellos son: [9]

- La reducción de costos y tiempo en el desarrollo de aplicaciones: SOA permite reutilizar los módulos de aplicaciones existentes para resolver problemas comunes a otras aplicaciones. Como consecuencia también se reducen los costos de mantenimiento y se logra aumentar la robustez del nuevo sistema, al utilizarse software ya probado.

- Facilita el proceso de integración: se interactúa con elementos que se abstraen de la ubicación de los servicios y tecnología con la que fueron creados los mismos.

Existen varias definiciones de componente, una de ellas plantea que en sentido general es una pieza de software que posee una interfaz y función bien definida. El desarrollo de software basado en componentes se centra en la construcción de aplicaciones complejas mediante ensamblado de módulos o componentes que han sido previamente diseñados por otras personas a fin de ser rehusados en múltiples aplicaciones. Entre las características más relevantes de esta tecnología de

programación se pueden citar la modularidad y la reusabilidad, rasgos en los que coincide con la tecnología orientada a objetos de la que se puede considerar una evolución. [10]

Algunas de las ventajas de desarrollar software basado en componentes son las que se comentan a continuación: [11]

- **Reutilización de software:** Nos lleva a alcanzar un mayor nivel de reutilización.

- **Simplifica las pruebas:** Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados.

- **Simplifica el mantenimiento del sistema:** Cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar las otras partes del sistema.

- **Mayor calidad:** Dado que un componente puede ser construido y luego mejorado continuamente por un experto u organización, la calidad de una aplicación basada en componentes mejorará continuamente.

1.6.2.2.1 Tecnología Servicios Web XML

La comunicación hacia y desde un servicio, es realizada utilizando mensajes y no llamadas a métodos, como se realiza en el paradigma de la Programación Orientada a Objetos. Estos mensajes deben contener o referenciar toda la información necesaria para ser comprendidos internamente por la aplicación que lo ha solicitado. Un Servicio es un sistema de software diseñado para soportar interacción máquina a máquina los cuales pueden ser descritos, publicados, localizados e invocados a través de la red utilizando protocolos estándar.

Los Servicios Web usan SOAP (Simple Object Access Protocol) como protocolo para invocar llamadas remotas debido a su simplicidad, se puede identificar un mensaje SOAP como un documento XML conformado por una envoltura obligatoria, un encabezamiento opcional y un cuerpo también obligatorio. Este permite la comunicación entre aplicaciones heterogéneas, de modo que clientes de

diferentes plataformas o lenguajes de programación pueden comunicarse entre sí de manera satisfactoria.

Alrededor de los Servicios Web existen protocolos y mecanismos adicionales para facilitar tareas como el descubrimiento de servicios distribuidos a lo largo de la red o UDDI (Universal Description, Discovery and Integration), una descripción del contenido de los mensajes o WSLD (Web Services Description Language) el cual describe la forma de comunicación, es decir, los requerimientos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Además se utiliza el protocolo HTTP (Hypertext Transfer Protocol) para el transporte de la mensajería, utilizando el puerto 80 ya que el mismo siempre se encuentra accesible.

1.6.2.1 Alta cohesión y bajo acoplamiento

Los patrones Alta cohesión y Bajo acoplamiento pertenecen a la familia de GRASP (General Responsibility Assignment Software Patterns). El primero plantea que debe haber pocas dependencias entre las clases; de tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las mismas; mientras que el segundo propone que cada elemento del diseño debe realizar una labor única dentro del sistema. Ejemplos de una baja cohesión son clases que tienen muchas responsabilidades. En todas las metodologías se considera la refactorización. Uno de los elemento a refactorizar son las clases saturadas de métodos. [12]

1.6.2.3 Arquitectura en tres capas

La arquitectura en tres capas es una especialización de la arquitectura Cliente - Servidor donde la carga se divide en tres partes (o capas) con un reparto bien definido de funciones: una capa para la presentación (también conocida como interfaz de usuario), otra para el modelado de las reglas del negocio y otra para la persistencia de los datos. Esta arquitectura esta basada en un nivel de abstracción ascendente lo cual permite a los desarrolladores de software dividir un problema complejo en una secuencia de pasos incrementales. También permite una amplia reutilización, debido a que los datos abstractos pueden ser utilizados por diferentes implementaciones de una misma capa en la medida que se soporten las mismas interfaces. [13]

1.6.2.4 Modelo Vista Controlador

Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes diferentes. El patrón MVC se ve con mayor frecuencia en aplicaciones Web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es la Base de Datos y el controlador constituye la Lógica de Negocio. En este sentido se pueden describir los elementos fundamentales del MVC de la siguiente forma: [14]

- **Modelo:** Administra el comportamiento y los datos del dominio de la aplicación, responde a requerimientos de información sobre su estado, usualmente formulados desde la vista, respondiendo a instrucciones de cambio para cambiar el estado de estos datos, habitualmente desde el controlador.

- **Vista:** Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

- **Controlador:** Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

1.6.2.5 Single Sign On (SSO)

Los usuarios para identificarse ante varias aplicaciones informáticas son forzados a recordar numerosas contraseñas por lo que en la mayoría de los casos eligen contraseñas sencillas poniendo potencialmente en riesgo la seguridad del sistema. Existen muchas organizaciones que poseen arquitecturas que han sido diseñadas con el objetivo de dar acceso a los usuarios a múltiples servicios Web y/o aplicaciones. En la mayoría de los casos se encuentra que cada uno de los servicios o aplicaciones cuenta con su propio componente o mecanismo de seguridad, lo que puede comprometer la seguridad de todo el sistema. El nivel de seguridad de todo un sistema es igual al nivel de seguridad del componente más inseguro que lo compone.

Precisamente como alternativa a estos problemas de seguridad, surge el Single Sign On (SSO) que no es más que es un proceso de autenticación de sesión/usuario; que permite mediante un solo conjunto de credenciales acceder a diferentes aplicaciones, esto es principalmente utilizado en ambientes distribuidos. El proceso autentica al usuario para todas las aplicaciones a los que les han dado derechos y elimina todos los mensajes de autenticación que se generan cuando se cambia de interfaz durante una sesión particular.

Este es un elemento de mucha importancia a tener en cuenta en el proceso de desarrollo del Componente de Seguridad. Permite el encapsulamiento de la infraestructura de seguridad subyacente, posibilitando que los procesos de implementación, despliegue y mantenimiento sean más fáciles; ninguna de las partes comunicantes en el sistema distribuido, necesita implementar individualmente todos los mecanismos de seguridad. Este tipo de proceso permite a los desarrolladores y organizaciones centrarse en el desarrollo de la lógica de negocio asociada a un componente en particular, obviando los mecanismos de seguridad, debido a estos serán proporcionados de manera automática por el Componente de Seguridad.

1.6.3 Lenguajes utilizados para el proceso de desarrollo

En esta sección se realizará un análisis de los lenguajes que se utilizan para el proceso de implementación de las Capas de Negocio y Presentación del Componente de Seguridad. En todos los casos se abordará la definición de estos y se expondrán algunas de sus características, así como las ventajas que influyeron en su elección.

1.6.3.1 PHP

PHP (Hypertext Pre-processor) es un lenguaje de programación interpretado, es decir que no requiere compilación, creado en 1994 por Rasmus Lerdof. Su uso más frecuente es en la creación de páginas Web dinámicas y aplicaciones para servidores. Con frecuencia los scripts PHP se embeben en otros códigos como HTML aumentando las posibilidades del diseñador de páginas Web. La interpretación y ejecución de estos scripts se hacen en el servidor, el cliente (solicitud realizada desde un navegador Web) sólo recibe el resultado y jamás ve el código PHP. Permite conexión con todo tipo de bases de datos como MySQL, Postgre SQL, Oracle, DB2, Microsoft SQL Server. PHP es soportado por 7

plataformas, funciona en 11 tipos de servidores y ofrece soporte para varios Sistemas Gestores de Bases de Datos (SGBD).

Algunas de las capacidades más importantes de PHP son: [15]

- Integración con varias bibliotecas externas, permitiendo generar documentos Portable Document Format (PDF) y Microsoft Office Excel (XLS).

- Ofrece una solución simple y universal para las paginaciones dinámicas de fácil programación.

- Soportado por una gran comunidad de desarrolladores, como producto de código abierto, permitiendo que los fallos de funcionamiento se encuentren y reparen rápidamente, implicando menos costos.

- Gran número de funciones predefinidas. A diferencia de otros lenguajes de programación, PHP fue diseñado especialmente para el desarrollo de páginas Web dinámicas. Por ello, está dotado de un gran número de funciones que simplificará enormemente tareas habituales como descargar documentos, envío de correo electrónico, creación dinámica de imágenes y gráficos en el servidor, procesamiento de información en formularios, manipulación de cookies y sesiones, transporte de información mediante HTTP y análisis de documentos XML.

- Análisis léxico para reconocer el tipo de dato almacenado en una variable haciéndose automáticamente, permitiéndole al usuario no tener que separar las variables de sus valores.

- Posee un conjunto de funciones de seguridad que previenen la inserción de órdenes dentro de una solicitud de datos desde el cliente evitando por ejemplo, la ocurrencia de la conocida inyección de Código SQL.

- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos denominados extensiones.

- Permite la implementación mediante el Paradigma de la Programación Orientada a Objetos (POO).

- No requiere definición de tipos de variables.

Este potente lenguaje de programación también posee algunas desventajas entre ellas se pueden citar que todo el trabajo se realiza del lado del servidor, no delegando responsabilidades al cliente, puede que en un determinado momento la capacidad de respuesta sea ineficiente en la medida en que las peticiones al servidor aumenten considerablemente. Por sus características, promueve la creación de código desordenado y complejo de mantener. No posee una abstracción de base de datos estándar, sino bibliotecas especializadas para cada motor.

PHP actualmente se encuentra en su versión 5, aunque se espera próximamente el lanzamiento de su versión 6. El PHP 5, utiliza el motor Zend Engine II o Zend Engine 2 y fue lanzado el 13 de julio de 2004. La versión más reciente de PHP es la 5.2.5 estando centrada en mejorar la estabilidad con cerca de 60 errores solucionados, algunos de ellos relativos a la seguridad. Las ventajas que provee el nuevo Zend Engine 2 son:

- Soporte sólido y real para POO con PHP Data Objects.
- Mejoras de rendimiento.
- Mejor soporte para MySQL con extensión completamente reescrita.
- Mejor soporte a XML (XPath, DOM).
- Soporte nativo para SQLite.
- Soporte integrado para SOAP.
- Iteradores de datos.
- Manipulación del mecanismo de errores o excepciones.

1.6.3.2 JavaScript

JavaScript de forma similar al PHP es un lenguaje interpretado, fue inventado por Brendan Eich en la empresa Netscape Communications y fue implementado por primera vez en el explorador Netscape Navigator 2.0. El código JavaScript es embebido directamente en el código HTML o similares,

haciendo fácil la creación de páginas Web con contenido dinámico. Está diseñado para controlar la apariencia y manipular los eventos de los objetos contenidos en un formulario, además de ser soportado por la mayoría de los navegadores existentes (Konqueror, Internet Explorer y Netscape Navigator, Opera, Mozilla.). El mismo comparte muchos elementos con lenguajes de alto nivel como C, Java o PHP, tanto en su formato como en su sintaxis. [16]

El código Javascript se ejecuta en el cliente logrando de esta forma que el servidor de aplicaciones sea solicitado solo cuando sea necesario; mejorando de esta forma su rendimiento debido a que los servidores de capacidades más limitadas podrían resentir de una continua solicitud por un mayor número de usuarios; la seguridad en este lenguaje es casi total, sólo en su primera versión se le señalaron problemas de leve entidad, entre ellos la lectura de la caché y de los sitios visitados, de la dirección e-mail y de los archivos presentes en el disco. Sin embargo, estos fallos se corrigieron ya en las versiones de Netscape sucesivas a la 2.0. También se debe tener en cuenta que los programas escritos en este lenguaje tienden a ser pequeños y compactos, lo cual influye en que no se requiera ni memoria ni tiempo adicional para su transmisión.

1.6.3.3 YUI

Las YUI (Yahoo User Interface) constituyen una serie de librerías escritas en JavaScript y Cálculo de Sistemas Comunicantes de código abierto, para la construcción de aplicaciones, liberadas bajo licencia BSD por parte de la compañía Yahoo. Estas librerías son utilizadas para el desarrollo Web específicamente en la programación de aplicaciones de escritorio, con componentes vistosos y personalizables y con una amplia implementación con AJAX. Provee una gran cantidad de utilidades en JavaScript y controles de interfaz de usuario, lo que permite la homogenización de las interfaces visuales de los productos de software. Entre sus características principales se encuentran: [17]

- La capacidad de generar código ordenado para el programador.
- La posibilidad de reutilizar controladores, el manejo de AJAX.
- La personalización de los componentes y el soporte por parte de Yahoo Developers.
- Posee elementos para depurar, una sección de pruebas.
- Posee la alta cantidad de widgets y componentes pre-hechos.

1.6.3.4 Lenguajes de Marcas

El intercambio de información siempre ha constituido un problema cuando se utilizan lenguajes y sistemas operativos incongruentes y heterogéneos, debido a esto fue necesaria la creación de mecanismos para lograr el intercambio fluido de información entre diferentes sistemas, los lenguajes de marcas. En sentido general, estos son una forma de codificar un documento que, junto con el texto, incorpora marcas que contienen información adicional acerca de la estructura del texto o su presentación. Proporcionan un conjunto de etiquetas o tags, que se entremezclan con el propio contenido en un único archivo o flujo de datos.
Existen tres clases de lenguajes de marcado, aunque en la práctica pueden combinarse varias clases en un mismo documento siendo estas el marcado de presentación, el marcado procedimental y el marcado descriptivo o semántico, los cuales se comentan a continuación:

- **Marcado de presentación** es aquel que indica el formato del texto. Este tipo de marcado es útil para maquetar la presentación de un documento para su lectura, pero resulta insuficiente para el procesamiento automático de la información. El marcado de presentación resulta más fácil de elaborar, sobre todo para cantidades pequeñas de información. Sin embargo resulta complicado de mantener o modificar, por lo que su uso se ha ido reduciendo en proyectos grandes en favor de otros tipos de marcado más estructurados.

- **Marcado procedimental** está enfocado hacia la presentación del texto, sin embargo, también es visible para el usuario que edita el texto. El programa que representa el documento debe interpretar el código en el mismo orden en que aparece. Por ejemplo, para formatear un título, debe haber una serie de directivas inmediatamente antes del texto en cuestión, indicándole al software instrucciones tales como centrar, aumentar el tamaño de la fuente, o cambiar a negrita. Inmediatamente después del título deberá haber etiquetas inversas que reviertan estos efectos. En sistemas más avanzados se utilizan macros o pilas que facilitan el trabajo.

- **Marcado descriptivo o semántico** utiliza etiquetas para describir los fragmentos de texto, pero sin especificar cómo deben ser representados o en qué orden, lo cual le da al mismo una gran flexibilidad. El marcado descriptivo también simplifica la tarea de reformatear un texto, debido a que la información del formato está separada del propio contenido.

1.6.3.4.1 HTML

HTML (HyperText Markup Language), creado en 1989 por Tim Berners-Lee como un subconjunto del SGML (Standard Generalized Markup Language). Es un lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas Web. El proyecto inicial se basaba en una colección de etiquetas que permitían describir documentos de texto y vínculos de hipertexto que hacían posible el desplazamiento en forma jerárquica entre diferentes documentos. La facilidad de su uso y la particularidad de no ser propiedad de nadie, hizo de HTML el sistema ideal para compartir información a través de Internet. [18]

1.6.3.4.2 XML

XML (eXtensible Markup Language), se deriva del SGML, optimizado para su uso en la Web, es un lenguaje de marcas, diseñado especialmente para la comunicación entre aplicaciones a través de la red. Presenta un grupo de características que permiten que los desarrolladores creen sus propias etiquetas, permitiendo la definición, transmisión, validación e interpretación de datos enviados y recibidos, independientemente de las tecnologías que se utilicen. XML logra que la información sea más accesible y reutilizable, porque la flexibilidad de las etiquetas puede utilizarse sin tener que adaptarse a reglas específicas de un fabricante, como es el caso de HTML. Es un lenguaje fácil de implantar, programar y aplicar a los distintos sistemas en los que se decida utilizar. [19]

1.6.3.4.3 XHTML

XHTML (eXtensible Hypertext Markup Language) es el lenguaje de marcado, pensado para sustituir a HTML como estándar para las páginas Web. XHTML es la reformulación de HTML 4.01 de acuerdo a las reglas de XML 1.0, por lo que tiene, básicamente, las mismas funcionalidades, pero cumple las especificaciones más estrictas de XML. Su objetivo es avanzar en el proyecto del W3C (World Wide Web Consortium) para lograr una Web semántica, donde la información, y la forma de presentarla estén separadas. En este sentido, XHTML serviría únicamente para transmitir la información que contiene un documento, dejando para hojas de estilo y JavaScript su aspecto y diseño en distintos medios como computadoras, ordenadores de mano (PDA) y teléfonos móviles, por citar algunos ejemplos. [20]

El lenguaje XHTML presenta un conjunto de ventajas sobre otros formatos, algunas de ellas son:

- Compatibilidad parcial con navegadores antiguos: la información se visualiza, aunque sin formato. XHTML 1.0 fue diseñado expresamente para ser mostrado en navegadores que soportan HTML de base.
- Un mismo documento puede adoptar diseños radicalmente distintos en diferentes dispositivos, pudiendo incluso escogerse entre varios diseños para un mismo medio.
- Facilidad de edición directa del código y de mantenimiento.

- Formato abierto, compatible con los nuevos estándares que actualmente está desarrollando el W3C como recomendación para futuros agentes de usuario o navegadores.

- Los documentos escritos conforme a XHTML 1.0 pueden potencialmente presentar mejor rendimiento en las actuales herramientas Web que aquellos escritos conforme a HTML.

1.6.3.5 Protocolo HTTPS

HTTPS (Hypertext Transfer Protocol Secure), es un protocolo de red basado en el protocolo HTTP, destinado a la transferencia segura de datos de hipertexto, es decir, es la versión segura de HTTP.

El sistema HTTPS utiliza un cifrado basado en las SSL (Secure Socket Layers) para crear un canal cifrado -cuyo nivel de cifrado depende del servidor remoto y del navegador utilizado por el cliente- más apropiado para el tráfico de información sensible que el protocolo HTTP. Es válido destacar que el uso del protocolo HTTPS no impide que se pueda utilizar HTTP. [21]

Los protocolos HTTPS son utilizados por navegadores como:

- Safari.
- Internet Explorer.
- Mozilla Firefox.
- Opera.

1.6.4 Tecnologías utilizadas en el proceso de desarrollo

1.6.4.1 Smarty

Es común que en grandes proyectos el rol de diseñador gráfico y el de programador sean cubiertos por personas distintas, sin embargo la programación utilizando PHP tiende a combinar estas dos labores en una persona y dentro del mismo código lo que trae consigo grandes dificultades a la hora de cambiar alguna parte del diseño de la página, Smarty tiene como objetivo principal solucionar este problema. Es un motor de plantillas para PHP que se encarga de separar el código PHP, como lógica de negocios, del código HTML, como lógica de presentación y genera contenidos Web mediante la colocación de etiquetas Smarty en un documento. Se encuentra bajo la licencia LGPL por lo que puede ser usado libremente. [22]

Smarty permite programar plantillas con un gran número de funcionalidades, como:

- Archivos de configuración.
- Filtros.
- Modificadores de variables.
- Funciones creadas por el usuario.
- Plugins.
- Caching (uso de la caché).

1.6.4.2 AJAX

AJAX (Asynchronous JavaScript And XML) es una metodología para el desarrollo de aplicaciones Web interactivas o RIA (Rich Internet Applications) que no constituye una tecnología en sí, sino una combinación de cuatro tecnologías ya existentes: [23]

- XHTML (o HTML) y hojas de estilos en cascada (CSS) para el diseño que acompaña a la información.

- DOM (Document Object Model) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones ECMAScript como JavaScript y JScript, para mostrar e interactuar dinámicamente con la información presentada.

- El objeto XMLHttpRequest para intercambiar datos asincrónicamente con el servidor Web. En algunos frameworks y en algunas situaciones concretas, se usa un objeto iframe en lugar del XMLHttpRequest para realizar dichos intercambios.

- XML es el formato usado comúnmente para la transferencia de vuelta al servidor, aunque cualquier formato puede funcionar, incluyendo HTML preformateado, texto plano, JSON y hasta EBML.

El código AJAX se ejecuta en el cliente, es decir, en el navegador de los usuarios y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla, permitiendo el aumento de la interactividad, velocidad y usabilidad en ella. Es soportado por un gran número de los navegadores más populares como: Mozilla Firefox, Netscape versión 7.1 y superiores Microsoft Internet Explorer para Windows versión 5.0 y superiores.

1.6.5 Sistema de Gestión de Base de Datos

Un Sistema de Gestión de Base de Datos (SGBD) es un conjunto de programas que permiten crear y mantener una Base de Datos. Por lo tanto el SGBD es un software de propósito general que facilita a un usuario procesos como definición de una base de datos (especificación de tipos, estructuras y restricciones de datos), construcción (guardar los datos en algún medio controlado por el mismo SGBD) y manipulación (realizar consultas, actualizarla, generar informes) asegurando al mismo tiempo la integridad, confidencialidad y seguridad de la Base de datos.

Existen diversos tipos de SGBD algunos ejemplos son: Oracle, Microsoft SQL Server, Postgre SQL, MySQL, Firebird pero en sentido general todos convergen en un conjunto de objetivos que les permitirá manejar de una forma clara, sencilla y ordenada una colección de datos que posteriormente se convertirá en información, los cuales se comentan a continuación: [24]

- **Abstracción de la información**. Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. El hecho de que una base de datos ocupe uno o cientos de archivos se hace transparente al usuario.

- **Redundancia mínima**. Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante.

- **Consistencia**. En aquellos casos en los que no se ha logrado una redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.

- **Seguridad**. La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra segura frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero despistado.

- **Integridad**. Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.

- **Respaldo y recuperación**. Los SGBD deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.

- **Control de la concurrencia**. En la mayoría de entornos es habitual que sean muchas las personas que acceden a una base de datos de forma simultánea, tanto para recuperar o almacenar información. El SGBD debe ser capaz de controlar estos accesos concurrentes a la información, que podría derivar en inconsistencias.

- **Tiempo de respuesta**. Es deseable que el tiempo de respuesta del SGBD ante diferentes solicitudes sea mínimo.

1.6.5.1 PostgreSQL

PostgreSQL es uno de los SGBD más populares desarrollado bajo la filosofía de código abierto, liberado bajo la BSD (Licencia de Software Distribuidos) que aproxima sus datos a un modelo objeto - relacional. Este gestor de Bases de Datos además de ser potente, soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes y cadenas de bits, también permite la creación de tipos propios e incorpora una estructura de datos array que constituye un conjunto o agrupación de variables del mismo tipo cuyo acceso se realiza por índices. Entre sus principales ventajas constan las siguientes: [25]

- **Añade funciones de diversa índole:** Manejo de fechas, geométricas y orientadas a operaciones con redes.

- **Permite la declaración de funciones propias:** Así como la definición de disparadores.

- **Soporta el uso de índices, reglas y vistas:** Incluye herencia entre tablas (aunque no entre objetos, ya que no existen) por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.

- **Admite la gestión de diferentes usuarios:** Como también los permisos asignados a cada uno de ellos

.
Por otro lado, la velocidad de respuesta que ofrece este gestor con bases de datos relativamente pequeñas puede parecer un poco deficiente, aunque esta misma velocidad la mantiene al gestionar bases de datos realmente grandes, cosa que resulta loable.

1.6.5.2 Servidor Web Apache

Es uno de los servidores Web más usados del mundo, desarrollado bajo la filosofía de código abierto y capaz de funcionar en diversas plataformas como: Unix (BSD, GNU/Linux) Windows, Macintosh y otras, implementa el protocolo HTTP/1.1 y la implementación de host virtuales. Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido. Representa un complemento perfecto para desarrollo de páginas Web

dinámicas con PHP y PostgreSQL porque comparte muchas de sus características como: gratuidad, popularidad, su sencillez de manejo y versatilidad.

1.6.5.3 Proceso Unificado de Desarrollo de Software (RUP)

Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requerimientos de un usuario en un software. Sin embargo, RUP es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, tipos de organizaciones, niveles de aptitud y tamaños de proyecto. El Proceso Unificado está basado en componentes, lo cual quiere decir que el sistema o software en construcción está formado por componentes interconectados a través de interfaces.

RUP utiliza el UML para representar todos los esquemas del sistema de software a desarrollar, de hecho UML es una parte esencial de RUP, pero sus verdaderos aspectos definitorios se resumen en tres ideas claves, dirigidas por casos de uso, centrado en la arquitectura e iterativas e incrementales. Es decir, un sistema de software se define para dar servicios a sus usuarios, por lo tanto para construirlo con éxito se debe conocer qué es lo que los futuros usuarios necesitan así como que para llevar a cabo el proceso de desarrollo es conveniente dividir el trabajo en partes pequeñas o miniproyectos, representando cada uno de ellos lo que se conoce como iteración. [26]

1.6.5.4 Lenguaje Unificado de Modelado (UML)

UML es un lenguaje estándar de modelado para software, que permite la visualización, especificación, construcción y documentación de los artefactos de sistemas donde se manejan conceptos orientados a objetos. Básicamente UML permite a los desarrolladores visualizar los resultados de su trabajo en esquemas y diagramas estandarizados. UML ofrece un estándar para describir un modelo del sistema, incluyendo aspectos conceptuales tales como procesos de negocios, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. Nació en el año 1994 por iniciativa de Grady Booch y Jim Rumbaug. [27]

1.6.6 Herramientas a utilizar

Con los argumentos dados en los epígrafes anteriores y con un conocimiento del problema en cuestión, se pueden definir las herramientas que se utilizan para el desarrollo del Componente de Seguridad. En primer lugar se utilizará el Rational Rose Enterprise Edition 2003 como herramienta CASE (Computer-Aided Software Engineering) basada en UML, que permite crear los diagramas que se generen como parte de la documentación. Para la creación de las páginas Web, Dreamweaver 8 debido a que soporta el lenguaje de marcas XHTML 1.0, para la edición de código PHP, se utiliza el Zend Studio 5 y para administrar la base de datos el SQL Manager 2005 for PostgreSQL, debido a que es un ambiente gratuito de sencilla utilización que posee una interfaz muy intuitiva.

1.7 Conclusiones

En este capítulo, se profundizó en los conceptos y definiciones necesarias para comprender el proceso de desarrollo del Componente de Seguridad. Este es un sistema de administración y seguridad, que favorece el funcionamiento de las aplicaciones del Área Temática Sistemas de Apoyo a la Salud. Además se realizó, un análisis de las tecnologías, metodologías, lenguajes y herramientas que se tienen en cuenta para llevar a cabo proceso de desarrollo, basado completamente en software no propietario, lo cual obedece a las políticas definidas por el Área Temática SAS y la Facultad.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA.

2.1 Introducción

En este capítulo se realiza una descripción de la propuesta de solución para el Componente de Seguridad. Debido a la poca organización de los procesos de negocio y para poder comprender mejor el contexto en el cual se desarrolla el sistema, se acuerda desarrollar un Modelo de Dominio donde se expone un marco conceptual y se establecen las relaciones entre los conceptos que lo conforman. Por otra parte, se especifican los requerimientos funcionales y no funcionales, agrupándose los primeros en casos de uso, con el fin de estructurar el Diagrama de Casos de Uso del sistema. De la misma forma se justifican los usuarios finales que interactuarán con el Componente de Seguridad.

2.2 Modelo de Dominio

El Modelo de Dominio o Modelo Conceptual, constituye una representación visual para el usuario de los conceptos u objetos del mundo real significativos para un problema o área de interés. Representa conceptos del mundo real, no de los componentes de software. A diferencia de los otros modelos propuestos por la metodología de desarrollo RUP, este no está formado por un conjunto de diagramas que describen clases u objetos de software con responsabilidades, sino que puede considerarse como un diccionario visual de las abstracciones más relevantes en el dominio de definición del producto. Aprovechando las bondades de los Diagramas UML para representar conceptos, el Modelo de Dominio se muestra en forma de diagramas de clases donde figuran las principales definiciones y roles del sistema en cuestión.

Se decide realizar un Modelo de Dominio debido a que no pueden ser identificados los procesos de negocio asociados al contexto del sistema, como consecuencia de la poca estructuración y fronteras de los mismos. Las definiciones que se interrelacionan fueron obtenidas sobre la base del conocimiento de especialistas del Área Temática SAS y de la experiencia asociada a la versión actual del Centro de Control. También fue recomendable realizar un Modelo de Dominio debido a que los conceptos que serán expuestos representan objetos del mundo real a los cuales el Componente de Seguridad debe darles un estricto seguimiento en su proceso de desarrollo.

2.2.1 Conceptos fundamentales

En el siguiente epígrafe se proporciona un marco conceptual con las definiciones identificadas en el contexto del proceso de desarrollo del Componente de Seguridad, para ayudar a una mejor comprensión del Diagrama del Modelo de Dominio:

Administrador General: Actor que en dependencia de su nivel de acceso sobre un organismo cuyos requerimientos de seguridad son gestionados por el Componente de Seguridad, tiene los permisos necesarios para gestionar la información de los usuarios que en la estructura jerárquica de niveles defina por su organismo, posean un nivel igual o inferior al suyo. Esta gestión incluye crear usuarios, modificar sus datos y privilegios, eliminarlos y realizar búsquedas a partir de diferentes parámetros. Por otra parte puede realizar una auditoría estricta a través de las trazas almacenadas por el sistema, conociendo qué usuario ha participado en cada transacción.

Administrador Configuración: Actor que tiene permiso total sobre la configuración del sistema. Único encargado de la gestión de organismos, niveles, ubicaciones, componentes, servicios y roles.

Usuario: Actor que interactúa directamente con el Componente de Seguridad una vez que se hayan superado las fases de desarrollo correspondientes, con el objeto de consultar, modificar o eliminar la información gestionada por el mismo.

Componente: Es una unidad ejecutable que representa el núcleo de la aplicación, la cual puede ser implantada independientemente y ser a la vez sujeto de composición de terceras partes, es decir, se puede tomar el componente y agregarlo a otro componente en desarrollo o simplemente consumir algunos de los servicios que brinda.

Certificado: Conjunto de datos y privilegios de un usuario determinado que se crea de forma automática durante los procesos de autenticación y autorización. El mismo posee un identificador único de treinta y dos caracteres que se genera de manera aleatoria, contiene el nivel de acceso del usuario, el identificador del nivel de acceso y un listado de los componentes a los que tiene derecho de acceso, así como los privilegios de ejecución correspondientes en estos componentes.

Nivel de Actividad: Clasificación del usuario que permite diferenciar entre un usuario activo (que está autorizado a utilizar el sistema dentro de un período de tiempo determinado) y uno de tipo inactivo (contrarresta la definición anterior).

Traza: Historial donde se almacenan todos los eventos realizados por el usuario al interactuar con la información perteneciente al sistema.

Rol: Papel que cumple un usuario dentro de un componente y que limita el conjunto de funcionalidades que puede desempeñar en ese ámbito dentro del sistema.

Servicio: Operación proporcionada por un componente determinado.

Organismo: Conjunto de dependencias, oficinas o empleos que cumplen con determinadas leyes, usos y costumbres y que a la vez forman una institución social.

Nivel: Concepto asociado a las diferentes instancias de dirección administrativa de un organismo determinado.

Ubicación: Recoge la ubicación exacta del usuario en cada uno de los niveles en que se desempeñe.

Grupo de Nivel: Especifica los diferentes niveles que tiene cada uno de los organismos.

Grupo de Rol: Se recogen todos los roles que pertenecen a cada organismo.

Período de Actividad: Es el tiempo durante el cual el usuario va a estar en estado activo dentro de la aplicación.

Historial de Traza: Se especifica los tipos de trazas definidos que pueden dejar los usuarios.

2.2.2 Diagrama del Modelo de Dominio

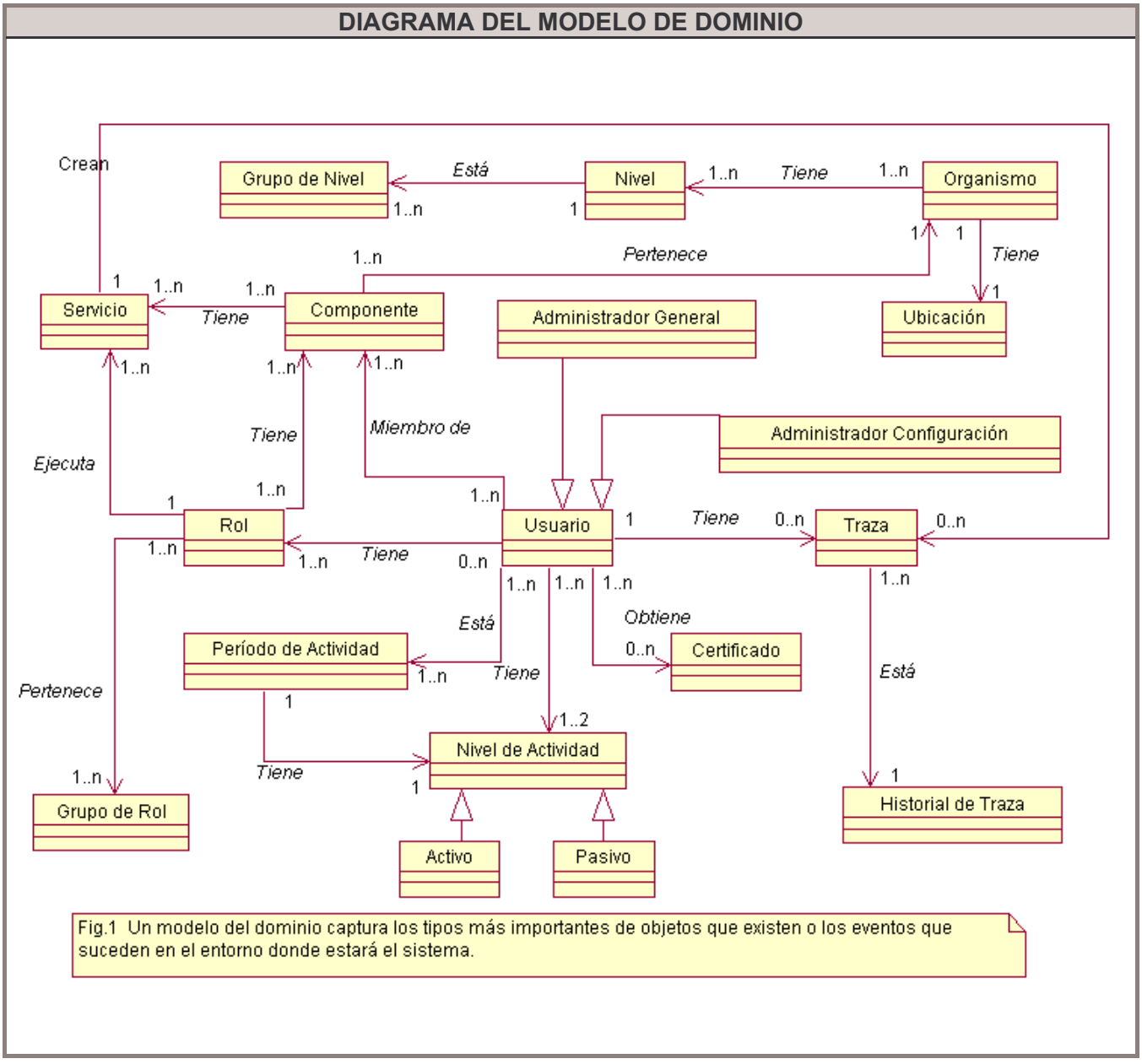


Figura 2.1 Diagrama del Modelo del Dominio.

2.3 Propuesta de Sistema

Un sistema informático está formado por varios elementos como: hardware, software y soporte humano. Generalmente la elaboración de un sistema comienza conformando una visión global del ambiente donde será implantado, el cual está comprendido en el Modelo de Dominio, de gran utilidad para analizar cada uno de los conceptos reconocidos, así como sus interrelaciones con el fin de identificar todos los requerimientos que se deben cumplir. Una vez que estos hayan sido identificados, el Modelo del Sistema puede ser realizado.

2.3.1 Especificación de los Requerimientos de Software

La especificación de los requerimientos de software constituye un elemento de vital importancia para la elaboración de un software de calidad superior. El IEEE Standard Glossary of Software Engineering Terminology define los requerimientos de software como condiciones o capacidades que deben estar presentes en un sistema o componentes de este, para satisfacer un contrato, estándar, especificación u otro documento formal. Es necesario hacer énfasis en la precisión con que se debe realizar esta tarea por cumplir un papel primordial en el proceso de producción de software, pues se enfoca en un área fundamental: la definición de lo que se desea producir, permitiendo describir con mayor claridad el comportamiento del sistema minimizando los problemas derivados de su desarrollo.

2.3.1.1 Requerimientos Funcionales

Los requerimientos funcionales son condiciones o capacidades que el sistema debe cumplir. Estos definen el comportamiento interno de un software: cálculos, manipulación de datos y otras funcionalidades específicas. Una vez examinados los conceptos fundamentales descritos con mayor precisión en el Modelo de Dominio, se obtuvo el siguiente levantamiento de requerimientos funcionales.

LISTADO DE REQUERIMIENTOS FUNCIONALES	
RF1- Autenticar	**RF29**- Eliminar Ubicación
RF2- Autorizar	**RF30**- Listar Componente
RF3- Crear Certificado Digital	**RF31**- Buscar Usuario
RF4- Crear Lista de Privilegio	**RF32**- Buscar Usuario Histórico
RF5- Registrar Traza	**RF33**- Listar Organismo
RF6- Adicionar Componente	**RF34**- Listar Tipo Traza
RF7- Modificar Componente	**RF35**- Buscar Traza
RF8- Eliminar Componente	**RF36**- Buscar Traza Histórica
RF9- Adicionar Usuario	**RF37**- Listar Rol
RF10- Modificar Usuario	**RF38**- Listar Nivel
RF11- Eliminar Usuario	**RF39**- Listar Servicios
RF12- Adicionar Organismo	**RF40**- Listar Ubicación
RF13- Modificar Organismo	**RF41**- Cambiar Contraseña
RF14- Eliminar Organismo	**RF42**- Imprimir Búsqueda Usuario
RF15- Adicionar Tipo Traza	**RF43**- Imprimir Búsqueda Traza
RF16- Modificar Tipo Traza	**RF44**- Crear Fichero Traza
RF17- Eliminar Tipo Traza	**RF45**- Recuperar Fichero Traza
RF18- Adicionar Rol	**RF46**- Asignar Componente
RF19- Modificar Rol	**RF47**- Asignar Organismo
RF20- Eliminar Rol	**RF48**- Asignar Rol
RF21- Adicionar Nivel	**RF49**- Asignar Nivel
RF22- Modificar Nivel	**RF50**- Asignar Ubicación
RF23- Eliminar Nivel	**RF51**- Asignar Tipo Traza
RF24- Adicionar Servicio	**RF52**- Asignar Usuario
RF25- Modificar Servicio	**RF53**- Asignar Servicio
RF26- Eliminar Servicio	**RF54**- Eliminar Traza
RF27- Adicionar Ubicación	**RF55**- Modificar Perfil
RF28- Modificar Ubicación	

Tabla 2.1 Listado de Requerimientos Funcionales.

A continuación los requerimientos funcionales serán agrupados en casos de uso del sistema. Estos últimos representan fragmentos de funcionalidades que el sistema ofrece y al mismo tiempo constituyen la forma en que los usuarios lo utilizan. En este sentido los casos de uso identificados son los siguientes:

DEFINICIÓN DE CASOS DE USO	
CUS1: Autenticar **RF1**- Autenticar **RF2**- Autorizar **RF3**- Crear Certificado Digital **RF4**- Crear Lista de Privilegios	**CUS2: Modificar_Perfíl** **RF41**- Cambiar Contraseña **RF55**- Modificar Perfíl
CUS3: Gestionar_Usuario **RF9**- Adicionar Usuario **RF10**- Modificar Usuario **RF11**- Eliminar Usuario **RF31**- Buscar Usuario **RF32**- Buscar Usuario Histórico **RF42**- Imprimir Búsqueda Usuario **RF51**- Asignar Usuario **RF52**- Asignar Usuario	**CUS4: Gestionar_Componente** **RF6**- Adicionar Componente **RF7**- Modificar Componente **RF8**- Eliminar Componente **RF30**- Listar Componente **RF46**- Asignar Componente
CUS5: Gestionar_Organismo **RF12**- Adicionar Organismo **RF13**- Modificar Organismo **RF14**- Eliminar Organismo **RF33**- Listar Organismo **RF47**- Asignar Organismo	**CUS6: Gestionar_Tipo_Traza** **RF5**- Registrar Tipo Traza **RF15**- Adicionar Tipo Traza **RF16**- Modificar Tipo Traza **RF17**- Eliminar Tipo Traza **RF34**- Listar Tipo Traza **RF36**- Buscar Traza Histórica **RF43**- Imprimir Búsqueda Traza **RF44**- Crear Fichero Traza **RF45**- Recuperar Fichero Traza **RF51**- Asignar Tipo Traza

CUS7: Gestionar_ Servicio **RF24**- Adicionar Servicio **RF25**- Modificar Servicio **RF26-** Eliminar Servicio **RF39**- Listar Servicios **RF53**- Asignar Servicio	**CUS8: Gestionar_ Nivel** **RF21**- Adicionar Nivel **RF22**- Modificar Nivel **RF23**- Eliminar Nivel **RF38**- Listar Nivel **RF49**- Asignar Nivel
CUS9: Gestionar_ Rol **RF18**- Adicionar Rol **RF19**- Modificar Rol **RF20**- Eliminar Rol **RF37**- Listar Rol **RF48**- Asignar Rol	**CUS10: Gestionar_Ubicación** **RF27**- Adicionar Ubicación **RF28**- Modificar Ubicación **RF29**- Eliminar Ubicación **RF40**- Listar Ubicación **RF50**- Asignar Ubicación
CUS11: Buscar_Usuario **RF31**- Buscar Usuario	**CUS12: Registrar_Traza** **RF5**- Registrar Traza
CUS13: Eliminar_Traza **RF54**- Eliminar Traza	

Tabla 2.2 Definición de Casos de Uso del Sistema.

2.3.1.2 Requerimientos No Funcionales

Los requerimientos no funcionales son cualidades o propiedades que el producto debe tener, es decir, restricciones en el producto que está siendo desarrollado. Estos no describen lo que el software hará, sino cómo lo hará y son muy importantes debido a que permite a los clientes valorar las características no funcionales del producto como: usabilidad, rendimiento, portabilidad, que junto a las funcionalidades esperadas del software ayudarán a marcar la diferencia entre un producto bien aceptado y uno con poca aceptación. Es preciso tener en cuenta que la definición de estos cobra un valor adicional porque los errores debidos a requerimientos no funcionales son difíciles y caros de resolver.

2.3.1.2.1 Usabilidad

RNF1: El sistema debe garantizar un acceso fácil y rápido. Podrá ser usado por cualquier usuario que posea pocos conocimientos informáticos y de un ambiente Web en sentido general.

2.3.1.2.2 Rendimiento

RNF2: El sistema debe tener una similitud entre sus páginas y estar poco cargado con imágenes, posibilitando que se devuelvan respuestas de una manera eficiente en un tiempo mínimo, siendo más sencillo de entender y utilizar por el usuario.

2.3.1.2.3 Soporte

RNF3: Una vez terminado el desarrollo del Componente de Seguridad se llevarán a cabo los procesos de despliegue, capacitación y mantenimiento de software con motivo de asistir a los clientes del producto, así como lograr su mejoramiento progresivo, evolución y sostenibilidad en el tiempo. Además se contará con un grupo de trabajo designado para el mantenimiento y actualización continuos del sistema.

2.3.1.2.4 Portabilidad

RNF4: Permitir que el sistema se ejecute sobre el Sistema Operativo Linux, Windows 98 o superior.

2.3.1.2.5 Seguridad

RNF5: Disponer de un mecanismo de seguridad basado en el modelo de Autenticación, Autorización y Auditoría (AAA).

- **Confiabilidad:** La información manejada por el sistema está protegida contra acceso no autorizado.

- **Integridad:** Que la información sea modificada -incluyendo su creación y eliminación- sólo por personal autorizado. Se implementarán políticas de resguardo de información, así como la realización de copias periódicas de seguridad. Se permitirá la creación de copias de respaldo que puedan restaurar el sistema en caso de fallo crítico o pérdida total de la información.

- **Disponibilidad:** Los usuarios autorizados tendrán acceso a la información en todo momento.

RNF5.1: La autenticación será la primera acción del usuario en el sistema y consiste en suministrar un nombre único de usuario y una contraseña que debe ser de conocimiento exclusivo de la persona que se está involucrada en esta operación.

RNF5.2: Cada petición de usuario autorizada o no será registrada, almacenándose día, mes, año, hora, minuto y segundo en que fue solicitada dicha petición.

RNF5.3: El sistema controlará la cantidad de caracteres mínimo y máximo para el nombre de usuario y contraseña, el registro de las cuentas creadas por el administrador y no permitirá conexiones concurrentes con un mismo conjunto de credenciales.

RNF5.4: Se utilizará el protocolo HTTPS para el transporte de información a través de la Web.

RNF5.5: Aplicar políticas de seguridad y acceso a los servidores donde sea desplegado el Componente de Seguridad.

2.3.1.2.6 Apariencia o Interfaz Externa

RNF6: La interfaz debe ser sencilla y amigable ya que el usuario no es experto en el uso de las aplicaciones Web. Diseño sencillo, con pocas entradas, permitiendo que no sea necesario mucho entrenamiento para utilizar el sistema, además que exista paginación de reportes de búsqueda y listados.

RNF6.1: El sistema debe presentar un diseño que permita una clara distinción entre los elementos de las ventanas, formularios a través del uso de colores, tamaño de las fuentes, iconografía, así como otros recursos.

2.3.1.2.7 Ayuda y documentación en línea

RNF7: Se contará con una Ayuda en línea así como un Manual de Usuario en formato PDF que indicará al usuario cómo interactuar con las funcionalidades del sistema.

2.3.1.2.8 Software

RNF8.1: Para clientes: Se tendrá acceso al Componente de Seguridad a través de cualquier navegador Web. Recomendados: Firefox Mozilla 1.5, Internet Explorer 5.0 o superior que soporte DHTML y CSS2.

RNF8.2: Para Capa de Presentación: El servidor debe tener PHP Versión 5.2.5 o superior. Servidor HTTP preferiblemente Apache. Sistema Operativo LINUX distribución Debian 4 Etch.

RNF8.3: Para Capa de Datos: Servidor de Base de Datos PostgreSQL Versión 8.2. Sistema Operativo LINUX, distribución Debian 4 Etch.

2.3.1.2.9 Hardware

RNF9.1: Requerimientos mínimos para clientes:

- Ordenador Pentium o superior.
- 64 MB de Memoria RAM.
- Monitor VGA o superior.
- Teclado y Mouse.
- Procesador 486DX / 66 MHZ o superior.
- Disco duro de 20 GB.
- Impresora de puntos.
- Insumos. (Disquetes, CD- RW, Papel continuo y cintas de impresora).
- La PC de trabajo debe estar conectada a una Red de Área Local (LAN).

2.3.2 Modelo de Casos de Uso del Sistema

El Modelo de Casos de Uso del Sistema es un artefacto de Ingeniería de Software que describe, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario, permitiendo de esta forma el establecimiento de un acuerdo entre clientes y desarrolladores sobre las condiciones y requerimientos que debe cumplir el sistema. Este modelo está formado por actores, casos de usos y las relaciones que se establecen entre estos, es decir representa gráficamente a los procesos y su interacción con los actores y constituye una entrada de gran valor para las siguientes fases de construcción de un software.

2.3.2.1 Definición de los actores

Un actor del sistema es una persona o la abstracción de un software que interactúa de alguna manera con el sistema: puede intercambiar información con él, ser un recipiente pasivo de información, representar el rol que juegan una o varias personas, un equipo o un sistema automatizado. A continuación se definen los actores del sistema en desarrollo:

ACTORES	JUSTIFICACIÓN
Usuario	Actor que interactúa directamente con el Componente de Seguridad una vez que haya superado sus fases de desarrollo correspondientes, con el objeto de consultar, modificar o eliminar la información gestionada por el mismo.
Administrador General	Actor que en dependencia de su nivel de acceso sobre un organismo cuyos requerimientos de seguridad son gestionados por el Componente de Seguridad, tiene los permisos necesarios para gestionar la información de los usuarios que en la estructura jerárquica de niveles defina por su organismo, posean un nivel igual o inferior al suyo. Esta gestión incluye crear usuarios, modificar sus datos y privilegios, eliminarlos y realizar búsquedas a partir de diferentes parámetros. Por otra parte puede realizar una auditoría estricta a través de las trazas almacenadas por el sistema, conociendo qué usuario ha

	participado en cada transacción.
Administrador Configuración	Actor que tiene permiso total sobre la configuración del sistema. Único encargado de la gestión de organismos, niveles, ubicaciones, componentes, servicios y roles.

Tabla 2.3 Justificación de Actores del Sistema.

2.3.2.1.1 Vista global de los Actores del Sistema

En el siguiente diagrama, se puede observar cómo han sido separados en dos usuarios diferentes, la administración del sistema. En la propuesta de actores del Componente de Seguridad, se propone separar las labores de administración en actores diferentes: el Administrador General y el Administrador de Configuración, con el objetivo de eliminar la complejidad que supone la gestión de toda la información que maneja el componente por un mismo usuario.

El Administrador General será el encargado de la gestión de usuarios y las trazas, mientras que el Administrador de Configuración está facultado a gestionar el resto de las configuraciones necesarias para lograr el funcionamiento correcto del Componente de Seguridad, entre las cuales se encuentra la gestión de organismos, niveles, ubicaciones, componentes, servicios y roles.

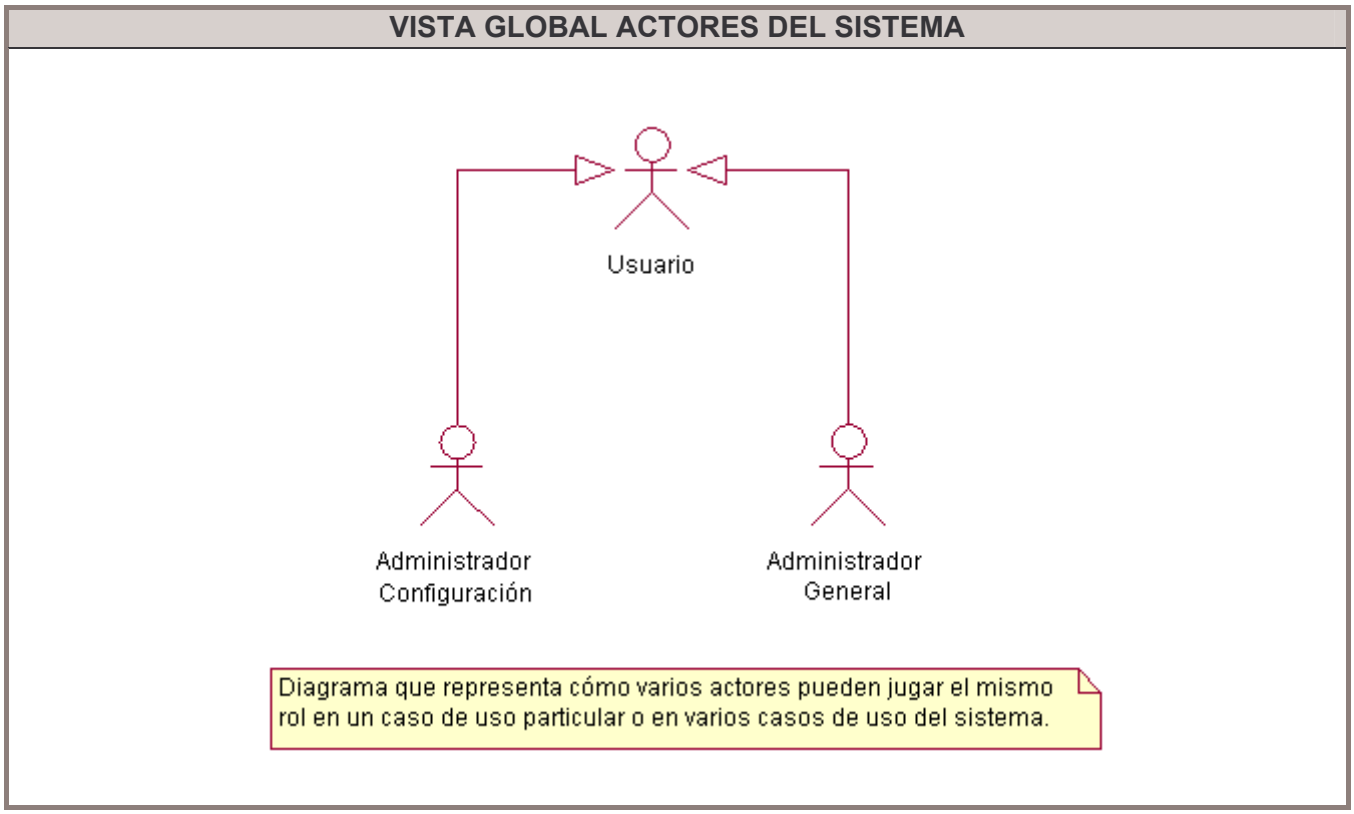


Figura 2.1 Vista Global Actores del Sistema.

2.3.2.2 Diagrama de Casos de Uso

En este importante diagrama se representan las relaciones entre actores y casos de uso del sistema. A continuación se muestra el Diagrama de Casos de Uso del Componente de Seguridad:

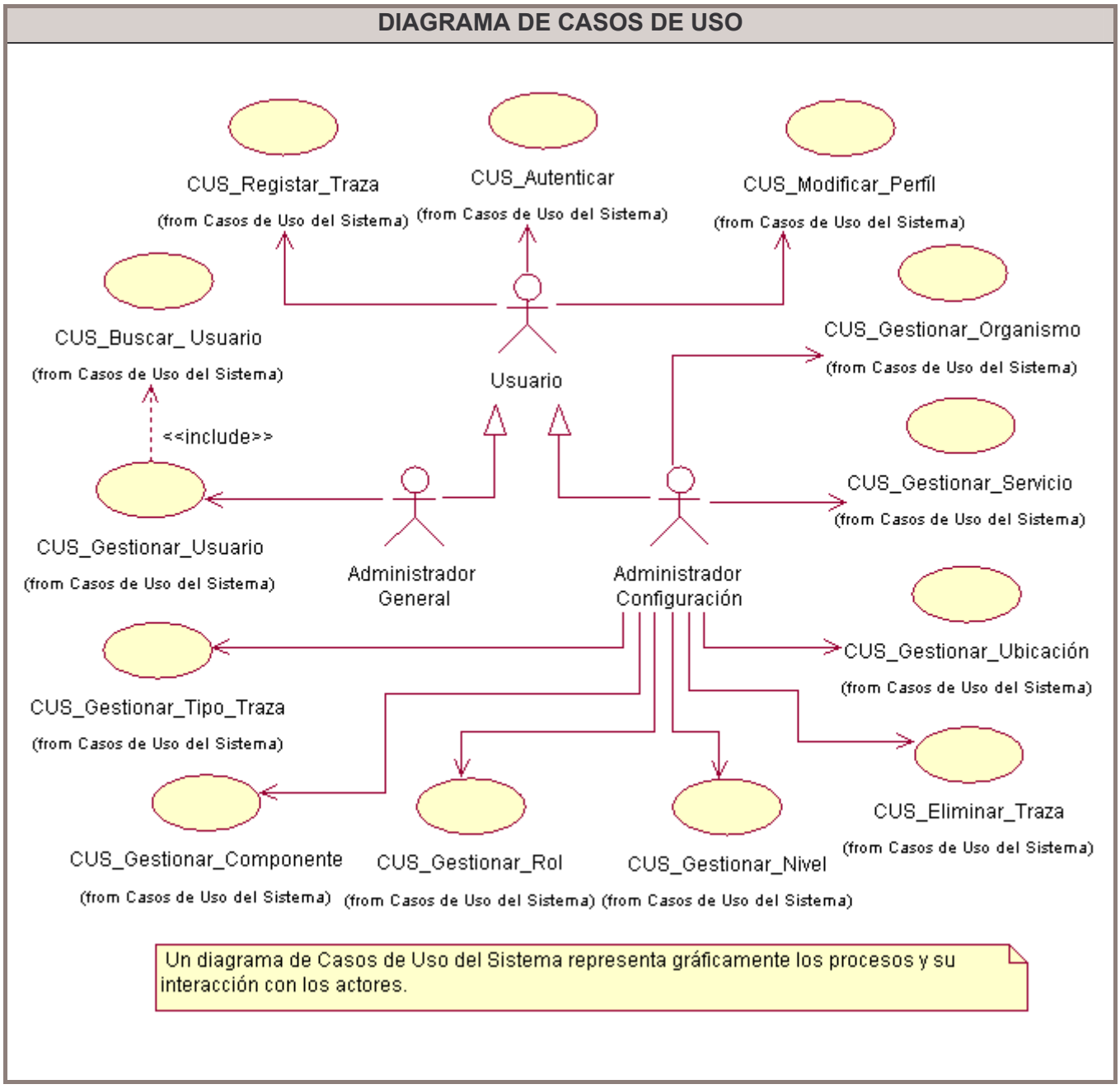


Figura 2.2 Diagrama de Casos de Uso.

2.3.2.3 Descripción Textual de los Casos de Uso

A continuación se describen los casos de uso identificados, mediante los cuales se satisfacen los requerimientos funcionales del sistema en desarrollo:

CUS1_Autenticar	
Propósito:	Permitir la entrada al sistema.
Actores:	**Usuario** (Inicia)
Resumen:	El caso de uso inicia cuando un usuario intenta acceder al Componente de Seguridad. Este introduce su nombre de usuario y contraseña, accediendo al sistema con los privilegios que le fueron otorgados. El caso de uso termina cuando el actor haya accedido al sistema satisfactoriamente.
Precondición:	El usuario debe conocer el nombre de usuario y contraseña que le fue asignado en el momento en que el Administrador General haya registrado a dicho usuario.
Poscondición:	Permite la entrada del usuario al Componente de Seguridad. Si no se realiza la operación, el sistema muestra un mensaje al actor.
Referencias:	**RF1, FR2, RF3, RF4**
Prioridad:	Crítico

Tabla 2.4 Descripción textual *CUS1_ Autenticar.*

CUS2_ Modificar_Perfíl	
Propósito:	Permitir a un usuario determinado cambiar su contraseña.
Actores:	**Usuario** (Inicia)
Resumen:	El caso de uso inicia cuando un usuario desea cambiar su contraseña, escribe la nueva contraseña y la rectifica, el sistema muestra un mensaje de confirmación de la realización de la operación, terminado de esta forma el caso de uso.
Precondición:	El usuario debe haber sido autenticado en el sistema.
Poscondición:	Permite que la contraseña sea modificada. Si no se realiza la operación, el sistema presenta un mensaje al actor.
Referencias:	**RF41, RF55**
Prioridad:	Auxiliar

Tabla 2.5 Descripción textual *CUS2_Modificar_Perfil.*

CUS3_ Gestionar_Usuario [include: CUS_Buscar_Usuario]	
Propósito:	Gestionar la información relacionada con los usuarios del Sistema de Seguridad.
Actores:	**Administrador General** (Inicia)
Resumen:	El caso de uso inicia cuando el actor accede al Sistema de Seguridad para realizar acciones sobre un usuario: insertarlo, buscarlo, eliminarlo o modificarlo. Si desea insertar uno nuevo debe especificar algunos datos como el período de actividad, correo electrónico, nombre de usuario (login) especificando si este usuario es administrador o no, en caso de no serlo se le asignan los privilegios. Si el actor desea eliminar o modificar un usuario primeramente tiene que buscarlo, una vez localizado, es realizada la acción deseada. Si el actor desea puede exportar los resultados de las búsquedas realizadas. El caso de uso termina cuando el actor termine las opciones de gestión deseadas sobre los usuarios.
Precondición:	El usuario debe haber sido autenticado en el sistema.
Poscondición:	El sistema deja actualizada la información perteneciente al usuario. Si no se realiza la operación, el sistema muestra un mensaje al actor.
Referencias:	**RF9, R10, RF11, RF31, RF32, RF42, RF51, RF52.**
Prioridad:	Crítico

Tabla 2.6 Descripción textual *CUS3_Gestionar_Usuario [include: CUS_Buscar_Usuario].*

CUS4_Gestionar_Componente	
Propósito:	Gestionar la información relacionada con los componentes.
Actores:	**Administrador Configuración** (Inicia)
Resumen:	El caso de uso inicia cuando el actor intenta realizar todas las operaciones relacionadas con los componentes: insertarlo, buscarlo, eliminarlo o modificarlo. Si el actor desea crear un componente, llena todos los datos generales que lleva este en su creación. Si desea eliminar o modificar un componente ya insertado, primeramente tiene que buscarlo para realizar la acción deseada, concluyendo así el caso de uso.
Precondición:	El usuario debe haber sido autenticado en el sistema.
Poscondición:	El sistema deja actualizada la información perteneciente al componente. Si no se realiza la operación, el sistema muestra un mensaje al actor.
Referencias:	**RF6, RF7, RF8, RF30, RF46.**
Prioridad:	Crítico

Tabla 2.7 Descripción textual *CUS4_ Gestionar_Componente.*

CUS5_ Gestionar_Organismo	
Propósito:	Gestionar la información relacionada con los organismos.
Actores:	**Administrador Configuración** (Inicia)
Resumen:	El caso de uso inicia cuando el actor intenta realizar todas las operaciones relacionadas con los organismos: adicionarlos, modificarlos, eliminarlos y buscarlos. Si desea crear un organismo, llena todos los datos generales necesarios para su creación, debe insertarle los niveles así como las ubicaciones para cada uno de los niveles. Si desea eliminar o modificar un organismo ya insertado, primeramente tiene que buscarlo para realizar la acción deseada, concluyendo así el caso de uso.
Precondición:	El usuario debe haber sido autenticado en el sistema.
Poscondición:	El sistema deja actualizada la información perteneciente al organismo. Si no se realiza la operación, el sistema muestra un mensaje al actor.
Referencias:	**RF12, RF13, RF14, RF33, RF47.**
Prioridad:	Crítico

Tabla 2.8 Descripción textual *CUS5_Gestionar_Organismo.*

CUS6_Gestionar_Tipo_Traza	
Propósito:	Obtener las trazas registradas en el sistema.
Actores:	**Administrador General** (Inicia)
Resumen:	El caso de uso inicia cuando el actor intenta realizar todas las operaciones relacionadas con las trazas como: insertarlo, buscarlo, eliminarlo o modificarlo. si desea obtener un listado de trazas registradas puede realizar la búsqueda por varios parámetros, una vez llenado estos parámetros, se le mostrará las trazas que tiene registradas el sistema que coinciden con estos parámetros, si el actor desea puede imprimir el resultado obtenido, concluyendo el caso de uso.
Precondición:	El usuario debe haber sido autenticado en el sistema.
Poscondición:	El actor obtiene las trazas. Si no se encontró ningún resultado el sistema le muestra un mensaje al actor.
Referencias:	**RF5, RF15, RF16, RF17, RF34, RF36, RF43, RF44, RF45.**
Prioridad:	Crítico

Tabla 2.9 Descripción textual *CUS6_Gestionar_Tipo_Traza.*

CUS7_Gestionar_ Servicio	
Propósito:	Gestionar la información relacionada con los servicios.
Actores:	**Administrador Configuración** (Inicia)
Resumen:	El caso de uso inicia cuando el actor intenta realizar todas las operaciones relacionadas con los servicios como: insertarlo, buscarlo, eliminarlo o modificarlo., si el actor desea adicionar un servicio, llena todos los datos necesarios para su creación, además debe especificar el rol que puede desempeñar este servicio, si desea eliminar o modificar un servicio ya insertado, primeramente tiene que buscarlo para realizar la acción deseada, concluyendo así el caso de uso.
Precondición:	El usuario debe haber sido autenticado en el sistema.
Poscondición:	El sistema deja actualizada la información perteneciente a los servicios. Si no se realiza la operación, el sistema muestra un mensaje al actor.
Referencias:	**RF24, RF25, RF26, RF39, RF53.**
Prioridad:	Crítico

Tabla 2.10 Descripción textual *CUS7_Gestionar_Servicio.*

CUS8_Gestionar_Nivel	
Propósito:	Gestionar la información relacionada con los niveles.
Actores:	**Administrador Configuración** (Inicia)
Resumen:	El caso de uso inicia cuando el actor intenta realizar todas las operaciones relacionadas con los niveles como: insertarlo, buscarlo, eliminarlo o modificarlo., si el actor desea adicionar un nivel, llena todos los datos necesarios que lleva el nivel en su creación, además debe especificar a qué organismo va a pertenecer, si desea eliminar o modificar un nivel ya insertado, primeramente tiene que buscarlo para realizar la acción deseada, concluyendo así el caso de uso.
Precondición:	El usuario debe haber sido autenticado en el sistema.
Poscondición:	El sistema deja actualizada la información perteneciente a los niveles. Si no se realiza la operación, el sistema muestra un mensaje al actor.
Referencias:	**RF21, RF22, RF23, RF38, RF49.**
Prioridad:	Crítico

Tabla 2.11 Descripción textual *CUS8_Gestionar_Nivel.*

CUS9_Gestionar_Rol	
Propósito:	Gestionar la información relacionada con los roles.
Actores:	**Administrador Configuración** (Inicia)
Resumen:	El caso de uso inicia cuando el actor intenta realizar todas las operaciones relacionadas con el rol como: insertarlo, buscarlo, eliminarlo o modificarlo, si el actor desea adicionar un rol, llena todos los datos necesarios que lleva el rol en su creación donde debe especificar a qué organismo va a pertenecer, si desea eliminar o modificar un rol ya insertado, primeramente tiene que buscarlo para realizar la acción deseada, concluyendo así el caso de uso.
Precondición:	El usuario debe haber sido autenticado en el sistema.
Poscondición:	El sistema deja actualizada la información perteneciente a los roles. Si no se realiza la operación, el sistema muestra un mensaje al actor.
Referencias:	**RF18, RF19, RF20, RF37, RF48.**
Prioridad:	Crítico

Figura 2.12 Descripción textual *CUS-9: Gestionar_Rol.*

CUS10_Gestionar_Ubicación	
Propósito:	Gestionar la información relacionada con la ubicación.
Actores:	**Administrador de Configuración** (Inicia)
Resumen:	El caso de uso inicia cuando el actor intenta realizar todas las operaciones relacionadas con la ubicación como: insertarlo, buscarlo, eliminarlo o modificarlo y además asignarlas a las usuarios, si el actor desea adicionar una nueva ubicación a un organismo tiene que llenar todos los datos necesarios que lleva la ubicación en su creación, si desea eliminar o modificar una ubicación ya insertada, primeramente tiene que buscarla para realizar la acción deseada, concluyendo así el caso de uso.
Precondición:	El usuario debe haber sido autenticado en el sistema.
Poscondición:	El sistema deja actualizada la información perteneciente a la ubicación. Si no se realiza la operación, el sistema muestra un mensaje al actor.
Referencias:	**RF27, RF28, RF29, RF40, RF50.**
Prioridad:	Crítico

Tabla 2.13 Descripción textual *CUS10_Gestionar_Ubicación.*

CUS11_Buscar_Usuario	
Propósito:	Buscar la información relacionada con los usuarios del sistema.
Actores:	**Administrador General** (Inicia)
Resumen:	El caso de uso inicia cuando el actor intenta realizar la búsqueda de los usuarios, una vez obtenido un listado de los usuarios puede imprimir el listado, concluyendo así el caso de uso.
Precondición:	El usuario debe haber sido autenticado en el sistema.
Poscondición:	El sistema muestra al usuario la información perteneciente a la búsqueda de los usuarios. Si no se realiza la operación, el sistema muestra un mensaje al actor.
Referencias:	**RF31**
Prioridad:	Secundario

Tabla 2.14 Descripción textual *CUS11_ Buscar Usuario.*

CUS12_ Registrar_Traza	
Propósito:	Permite registrar las trazas por el usuario.
Actores:	**Usuario** (Inicia)
Resumen:	El caso de uso inicia cuando el actor registra las trazas realizadas en el sistema, concluyendo así el caso de uso.
Precondición:	El usuario debe haber sido autenticado en el sistema.
Poscondición:	El sistema deja actualizada la información perteneciente al registro de las trazas. Si no se realiza la operación, el sistema muestra un mensaje al actor.
Referencias:	**RF5**
Prioridad:	Secundario

Tabla 2.15 Descripción textual *CUS12_Registrar_Traza.*

CUS13_ Eliminar_Traza	
Propósito:	Permite eliminar las trazas por el administrador configuración.
Actores:	**Administrador Configuración** (Inicia)
Resumen:	El caso de uso inicia cuando el administrador elimina las trazas realizadas en el sistema, concluyendo así el caso de uso.
Precondición:	El usuario debe haber sido autenticado en el sistema.
Poscondición:	El sistema deja actualizada la información perteneciente cuando se elimina alguna traza. Si no se realiza la operación, el sistema muestra un mensaje al actor.
Referencias:	**RF54**
Prioridad:	Secundario

Tabla 2.16 Descripción textual *CUS13_Eliminar_Traza.*

2.4 Conclusiones

El desarrollo de este capítulo ha permitido alcanzar un mejor entendimiento del sistema que se desea construir así como las restricciones que deben existir para satisfacer las necesidades de los clientes. Se especificaron todos los requerimientos funcionales y no funcionales del sistema, se identificaron los actores que intervienen, así como todos los casos de uso, que fueron descritos de forma detallada reflejando las funcionalidades recogidas previamente en los requerimientos. El desarrollo de este flujo de trabajo y los artefactos obtenidos a partir del mismo, nos permite comenzar con el Flujo de Trabajo de Diseño, que requiere de mayor esfuerzo por parte del equipo de desarrollo para la construcción de la solución de software propuesta.

CAPÍTULO 3: DISEÑO DEL SISTEMA

3.1 Introducción

En el Diseño se modela y adquiere una forma del sistema para que soporte los requerimientos funcionales o no funcionales, contribuyendo a obtener una arquitectura sólida y estable para la futura implementación del sistema de software. En tal sentido, el propósito de este capítulo está encaminado a adquirir una comprensión de los aspectos relacionados con los requerimientos, lenguajes de programación, componentes reutilizables, tecnologías de distribución y concurrencia. Entre los artefactos que serán mostrados en el presente capítulo se encuentran: Modelo de Diseño, especificándose la estructura y la definición de los elementos que este posee; Diagramas de Clases y descripción de las clases del diseño.

3.2 Modelo de Diseño

Un modelo de Diseño es un modelo de objetos que describe la realización física de los casos de uso, centrándose en cómo los requerimientos funcionales y no funcionales junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema. Este artefacto constituye la entrada fundamental utilizada para el correcto desarrollo de las entradas de implementación.

3.2.1 Estructuración

Antes de comenzar a desarrollar el Modelo de Diseño es necesario definir la descomposición de este en subsistemas, con sus interfaces y las dependencias. Esta descomposición es muy significativa para la arquitectura en general, debido a que los subsistemas y sus interfaces constituyen la estructura fundamental del producto de software.

En el caso que ocupa el desarrollo del Componente de Seguridad se ha modelado un Diagrama de Subsistemas de Servicios; entiéndase este diagrama como un conjunto de clases agrupadas por compartir funcionalidades similares y las relaciones que se establecen entre sí. Del mismo modo, una de las características primordiales de estos subsistemas es su capacidad de reutilización, lo cual se

ajusta a la arquitectura definida para el Componente de Seguridad. A continuación se muestran estos subsistemas los cuales serán detallados posteriormente:

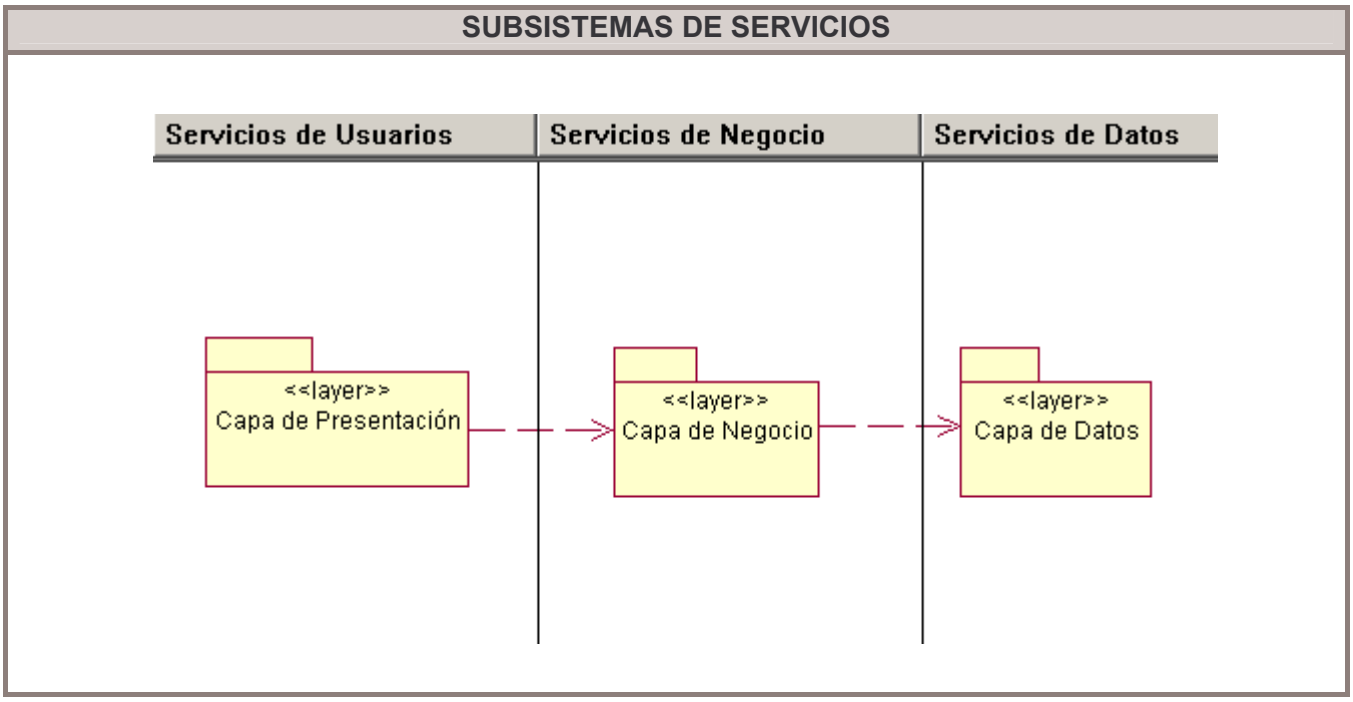


Figura 3.1 Diagrama de Subsistemas de Servicios.

A continuación, se describirá el contenido de cada uno de estos subsistemas de servicios, posibilitando una adecuada comprensión de los Diagramas de Clases del Diseño:

Capa de Presentación: Las funcionalidades de la capa de presentación consisten en intercambiar información con los actores, por lo tanto es la única capa con la que interactúa directamente un usuario. Contiene los ficheros mediante los cuales son invocados los métodos de la Capa de Negocio, entre ellos pueden ser mencionados los documentos JavaScript que se emplean en las validaciones y tratamiento de errores o excepciones y el motor de plantilla Smarty utilizado para la transformación de los ficheros HTML.

Capa de Negocio: Esta capa también se conoce como capa de Dominio o Capa de Lógica de Negocio. Establece la comunicación entre la capa de presentación y la capa de datos, encargada de recibir y responder cada petición de los usuarios. Los ficheros que la conforman reciben las solicitudes de los clientes, se comunican con la capa de datos actualizando o recuperando información, emitiendo

una respuesta. Constituye la parte del sistema donde se establecen todas las reglas de negocio que deben cumplirse.

Capa de Datos: Representa las tablas de la Base de Datos del Componente de Seguridad; es el repositorio físico de la información gestionada por el sistema, la cual puede ser recuperada, actualizada o eliminada a través de la Capa de Negocio.

3.2 .1 Definición de elementos de diseño

Para la realización de los Diagramas de Clases del Diseño será utilizada la extensión de UML para el modelado de aplicaciones Web, publicada por Jim Conallen a fines de 1999, en el artículo "Modelling Web Application Architectures with UML". Así, esta extensión presenta como elementos más significativos a tres clases UML estereotipadas "Server Page", "Client Page" y "Form" empleadas para el código servidor, código cliente y formularios respectivamente; permitiendo además representar ficheros contenedores de sentencias script como por ejemplo PHP y JavaScript. Entre estos elementos de diseño se establece un conjunto de relaciones las cuales son especificadas en la siguiente tabla: [27]

Hasta / Desde	Client Page	Form	Server Page
Client Page	<<link>>, <<redirect>>	aggregation	<<link>>, <<redirect>>
Form	aggregated by		<<submit>>
Server Page	<<build>>, <<redirect>>		<<redirect>>, <<include>>

Tabla 3.1 Relaciones entre las clases principales que conforman la extensión de UML para Web.

El código servidor se encarga de construir o generar el resultado XHTML que conforma el código cliente (<<*build*>>), los formularios envían sus datos al código servidor para ser procesados los pedidos (<<*submit*>>), además forman parte del código cliente o resultado XHTML, es por esto que la relación entre la clase empleada para el código cliente y la clase empleada para el formulario es de agregación. Entre las páginas clientes pueden existir vínculos (<<*link*>>) o redireccionamientos (<<*redirect*>>). Es importante destacar que una página cliente es construida por una sola página servidora; esta a su vez, puede completar su funcionamiento incluyendo código existente en otra

página de este mismo tipo, utilizando la relación de inclusión (<<*include*>>*)*, que aunque no es propia de la extensión de UML, las herramientas de modelado la consideran para representar todas las relaciones existentes en el modelo.

Para la nomenclatura de estas clases en los Diagramas de Clases del Diseño del Componente de Seguridad, se siguió la siguiente estructura: CL_<NombreClaseCliente>, FR_<NombreFormulario>, SP_<NombreClaseServidora> y <NombreMétodo>.php, para las páginas clientes, formularios, clases servidoras ubicadas en la Capa de Presentación y métodos de la Lógica de Negocio respectivamente.

3.2 .3 Diagramas de Clases del Diseño

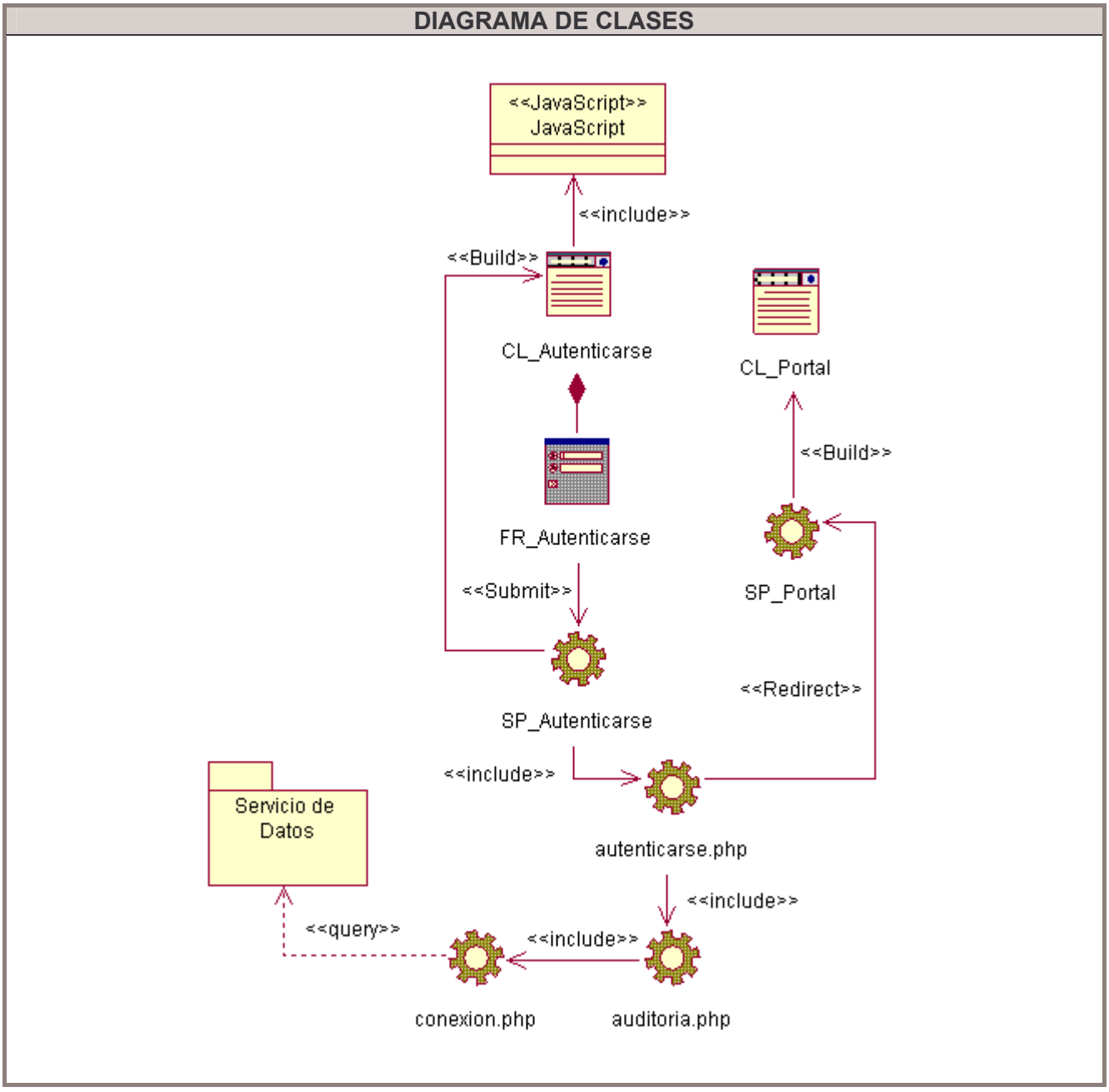


Figura 3.2 Diagrama de Clases del Diseño *CUS1_ Autenticar.*

DIAGRAMA DE CLASES

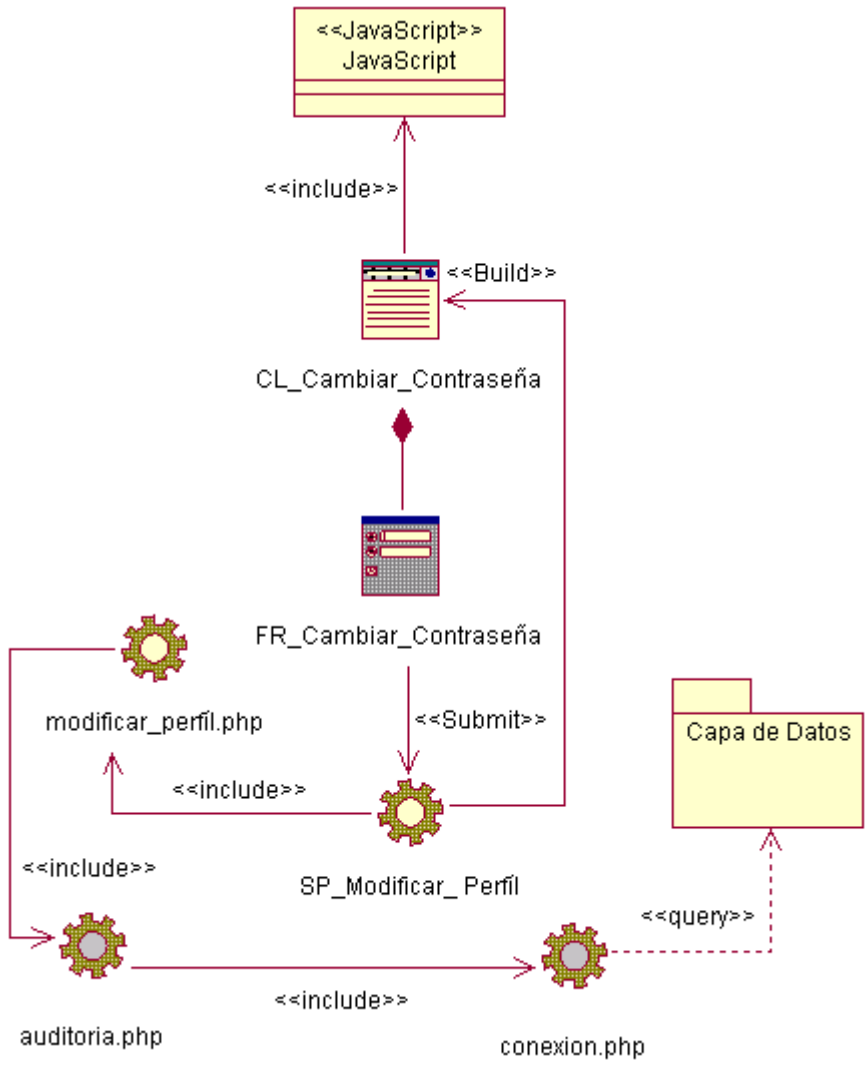


Figura 3.3 Diagrama de Clases del Diseño *CUS2_ Modificar_Perfil.*

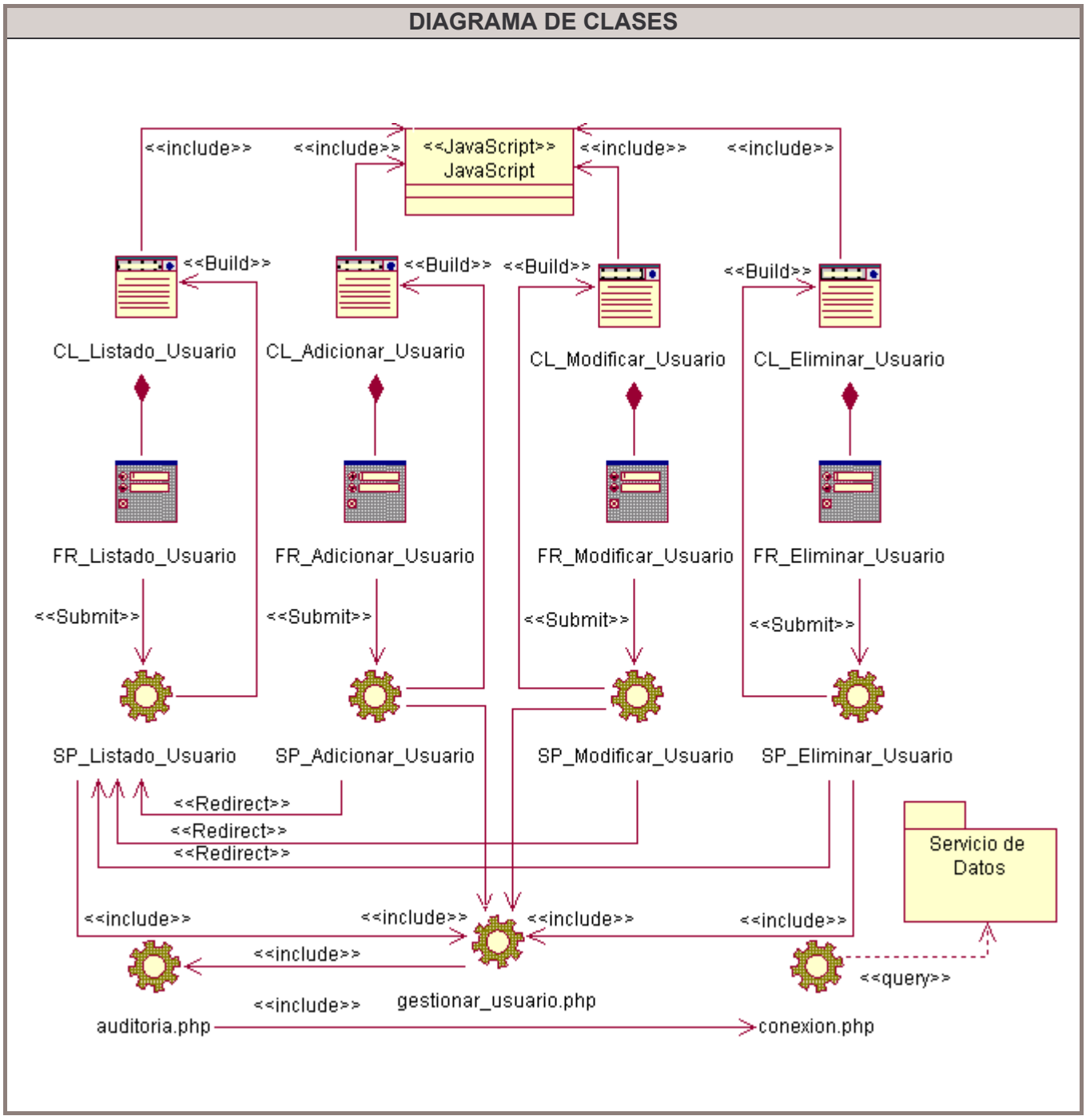


Figura 3.4 Diagrama de Clases del Diseño *CUS3_Gestionar_Usuario.*

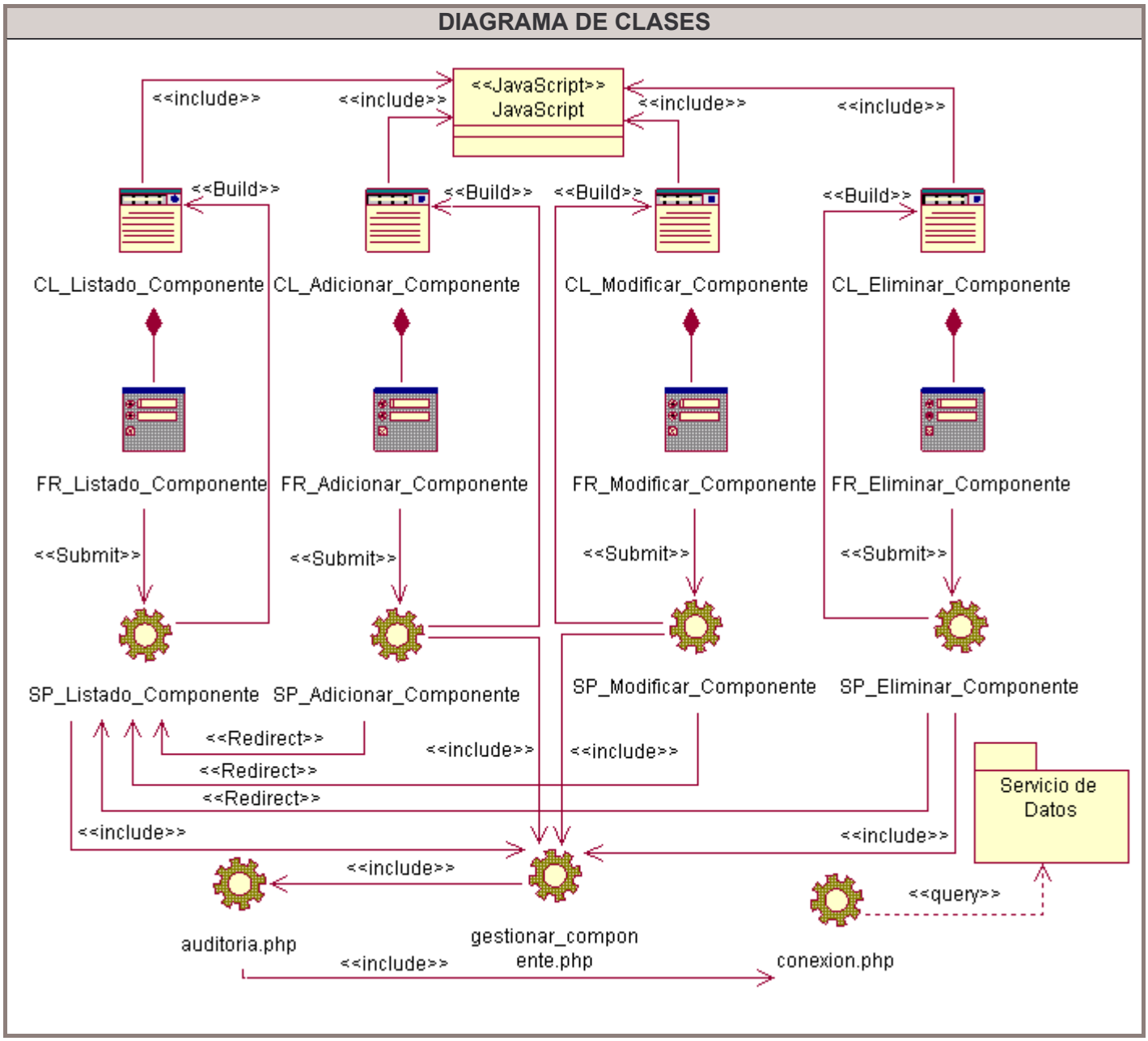


Figura 3.5 Diagrama de Clases del Diseño *CUS4_Gestionar_Componente.*

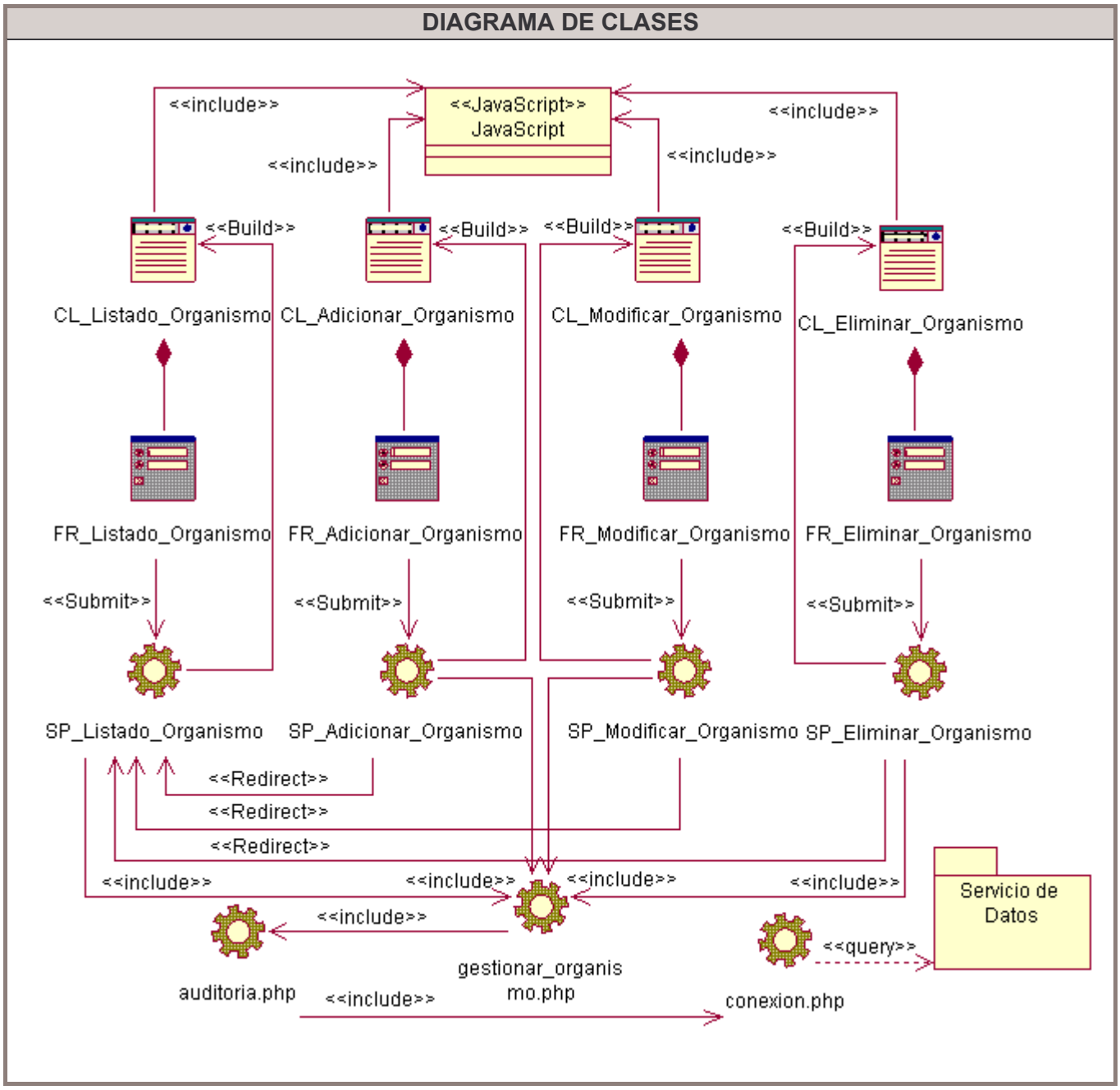


Figura 3.6 Diagrama de Clases del Diseño *CUS5_Gestionar_Organismo.*

DIAGRAMA DE CLASES

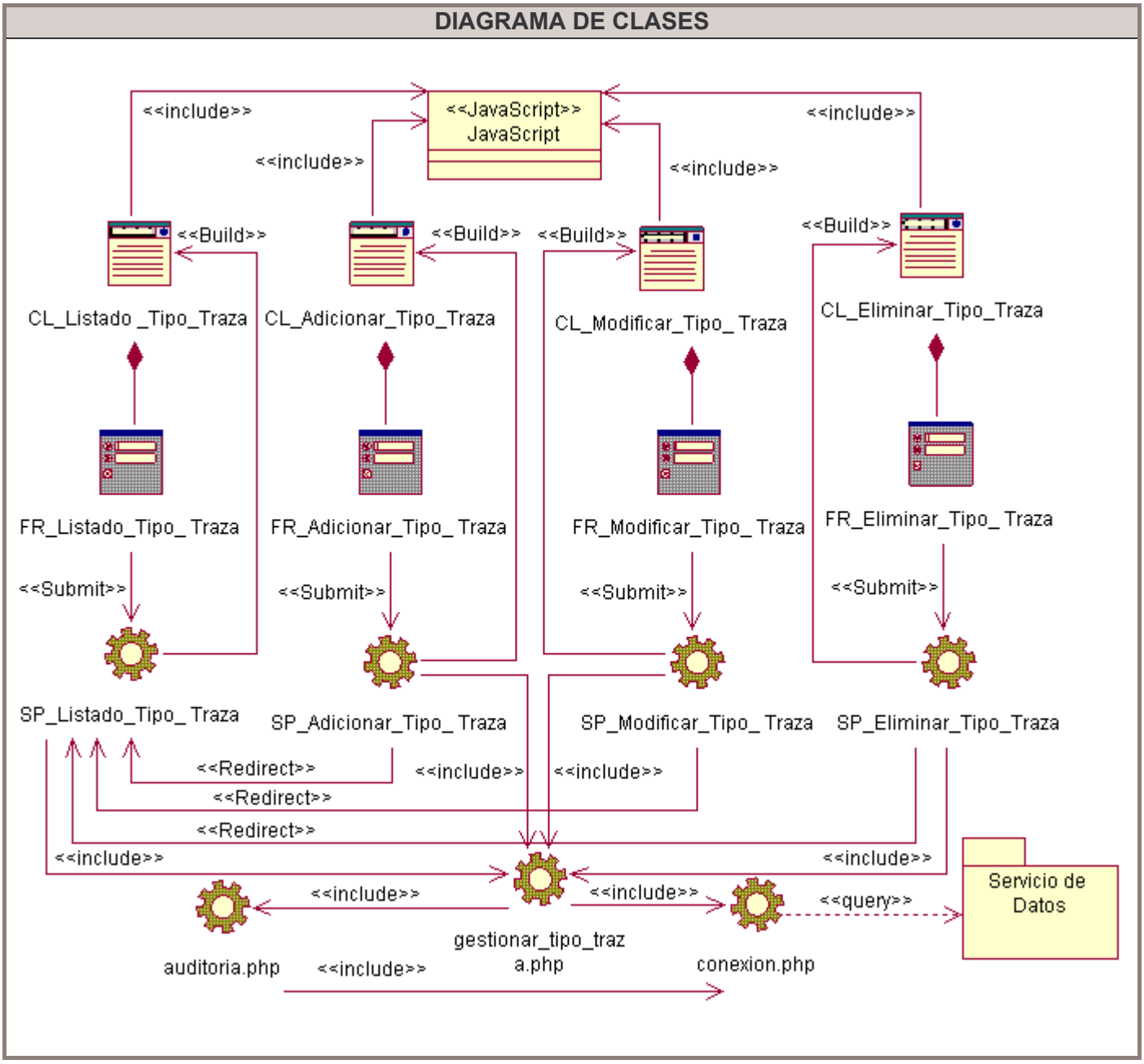


Figura 3.7 Diagrama de Clases del Diseño *CUS6_Gestionar_Tipo_Traza*.

DIAGRAMA DE CLASES

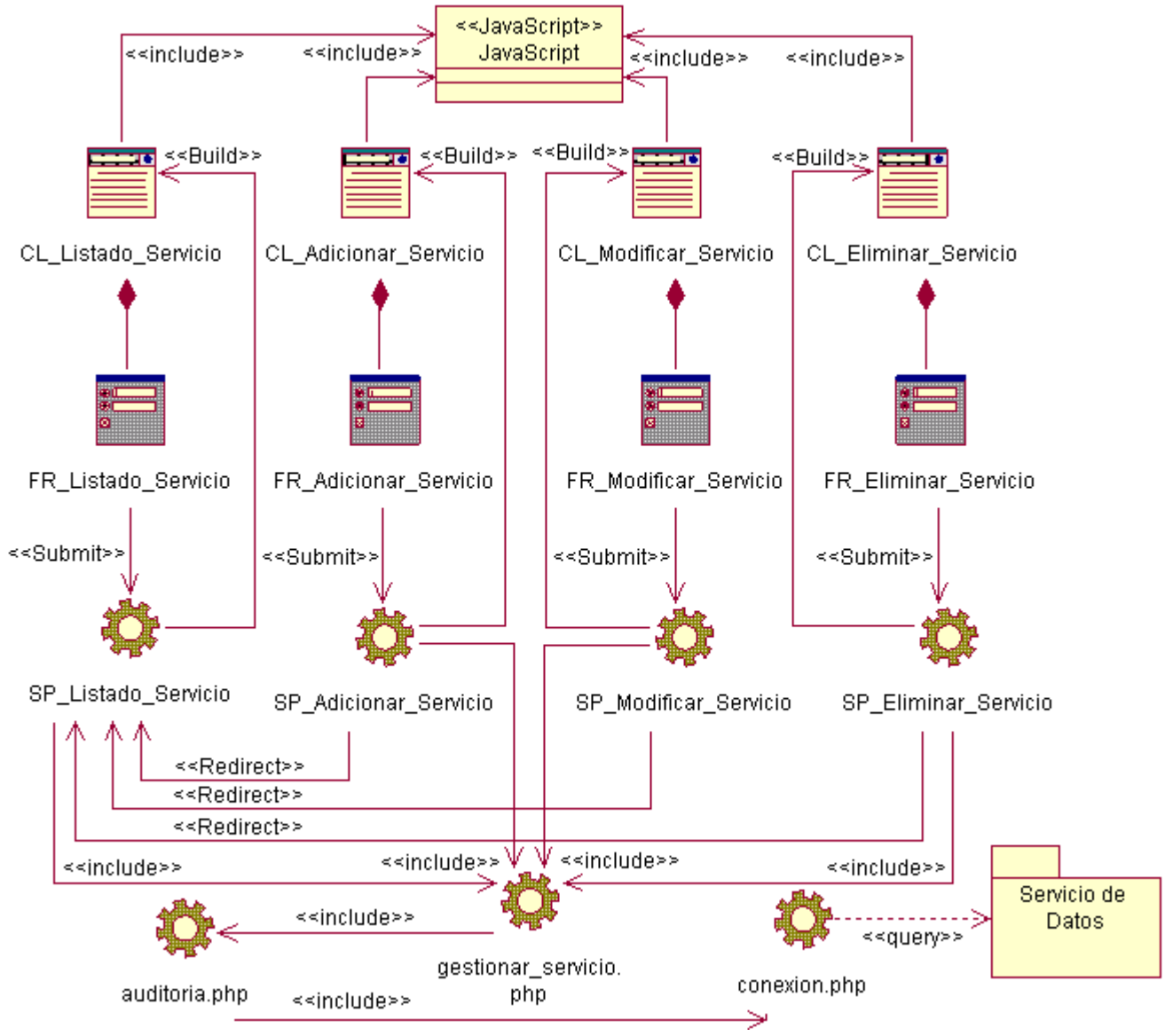


Figura 3.9 Diagrama de Clases del Diseño *CUS7_Gestionar_Servicio.*

DIAGRAMA DE CLASES

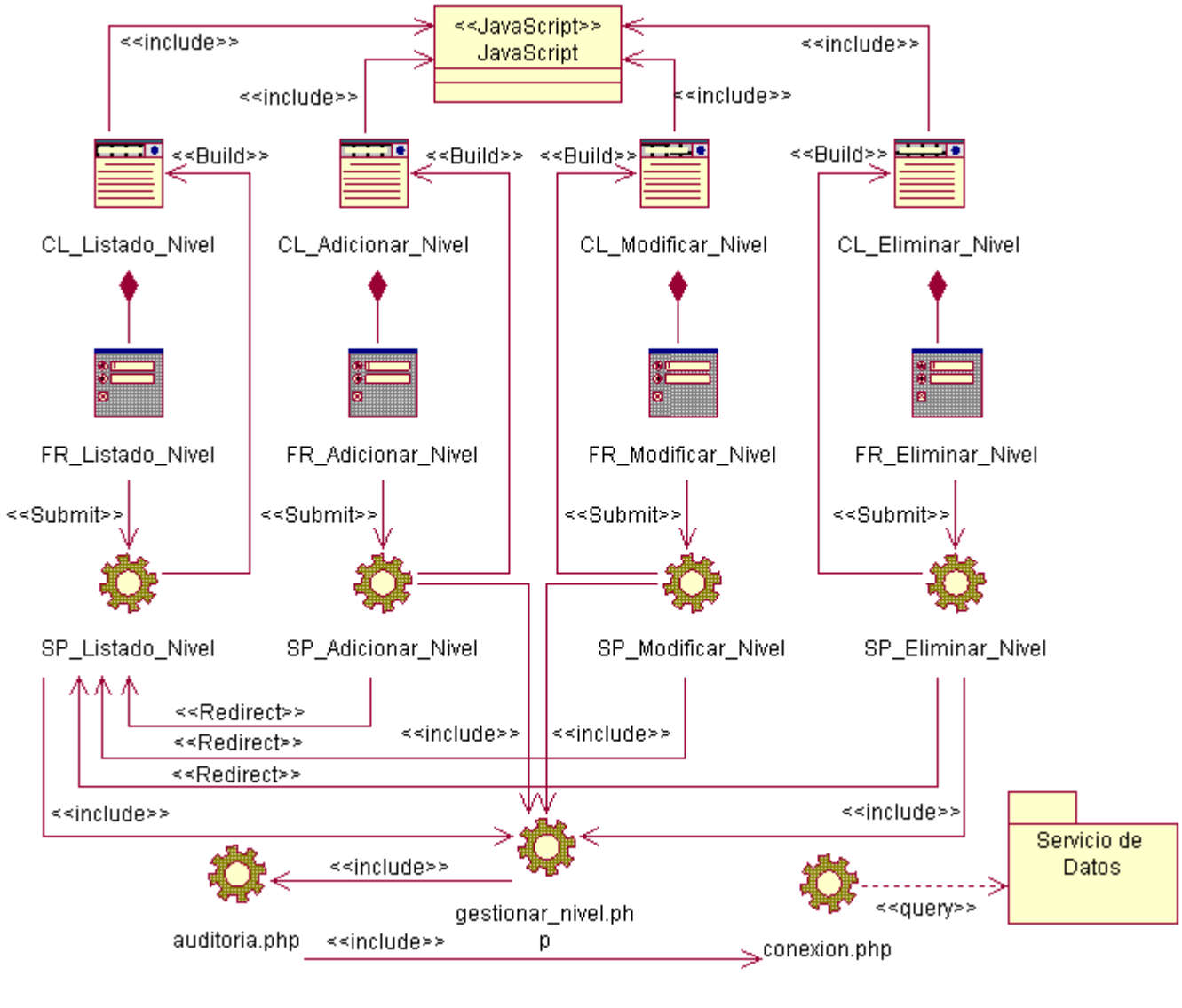


Figura 3.9 Diagrama de Clases del Diseño *CUS8_ Gestionar_Nivel*

DIAGRAMA DE CLASES

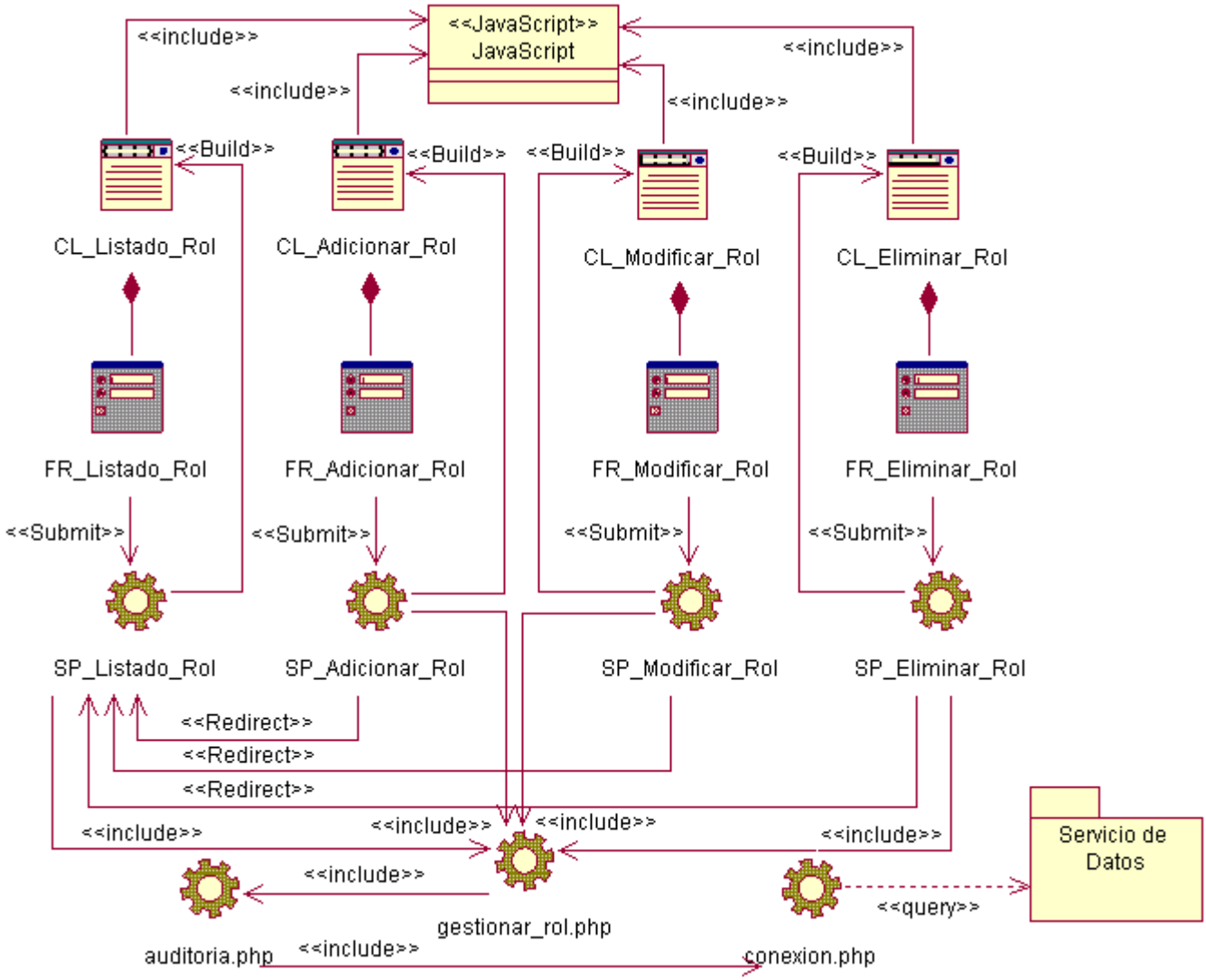


Figura 3.10 Diagrama de Clases del Diseño *CUS9_Gestionar_Rol.*

DIAGRAMA DE CLASES

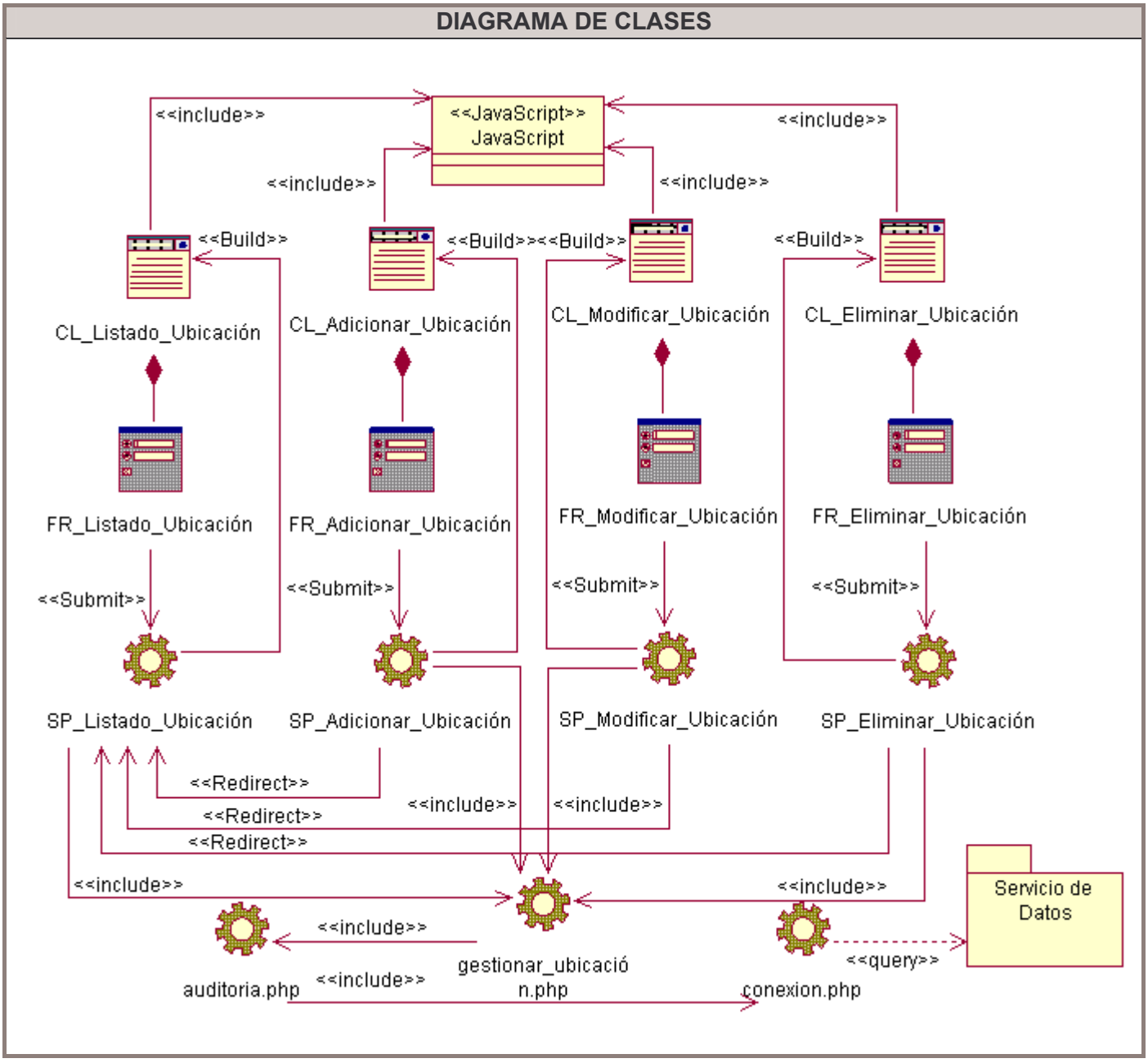


Figura 3.11 Diagrama de Clases del Diseño *CUS10_Gestionar_Ubicación.*

3.2 .4 Descripción de clases y atributos

Seguidamente serán explicadas algunas de las clases que han sido identificadas para su futura implementación. La estructura general de la Capa de Presentación para cada uno de los casos de uso identificados en el Componente de Seguridad es similar, contando con las mismas páginas clientes y servidoras. Del mismo modo serán descritas algunas de las responsabilidades que realizarán las páginas servidoras que responden a la Lógica de Negocio. De esta manera se tendrá una comprensión mayor del funcionamiento que tendrá el sistema en desarrollo.

3.2 .4.1 Descripción de páginas clientes

Nombre: *CL_Autenticarse*
Tipo de Clase: Client Page
Descripción General: Constituye la primera página Web con la que interactúa un usuario, la misma tiene el objetivo de capturar el nombre de usuario y contraseña para poder acceder al sistema, enviando al mismo a la página de bienvenida. Es utilizada en el Caso de Uso *Autenticar*.

Tabla 3.2 Descripción página cliente *CL_Autenticarse.*

Nombre: *CL_Cambiar_ Contraseña*
Tipo de Clase: Client Page
Descripción General: Esta página le brinda la posibilidad al usuario de cambiar su contraseña del sistema, la cual inicialmente es creada por un Administrador General, consta de tres campos de texto utilizados para el nombre de usuario, el cual aparece por defecto sin poder ser modificado y otros dos para la nueva contraseña y la confirmación de esta.

Tabla 3.3 Descripción página cliente *CL_Cambiar_Contraseña.*

Nombre: *CL_Listado*
Tipo de Clase: Client Page
Descripción General: La clase *CL_Listado* es una página Web que se ejecuta del lado del cliente sobre un navegador Web. Permite a los administradores la realización de listados y búsqueda de información específica, permitiendo generar documentos Portable Document Format (PDF). A través de esta página se puede registrar, modificar o eliminar la información almacenada o simplemente visualizar información. El resultado de las búsquedas es paginado cada 7 elementos, permitiendo la movilidad por tales resultados, inclusive ir directamente a la última página. Es utilizada en los siguientes Casos de Uso del Sistema: o Buscar_Usuario. o Gestionar_Componente. o Gestionar_Nivel. o Gestionar_Organismo. o Gestionar_Rol. o Gestionar_Servicio. o Gestionar_Tipo_Traza. o Gestionar_Ubicación. o Gestionar_Usuario. o Registrar_Traza.

Tabla 3.4 Descripción página cliente *CL_Listado.*

Nombre: *CL_Adicionar*
Tipo de Clase: Client Page
Descripción General: La clase *CL_Adicionar* es una página Web que se ejecuta del lado del cliente sobre un navegador Web, captura los datos que serán registrados para su posterior almacenamiento en la base de datos del Componente de Seguridad, posee un conjunto de validaciones JavaScript que impiden la realización de peticiones innecesarias y por lo tanto se incrementa la usabilidad. Es utilizada en los siguientes Casos de Uso del Sistema: o Buscar_Usuario. o Gestionar_Componente. o Gestionar_Nivel. o Gestionar_Organismo. o Gestionar_Rol.

- Gestionar_Servicio.
- Gestionar_Tipo_Traza.
- Gestionar_Ubicación.
- Gestionar_Usuario.
- Registrar_Traza.

Tabla 3.5 Descripción página cliente *CL_Adicionar.*

Nombre: *CL_Modificar*
Tipo de Clase: Client Page
Descripción General: La clase *CL_Modificar* es una página Web que se ejecuta del lado del cliente. Muestra los datos del elemento que ha sido previamente seleccionado para modificar desde la página cliente *CL_Listado*. Permite modificar la información que se visualiza en la misma. Es utilizada en los siguientes Casos de Uso del Sistema: o Buscar_Usuario. o Gestionar_Componente. o Gestionar_Nivel. o Gestionar_Organismo. o Gestionar_Rol. o Gestionar_Servicio. o Gestionar_Tipo_Traza. o Gestionar_Ubicación. o Gestionar_Usuario. o Registrar_Traza.

Tabla 3.6 Descripción página cliente *CL_Modificar.*

3.2 .4.2 Descripción de páginas servidoras

Nombre: *SP_Autenticarse*
Tipo de Clase: Server Page
Descripción General: Página servidora ubicada en la Capa de Presentación. Su tarea fundamental es construir la página cliente *CL_Autenticarse*. Invoca al método del negocio autenticacion.php para obtener la lista de derechos de los usuarios desglosados por componentes. Además a través de la invocación del método auditoria.php se registra el acceso del usuario al sistema.

Tabla 3.7 Descripción página servidora *SP_Autenticarse.*

Nombre: *SP_Modificar_ Perfíl*
Tipo de Clase: Server Page
Descripción General: Clase servidora ubicada en la Capa de Presentación. Su tarea fundamental es construir la página cliente *CL_Modificar_Perfíl*.

Tabla 3.8 Descripción página servidora SP_*Modificar_Perfíl.*

Nombre: *SP_Listado*
Tipo de Clase: Server Page
Descripción General: La clase *SP_Listado* es una clase que se ejecuta del lado del servidor en la Capa de Presentación. Su actividad fundamental es construir la página cliente *CL_Listado*. Es utilizada en los siguientes Casos de Uso del Sistema: o Buscar_Usuario. o Gestionar_Componente. o Gestionar_Nivel. o Gestionar_Organismo. o Gestionar_Rol. o Gestionar_Servicio. o Gestionar_Tipo_Traza. o Gestionar_Ubicación. o Gestionar_Usuario. o Registrar_Traza.

Tabla 3.9 Descripción página servidora *SP_Listado.*

Nombre: *SP_Adicionar*
Tipo de Clase: Server Page
Descripción General: La clase *SP_Adicionar* es una página que se ejecuta del lado del servidor en la Capa de Presentación. Recibe y valida los datos que se envían desde la página cliente *CL_Adicionar*. Codifica la información y construye las estructuras que serán enviadas a la Capa de Negocio. Se invoca al método del negocio para el registro de los nuevos datos y una vez concluida la ejecución de sus responsabilidades redirecciona su ejecución a la clase *SP_Listado.* Es utilizada en los siguientes Casos de uso. o Buscar_Usuario. o Gestionar_Componente. o Gestionar_Nivel. o Gestionar_Organismo. o Gestionar_Rol. o Gestionar_Servicios. o Gestionar_Tipo_Traza. o Gestionar_Ubicación. o Gestionar_Usuario. o Registrar_Traza.

Tabla 3.10 Descripción página servidora SP_*Adicionar.*

Nombre: *SP_Modificar*
Tipo de Clase: Server Page
Descripción General: La clase *SP_Modificar* es una página que se ejecuta del lado del servidor en la Capa de Presentación. Recibe y valida los datos que se envían desde la página cliente *CL_Modificar*. Codifica la información y construye las estructuras que serán enviadas a la Capa de Negocio. Se invoca al método del negocio para la modificación del los datos involucrados y una vez concluida la ejecución de sus responsabilidades redirecciona su ejecución a la clase *SP_Listado.* Es utilizada en los siguientes Casos de Uso. o Buscar_Usuario. o Gestionar_Componente. o Gestionar_Nivel. o Gestionar_Organismo.

○ Gestionar_Rol. ○ Gestionar_Servicio. ○ Gestionar_Tipo_Traza. ○ Gestionar_Ubicación. ○ Gestionar_Usuario. ○ Registrar_Traza.

Tabla 3.11 Descripción página servidora SP_*Modificar.*

Nombre: *SP_Eliminar*
Tipo de Clase: Server Page
Descripción General: La clase *SP_Eliminar* es una página que se ejecuta del lado del servidor en la Capa de Presentación. Recibe el identificador del elemento que se desea eliminar desde la página *SP_Listado*. Invoca al método del negocio necesario y una vez concluida la ejecución de sus responsabilidades redirecciona su ejecución a la clase *SP_Listado.* Es utilizada en los siguientes Casos de Uso. ○ Buscar_Usuario. ○ Gestionar_Componente. ○ Gestionar_Nivel. ○ Gestionar_Organismos. ○ Gestionar_Rol. ○ Gestionar_Servicio. ○ Gestionar_Tipo_Traza. ○ Gestionar_Ubicación. ○ Gestionar_Usuario. ○ Registrar_Traza. ○ Eliminar_Traza

Tabla 3.12 Descripción página servidora SP_*Eliminar.*

3.2 .4.3 Descripción de métodos de negocio

Nombre: *autenticacion.php*	
Tipo de Clase: Server Page	
Caso de Uso: Autenticar	
Descripción General: Permite el acceso de los usuarios al sistema, comprueba si el usuario que está solicitando el acceso se encuentra registrado en el Componente de Seguridad y si su cuenta de usuario está habilitada, es decir, si es un usuario activo. Si no se cumple alguna de estas condiciones se reporta un error de acceso. En el caso contrario es generado su certificado de acceso de manera aleatoria, registrándose la fecha y hora en que se autorizó el acceso al usuario, almacenándose mediante la página auditoria.php una traza de esta operación, devuelve una estructura necesaria para la ejecución de las funcionalidades del sistema y una Lista de Derechos que contiene sus privilegios desglosados por módulos, tipos de usuario, niveles y servicios.	
PARÁMETROS DE ENTRADA	
Descripción	**Tipo**
Usuario	String
Contrasena	String

Tabla 3.13 Descripción del método *autenticacion.php.*

Nombre: *auditoria.php*	
Tipo de Clase: Server Page	
Caso de Uso: Autenticar, Buscar_Usuario, Gestionar_Componente, Gestionar_Nivel, Gestionar_ Organismo, Gestionar_Rol, Gestionar_Servicio, Gestionar_Tipo_Traza, Gestionar_Ubicación, Gestionar_Usuario, Registrar_Traza y Eliminar_Traza.	
Descripción General: Esta es una página que se ejecuta del lado del servidor formando parte de la lógica de negocio del sistema. Permite el registro y búsqueda de las trazas en el sistema.	
PARÁMETROS DE ENTRADA	
Descripción	**Tipo**
Usuario	String
Componente	String
TipoTraza	String
DirecciónIP1	String
DirecciónIP2	String
Fecha1	Date
Fecha2	Date
Hora1	String
Hora2	String

Tabla 3.14 Descripción del método *auditoria.php*

Nombre: *modificar_perfil.php*	
Tipo de Clase: Server Page	
Caso de Uso: Modificar_Perfíl	
Descripción General: Esta es una página que se ejecuta del lado del servidor formando parte de la lógica de negocio del sistema. Es invocada por la página servidora SP_Modificar Perfil, permite al usuario cambiar su contraseña.	
PARÁMETROS DE ENTRADA	
Descripción	**Tipo**
Usuario	String
Contrasena	String

Tabla 3.15 Descripción del método *modificar_perfil.php.*

Nombre: *organismo.php*	
Tipo de Clase: Server Page	
Caso de Uso: Gestionar_Organismo	
Descripción General: Página servidora que es invocada por SP_Insertar Organismos. Permite registrar un nuevo organismo en el Componente de Seguridad. Del mismo son recogidos un conjunto de datos generales.	
PARÁMETROS DE ENTRADA	
Descripción	**Tipo**
Pais	String
Nombre	String
Correo	String
Telefono	String
Direccion	String
Descripcion	String
Ldap	Integer

Tabla 3.16 Descripción del método *organismos.php.*

Nombre: *componente.php*	
Tipo de Clase: Server Page	
Caso de Uso: Gestionar_Componente	
Descripción General: Página servidora que es invocada por SP_Insertar Componentes. Permite registrar un nuevo componente perteneciente a un organismo previamente registrado en el Componente de Seguridad. Del mismo son recogidos un conjunto de datos generales.	
PARÁMETROS DE ENTRADA	
Descripción	**Tipo**
Organismo	Integer
Nombre	String

UrlServicioWeb	String
UrlComponente	String
Descripcion	String

Tabla 3.17 Descripción del método *componentes.php.*

Nombre: *usuarios.php*	
Tipo de Clase: Server Page	
Caso de Uso: Gestionar_Usuario	
Descripción General: Página servidora que es invocada por SP_Insertar Usuarios. Permite registrar un nuevo usuario en el Componente de Seguridad. Del mismo son recogidos un conjunto de datos generales.	
PARÁMETROS DE ENTRADA	
Descripción	**Tipo**
Organismo	Integer
Componente	Integer
Nombre	String
PrimerApellido	String
SegundoApellido	String
CorreoElectronico	String
Usuario	String
Contrasena	String

Tabla 3.17 Descripción del método *usuarios.php.*

3.3 Modelo de Datos

Un Modelo de Datos es aquel que describe de forma abstracta cómo se representan los datos. Básicamente consiste en una descripción de algo conocido como contenedor de datos, así como de los métodos para almacenar y recuperar información de esos contenedores. El Modelo de Datos tiene gran importancia en el ciclo de desarrollo de software, debido a que define formalmente las estructuras permitidas y las restricciones a fin de representar los datos y constituye un elemento básico para el desarrollo de la metodología de diseño de la base de datos. Este modelo esta formado por Objetos (entidades que existen y que se manipulan), Atributos (características básicas de estos objetos) y Relaciones (forma en que enlazan los distintos objetos entre si).

MODELO DE DATOS

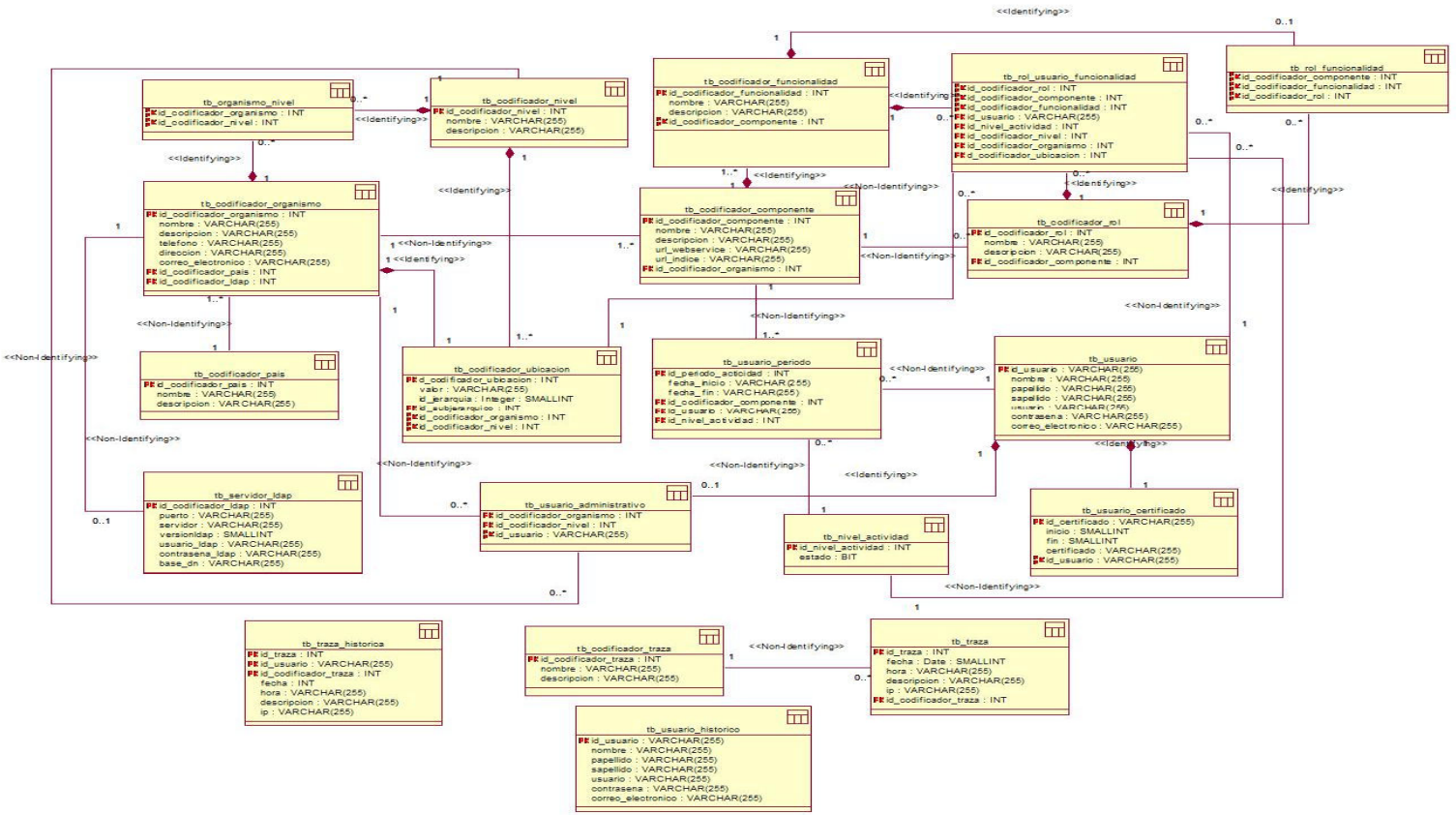


Figura 3.12 Modelo de Datos.

3.3.1 Descripción de las tablas de la Base de Datos

Nombre: *tb_usuario*	
Descripción General: En esta tabla se almacenan los datos generales del usuario registrado en el Componente de Seguridad.	
ATRIBUTOS	
Nombre	**Tipo**
id_usuario	String
nombre	String
papellido	String
sapellido	String
usuario	String
contrasena	String
correo_electronico	String
ldap	Integer

Tabla 3.18 Descripción de la tabla de la Base de Datos: *tb_usuario.*

Nombre: *tb_codificador_organismo*	
Descripción General: En esta tabla se almacenan los datos generales del organismo cuyos requerimientos de seguridad son gestionados por el Componente de Seguridad.	
ATRIBUTOS	
Nombre	**Tipo**
id_codificador_organismo	Integer
nombre	String
descripcion	String
pais	String
telefono	String
direccion	String
correo_electronico	String

Tabla 3.19 Descripción de la tabla de la Base de Datos: *tb_codificador_organismo.*

Nombre: *tb_codificador_componente*	
Descripción General: En esta tabla se almacenan los datos generales del componente cuyos requerimientos de seguridad son gestionados por el Componente de Seguridad.	
ATRIBUTOS	
Nombre	**Tipo**
id_codificador_componente	Integer
id_codificador_organismo	Integer
nombre	String
descripcion	String
url_webservice	String
url_indice	String

Tabla 3.20 Descripción de la tabla de la Base de Datos: *tb_codificador_componente.*

Nombre: *tb_traza*	
Descripción General: En esta tabla se almacena un conjunto de información sobre las acciones y peticiones de los usuarios en el Componente de Seguridad.	
ATRIBUTOS	
Nombre	**Tipo**
id_traza	Integer
funcionalidad	String
componente	String
usuario	String
tipo_traza	String
fecha	Date
hora	String
ip	String
descripcion	String

Tabla 3.21 Descripción de la tabla de la Base de Datos: *tb_traza.*

3.4 Conclusiones

Al concluir este capítulo donde fue obtenido el Modelo de Diseño, se describió la realización física de los casos de uso, centrándose en cómo los requerimientos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a desarrollar. Además, constituyó una abstracción para la implementación y es de este modo utilizado como una entrada fundamental en las actividades que se realizan en el Flujo de Trabajo de Implementación.

CAPÍTULO 4: IMPLEMENTACIÓN

4.1 Introducción

El presente capítulo constituye la secuencia lógica del Diseño, a través del mismo se implementarán las clases y subsistemas en términos de componentes. Se obtendrá el Diagrama de Despliegue como parte del Modelo de Implementación, que indicará cómo la solución implementada estará distribuida físicamente. Finalmente se proporcionará una detallada explicación de la solución dada al mantenimiento de la integridad referencial en una Arquitectura Orientada a Servicios.

4.2 Modelo de Implementación

La implementación es el centro de las iteraciones durante la fase de construcción, aunque también se lleva a cabo la implementación durante la fase de elaboración, para crear la línea base de la arquitectura y durante la fase de transición, para tratar los defectos tardíos como los encontrados en su distribución. El Modelo de Implementación describe cómo los elementos del Modelo de Diseño serán implementados en términos de componentes describiendo además su organización de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y el lenguaje o lenguajes de programación utilizados así como la dependencia que se establece entre estos componentes. [28]

4.2.1 Diagramas de Componentes

Un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo del diseño. Alguno de sus estereotipos son <<file>>, <<library>> y <<table>>. Los componentes tienen relaciones de traza con los elementos del modelo de diseño que implementan. A continuación serán expuestos los Diagramas de Componentes asociados a varios de los subsistemas de implementación identificados para el Componente de Seguridad.

Como ha sido mencionado anteriormente el Componente de Seguridad es desarrollado sobre una arquitectura en tres capas, en tal sentido su estructuración en subsistemas de implementación es la siguiente:

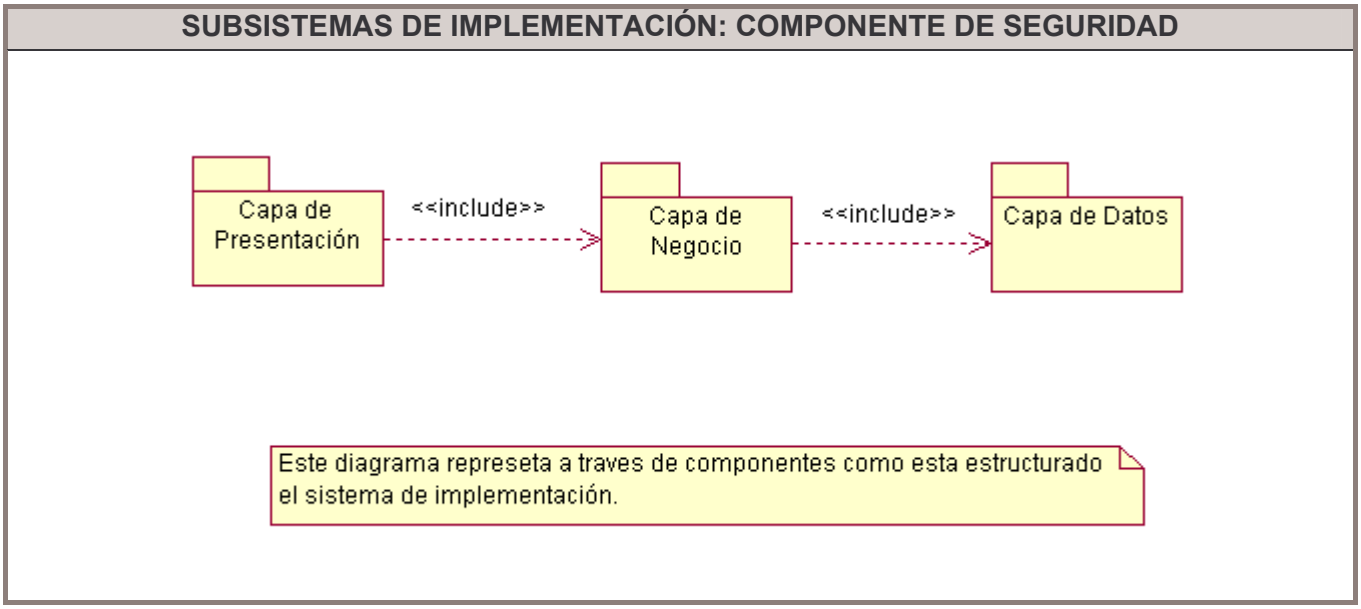


Figura 4.1 Estructura en subsistemas de Implementación.

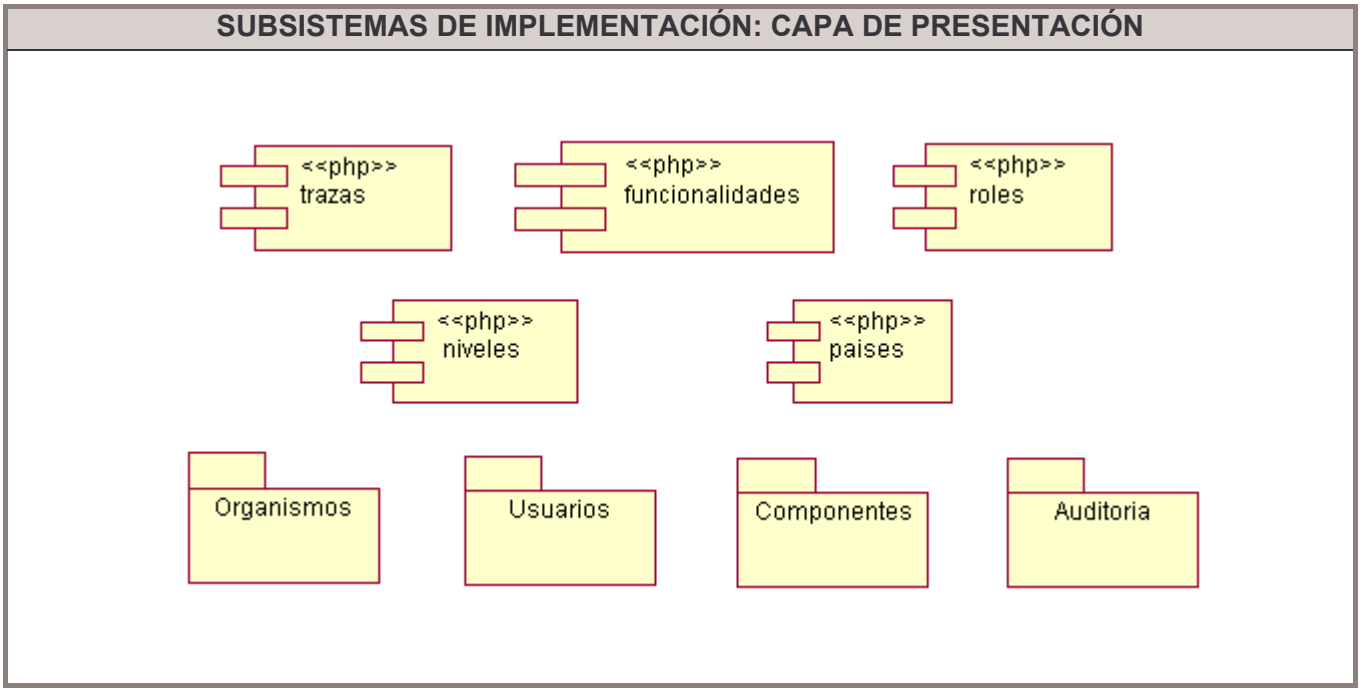


Figura 4.2 Estructura en subsistemas de Implementación: *Capa de presentación.*

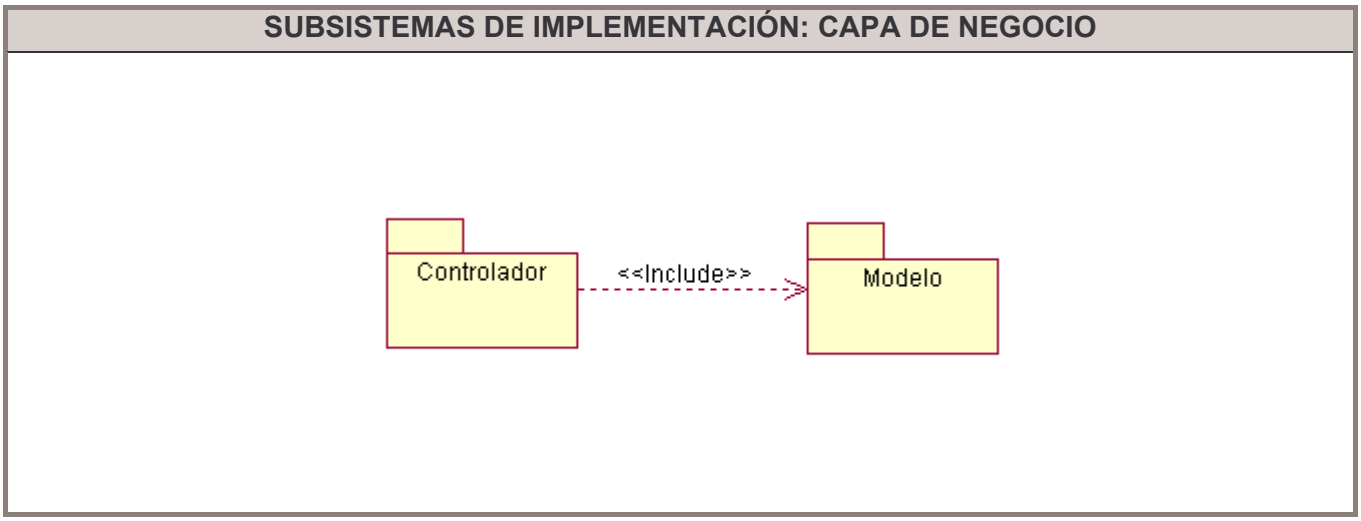


Figura 4.3 Estructura en subsistemas de Implementación: *Capa de Negocio.*

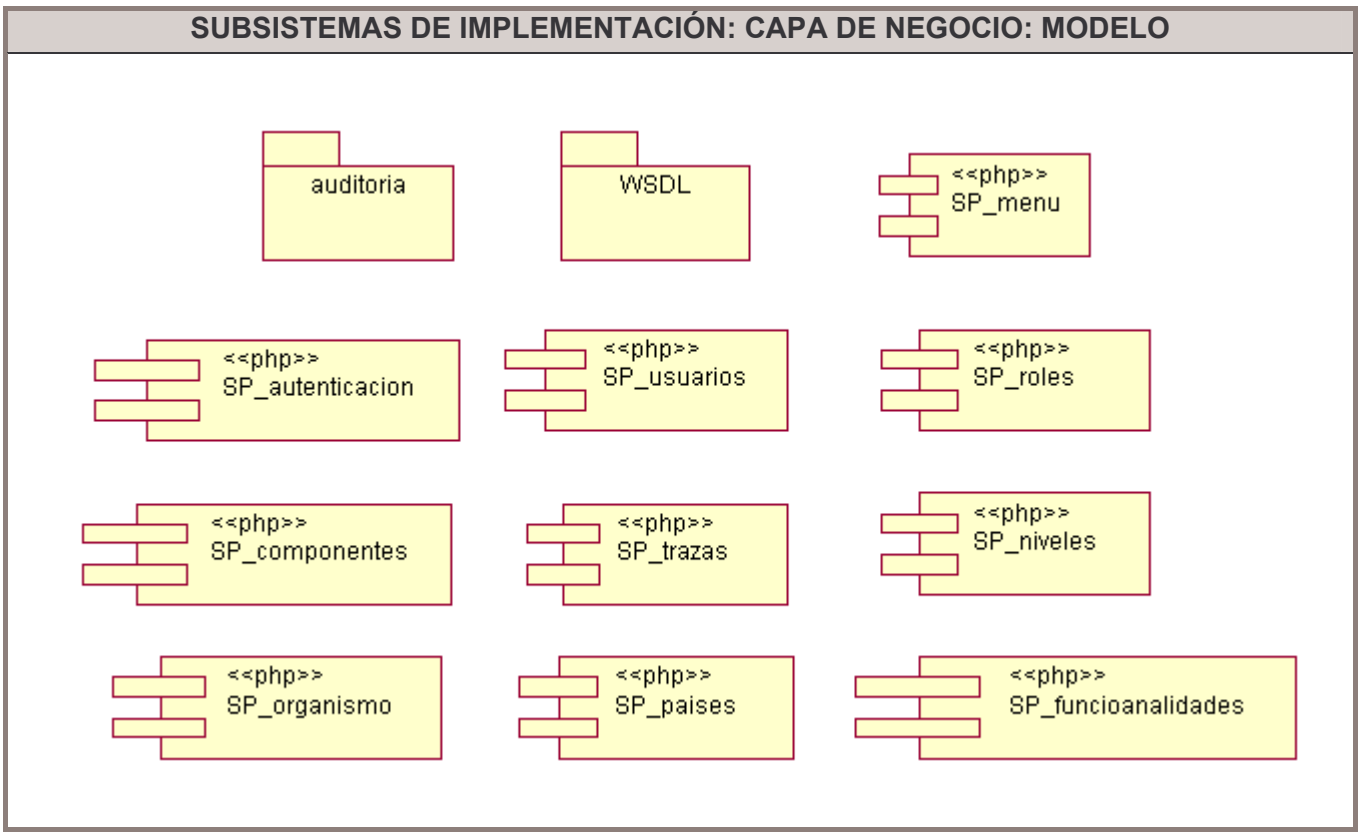


Figura 4.4 Estructura en subsistemas de Implementación: *Capa de Negocio: Modelo.*

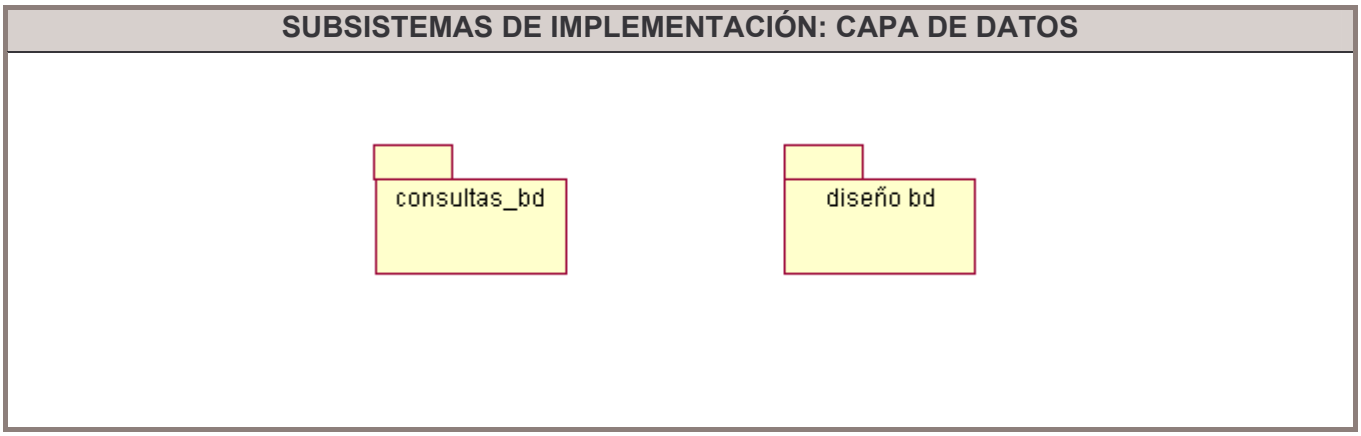


Figura 4.5 Estructura en subsistemas de Implementación: *Capa de Datos.*

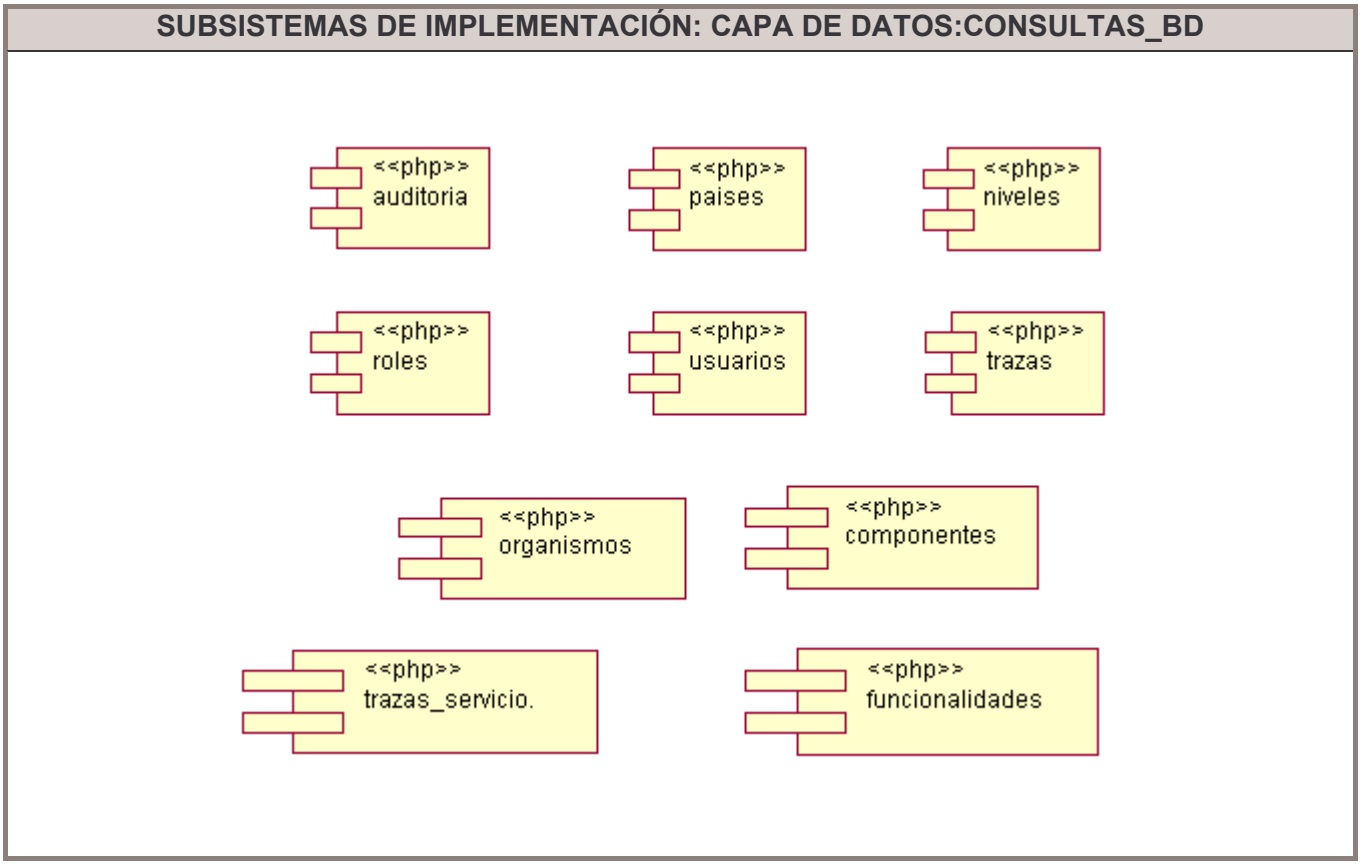


Figura 4.6 Estructura en subsistemas de Implementación: *Capa de Datos: consultas_bd.*

4.2.2 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 de cómputo. Por esta razón, tal distribución tiene una influencia principal para las actividades de implementación. Cada nodo representa un recurso de cómputo, en el caso del Componente de Seguridad, este será distribuido en tres servidores, el primero de ellos conteniendo la Capa de Presentación conectado a una PC cliente. El servidor de la Lógica de Negocio tendrá conectividad punto a punto con el servidor de datos, todos estos nodos de procesamiento funcionarán con sistema operativo LINUX distribución Debian 4 Etch y la comunicación se realizará mediante SOAP utilizando el protocolo de transporte HTTPS.

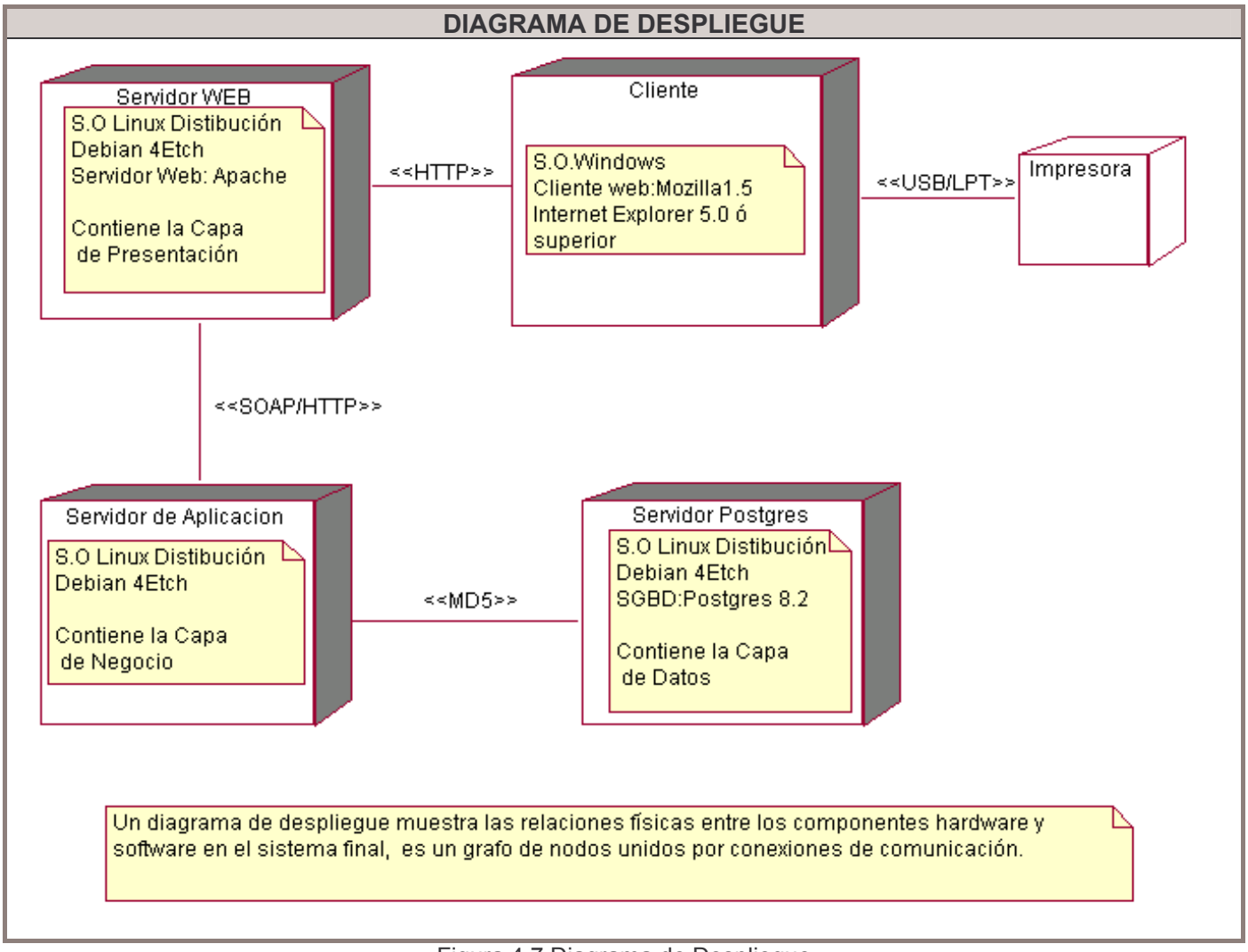


Figura 4.7 Diagrama de Despliegue

4.3 Seguridad en las comunicaciones

Desde la concepción inicial del Componente de Seguridad se identificó la necesidad de contar con un canal seguro de comunicación, debido a la importancia de la información que el mismo gestiona y a las consecuencias nefastas que se podrían originar en caso de no tomar las precauciones necesarias para garantizar la seguridad de la información que viaja entre el servidor Web y el usuario. En caso de no tener presente este aspecto, el sistema podría ser victima de un ataque electrónico que permitiría interceptar el contenido de las comunicaciones TCP/IP comprometiendo la seguridad de las aplicaciones Web que utilicen los servicios que brinda el Componente de Seguridad.

Por esta razón es necesaria la utilización de protocolos seguros de comunicación como el HTTPS. Este protocolo no forma parte del proceso de elaboración propio del Componente de Seguridad, sino que debe configurarse en el servidor Web donde va a ser instalado el sistema una vez que haya rebasado sus fases de construcción. A continuación se presenta una pequeña guía sobre la configuración de este protocolo en la plataforma Linux específicamente sobre la distribución Debian 4 Etch.

4.3.1 Configuración de HTTPS en Debian 4 Etch

Primeramente se debe instalar OpenSSL que es un robusto paquete de herramientas de administración y librerías relacionadas con la criptografía, que suministran las funciones criptográficas necesarias al navegador web y permite crear los certificados digitales que podremos aplicar a nuestro servidor. Utilizando el comando:

```
#apt-get install openssl ssl-cert
```

El próximo paso es crear los certificados digitales, para esto es necesario usar OpenSSL desde la consola de comandos e introducir la siguiente instrucción:

```
#openssl req $@ -new -x509 -days 365 -nodes -out /etc/apache2/apache.pem -keyout /etc/apache2/apache.pem
```

Al ejecutar el comando anterior se debe ingresar la información sobre la organización y de contacto que aparecerán en el certificado, aunque la misma puede ser opcional.

EJEMPLO DE CREACIÓN DE UN CERTIFICADO DIGITAL

```
Generating a 1024 bit RSA private key
.....++++++
.....++++++
writing new private key to '/etc/apache2/apache.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a
DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:
Email Address []:
```

Figura 4.8 Ejemplo de Creación de un Certificado Digital.

Luego se debe modificar el permiso para evitar que los usuarios del sistema puedan tener acceso al fichero /etc/apache2/apache.pem que es donde se guarda el certificado cifrado en forma de texto.

```
# chmod 600 /etc/apache2/apache.pem
```

Por defecto el servidor escuchará peticiones HTTP por el Puerto 80, y no las conexiones SSL por el puerto 443. Lo que hace necesario habilitar el soporte SSL. Mediante la modificación del archivo /etc/apache2/ports.conf agregando el puerto 443 de escucha para SSL.

```
Listen 443
```

Se activa el modulo de apache para soporte de SSL

```
# a2enmod ssl
```

Finalmente se deben realizar algunas configuraciones de servidor:

Se crea el archivo que contendrá la configuración del sitio: /etc/apache2/sites-available/default-ssl donde: [29]

- La expresión *:443 indica que el sito recibirá las llamadas al puerto 443.
- El directorio /var/www-ssl/ contendrá las páginas que utilizaran el canal SSL cifrado. Si no existe se debe crear.
- La directiva SSLEngine on activa el soporte SSL
- La directiva SSLCertificateFile /etc/apache2/apache.pem indica el archivo que contiene el certificado del sitio.
- ErrorLog y CustomLog guardan la información de la bitácora en los archivos especificados.

4.4 Integración con servidores LDAP

Generalmente los organismos o instituciones tienen su propia red informática, para la comunicación de los diferentes servicios que pudieran brindar o consumir, así como sus propios usuarios ya establecidos con una estructura jerárquica definida. Para acceder a los datos de los usuarios utilizan un servidor LDAP, que puede ser utilizado desde distintas plataformas, debido a que su implementación esta basada en estándares internacionales y hace que los procesos de búsqueda, y de autenticación sean mucho mas rápidos y eficientes que un SGBD convencional.

Teniendo en cuenta estas características, el Componente de Seguridad, para la gestión de la información de sus usuarios, brinda de forma adicional la posibilidad de conectarse a servidores LDAP, posibilitando la reutilización de la información y beneficiándose de las ventajas que proporciona el uso del protocolo LDAP.

Para la conexión a servidores LDAP, se especifican una serie de parámetros de entrada, que permiten la comunicación de manera estándar, independientemente del lugar y estructura del servidor que provee la información. Estos parámetros son:

- Dirección IP donde se encuentra el servidor.
- Puerto por el cual se va a establecer la conexión.
- Versión del LDAP.

- Base DN (Nombre Distinguido), estructura en forma de árbol jerárquico que define la concatenación de los DNS (Sistema de Nombres de Dominio) relativos de las entradas “padre” hasta llegar a la entrada “raíz” del árbol. Un ejemplo de DN pudiera ser (cn=Pedro Pérez, ou= UCI Domain Users, o=uci ,c=cu)
- Usuario y Contraseña registrados en el LDAP para establecer la conexión.
- Filtro de conexión: filtro que permite restringir la búsqueda de una persona en el directorio.

Luego de haber establecido la conexión con el servidor LDAP, se obtienen una serie de parámetros de salida estándares, lo cuales pueden ser utilizados en dependencia de las necesidades del usuario o sistema que consulta el Servidor LDAP. Algunos de estos parámetros son:

- givenname: Devuelve el nombre del usuario.
- sn: Apellidos del usuario.
- cn: Nombre completo del usuario.
- mail: Dirección de correo electrónico del usuario.
- mailnickname: Nombre de usuario.

A continuación se muestra un ejemplo de cómo se establece la conexión y se consulta un servidor LDAP:

EJEMPLO DE CÓMO SE ESTABLECE LA CONEXIÓN Y SE CONSULTA UN SERVIDOR LDAP

```
function ConectarLdap()
{
    $user     = 'drojas';
    $password = '**********';
    $puerto = 'uci.cu';
    $ds = ldap_connect("ldap://$puerto");
    $r = @ldap_bind($ds,$user."@$puerto",$password);

    if($r!=1)
    {
        echo 'no se conecto';
    }

    $cadena = trim($user);
    $filtro = "mail=$cadena@*";

    $sr = @ldap_search($ds,"OU=UCI Domain Users,DC=uci,DC=cu", $filtro);
    echo '<ul>';

    if(ldap_count_entries($ds,$sr)<1)
    echo '<li>No hay resultados</li>';

    $info  = ldap_get_entries($ds, $sr);
    $count = $info["count"];
    for($i=0;$i<$count;$i++)
    {
        echo '<li><strong>'.$info[$i]['mailnickname'][0].'</strong></li>';//usuario
        echo '<li><strong>'.$info[$i]['cn'][0].'</strong></li>';//nombre completo
        echo '<li><strong>'.$info[$i]['givenname'][0].'</strong></li>';//nombre solo
        echo '<li><strong>'.$info[$i]['sn'][0].'</strong></li>';// apellidos
        echo '<li><strong>'.$info[$i]['mail'][0].'</strong></li>';// correo
    }
    ldap_unbind($ds);
    echo '</ul>';
}
```

Figura 4.10 Ejemplo de cómo se establece la conexión y se consulta un Servidor LDAP.

4.5 Publicación de Servicios Web

La implementación de Servicios Web juega un papel protagónico en el Componente de Seguridad, debido a que brinda la posibilidad a componentes externos, de consumir algunas de sus principales funcionalidades independientemente de la plataforma o lenguaje en el que hayan sido desarrollados.

Para publicar un Servicio Web se agrupan las funcionalidades que se deseen brindar como servicios en una o varias clases contenedoras, posteriormente se especifica la descripción para cada tipo de datos, tanto de entrada como de salida de cada función. Luego utilizando el IDE de Desarrollo para PHP, ZendStudio, se especifica la dirección del fichero donde se encuentran la clases contenedoras, a continuación se selecciona la clase que contiene las funciones que se desean publicar y finalmente se describe el servicio en un fichero .wsdl que es la interfaz entre el Servicio Web y los sistemas o clientes que necesitan consumir alguno de los servicios publicados.

4.5.1 Servicios Web

El Componente de Seguridad brinda los Servicios Web: Autenticar, Autorizar, Búsqueda de Usuarios, Búsqueda de Componentes, Adicionar Traza, Búsqueda de Traza. Seguidamente se muestran datos de los servicios Autenticar, Búsqueda de Usuarios, y Adicionar Traza respectivamente, entre los que se incluyen: una breve descripción, parámetros de entrada y salida, ejemplos sobre cómo consumirlos así como la estructura de la información que devuelven.

Nombre del Servicio Web: *Autenticar*	
Descripción General: Este servicio es el encargado de verificar la identidad digital de un usuario que intenta acceder a un sistema cuyos requerimientos de seguridad son gestionados por el Componente de Seguridad.	
PARÁMETROS DE ENTRADA	
Descripción	**Tipo**
Usuario	String
Contrasena	String
PARÁMETROS DE SALIDA	
Descripción	**Tipo**
IDUsuario	String
Usuario	String
Nombre	String

CorreoElectronico	String
Certificado	String
ListaDerechos	Array

Tabla 4.1 Descripción del Servicio Web: *Autenticar.*

EJEMPLO DE IMPLEMENTACIÓN PARA CONSUMIR EL SERVICIO WEB AUTENTICAR

```
try
{
    $cliente = new SoapClient("http://localhost:5800/Seguridad/modelo/wsdl/servicios_usuario.wsdl");
    $arg=array("user"=>"root","pass"=>"1234567");
    $r= $cliente->__call("autenticar",$arg);
    print_r ($r);
}
catch (SoapFault  $f)
{
    print_r($f);
}
```

Figura 4.10 Ejemplo de implementación para consumir el Servicio Web: *Autenticar*.

ESTRUCTURA DE LA INFORMACIÓN QUE DEVUELVE EL SERVICIO WEB AUTENTICAR

```
[idusuario] => 0_158hmsolis
[usuario] => hmsolis
[nombre] => Hector Manuel Solis Mulet
[correo] => hmsolis@estudiantes.uci.cu
[certificado] => a4cedf64b4debf70f72b027f934a1b03
[ListaDerechos] => Array
    (
        [3] => stdClass Object
            (
                [modulo] => 113
                [nombremodulo] => docencia
                [url] =>
                [ListaRoles] => Array
                    (
                        [0] => stdClass Object
                            (
                                [rol] => 158
                                [nombrerol] => visualizador
                                [ListaNiveles] => Array
                                    (
                                        [0] => stdClass Object
                                            (
                                                [nivel] => 201
                                                [nombrenivel] => Nacional
                                                [idnivel] =>
                                                [ListaServicios] => Array
                                                    (
                                                        [0] => stdClass Object
                                                            (
                                                                [servicio] => 270
                                                                [nombreservicio] => dar baja estudiante
                                                            )
                                                    )
                                            )
                                    )
                            )
                    )
            )
    )
)
```

Figura 4.11 Estructura de la información que devuelve el Servicio Web: *Autenticar*.

Nombre del Servicio Web: Búsqueda de Usuarios	
Descripción General: Este servicio es el encargado de buscar los usuarios registrados en el Componente de Seguridad.	
PARÁMETROS DE ENTRADA	
Descripción	**Tipo**
Usuario	String
Nombre	String
PrimerApellido	String
SegundoApellido	String
Componente	Integer
Rol	Integer
Nivel	Integer
Servicio	Integer
Certficado	String
PARÁMETROS DE SALIDA	
Descripción	**Tipo**
Usuarios	Array

Tabla 4.2 Descripción del Servicio Web: *Búsqueda de Usuarios.*

EJEMPLO DE IMPLEMENTACIÓN PARA CONSUMIR EL SERVICIO BÚSQUEDA DE USUARIOS

```
try{
    $cliente = new SoapClient("http://localhost:5800/Seguridad/modelo/wsdl/servicios_usuario.wsdl");
    $sCert = $_SESSION['certificado'];
    $arg=array("sUsuario"=>"","sNombre"=>"","sPa"=>"","sSa"=>"","sCorreo"=>"","iOrganismo"=>0,
    "iComponente"=>'0',"iRol"=>'0',"iNivel"=>'0',"iServicio"=>'0',"sCertificado"=>$sCert);
    $r= $cliente->__call("BuscarUsuarios",$arg);
    print_r ($r);
}
catch (SoapFault  $f)
{
    print_r($f);
}
```

Figura 4.11 Ejemplo de implementación para consumir el Servicio Web: *Búsqueda de Usuarios*.

ESTRUCTURA DE INFORMACIÓN QUE DEVUELVE EL SERVICIO BÚSQUEDA DE USUARIOS

```
Array
(
    [0] => stdClass Object
        (
            [idusuario] => 0_158hmsolis
            [usuario] => hmsolis
            [nombre] => Hector Manuel Solis Mulet
            [correo] => hmsolis@estudiantes.uci.cu
            [certificado] => a4cedf64b4debf70f72b027f934a1b03
            [ListaDerechos] => Array
                (
                    [3] => stdClass Object
                        (
                            [modulo] => 113
                            [nombremodulo] => docencia
                            [uri] =>
                            [ListaRoles] => Array
                                (
                                    [0] => stdClass Object
                                        (
                                            [rol] => 158
                                            [nombrerol] => visualizador
                                            [ListaNiveles] => Array
                                                (
                                                    [0] => stdClass Object
                                                        (
                                                            [nivel] => 2D1
                                                            [nombrenivel] => Nacional
                                                            [idnivel] =>
                                                            [ListaServicios] => Array
                                                                (
                                                                    [0] => stdClass Object
                                                                        (
                                                                            [servicio] => 270
                                                                            [nombreservicio] => dar baja estudiante
                                                                        )
                                                                )
                                                        )
                                                )
                                        )
                                )
                        )
                )
        )
)
```

Figura 4.12 Estructura de la información que devuelve el Servicio Web: *Búsqueda de Usuarios*.

Nombre del Servicio Web: *Adicionar Traza*	
Descripción General: Este servicio es el encargado de adicionar las trazas de interés de los sistemas cuyos requerimientos de seguridad son gestionados por el Componente de Seguridad.	
PARÁMETROS DE ENTRADA	
Descripción	**Tipo**
Servicio	String
Componente	String
Usuario	String
TipoTraza	String
Descripcion	String
Certficado	String
PARÁMETROS DE SALIDA	
Descripción	**Tipo**
Resultado	Boolean

Tabla 4.3 Descripción del Servicio Web: *Adicionar Traza.*

EJEMPLO DE IMPLEMENTACIÓN PARA CONSUMIR EL SERVICIO WEB ADICIONAR TRAZA

```
try{

$cliente = new SoapClient("http://localhost:5800/Seguridad/modelo/wsdl/servicios_auditoria.wsdl",array("use"=> UTF-8));

$sCert = $_SESSION['certificado'];
$arg=array("sFuncionalidad"=>"Insertar servicio","sComponente"=>"Componente de Seguridad","sUsuario"=>"hmsoliss",
          "sCodificadorTraza"=>'acceso denegado',"txDescripcion"=>"","sCertificado"=>$sCert);
$r= $cliente->__call("AdicionarTraza",$arg);
if($r)
return true;
return false;
}
catch (SoapFault  $f)
{
print_r($f);
}
```

Figura 4.13 Ejemplo de implementación para consumir el Servicio Web: *Adicionar Traza*.

4.6 Conclusiones

Una vez concluido el proceso de implementación del Componente de Seguridad, se ha obtenido un producto con la totalidad de las funcionalidades previstas para su correcto funcionamiento. Se les brinda a los administradores del sistema una solución mucho más funcional que la anterior versión del Módulo de Administración, incorporándosele a esta nueva versión un grupo de operaciones fundamentalmente orientadas hacia una eficiente gestión de usuarios.

CONCLUSIONES

- Ha sido realizado un estudio de las tendencias y tecnologías actuales para el desarrollo del Componente de Seguridad, asimilando la arquitectura definida por el MINSAP para el desarrollo de sus aplicaciones (Orientada a Servicios y Basada en Componentes (SOA-CBA)).

- Se han analizado los procesos de negocio actuales de Autenticación, Autorización y Auditoria (AAA) en los componentes desarrollados en el Área Temática SAS, así como los servicios públicos del Componentes SAAA.

- Se ha desarrollado un Componente de Seguridad para los productos del Área Temática cualitativamente superior a sus antecesores, perfeccionándose y estandarizándose los procesos de Autenticación, Autorización y Auditoría.

- Se desarrollaron los Servicios Web XML públicos: Autenticar, Autorizar, Adicionar Traza, Buscar Traza, Búsqueda de Usuarios, Búsqueda de Componentes para los componentes desarrollados por el Área Temática con sus respectivas descripciones (WSDL).

- La información gestionada será objeto de cuidadosa protección contra estados corruptos e inconsistentes, teniendo acceso a ella solo el personal autorizado.

- Durante el proceso de transición del sistema informático obtenido tendrán que llevarse a cabo importantes acciones de capacitación a los usuarios finales para su exitoso despliegue.

RECOMENDACIONES

- Someter al Componente de Seguridad a un proceso de pruebas por el Grupo de Calidad de la Facultad, emitiéndose de este modo la certificación requerida.

- Utilizar el Componente de Seguridad como sistema que gestione la Autenticación, Autorización y Auditoría para todos los proyectos desarrollados por el Área Temática Sistemas de Apoyo a la Salud.

- Realizar una futura versión del Componente de Seguridad utilizando el framework Symfony por las ventajas que el mismo proporciona.

- Implementar un patrón Proxy para manipular todas las peticiones realizadas entre los componentes integrados con el Componente de Seguridad.

REFERENCIAS BIBLIOGRÁFICAS

[1]. Espinosa Eduardo. *Seguridad en la Web*. Ing. en Computación. Disponible en: http://www.espina.info/papers/seguridadenlaWeb.pdf.

[2]. Ataques más comunes sobre aplicaciones Web, 2 de mayo de 2006. Disponible en: http://www.hispasec.com/corporate/noticias/94

[3]. ¿Qué es LDAP?, 10 de diciembre de 2004. Disponible en: http://www.ldap-es.org/contenido/04/12/1.-%C2%BFque-es-ldap%3F

[4]. Ídem a la referencia 3.

[5]. Ídem a la referencia 3.

[6]. Pressman, Roger S. *Ingeniería de Software, un enfoque práctico. Parte 1*. La Habana, Cuba. Editorial Félix Varela. 2005.

[7]. Gudiño Fleites, Pedro. *Tutorial de Sistemas Distribuidos I*. Departamento de Sistemas y Computación. Instituto Tecnológico de Colima. México. 2004. Disponible en: http://www.itcolima.edu.mx/profesores/tutoriales/sistemas_distribuidos_I/sd_u1_1.htm

[8]. Gómez Karel, González Leonardo, Arencibia Annia. *Centro de Control para el Sistema de Información para la Salud.* Trabajo de Diploma para optar por el título de Ingeniero Informático. Universidad de las Ciencias Informáticas. La Habana, Cuba, junio de 2007.

[9]. SOFTEL. *Documento sobre la Arquitectura de Software a emplear en los componentes del Sistema de Información para la Salud. La Habana. Cuba. 2006.*

[10]. Aruquipa Chambi Marcelo G., Márquez Granado Edwin P. *Desarrollo de Software Basado en Componentes.* Universidad Mayor de San Andrés. La Paz Bolivia, agosto de 2007.

Disponible en: http://pgi.umsa.bo/enlaces/investigacion/pdf/INGSW3_23.pdf

[11]. Maldonado Segura, José Alberto et al. *Tecnologías de la información al servicio de la historia clínica electrónica. Sociedad Española de Informática para la Salud*. España.2002. p 154

[12]. Reynoso, Carlos; Kicillof, Nicolás. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. Universidad de Buenos Aires 2004.

[13]. Larman. Craig. Patrones Grasp. 2005. Disponible en: http://jorgesaavedra.wordpress.com.

[14]. Ídem a la referencia 12.

[15]. Rodas Hinostroza. Raúl. Usuarios y Grupos en Linux. Octubre 2006. Disponible en: http://www.linuxcentro.net/linux/staticpages/index.php?page=CaracterísticasPHP

[16]. *¿Qué es JavaScript?* Disponible en: http://www.librosweb.es/JavaScript/capitulo1.html

[17]. *The Yahoo! User Interface Library (YUI).* Disponible en: http://developer.yahoo.com/yui/

[18]. *¿Qué es HTML?* Disponible en: http://www.librosweb.es/xhtml/capitulo1.html

[19]. Orivers. *XML Copy Editor es un editor de XML para Linux.* Noviembre 2007. Disponible en: http://www.entrebits.cl/index.php?option=com_content&task=view&id=463

[20]. Minnick, Chris; Valentine, Chelsea. *XHTML Serie Práctica. Nueva*, Estados Unidos.2000.p 7

[21]. Protocolo HTTPS:
Disponible en: http://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol_Secure

[22]. Pérez, Javier. *Maestros del Web¿Qué es Smarty.* Disponible en: http://www.maestrosdelweb.com/editorial/que-es-smarty/

[23]. James Garrett, Jesse. *Ajax. Un nuevo acercamiento a las aplicaciones Web*. Julio 2005. Disponible en: http://www.maestrosdelweb.com/editorial/ajax/

[24]. *Programación en PHP y Bases de Datos*. Disponible en: http://www.iescamp.es/tutoriales/php/tema1/tema1p21.htm

[25]. *PostgreSQL vs. MySQL.* Disponible en: http://www.netpecos.org/docs/mysql_postgres/x15.html

[26]. Jacobson, Ivar; Booch, Grady; Rumbaugh, James. *El Proceso Unificado de Desarrollo de Software. La Habana. Cuba. Editorial Félix Varela*. 2004. Pág. 4, 5, 6 y 7

[27]. Navarro Franco, Ángel José. *UML en acción. Modelando Aplicaciones Web.* Instituto Superior Politécnico José Antonio Echeverría. La Habana, Cuba, Mayo 2005.

[28]. Ídem a la referencia 8.

[29]. SSL en Apache2 Debian Etch. Disponible en: http://www.pacheco.org.mx/SoftwareLibre/ApacheSSLEnDebianEtch

BIBLIOGRAFÍA

Aruquipa Chambi Marcelo G., Márquez Granado Edwin P. *Desarrollo de Software Basado en Componentes.* Universidad Mayor de San Andrés. La Paz Bolivia, agosto de 2007.

Ataques más comunes sobre aplicaciones Web, 2 de mayo de 2006. Disponible en: http://www.hispasec.com/corporate/noticias/94

Espinosa Eduardo. *Seguridad en la Web*. Ing. en Computación. Disponible en: http://www.espina.info/papers/seguridadenlaWeb.pdf.

Gómez Karel, González Leonardo, Arencibia Annia. *Centro de Control para el Sistema de Información para la Salud.* Trabajo de Diploma para optar por el título de Ingeniero Informático. Universidad de las Ciencias Informáticas. La Habana, Cuba, junio de 2007.

Gudiño Fleites, Pedro. *Tutorial de Sistemas Distribuidos I*. Departamento de Sistemas y Computación. Instituto Tecnológico de Colima. México. 2004. Disponible en: http://www.itcolima.edu.mx/profesores/tutoriales/sistemas_distribuidos_I/sd_u1_1.htm

Jacobson, Ivar; Booch, Grady; Rumbaugh, James. *El Proceso Unificado de Desarrollo de Software. La Habana. Cuba. Editorial Félix Varela*. 2004. Pág. 4, 5, 6 y 7

James Garrett, Jesse. *Ajax. Un nuevo acercamiento a las aplicaciones Web*. Julio 2005. Disponible en: http://www.maestrosdelweb.com/editorial/ajax/

Larman. Craig. Patrones Grasp. 2005. Disponible en: http://jorgesaavedra.wordpress.com.

Maldonado Segura, José Alberto et al. *Tecnologías de la información al servicio de la historia clínica electrónica. Sociedad Española de Informática para la Salud*. España.2002. p 154

Minnick, Chris; Valentine, Chelsea. *XHTML Serie Práctica. Nueva*, Estados Unidos.2000.p 7

Navarro Franco, Ángel José. *UML en acción. Modelando Aplicaciones Web.* Instituto Superior Politécnico José Antonio Echeverría. La Habana, Cuba, Mayo 2005.

Orivers. *XML Copy Editor es un editor de XML para Linux.* Noviembre 2007. Disponible en: http://www.entrebits.cl/index.php?option=com_content&task=view&id=463

Pérez, Javier. *Maestros del Web¿Qué es Smarty.* Disponible en: http://www.maestrosdelweb.com/editorial/que-es-smarty/

PostgreSQL vs. MySQL. Disponible en: http://www.netpecos.org/docs/mysql_postgres/x15.html

Pressman, Roger S. *Ingeniería de Software, un enfoque práctico. Parte 1*. La Habana, Cuba. Editorial Félix Varela. 2005

Programación en PHP y Bases de Datos. Disponible en: http://www.iescamp.es/tutoriales/php/tema1/tema1p21.htm

Protocolo HTTPS:
Disponible en: http://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol_Secure

¿Qué es HTML? Disponible en: http://www.librosweb.es/xhtml/capitulo1.html

¿Qué es JavaScript? Disponible en: http://www.librosweb.es/JavaScript/capitulo1.html

¿Qué es LDAP?, 10 de diciembre de 2004. Disponible en: http://www.ldap-es.org/contenido/04/12/1.-%C2%BFque-es-ldap%3F

Reynoso, Carlos; Kicillof, Nicolás. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. Universidad de Buenos Aires 2004.

Rodas Hinostroza. Raúl. Usuarios y Grupos en Linux. Octubre 2006. Disponible en: http://www.linuxcentro.net/linux/staticpages/index.php?page=CaracterísticasPHP

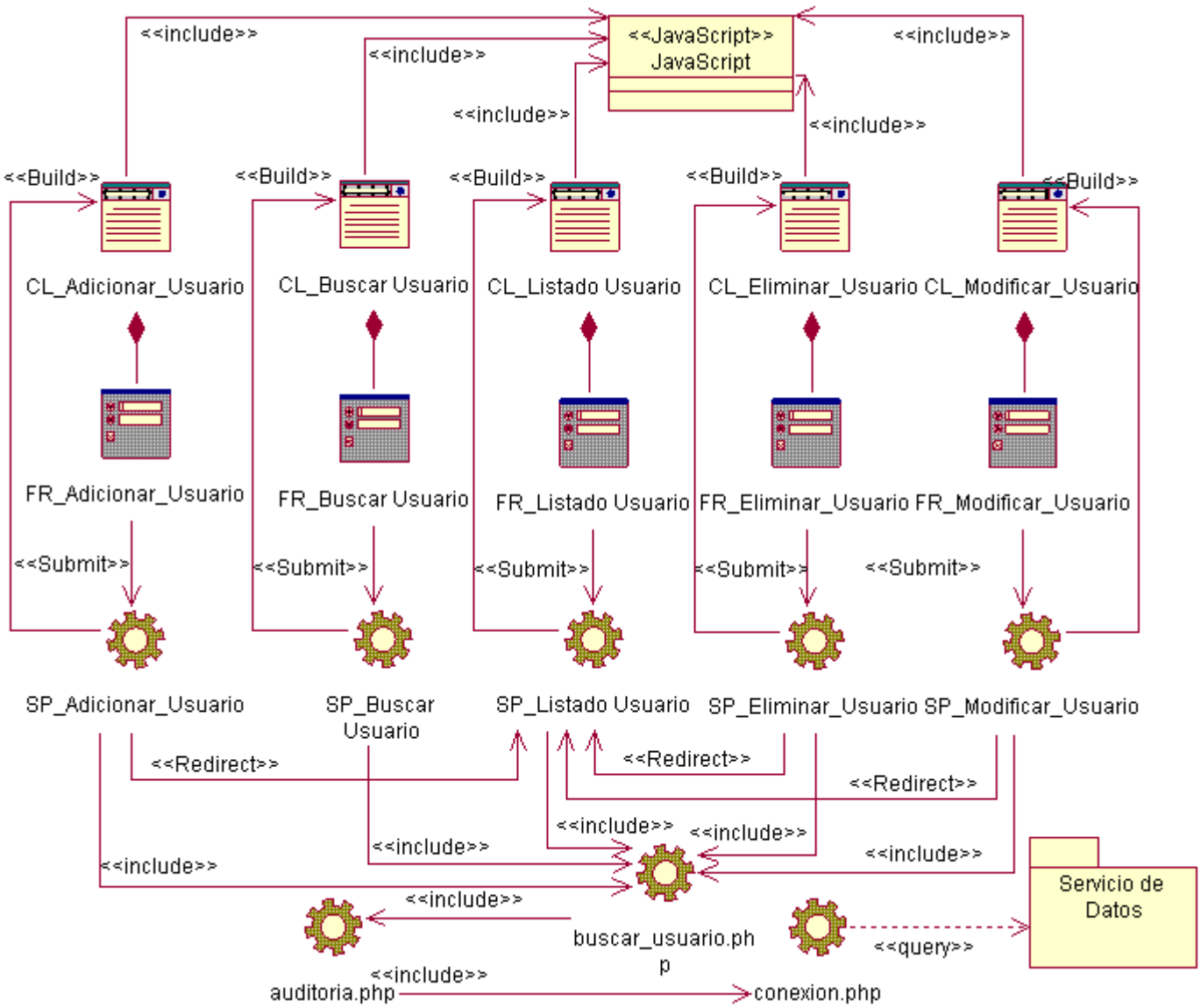
SOFTEL. *Documento sobre la Arquitectura de Software a emplear en los componentes del Sistema de Información para la Salud. La Habana. Cuba. 2006.*
Disponible en: http://pgi.umsa.bo/enlaces/investigacion/pdf/INGSW3_23.pdf

SSL en Apache2 Debian Etch. Disponible en:
http://www.pacheco.org.mx/SoftwareLibre/ApacheSSLEnDebianEtch

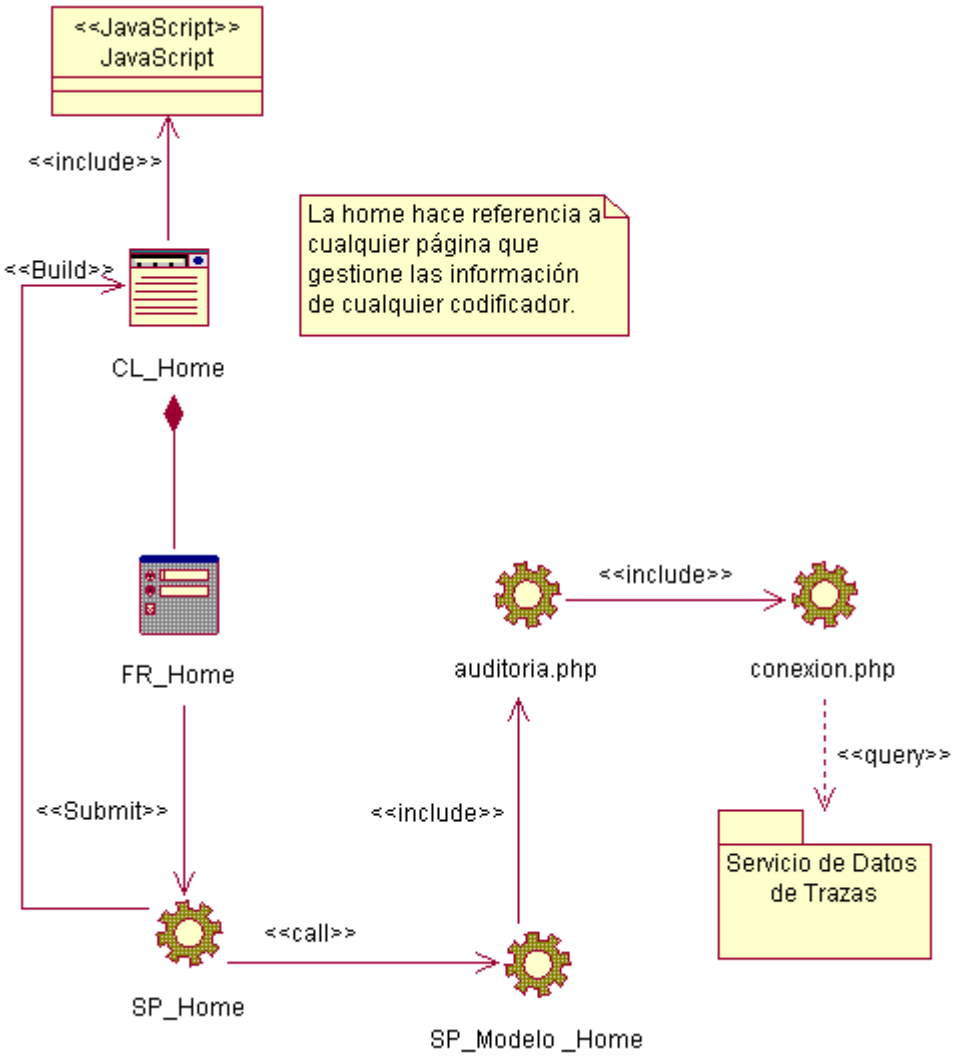
The Yahoo! User Interface Library (YUI). Disponible en: http://developer.yahoo.com/yui/

ANEXOS

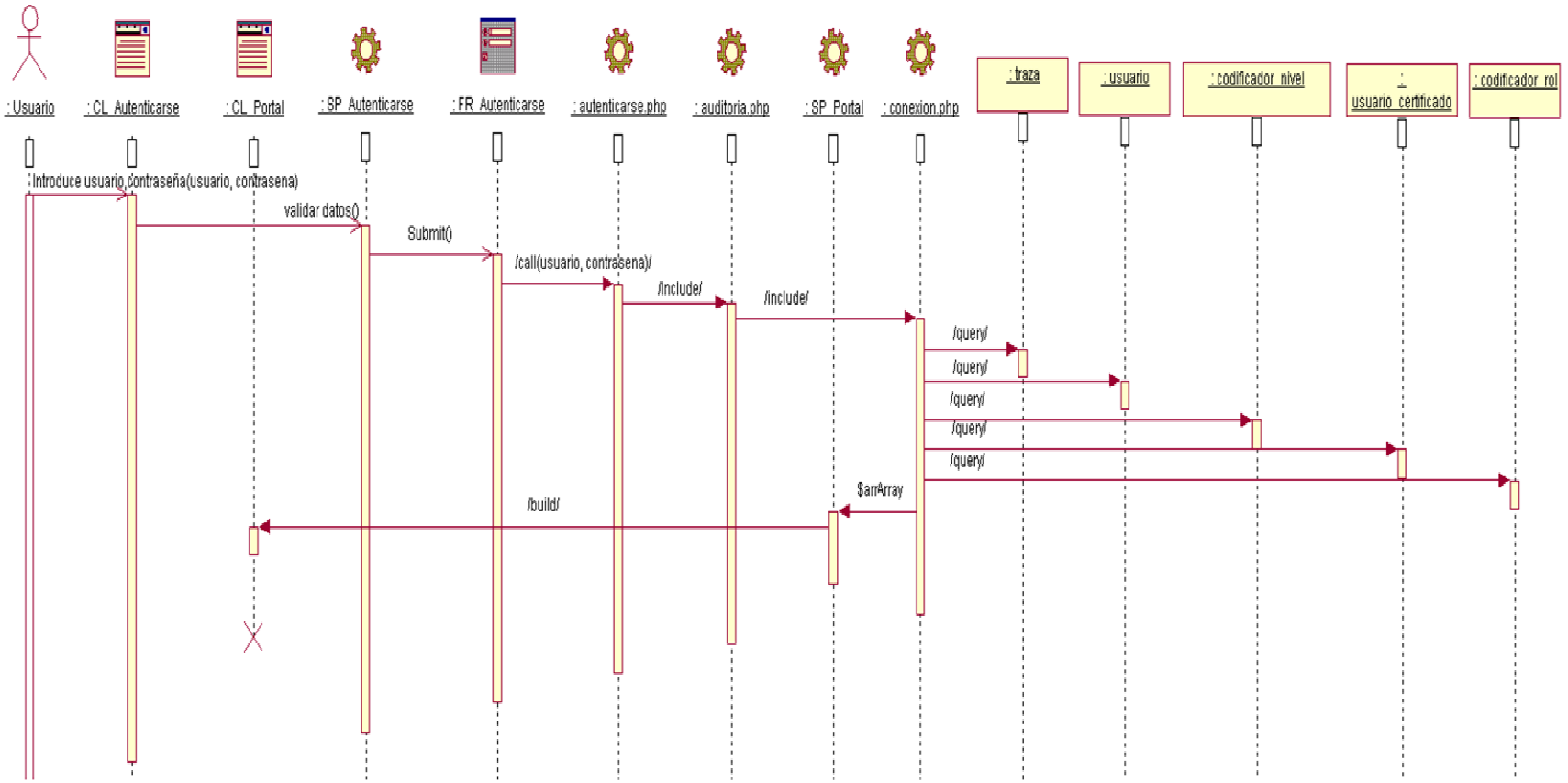
ANEXO I: DIAGRAMA DE DISEÑO: *CUS11_Buscar_Usuario*



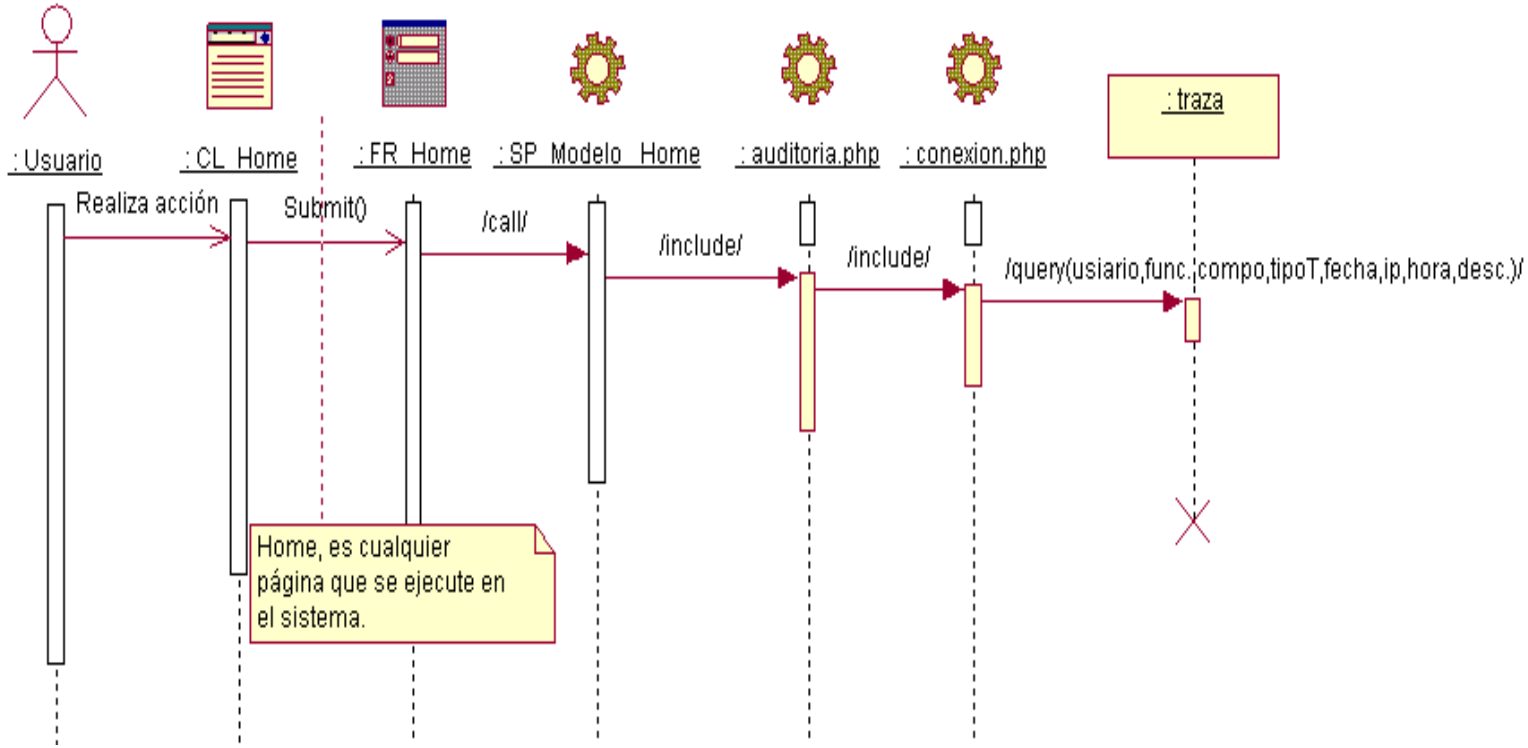
ANEXO II: DIAGRAMA DE DISEÑO: *CUS12_Registrar_Traza*



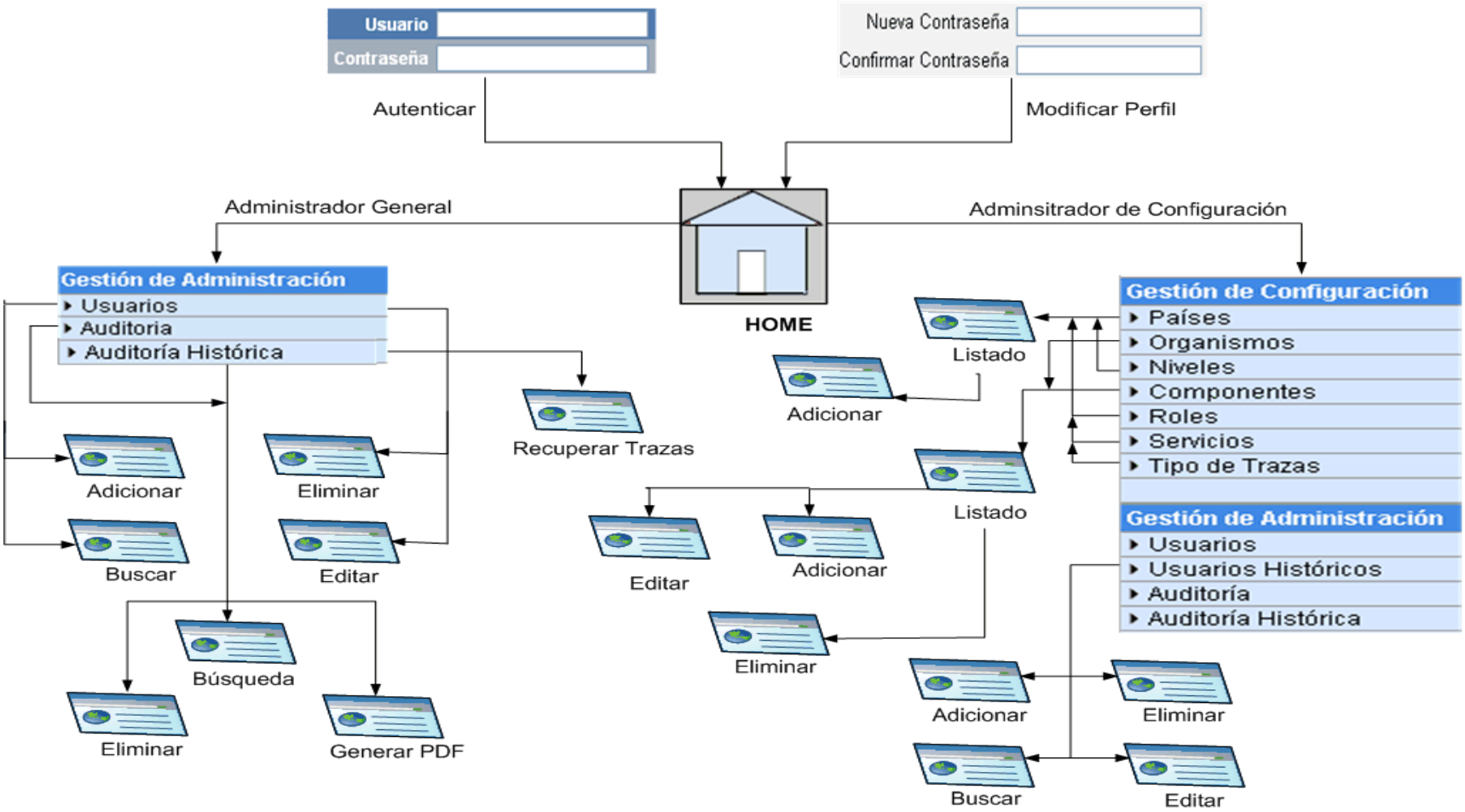
ANEXO III: DIAGRAMA DE SECUENCIA: *CUS1_Autenticar*



ANEXO VI: DIAGRAMA DE SECUENCIA: *CUS12_Registrar_Traza*

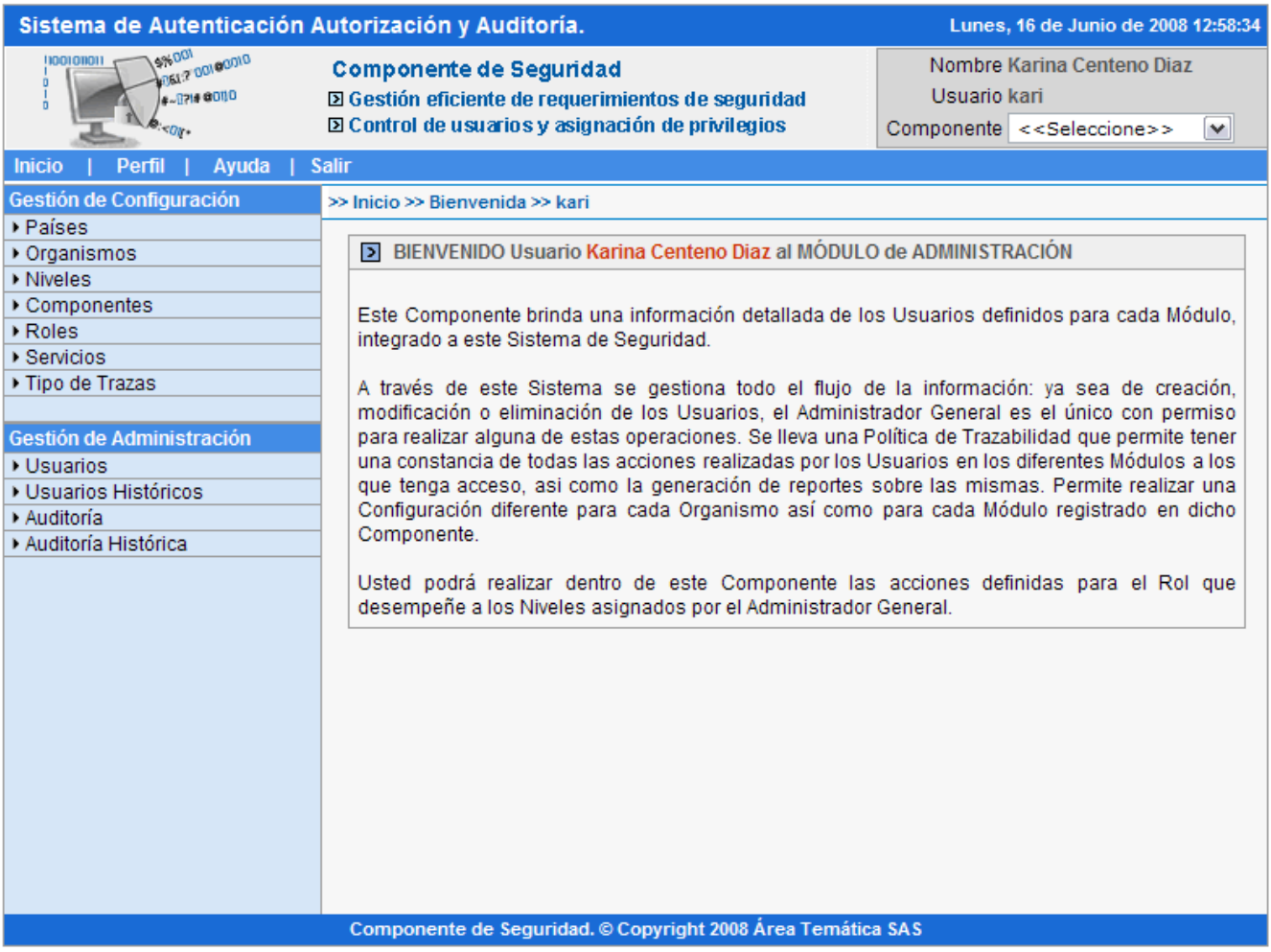


ANEXO V: MAPA DE NAVEGACIÓN



ANEXO VI: PÁGINA DE AUTENTICACIÓN DEL COMPONENTE DE SEGURIDAD

ANEXO VII: PÁGINA PRINCIPAL DEL COMPONENTE DE SEGURIDAD (ADMINISTRADOR CONFIGURACIÓN)



GLOSARIO DE TÉRMINOS

FTP: Siglas de File Transfer Protocol. Es un protocolo de transferencia de archivos entre sistemas conectados a una red TCP basado en la arquitectura cliente - servidor, de manera que desde un equipo cliente nos podemos conectar a un servidor para descargar archivos desde él o para enviarle nuestros propios archivos independientemente del sistema operativo utilizado en cada equipo.

TCP: Siglas de Transmission Control Protocol. Es uno de los protocolos fundamentales en Internet. Muchos programas dentro de una red de datos compuesta por ordenadores pueden usarlo para crear conexiones entre ellos a través de las cuales puede enviarse un flujo de datos. El protocolo garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron.

Sockets: Designa un concepto abstracto por el cual dos programas -posiblemente situados en computadoras distintas- puede intercambiarse cualquier flujo de datos, generalmente de manera fiable y ordenada. Un *socket* queda definido por una dirección IP, un protocolo y un número de puerto.

RPC: Siglas de Remote Procedure Call. Es un protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos. El protocolo es un gran avance sobre los sockets usados hasta el momento. De esta manera el programador no tendría que estar pendiente de las comunicaciones, estando éstas encapsuladas dentro de las RPC.

Cookie: Es un fragmento de información que se almacena en el disco duro del visitante de una página Web a través de su navegador, a petición del servidor de la página. Esta información puede ser luego recuperada por el servidor en posteriores visitas.

Código SQL: Es una vulnerabilidad informática en el nivel de la validación de las entradas a la base de datos de una aplicación. El origen es el filtrado incorrecto de las variables utilizadas en las partes del programa con código SQL.

XPath: Siglas de Path Language. Es un lenguaje que permite construir expresiones que recorren y procesan un documento XML. La idea es parecida a las expresiones regulares para seleccionar partes

de un texto sin atributos. Permite buscar y seleccionar teniendo en cuenta la estructura jerárquica del XML.

DOM: Siglas de Document Object Model. Es esencialmente un modelo computacional a través de la cual los programas y scripts pueden acceder y modificar dinámicamente el contenido, estructura y estilo de los documentos HTML y XML. Su objetivo es ofrecer un modelo orientado a objetos para el tratamiento y manipulación en tiempo real -o en forma dinámica- a la vez que de manera estática de páginas de internet.

SQLite: Es un sistema de gestión de bases de datos relacional. A diferencia de los sistemas de gestión de base de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la librería SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones.

SGML: Siglas de Standard Generalized Markup Language. Consiste en un sistema para la organización y etiquetado de documentos. El lenguaje SGML sirve para especificar las reglas de etiquetado de documentos y no impone en sí ningún conjunto de etiquetas en especial.

HTTP: Siglas de HyperText Transfer Protocol. Es un protocolo de transferencia de hipertexto, define la sintaxis y la semántica que utilizan los elementos software de la arquitectura Web (clientes, servidores, proxies) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.

Proxi: Hace referencia a un programa o dispositivo que realiza una acción en representación de otro. La finalidad más habitual es la de servidor proxy, que sirve para permitir el acceso a Internet a todos los equipos de una organización cuando sólo se puede disponer de un único equipo conectado, esto es, una única dirección IP.

Plugin: Es una aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica, generalmente muy específica, como por ejemplo servir como controlador en una aplicación, para hacer así funcionar un dispositivo en otro programa. Ésta aplicación adicional es ejecutada por la aplicación principal.

Iframe: Es un elemento HTML que permite insertar o incrustar un documento HTML dentro de un documento HTML principal. Insertar un iframe entre una sección o bloque es semejante a insertar un elemento *object*. Esto permite que se pueda insertar un documento HTML dentro de otro, alineado de acuerdo a sus límites.

JSON: Es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML. La simplicidad de JSON ha dado lugar a la generalización de su uso, especialmente como alternativa a XML en AJAX.

EBML: Siglas de Extensible Binary Meta Language fue diseñado como una extensión binaria simplificada de XML, con el propósito de almacenar y manipular datos de forma jerárquica con campos de longitud variable.

PDP: Es un formato de almacenamiento de documentos. Está especialmente ideado para documentos susceptibles de ser impresos, ya que especifica toda la información necesaria para la presentación final del documento, determinando todos los detalles de cómo va a quedar, no requiriéndose procesos anteriores de ajuste ni de maquetación.

SSL: Es un protocol que proporciona cifrado de datos, autenticación de servidores, integridad de mensajes y, opcionalmente, autenticación de cliente para conexiones TCP/IP.

BSD: Siglas de Berkeley Software Distribution, es una Licencia otorgada principalmente para los sistemas BSD. Pertenece al grupo de licencias de software Libre. Esta licencia tiene pocas restricciones estando muy cercana al dominio público. La licencia BSD permite el uso del código fuente en software no libre.

Widget: Es una pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños que son ejecutados por un motor de *widgets* o Widget Engine. Entre sus objetivos están los de dar fácil acceso a funciones frecuentemente usadas y proveer de información visual.

LGPL: Es una licencia de software creada por la Free Software Foundation. Los contratos de licencia de la mayor parte del software están diseñados para jugar con su libertad de compartir y modificar dicho software.

RIA: Siglas de Rich Internet Applications es un nuevo tipo de aplicación con más ventajas que las tradicionales aplicaciones Web. Esta surge como una combinación de las ventajas que ofrecen las aplicaciones Web y las aplicaciones tradicionales.

ECMAScript*:* Es una especificación de lenguaje de programación publicada por ECMA International. El desarrollo empezó en 1996 y estuvo basado en el popular lenguaje JavaScript propuesto como estándar por Netscape Communications Corporation. Actualmente está aceptado como el estándar ISO 16262.

XMLHttpRequest: Es una interfaz empleada para realizar peticiones HTTP y HTTPS a servidores WEB. Para los datos transferidos se usa cualquier codificación basada en texto, incluyendo: texto plano, XML, JSON, HTML y codificaciones particulares específicas. La interfaz se presenta como una clase de la que una aplicación cliente puede generar tantas instancias como necesite para manejar el diálogo con el.