Universidad de las Ciencias Informáticas

Facultad 10

TÍTULO: Componente de Seguridad para la Arquitectura de la Informatización de la UCI**.**

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

Autor: Aluisco Miguel Ricardo Mastrapa.

Tutor: Ing. Yunier Saborit Ramírez.
Ing. Dayron Cruz Iñigo.

Ciudad de La Habana, Cuba

Junio, 2009

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 firmo la presente a los ____ días del mes de _____ del año _____.

_____
Aluisco Miguel Ricardo Mastrapa

_____ _____
Ing. Dayron Cruz Iñigo Ing. Yunier Saborit Ramírez

OPINIÓN DEL USUARIO DEL TRABAJO DE DIPLOMA.

El Trabajo de Diploma, titulado “Componente de Seguridad para la Arquitectura de Informatización de la UCI.”, fue realizado en la Universidad de las Ciencias Informáticas (UCI) de la provincia de Ciudad Habana. Esta entidad considera que, en correspondencia con los objetivos trazados, el trabajo realizado le satisface

__ Totalmente

__ Parcialmente en un ____ %

Los resultados de este Trabajo de Diploma le reportan a esta entidad los beneficios siguientes:

Como resultado de la implantación de este trabajo se reportará un efecto económico que asciende a:

_____

Y para que así conste, se firma la presente a los días ___ del mes ___ del año 2008

_____ _____

Representante de la entidad Cargo

_____ _____

Firma Cuño

OPINIÓN DEL USUARIO DEL TRABAJO DE DIPLOMA.

Título: Componente de Seguridad para la Arquitectura de la Informatización de la UCI (SegRED).

Autor: Aluisco Miguel Ricardo Mastrapa

El tutor del presente Trabajo de Diploma considera que durante su ejecución el estudiante mostró las cualidades que a continuación se detallan.

Por todo lo anteriormente expresado considero que el estudiante está apto para ejercer como Ingeniero Informático; y propongo que se le otorgue al Trabajo de Diploma la calificación de: ___

Ing. Yunier Saborit Ramírez

Tutor

_____ _____

Firma Fecha

Ing. Dayron Cruz Iñigo

Tutor

_____ _____

Firma

AGRADECIMIENTO

A mis padres que siempre han estado cerca para apoyarme siempre que lo necesito.

A mi hermano que aunque está lejos siempre se preocupa y me aconseja.

A todos aquellos que compartieron conmigo estos 5 años.

A mis amigos, incontables que tengo y se encuentran aquí.

A Dayron por ser además de mi tutor mi amigo y estar siempre en las noches de desvelo, por apoyarme siempre que lo necesite.

A Julio por ser el hombre sonrisas, por siempre animar nuestras noches de desvelo con sus chistes y bromas además de brindar sus ideas.

A Yanet por atenderme aun cuando siempre ha estado muy ocupada con su trabajo, siempre me guardo un pedacito de su tiempo para ayudarme.

A Yunier por ser la persona de las ideas, por tutor y amigo.

A toda mi familia por estar siempre pendiente de mí y de ayudarme.

A todos aquellos que de una forma u otra hicieron posible que hoy me encuentre aquí.

A todos, ¡Muchas Gracias!

DEDICATORIA

A nuestro Comandante Fidel Castro por ser guía de esta revolución que ha hecho realidad este sueño tan bonito.

A mis padres Vicente y Nancy por siempre apoyarme y brindarme sus experiencias para que saliera adelante en la vida.

A Dayron y Yanet que sin ellos nunca hubiera sido posible este momento.

A mi tutor Yunier por saber siempre entender todos mis apuros.

A Eliurkis por su gran sabiduría y apoyo.

A mi hermano por confiar en mí.

A todos aquellos que confiaron en que este día llegaría.

RESUMEN

El mecanismo de autenticación – autorización de las diferentes plataformas en la Universidad de las Ciencias Informáticas (UCI) es un componente se seguridad que asegura la autenticidad de los usuarios. Actualmente este proceso se hace engorroso al tener que autenticar a cada sitio web que se accede ya que el sistema de autenticación de los mismos es de forma independiente. Esto hace que el usuario pierda gran cantidad del su tiempo autenticando. Muchos de estos sitios al no ser desarrollados bajo una misma plataforma (Framework) crea que su mantenimiento conlleve una gran pérdida de tiempo.

En la Universidad el flujo de accesos a las diferentes aplicaciones es de gran envergadura, por lo que se hace necesaria la implementación de un sistema capaz de controlar de forma única todos estos accesos, garantizando además la autenticidad de los usuarios y los niveles de acceso que estos tengan en cada una de las aplicaciones a las que se acceda. Esto provoco la necesidad de crear la implementación del Componente de Seguridad para la Arquitectura de la Informatización de la UCI (SegRED), sistema que brinda la posibilidad de que el usuario autentique una única vez mediante un *Single Sign On* (SSO) y establece los permisos que este tendrá en las aplicaciones. Como también soporta la posibilidad de establecer el mecanismo de búsqueda de usuarios, obtener información de las aplicaciones, funcionalidades, roles y el cambio de permisos por grupo de usuarios de la Universidad.

El proceso de autenticación mediante el SSO se realizaría con el objetivo de gestionar las sesiones de los usuarios en un único proceso de autenticación invisible a los ojos de los mismos, para esto se utilizaría un sistema con fachada de servicios web capaz de gestionar las sesiones que creen los usuarios en el dominio UCI.

Este documento contiene los resultados de un estudio realizado en la Universidad de las Ciencias Informáticas (UCI) para la implementación de SegRED y se explican los conceptos relacionados con el mismo.

Palabras Claves: *Autenticación, Autorización, Seguridad Informática, Solución de Software Implantada, Funcionalidad, Usuario, Rol, Servicio Web.*

INDICE

INTRODUCCIÓN

Ante la demanda en la autopista de la información, la cual se llama Internet, muchas instituciones se dieron a la tarea de sacarle provecho a las múltiples ventajas que brinda la Tecnología Web, con el fin de garantizar una mejor colaboración y comunicación. Al ser Internet una red tan grande surge la idea de crear una red menor y más centralizada, la cual contará con un canal de comunicación unificado y coherente, para toda la información corporativa de una institución determinada. A este nuevo concepto en el mundo del desarrollo informático se le denominó “Intranet”.

En la actualidad existen muchas intranets que tanto como Internet se soportan bajo aplicaciones Web. La popularidad alcanzada en la Web se debe a la gran facilidad que ofrece al usuario, y expande el tiempo del desarrollador al no tener que distribuir e instalar el software.

La Universidad de las Ciencias Informáticas (UCI), siendo uno más de los grandes proyectos de la Revolución, nació para contribuir al proceso de informatización de la sociedad cubana, representa además una capacidad para generar soluciones integrales y servicios de software.

La Universidad dispone además de una Intranet bastante amplia, siendo la red de más importancia en la UCI pues comparte información y recursos entre los distintos usuarios de la misma. Actualmente cuenta con alrededor de 60 aplicaciones Web en funcionamiento. Las mismas manejan un volumen considerable de información el cual establece en muchos casos un acceso limitado. Para garantizar lo antes expuesto cada una de las aplicaciones cuenta con un módulo de seguridad, el cual entre sus funciones principales establece la autenticación y autorización de usuarios.

Por lo general todas las aplicaciones que se encuentran en la Intranet autentican directamente al dominio uci.cu, otras cuentan con bases de datos propias para el manejo de usuarios. Esto hace que navegar entre las distintas aplicaciones haga que los usuarios se registren continuamente al acceder a cada una de ellas. Si contamos con que la Universidad tiene un total de más de 16000 cuentas de usuario y el proceso de autenticación – autorización de estas aplicaciones no se encuentra centralizada, hay un gran derroche de los recursos destinados a estas labores.

El hecho de que cada aplicación particularice este proceso, trae como consecuencia además, que no todas tengan la seguridad que necesitan según la información que manejan y que no exista un control centralizado y actualizado de la cantidad de personas que tienen acceso a un sitio específico, ni de la cantidad de aplicaciones a las que puede acceder una determinada persona, considerándose ambos listados sumamente importantes.

INTRODUCCIÓN

La Universidad se encuentra inmersa en un proceso de migración hacia el software libre, los módulos de seguridad que fueron implementados, para una plataforma o sistema de bases de datos, necesitan un gran mantenimiento para migrar. Cada proyecto tiene que desarrollar un módulo de seguridad que se adapte a un lenguaje y métodos de accesos establecidos, esto lleva consigo una pérdida de tiempo en el estudio e implementación de dichos módulos, que son comunes para muchos sistemas.

Para darle respuesta a esta problemática, se planteó el siguiente **problema científico**: ¿Cómo centralizar el proceso de autenticación – autorización en las aplicaciones o sistemas de la Universidad?

Por tanto, como **objeto de estudio** se ha tomado la seguridad de aplicaciones Web en la UCI, desarrollándolos en un **campo de acción** centrado en el proceso de autenticación – autorización de aplicaciones de la UCI.

El objetivo general de la investigación es el desarrollo de un sistema en software libre que permita la gestión de manera centralizada de los permisos de usuarios a las diferentes aplicaciones en la Universidad.

Como **objetivos específicos de la investigación** están:

- Analizar el estado en que quedarán las aplicaciones en la Universidad.
- Desarrollar una aplicación que permita la configuración de SegRED en la UCI, con una interfaz amigable e interactiva, además del alto grado de usabilidad que debe poseer.
- Desarrollar un Servicio Web que ofrezca la funcionalidad del sistema a otras aplicaciones.

Para cumplir con los objetivos propuestos se desarrollaron las siguientes **tareas de la investigación:**

- Realizar un estudio del estado del arte de las aplicaciones de seguridad existentes.
- Analizar la propuesta de solución de software ya realizada en el Sistema de Gestión de Sesiones y en el Sistema de Gestión de Accesos.
- Verificar las necesidades de gestión de seguridad en las aplicaciones de la institución.
- Refinar los requisitos funcionales y no funcionales con los que debe cumplir la primera versión del sistema.
- Escoger el lenguaje, las herramientas y las tecnologías a utilizar en el desarrollo del sistema.
- Realizar el proceso de desarrollo de la aplicación.

Dar solución al problema antes mencionado, permitiría tener un mayor control de los diferentes permisos de cada usuario y los accesos a cada aplicación, teniendo en cuenta las funcionalidades de los distintos sistemas que brindan servicios en la UCI. Además, lograría un considerable ahorro en el trabajo, tiempo y recursos, dedicados al desarrollo y mantenimiento de las aplicaciones que lo utilicen, pues todo lo referente a la autenticación y autorización, se separaría de los proyectos. Por lo que se **defiende la idea** de que el desarrollo de un sistema automatizado, que centralice el proceso de autenticación – autorización de aplicaciones en la UCI, logrará una mayor seguridad en la intranet universitaria, e influirá positivamente en el crecimiento de la producción de software.

Para cumplir con la meta trazada y satisfacer las necesidades de la organización cliente, es necesario hacer una profunda investigación sobre el tema en cuestión. La recopilación de información relacionada con dicha investigación se hace a partir de la definición de la muestra poblacional, para ello se seleccionó como población a todas las aplicaciones Web que están actualmente publicadas en el centro.

Métodos científicos de investigación

Para el desarrollo de la investigación se utilizarán métodos teóricos y empíricos.

Métodos teóricos

- **Análisis histórico-lógico:** A través de este método se estudia la trayectoria real de los elementos que se utilizan en la implementación de sistemas Web, dígase origen y evolución de los diferentes lenguajes de programación, frameworks, IDEs y Sistemas Gestores de Base de Datos.

- **Análisis y Síntesis:** Para la implementación del sistema se realizó una investigación previa de los procesos que intervienen en el desarrollo de software y los principales elementos que integran las metodologías como son las fases, disciplinas, actividades, roles, artefactos y otros. Los resultados alcanzados en conjunto con el conocimiento empírico nos permitió conseguir un dominio abarcador del tema al que propone la solución.

Métodos empíricos

- **Entrevista:** Se realizaron múltiples entrevistas a los analistas, jefes y demás implicados del proyecto en aras de obtener información referente al funcionamiento y dinámica del negocio en

el que se enmarca el problema, que ellos analizaron y cuya implementación se propone con este trabajo.

En la actualidad, aunque existen otras aplicaciones desarrolladas tanto a nivel nacional como internacional, que contribuyen a la garantía de un componente de seguridad, algunas cumplen con parte de las funcionalidades requeridas, pero presentan un grupo de limitaciones que hacen necesario desarrollar una solución diferente.

El presente documento está compuesto por tres capítulos, que incluyen todo lo relacionado con el trabajo investigativo, así como la arquitectura, la descripción de la solución propuesta y la implementación del sistema.

En el **Capítulo I Fundamentación Teórica:** Comprende un análisis de los sistemas que existen en la actualidad tanto a nivel Internacional, Nacional y en la Universidad y se vinculan con la investigación, además se realiza el estado del arte de las tecnologías y herramientas a utilizar en el desarrollo de la aplicación.

En el **Capítulo II Descripción de la Arquitectura:** Se muestran los requerimientos funcionales y no funcionales realizándose además una descripción de la arquitectura a utilizar y un análisis de posibles implementaciones, componentes o módulos ya existentes y que puedan ser rehusados. En este capítulo se muestra la vista de despliegue así como las estrategias de codificación y los estándares y estilos a utilizar.

En el **Capítulo III Descripción y análisis de la solución propuesta.**: Se realiza el análisis del diseño propuesto por el analista refinando los diagramas de clases del diseño y realizando una descripción de las nuevas clases del diseño, mostrándose además el modelo de datos y la vista del modelo de implementación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

La seguridad en entornos digitales es uno de los factores más importantes, la autenticación y autorización son dos factores esenciales para garantizar la seguridad. La implementación de un sistema que conjuntamente pueda garantizar que estas dos funcionalidades puedan ser autenticas y seguras lleva un conjunto de investigaciones con el fin de garantizar la calidad del producto final a obtener.

Durante el análisis del presente capítulo, se explica en detalle el proceso de investigación llevado a cabo para la implementación del software, además se describen algunos otros sistemas vinculados con el mismo, permitiendo así la fundamentación de la tecnología utilizada para la realización de la propuesta.

1.2 Estudio del estado del Arte.

Como parte de la investigación se realiza un estudio profundo y detallado de los sistemas informáticos existentes, tanto en el país como en el resto del mundo, que pudieran dar solución al problema planteado. A continuación se abordan los más significativos.

1.2.1 Soluciones existentes nivel Internacional.

A nivel Internacional son varias las soluciones ya implementadas y desplegadas que garantizan la autenticación de usuarios dentro de las más relevantes están:

1.2.1.1 Servicio Central de Autentificación (Central Authentication Service (CAS))

El CAS es un sistema de autentificación creado originalmente por la Universidad de Yale, con el objetivo de proporcionar un medio confiable a las aplicaciones para la autenticación de usuarios. Se convirtió en un proyecto del Grupo de Interés Especial de las Arquitecturas de Java (“JA-SIG”, por sus siglas en ingles) en diciembre del 2004. Actualmente ofrece servicios de Single Sign On (SSO), los usuarios se autentican al servidor CAS y solo necesitan hacerlo una sola vez por sesión del navegador. Trabaja sólo con aplicaciones y recursos accedidos vía Web, los accesos son interceptados con la ayuda de un servidor proxy o de un componente instalado en el servidor Web destino. Brinda la posibilidad a las aplicaciones de autenticación por Proxy a terceras capas de abastecedores de servicios ("back-end"), que eligen aceptar sus credenciales de Proxy. (1)

Este sistema aunque simplifica los procedimientos que siguen las aplicaciones para la autenticación, no cumple con todos los requerimientos para satisfacer las necesidades de la UCI, pues no cuenta con mecanismos que gestionen el proceso de autorización y no permite el trabajo con roles.

Considerando que la automatización del proceso de autorización es de vital importancia para la UCI, y que el trabajo con roles es una premisa fundamental dentro de dicho proceso, de aplicarse el CAS como solución, habrá que emplear mucho tiempo en su estudio para saber cómo está diseñado e implementado y lograr una exitosa modificación de sus elementos, que satisfaga las necesidades de la Universidad. Por lo que se ahorrará más tiempo realizando un sistema nuevo que permita gestionar la autenticación y la autorización que utilizando el CAS.

1.2.1.2 Cams

El Cams es un Single Sign-On (SSO) software para Web que centraliza la autenticación de usuarios, el control de acceso y la administración. Provee seguridad para recursos que están hospedados en todos los líderes Web y aplicaciones J2EE servidor, están incluidos Apache, Microsoft IIS, BEA WebLogic, IBM WebSphere, JBoss, Oracle 9iAs, Pramati y Tomcat. Los recursos protegidos por Cams pueden residir en una intranet corporativa, extranet o Internet, y pueden ser documentos estáticos y JSP/servlet, ASP.Net, PHP, Cold Fusion, y aplicaciones Web CGI. (2)

Este software satisface muchas de las necesidades de la UCI, pero presenta un elevado costo y cobran los servicios de soporte que ofrecen. Por esta razón, resulta mucho más factible dedicar tiempo, recursos y esfuerzos en crear un software que responda a la solución del problema planteado, que pagar un alto costo monetario por uno que luego hay que adaptar a las condiciones de la Universidad.

1.2.2 Soluciones existentes en Cuba.

1.2.2.1 Singles Authentication Authorization and Account (SAAA)

El SAAA es un sistema desarrollado por SOFTEL para garantizar la seguridad de Infomed (red de salud del país), pero esta empresa no está autorizada para compartir ningún tipo de información relacionada con dicho sistema, ni siquiera pueden dar a conocer los servicios que oferta el software. Pero se conoció que cuenta con un limitado nivel en cuanto a la gestión para la administración de usuarios y no permite validar y homologar los datos personales de los usuarios con un registro externo que contenga los datos con un registro externo que contenga los datos de los ciudadanos.

En la versión actual del SAAA el mantenimiento de la integridad referencial en el flujo de la información se realiza parcialmente, utilizando un grupo de tablas denominadas caché, en las bases de datos de los componentes, contenedoras de información no asociada a su negocio de frecuente utilización, las que tienen que ser objeto de constantes actualizaciones en forma manual. Asimismo no es posible configurar las restricciones de integridad establecidas entre los componentes ni especificar el tipo de dependencia, como restrictiva o en cascada.

Este componente de seguridad no cumple con las necesidades de la Universidad, pues no establece un control de usuarios por roles y no garantiza la integridad referencial de los datos.

1.2.3 Soluciones existentes en la UCI

En la Universidad son varias las soluciones propuestas pero que no se están aplicando porque son soluciones incompletas que no resuelven todas las necesidades de la UCI algunas de ellas se especifican a continuación.

1.2.3.1 Sistema de Autenticación de Aplicaciones en la Intranet

En el año 2006, en la UCI se realizó un trabajo de diploma con el nombre *"Sistema de Autenticación de Aplicaciones en la Intranet"*. El mismo proponía una solución informática para automatizar y centralizar el proceso de autenticación de usuarios, pero el desarrollo del proyecto nunca llegó a ser efectivo, por lo que no se terminó su implementación.

Actualmente, si se terminara el desarrollo de dicho sistema, éste no solucionaría el problema existente, ya que sólo propone centralizar la autenticación de usuarios y mantiene la autorización como responsabilidad de cada aplicación.

A pesar de esto se valoró la posibilidad de reutilizar la propuesta anterior y adaptarla a las nuevas necesidades, pero luego de un estudio sobre las herramientas y lenguajes utilizados, se decidió brindar una nueva solución utilizando una tecnología más apropiada para su desarrollo.

No obstante, los estudios hechos en aquel momento sirven de base para esta investigación.

1.2.3.2 Sistema de Gestión de Accesos (SGA)

En el año 2008 se realiza un trabajo de diploma con el nombre "Sistema de Gestión de Accesos (SGA)", basado en la necesidad de implementar un sistema que controle centralizadamente los accesos y garantice seguridad en la red y aplicaciones de la misma, el cual brinda principalmente, los servicios de autenticación y autorización de usuarios a las disímiles aplicaciones. Asimismo soporta los

mecanismos necesarios para, a través de él, obtener información de las aplicaciones, funcionalidades, roles, usuarios y grupos de usuarios de la Universidad.

Esta aplicación pudo haber sido una de las soluciones de software a implantar en la Universidad, pero tiene la desventaja de que no contiene un *Single Sign On (SSO)*, por lo tanto los usuarios tenían que estarse autenticando siempre que se accediera a otra aplicación.

Actualmente no se encuentra en funcionamiento y el estudio para el desarrollo de este y una terminación para corregir las deficiencias determinaba un tiempo considerable, por lo que solo se toma la idea central y se implementa una nueva solución de software.

1.2.3.3 Sistema de Gestión de Sesiones (SGS)

En el 2008 se realiza el trabajo de diploma “Sistema de Gestión de Sesiones (SGS)” el cual se convierte en una iniciativa viable de protección y control que permite ofrecer servicios de valor añadido con altos niveles de seguridad y confianza a los usuarios.

La implantación de este sistema, logrará mejorar la seguridad de la red en nuestra universidad, permitiendo a su vez a los usuarios disminuir su trabajo de autenticación y el tiempo que pierden los mismos en este proceso. Además en la universidad cada uno de los servicios o aplicaciones cuenta con su propio componente de seguridad, lo cual generalmente compromete la seguridad de todo el sistema, dado que el nivel de seguridad de todo un sistema es igual al nivel de seguridad del componente más inseguro que tenga.

Esta es una de las razones por las que no se usa esta solución de software pues no hace un control de roles ni de usuarios, pero si permitiría a los usuarios autenticarse una única vez y hacer un control de sesiones eficiente. De este software si se toma un gran porciento para la implementación de la nueva solución de software a implantar.

1.2.3.4 Sistema de Autentificación Única de RINDE

El Sistema de Autentificación Única de RINDE surge para el proyecto llamado *Red de Integración y Desarrollo Nacional de Software Libre de la República Bolivariana de Venezuela* (RINDE), unido al movimiento de Software Libre en la República de Venezuela. Es un SSO que logra la integración de tres subsistemas webs que lo conforman, eliminando redundancia de datos y garantizando un mayor grado de seguridad.

Este proyecto se desplegó exitosamente en la República Bolivariana de Venezuela.

Este sistema no da cumplimiento a los objetivos y necesidades que sigue la propuesta de solución de software, pues se le hicieron severas modificaciones a los tres subsistemas con los que cuenta para su integración, por lo que su despliegue en la universidad conlleva una modificación extensa de todos los subsistemas existentes, además de que no contiene un control de los roles y funcionalidades, solo hace posible la autenticación en los tres subsistemas funcionando como un Single Sign On.

1.3 Fundamentación de la Tecnología, Herramientas y Metodología Aplicada.

Con el objetivo de lograr un producto, no solo que solucione el problema existente, sino que además tenga la calidad requerida y que cumpla con las normas tecnológicas establecidas por la Dirección de Informatización de la UCI; se realiza un estudio minucioso sobre la arquitectura, herramientas, metodologías y lenguajes a utilizar en la confección de la propuesta de solución. A continuación se justifica la tecnología escogida para la confección del software.

1.3.1 Servicio Web.

Existen múltiples definiciones sobre lo que son los servicios Web, lo que muestra su complejidad a la hora de dar un adecuado concepto que englobe todo lo que son e implican. Una posible sería hablar de ellos como un conjunto de protocolos y estándares que sirven para intercambiar datos entre distintas aplicaciones de software, (3) en menos palabras, un conjunto de tecnologías con capacidad para interoperar en la Web.

Los servicios Web son la revolución informática de la nueva generación de aplicaciones. Su éxito se debe a que a través de ellos, las aplicaciones puedan hacer uso de las funcionalidades brindadas por otras aplicaciones, independientemente de cómo se hayan implementado, cuál sea el sistema operativo o la plataforma en que se ejecutan y cuáles sean los dispositivos utilizados para obtener acceso a ellas. Su funcionamiento es tan sencillo como que los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web. (Ver Anexo 1)

Los servicios Web se componen de 3 elementos principales:

1. SOAP (Simple Object Acces Protocol): Es el protocolo que se emplea para el intercambio de datos entre aplicaciones.
2. WSDL (Web Services Description Language): Es empleado para describir las funciones del servicio Web, su ubicación y la forma en que debe utilizarse.

3. UDDI (Universal Description Discovery and Integration): Viene a ser como las páginas amarillas de los servicios Web. Permite el registro y la búsqueda de servicios Web para su utilización. (4)

¿Por qué servicio Web?

Por lo general cuando se habla de SOA, se piensa en la utilización de servicios Web para su implementación, no obstante se puede implementar una SOA utilizando cualquier otra tecnología basada en servicios, como CORBA (Common Object Request Broker Architecture), RMI (Java Remote Method Invocation) o DCOM (Distributed Component Object Model). La preferencia sobre los servicios Web se debe a la facilidad de su implementación con respecto a los otros mecanismo y sobre todo a la incapacidad de estos últimos, de usar el protocolo http para sus comunicaciones, lo cual descarta el aprovechamiento de Internet.

En la UCI muchas de las aplicaciones que se utilizan en la intranet ofrecen o consumen servicios Web de otras, es decir, existe una colaboración entre los sistemas de la red para lograr la reutilización y la funcionalidad de estas. El sistema que se propone debe brindar un servicio Web que permita a las aplicaciones la autenticación y autorización de sus usuarios. Asimismo debe proveer a las aplicaciones de toda la información relacionada con sus funcionalidades, sus usuarios, y los roles que estos últimos, juegan dentro de ellas.

1.3.2 Servidor Web.

Un Servidor Web es un programa que se ejecuta continuamente en un ordenador (también se emplea el término para referirse al ordenador que lo ejecuta), manteniéndose a la espera de peticiones por parte de un cliente (un navegador web) y que responde a estas peticiones adecuadamente, mediante una página web que se exhibirá en el navegador o mostrando el respectivo mensaje si se detectó algún error.

Apache

Apache es el servidor Web más utilizado del mundo, encontrándose muy por encima de sus competidores, tanto gratuitos como comerciales. Es un software de código abierto que funciona sobre cualquier plataforma.

Desde su origen ha evolucionado hasta convertirse en uno de los mejores servidores en términos de eficiencia, funcionalidad y velocidad.

Apache presenta entre otras características, mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido.

¿Por qué Apache?

- Es gratuito, distribuido bajo la licencia Apache Software License, lo cual permite modificación y adaptación de código.
- Es flexible y extensible lo que da la gran posibilidad de, mediante módulos, ampliar sus capacidades y bondades.
- Es extremadamente popular, por lo que resulta muy fácil conseguir documentación o ayuda para su uso.
- Es altamente fiable pues aproximadamente el 90% de los servidores con más alta disponibilidad funcionan con él.
- Además de su eficiencia se destaca por su gran velocidad, característica de vital importancia cuando de autenticación y autorización se trata.

1.3.3 Base de datos

Una base de datos es un conjunto de datos que pertenecen al mismo contexto, almacenados sistemáticamente para su uso posterior. (5)

Utilizar una base de datos permite globalizar y compartir información, eliminar la redundancia e inconsistencia de datos, mejorar los mecanismos de privacidad y seguridad de los mismos, y mantener la integridad en la información. (5)

¿Para qué una base de datos?

Para desarrollar un software que gestione la autenticación y autorización de usuarios, es inevitable controlar de alguna forma toda la información relacionada con los usuarios, los roles que juegan y los permisos que tienen sobre las distintas aplicaciones; por lo que es fácil identificar la necesidad de utilizar una base de datos que almacene toda esta información.

1.3.4 Gestor de bases de datos.

Los sistemas gestores de base de datos (SGBD) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Están compuestos por un lenguaje de definición de datos, un lenguaje de manipulación de datos y un tercer lenguaje de consulta.

El objetivo principal de un SGBD es proporcionar una forma de almacenar y recuperar la información de una base de datos, de manera que sea tanto práctica como eficiente.

PostgreSQL

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos de software libre, publicado bajo la licencia BSD. (6)

Como muchos otros proyectos open source, el desarrollo de PostgreSQL no es manejado por una sola compañía sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo.

La *licencia BSD* es la licencia de software otorgada principalmente para los sistemas BSD (*Berkeley Software Distribution*). Pertenece al grupo de licencias de software Libre. Esta licencia tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre.

¿Por qué PostgreSQL?

Son muchas las razones para escoger PostgreSQL como solución para la administración de datos. En las siguientes líneas se muestran aquellas que fueron determinantes en esta elección.

- Puede ser usado bajo software libre, pues se encuentra publicado bajo la licencia BSD.
- Un sistema que gestione centralizadamente la autenticación y autorización de usuarios en un centro como la UCI, requiere en primer lugar de gran velocidad, punto en el que PostgreSQL supera a la mayoría de sus rivales.
- Resulta fácil de utilizar y de administrar. Además, gracias a su activa comunidad de desarrollo, se puede encontrar una gran cantidad de ayuda en la Web.
- Se ejecuta en la inmensa mayoría de sistemas operativos y en la mayor parte de los casos, los datos se pueden transferir de un sistema a otro sin dificultad.

1.3.5 Aplicación Web.

En la ingeniería software se denomina *aplicación web* a aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. En otras palabras, es una aplicación software que se codifica en un lenguaje soportado por los navegadores web (HTML, JavaScript, Java, etc.) en la que se confía la ejecución al navegador. (7)

Las aplicaciones web son populares debido a lo práctico del navegador web como cliente ligero, así como a la facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software a miles de usuarios potenciales.

¿Para qué una aplicación Web?

Es necesaria una aplicación Web que funcione como interfaz del sistema que se propone y permita la configuración del mismo.

1.3.6 Lenguaje de programación.

Un *lenguaje de programación* es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina. (8)

Permite a uno o más programadores especificar de *manera precisa* sobre qué datos debe operar una computadora, cómo estos datos deben ser almacenados o transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar *relativamente* próximo al lenguaje humano o natural, tal como sucede con el lenguaje Léxico. Una característica relevante de los lenguajes de programación es precisamente que más de un programador puedan tener un conjunto común de instrucciones que puedan ser comprendidas entre ellos para realizar la construcción del programa de forma colaborativa. (8)

PHP

PHP Hypertext Pre-processor (inicialmente PHP Tools, o, Personal Home Page Tools), es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas Web dinámicas.

Esta tecnología fue considerada durante mucho tiempo un juguete sobreutilizado por muchos de sus críticos, pues, como otros desarrollos OpenSource, nació como un pasatiempo en un garaje, pero su facilidad de uso, la inmensa documentación que tiene a su disposición, la rapidez de su motor y su perfecta integración con PostgreSQL lo han convertido en casi un estándar de la red para el desarrollo de aplicaciones Web. (9)

Otra de las principales ventajas que ofrece PHP es ser un lenguaje libre y abierto, pues su código fuente está disponible y es gratuito. Inicialmente esta tecnología fue diseñada para entornos UNIX por lo que ofrece más prestaciones en este sistema operativo, pero es perfectamente compatible con Windows, facilidades que aportan otro punto a su favor. Además este lenguaje interpretado y embebido en el HTML, es sumamente escalable.

Estas características provocaron que la elección del lenguaje a utilizar fuera sencilla, pues sin lugar a dudas PHP, es una de las mejores opciones.

PHP 5.0

PHP5 es una nueva versión de PHP que incorpora nuevas ventajas y mantiene las viejas, pretendiendo solucionar las carencias de las anteriores versiones y demostrando su rotunda superioridad sobre las mismas.

¿Por qué PHP5?

- PHP5 incorpora un soporte sólido y real para Programación Orientada a Objetos (POO), paradigma seleccionado para utilizar en la implementación de la aplicación que formará parte de la solución.
- Mejor soporte para PostgreSQL con extensión completamente rescrita, siendo éste el gestor que será utilizado para la base de datos de la solución a proponer.
- Presenta mejoras con respecto al tratamiento de excepciones de errores, característica que puede ser explotada, elevando la calidad del sistema a proponer.
- Contiene soporte integrado para SOAP, lo que resulta muy provechoso cuando se requiere trabajar con servicios Web, como es en este caso.

1.3.7 Framework.

Un framework o *generador de aplicaciones,* se puede considerar como una aplicación genérica incompleta y configurable, a la que se le pueden añadir las últimas piezas para construir una aplicación concreta (10). Su genialidad consiste en que simplifica y acelera considerablemente el proceso de desarrollo de una aplicación; ya que automatiza algunos de los patrones utilizados para resolver las tareas más comunes, mediante el encapsulamiento de operaciones complejas en instrucciones sencillas.

Todas estas ventajas hicieron irrevocable la decisión de utilizar un framework para el desarrollo de la solución de software, pues la reutilización de códigos y diseños que los mismos proporcionan, permitirá al desarrollador dedicarse por completo a los aspectos específicos de la aplicación en cuestión.

CodeIgniter

Desarrollado por Rick Ellis para EllisLab, Inc. Primera versión lanzada 28 de febrero de 2006.

El creador de PHP Rasmus Lerdof en la conferencia frOSCon de agosto del 2008, haciendo alusión al rendimiento de los frameworks de PHP existentes en la actualidad acotó que le gustaba de entre todos CodeIgniter porque es más rápido, ligero y el que menos se parece a un framework. (11)

De la traducción del manual oficial del framework se tiene que CodeIgniter es un conjunto de herramientas para personas que construyen su aplicación web usando PHP. Su objetivo es permitirle desarrollar proyectos mucho más rápido de lo que podría si lo escribiese desde cero, proveyéndole un rico juego de librerías para tareas comúnmente necesarias, así como una interfaz simple y estructura lógica para acceder a esas librerías. CodeIgniter le permite creativamente enfocarse en su proyecto minimizando la cantidad de código necesaria para una tarea dada. (8)

CodeIgniter se encuentra bajo licencia de código abierto Apache/BSD.

Características

- Sistema basado en Modelo-Vista-Controlador, compatible con PHP5 y clases de base de datos llenas de características con soporte para varias plataformas además de soporte de Active Record para Base de Datos, formulario y validación de datos. Seguridad y Filtro XSS. Manejo de Sesión.
- Clase de Envío de Email. Soporta Archivos Adjuntos, email de texto/HTML, múltiples protocolos (sendmail, SMTP, and Mail) y más.
- Librería de Manipulación de Imagen (cortar, redimensionar, rotar, etc.). Soporta GD, ImageMagick, y NetPBM. Clase de Carga (upload) de Archivo, clase de FTP, localización, paginación, encriptación de datos, puntos de referencia, cacheo de páginas enteras, historial de errores.
- Clase de calendario, clase de agente del usuario, clase de codificación zip, clase de motor de plantillas, librería XML-RPC, clase de prueba de unidad, URL's amigables a motores de búsqueda, ruteo de URI Flexible, soporte para ganchos, extensiones de clase y plugins, larga librería de funciones "asistentes".

¿Por qué CodeIgniter?

Es un Framework realmente liviano, que dispone de un rendimiento excepcional además de una gran compatibilidad que tiene con los servidores webs. Requiere apenas configuraciones y no precisa de un uso de herramientas para líneas de comando (consola). Su uso no va establecido a seguir por estrictas reglas de codificación.

Muy útil cuando no se necesita de librerías de gran escala como el repositorio de extensiones y aplicaciones de PHP, al igual que no se precisa ser forzado a aprender un lenguaje de plantilla para su

utilización, aunque el contenga uno propiamente, evitando así la complejidad, arribando a soluciones más simples.

Es un framework muy utilizado, por lo que se dispone de gran cantidad de documentación en internet.

1.3.8 Herramienta de desarrollo.

Zend Studio for Eclipse

Zend Studio es un ambiente de desarrollo integrado o Integrated Development Environment (IDE) destinado a desarrolladores profesionales. Incluye todos los componentes de desarrollo necesarios para ciclo de vida de aplicaciones PHP. A través de un comprensivo conjunto de herramientas de edición, depurado, análisis, optimización y bases de datos, agiliza el desarrollo Web y simplifica los proyectos complejos. Es compatible con las plataformas MAC, Windows y Linux. (12)

Zend acaba de lanzar una versión del entorno de desarrollo Zend Studio para Eclipse bajo el nombre en clave “Early Access”, basado en Zend Studio y el proyecto Eclipse PHP Developers Tools (PDT). Zend Studio for Eclipse se convierte así en el IDE para PHP más potente del mercado, ofreciendo al desarrollador profesional de PHP la potencia de Zend Studio, el soporte multilenguaje de Eclipse y su enorme conjunto de extensiones (plugins) (13).

¿Por qué Zend Studio for Eclipse?

Este poderoso IDE combina las ventajas de Zen Studio con las de Eclipse/PDT, reduciendo enormemente las razones para rechazarlo. A continuación se muestran sólo las principales características que se tuvieron en cuenta:

- Posee un excelente completamiento de código, lo que proporciona al programador centrarse en la lógica del sistema, agilizando en gran medida el desarrollo del mismo. Su ayuda contextual es tan inteligente que no solo contiene las funciones definidas en el lenguaje, sino que también reporta ayudas con las funciones que vayan creando los mismos desarrolladores.
- Posee soporte para servicios Web, y además un buen generador de wsdl (Web service description language). Esta ventaja no podía ser despreciada, pues, como fue explicado anteriormente, el sistema debe contener un servicio Web.
- Tiene soporte entre otros, para PHP5 y contiene manual de PHP, lenguaje seleccionado para la implementación del software.

1.3.9 Lenguaje de modelado.

En cualquier proyecto de ingeniería como en la construcción de un gran edificio, un avión, una represa hidroeléctrica, la construcción de un procesador de textos o un software de comunicaciones para Internet, requieren de etapas de modelamiento que permitan experimentar y visualizar el sistema que se construirá.

Uniendo varios conceptos y teorías, se puede conceptualizar un lenguaje de modelado como una estandarización de notaciones y reglas, que permitan diagramar o graficar un sistema, o parte de él.

La elección de un aceptado lenguaje de modelado es de vital importancia, pues un buen modelamiento del software influye determinantemente, en lograr una adecuada comunicación entre los desarrolladores y los clientes.

UML

Lenguaje Unificado de Modelado (**UML**, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. (14)

Como todo lenguaje, UML proporciona un vocabulario y unas reglas para permitir una comunicación. Éste particularmente, está compuesto por elementos que no son más que abstracciones que constituyen los bloques básicos de construcción, los cuales pueden unirse mediante relaciones para conformar los diagramas.

¿Por qué UML?

UML facilita la representación gráfica de un sistema, teniendo como objetivo sustancial, brindar un material de apoyo que le permita al lector poder definir diagramas propios, como también entender diagramas ya existentes. Sus principales funciones son visualizar, especificar, construir y documentar cada una de las partes que comprende el desarrollo de software.

UML tiene tres características que lo hacen ideal:

- Sirve para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas software como para la arquitectura hardware donde se ejecuten (15).

- Se pueden automatizar determinados procesos y permite generar código a partir de los modelos y a la inversa (a partir del código fuente generar los modelos) (15). Esto permite que el modelo y el código estén actualizados, por lo que siempre se puede mantener la visión en el diseño, de la estructura del proyecto.
- Es completamente independiente del lenguaje de implementación, de tal forma que los diseños realizados usando UML, se pueda implementar en cualquier lenguaje que soporte las posibilidades de UML (principalmente lenguajes orientados a objetos) (15).

Producto de todas estas ventajas, UML no solo es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; sino que además, se ha convertido en el estándar ansiado para describir un "plano" de los sistemas informáticos.

1.3.10 Metodología de desarrollo

Para desarrollar un software se necesita una forma ordenada de trabajo, un proceso que integre y guíe las múltiples facetas del desarrollo y que además, ofrezca criterios para el control y la medición de los productos. A esta clase de procesos se le conoce como metodología de desarrollo.

Todo desarrollo de software es riesgoso y difícil de controlar, pero si no se utiliza una metodología de por medio, lo que obtenemos es clientes insatisfechos con el resultado y desarrolladores aún más insatisfechos (16).

RUP

El Proceso Unificado de Desarrollo Software o simplemente Proceso Unificado es más que un simple proceso, es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organización, diferentes niveles de aptitud y diferentes tamaños de proyecto (17).

El refinamiento más conocido y documentado del Proceso Unificado es el Proceso Unificado de Rational (RUP). Este se ha hecho tan popular, que actualmente, los dos nombres suelen utilizarse para referirse indistintamente a un mismo concepto.

RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al culminar cada una de ellos, éstos a la vez se dividen en fases que finalizan con un hito, en donde se pone en práctica la toma de decisiones. Se caracteriza por tres prácticas esenciales que lo hace único: ser iterativo e incremental, dirigido por casos de uso y centrado en la arquitectura. Incluye artefactos, que son los productos

tangibles que conforman el producto final, y roles, papel que desempeña una persona dentro del proceso.

Estas características han hecho que junto con el UML, RUP constituya la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

¿Por qué RUP?

RUP lejos de ser un sistema con pasos firmemente establecidos, es un conjunto de metodologías lo suficientemente flexibles, como para adaptar el proceso de desarrollo, al contexto y a las necesidades de la organización cliente.

RUP utiliza el Lenguaje Unificado de Modelado para preparar todos esquemas de un sistema software. De hecho, UML es una parte esencial del Proceso Unificado; sus desarrollos fueron paralelos.

El proceso iterativo de RUP permite que en cada iteración se analice la opinión de los clientes y la estabilidad y eficacia del producto. Esta es una gran ventaja para cualquier proyecto, pues admite un incesante refinamiento del producto, así como una continua mitigación de los riesgos involucrados. Esto evidencia su capacidad de asegurar la calidad como parte del proceso de desarrollo y no de un grupo independiente.

RUP proporciona una forma disciplinada de asignar tareas y responsabilidades, característica que pesó muchísimo en su elección, puesto que los desarrolladores del sistema, necesitaban trabajar a distancia; situación excepcional que requiere de una extrema organización.

1.3.11 Herramientas de modelado.

El hecho de usar la notación UML para el intercambio de información de diseño e ideas, requiere de un software que ofrezca todas las herramientas necesarias, para hacer eficientemente este tipo de trabajo. Esta clase de software es a lo que se denomina herramienta de modelado y son usados para capturar, guardar, rechazar e integrar automáticamente información y diseño de documentación (18), labores difíciles de lograr con un simple procesador de texto.

Visual Paradigm

Existen herramientas Case de trabajo visuales como el Analise, el Designe, el Rational Rose y el Visual Paradigm que permiten realizar el modelado del desarrollo de los proyectos. Para la realización de este trabajo se ha decidido utilizar Visual Paradigm ya que a diferencia de otras herramientas de este tipo esta es una herramienta multiplataforma que utiliza “UML”: como lenguaje de modelaje.

Visual Paradigm para UML permite proporcionar un ambiente modelado visual para satisfacer la tecnología del software de hoy y necesidades de comunicación. Es muy fácil de usar y presenta un ambiente gráfico agradable para el usuario.

¿Por qué Visual Paradigm?

Permite generar código en lenguajes tan utilizados como: Java, C++, PHP, C #, Delphi, Perl

Permiten incorporar los dibujos de Visio® en cualquier diagrama de UML. Muchas herramientas CASE realizan solo diagramas de UML normales, Visual Paradigm posibilita modelar hardware dominio-específico, el software, conectando una red de computadoras los componentes usando sus iconos, más allá de las anotaciones de UML normales.

Es multiplataforma, Visual Paradigm está disponible en muchas plataformas incluso Windows, Linux, el Mac OS X, etc.

1.3.12 Conclusiones.

La metodología aplicada en la investigación permitió profundizar en aspectos necesarios para la realización de este trabajo. Todos los métodos aplicados fueron muy útiles, aportando conocimientos de significativa importancia para la misma.

El estudio realizado sobre las soluciones existentes en Cuba y el mundo enriqueció considerablemente la investigación, pues aportó un cúmulo de ideas aplicables al software que se propone como solución al problema planteado.

El estudio de las tecnologías, paradigmas y tendencias actuales, permitió escoger las metodologías, lenguajes y herramientas más adecuados para que la solución de software, cumpla con los requisitos establecidos y la calidad requerida.

CAPÍTULO 2: DESCRIPCIÓN DE LA ARQUITECTURA

2.1 Introducción

En este capítulo se ve lo referente a la especificación de requisitos funcionales o no funcionales con los que contara la solución de software propuesta. Además se expone la arquitectura y patrones que se emplearon y una pequeña descripción. Se propone una vista de despliegue a la solución de software y los estándares de código que se emplearon para su implementación.

2.2 Especificación de los requisitos.

Todo sistema informático, aún después de su puesta en marcha continúa desarrollándose. Por esto, continuamente salen al mercado nuevas versiones de los distintos productos, donde cada una supera a la anterior.

SegRED en la Universidad está en desarrollo de su primera versión, que a petición del cliente, se encargara de centralizar un flujo único para el proceso de autenticación – autorización de los usuarios en las aplicaciones de la UCI, profundizando en la seguridad de los protocolos de comunicación, de la base de datos, de la aplicación Web y del servicio Web.

A continuación se exponen los requisitos capturados para la primera versión de SegRED.

2.2.1 Requisitos funcionales.

Primero se mostrarán los requisitos funcionales capturados en el análisis y diseño realizado inicialmente a la propuesta de solución y seguidamente se mostrará un refinamiento de los mismos que son con los que se desarrolló la solución de software a implantar.

Requisitos Iniciales:

R1 Administrar Sistema de Gestión de Accesos.

1.1Autenticar usuario en el sistema.

1.2 Gestionar datos del usuario en el sistema

1.2.1 Registrar datos de usuario.

1.2.1.1 Otorgar rol al usuario.

1.2.2 Modificar datos de usuario.

1.2.3 Eliminar datos de usuario.

R2 Gestionar información.

2.1 Gestionar datos de solución de software implantada.

2.1.1 Registrar datos de solución de software implantada.

2.1.1.1 Cargar funcionalidades de solución de software implantada.

2.1.2 Modificar datos de solución de software implantada.

2.1.3 Eliminar datos de solución de software implantada.

2.1.4 Mostrar datos de una solución de software implantada.

2.1.5 Mostrar reporte de los datos de todas las soluciones de software implantadas.

2.1.6 Mostrar reporte de funcionalidades por solución de software implantada.

2.2 Gestionar roles.

2.2.1 Crear rol.

2.2.1.1 Otorgar funcionalidades de solución de software implantada a rol.

2.2.2 Eliminar rol.

2.2.3 Modificar datos de rol.

2.2.4 Mostrar datos de un rol determinado.

2.2.5 Mostrar reporte de roles por solución de software implantada.

2.3 Gestionar datos de usuario.

2.3.1 Registrar datos de usuario.

2.3.1.1 Cargar usuarios del dominio.

2.3.2 Modificar datos de usuario.

2.3.3 Adicionar usuario a solución de software implantada.

2.3.3.1 Otorgar rol a usuario.

2.3.4 Mostrar datos de un usuario determinado.

2.3.5 Mostrar reporte de usuarios por solución de software implantada.

2.4 Gestionar datos de grupo de usuarios

2.4.1 Registrar datos de grupo de usuarios.

2.4.1.1 Cargar grupo de estructura.

2.4.2 Modificar datos de grupo de usuarios.

2.4.3 Adicionar grupo de usuarios a solución de software implantada.

2.4.3.1 Otorgar rol a grupo de usuarios.

2.4.4 Mostrar datos de un grupo de usuarios determinado.

2.4.5 Mostrar reporte de grupos de usuarios por solución de software implantada.

R3 Gestionar servicios

3.1 Autenticar usuario.

3.2 Autorizar acceso de usuario.

3.3 Brindar Información

3.3.1 Brindar información de solución de software implantada.

3.3.2 Brindar información de funcionalidades.

3.3.3 Brindar información de roles.

3.3.4 Brindar información de usuarios.

3.3.5 Brindar información de grupo de usuarios.

Requisitos refinados y capturados para la implementación:

R1 Administrar SegRED en la UCI

1.3Autenticar usuario en el sistema.

R2 Gestionar información.

2.5 Gestionar datos de solución de software implantada.

2.5.1 Registrar datos de solución de software implantada.

2.5.1.1 Registrar funcionalidades de solución de software implantada.

2.5.2 Modificar datos de solución de software implantada.

2.5.3 Eliminar datos de solución de software implantada.

2.5.4 Mostrar datos de una solución de software implantada.

2.5.5 Mostrar reporte de los datos de todas las soluciones de software implantadas.

2.5.6 Mostrar reporte de funcionalidades por solución de software implantada.

2.6 Gestionar roles.

2.6.1 Crear rol.

2.6.1.1 Otorgar funcionalidades de solución de software implantada a rol.

2.6.2 Eliminar rol.

2.6.3 Modificar datos de rol.

2.6.4 Mostrar datos de un rol determinado.

2.6.5 Mostrar reporte de roles por solución de software implantada.

2.7 Gestionar datos de usuario.

2.7.1 Registrar datos de usuario.

2.7.1.1 Cargar usuarios del dominio.

2.7.2 Modificar datos de usuario.

2.7.3 Adicionar usuario a solución de software implantada.

2.7.3.1 Asociar usuario a un grupo.

2.7.4 Mostrar datos de un usuario determinado.

2.7.5 Mostrar reporte de usuarios por solución de software implantada.

2.8 Gestionar datos de grupo de usuarios

2.8.1 Registrar datos de grupo de usuarios.

2.8.2 Modificar datos de grupo de usuarios.

2.8.3 Adicionar grupo de usuarios a solución de software implantada.

2.8.3.1 Otorgar rol a grupo de usuarios.

2.8.4 Mostrar datos de un grupo de usuarios determinado.

2.8.5 Mostrar reporte de grupos de usuarios por solución de software implantada.

R3 Gestionar servicios

6.1 Autenticar usuario.

6.2 Autorizar acceso de usuario.

6.3 Brindar Información

6.3.1 Brindar información de solución de software implantada.

6.3.2 Brindar información de funcionalidades.

6.3.3 Brindar información de roles.

6.3.4 Brindar información de usuarios.

6.3.5 Brindar información de grupo de usuarios.

2.2.2 Requisitos no funcionales.

Estos requisitos se mantuvieron en una gran parte de la misma forma en que fueron propuestos en el análisis y diseño, solo se cambio el gestor de base de datos en el cual se propuso MySQL y la solución de software se implementó sobre PostgreSQL.

✓ Apariencia o interfaz externa.

- Diseño sencillo y con gran navegabilidad, que permita una fácil comprensión y utilización.
- Estará diseñado para resolución de 1024x768.

✓ Usabilidad.

- Deberá visualizarse en los navegadores más utilizados (Internet Explorer, Mozilla Firefox, Opera, Maxthon, Chrome).

- Estará bien documentado y proporcionará materiales de ayuda para hacer un mejor uso de todos los servicios que éste ofrece.
- El sistema debe ser independiente de la base de datos.

✓ Rendimiento.

- El sistema será imprescindible para el funcionamiento de muchas soluciones de software implantadas, por lo que debe ser eficiente, preciso, con tiempos de respuestas cortos y gran velocidad de procesamiento de la información.

✓ Soporte.

- El sistema debe ser de fácil instalación, adaptable a numerosas plataformas y de fácil mantenimiento.

✓ Portabilidad.

- Al sistema se debe acceder desde cualquier plataforma.

✓ Seguridad

- El sistema debe garantizar la confidencialidad e integridad de los datos.
- Verificar sobre acciones irreversibles (eliminaciones).
- La información manejada por el sistema deberá estar protegida de acceso no autorizado.
- Garantizar que la información sea vista únicamente por quien tiene derecho a verla.
- Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.
- Garantía de un tratamiento adecuado de las excepciones y validación de las entradas del usuario.

✓ Restricciones en el diseño y la implementación

- Implementación con PHP y sistema gestor de base de datos PostgreSQL.
- Utilizar los estándares de diseño y codificación establecidos.
- Usar el Framework CodeIgniter.
- Para el análisis y el diseño del sistema debe ser utilizada la metodología RUP, usando el lenguaje de modelación UML y como herramienta para llevarlo a cabo el Visual Paradigm.
- Para la implementación de la aplicación Web deberá utilizarse el patrón Modelo-Vista-Controlador.
- El sistema debe desarrollarse bajo una Arquitectura Orientada a Servicios (SOA).

✓ Confiabilidad.

- El sistema podrá ser usado en todo momento por lo que deberá tener un 100% de disponibilidad.
- El tiempo medio de reparación debe ser menor de 1 día, debido a que del funcionamiento correcto del sistema dependen otras soluciones de software implantadas
- Todas las salidas del sistema tienen que tener el 100% de veracidad y precisión.
- Debe ser capaz de recuperarse rápidamente a las fallas del sistema.
- Deben montarse sistemas de respaldo eléctrico en los locales de los servidores para mantener la vitalidad de los servicios.

✓ Políticos-culturales.

- Contará con logotipos e imágenes que se encuentren en correspondencia con el carácter científico y profesional de la UCI.
- Toda modificación al funcionamiento establecido en los requerimientos será realizada por el Departamento de Informatización de la UCI.

✓ Legales.

- El sistema se basa en un estándar que se rige por normas internacionales y cumple con las normas y leyes establecidas en Cuba y la UCI.

✓ Interfaz interna

- Debe definirse una Portada personalizada para cada usuario del Sistema, dándole la posibilidad de interactuar con un conjunto de funcionalidades en dependencia de su rol.
- Interfaz de usuario: Para el diseño de la interfaz de usuario se utilizará un diseño personalizado.
- Interfaz de hardware: Debe estar soportado por una red.
- Interfaz de software: La aplicación se realizará en ambiente Web.
- Interfaz de comunicación: La aplicación se comunica con los usuarios del sistema a través de la red de área local.

✓ Software.

- Navegador compatible o superior con Internet Explorer 5, o Firefox 1.5

✓ Hardware.

- Un servidor.
- Una red de computadoras.
- Conexión de red de 10 Mbps a 1 Gbps.

2.3 Arquitectura y patrones.

Una Arquitectura de Software, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software (19). Establece los fundamentos para que los trabajadores del software usen una línea común y define, de manera abstracta, los componentes que deben ser utilizados, sus interfaces y la comunicación entre ellos. Toda arquitectura de software debe ser implementable en una arquitectura física, que consiste simplemente en determinar qué computadora tendrá asignada cada tarea. (20)

Por su parte, un patrón es un modelo que se puede seguir para realizar algo. Los patrones surgen de la experiencia de seres humanos de tratar de lograr ciertos objetivos. Los patrones capturan la experiencia existente y probada para promover buenas prácticas. (21)

SOA

SOA es el acrónimo de *Arquitectura Orientada a Servicios* o en inglés, *Service Oriented Architecture*. Es un nuevo concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requerimientos de software del usuario. (22)

En un ambiente SOA, los nodos de la red hacen disponibles sus recursos a otros participantes en la red, como servicios independientes a los que tienen acceso de un modo estandarizado. (22)

Una arquitectura SOA está formada por tres partes: un proveedor, un intermediario y un cliente que no presentan ningún acoplamiento entre ellos.

El proveedor ofrece un servicio determinado que el cliente no tiene porque conocer directamente. El cliente aprende cómo utilizar el servicio a partir de la información que le ofrece el intermediario, que normalmente simplifica el uso de dicho servicio. El cliente sólo sabe cómo utilizar el servicio, es decir, cómo enviar y recibir datos, pero no conoce ningún detalle de su implementación interna.

¿Por qué SOA?

Las ventajas de SOA son múltiples, pero todas están relacionadas con los conceptos de simplicidad, flexibilidad, interoperabilidad, capacidad de mantenimiento, reutilización a gran escala y bajo acoplamiento.

La Dirección de Informatización de la Universidad de la Ciencias Informáticas promueve el desarrollo de aplicaciones bajo una arquitectura orientada a servicios. Esto se debe a que la UCI trabaja pensando en el futuro y si bien que no es posible garantizar que la construcción de un producto bajo SOA sea más barata o más rápida; lograr una adecuada colaboración entre los distintos sistemas de la red, reducirá considerablemente el costo, así como la complejidad y el tiempo de desarrollo y mantenimiento de las futuras aplicaciones.

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 distintos: (23)

- El modelo: representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La vista: transforma el modelo en una página Web que permite al usuario interactuar con ella.
- El controlador: se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista. (11)

¿Por qué Modelo-Vista-Controlador?

El uso de este patrón es recomendado para el desarrollo de aplicaciones Web, puesto que la división que propone, facilita cambios en una de las partes sin necesidad de modificar el resto y proporciona un mantenimiento más sencillo de las mismas.

2.4 Vista de Despliegue.

Un diagrama de despliegue se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. En el siguiente diagrama se modela la distribución de las diferentes partes que conforman el SegRED.

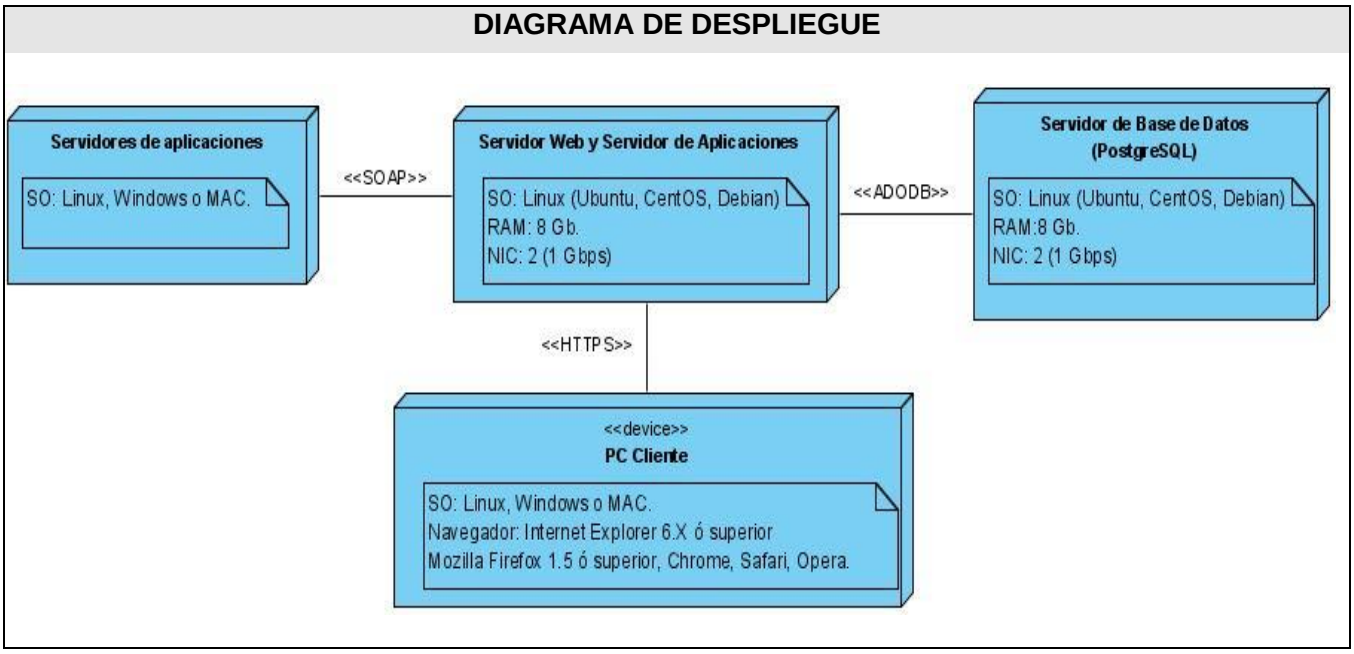


Fig. 1 Diagrama de Despliegue de Componentes de SegRED.

2.5 Estrategias de codificación. Estándares y estilos a utilizar.

Los estándares de código son importantes y de obligatorio cumplimiento para los proyectos en nuestra Universidad, ya que los equipos de desarrollo se desintegran con el tiempo y los sistemas deben ser mantenidos por equipos de soporte, que necesitan entender el código para posibles cambios. Se utilizarán los estándares definidos o internacionalmente aceptados según el lenguaje de programación utilizado.

Para el desarrollo de esta solución de software se tomó como base de estándar de codificación la documentación de PEAR y los estándares que proponen. A continuación se ofrecen algunos ejemplos de los usados: (24)

Sangrías y Longitud de línea

Usar una sangría de 4 espacios sin usar la tecla “tab”. Esto ayuda a evitar problemas con diffs, patches, historia del CVS y anotaciones. (24)

Estructuras de Control

Estas incluyen: if, for, while, switch, etc. Aquí está el ejemplo de la postura de “if”, porque es el más complicado de todas ellas. (24)

```php
<?php
if ((condition1) || (condition2)) {
 action1;
} elseif ((condition3) && (condition4)) {
 action2;
} else {
 defaultaction;
}
?> (24)
```

Los estatutos de control deben de tener un espacio entre la palabra clave de control y el inicio de paréntesis "(", para distinguirlos de las llamadas de función. (24)

Deben de utilizarse las llaves en cualquier caso "{ }", incluso en situaciones donde son técnicamente opcionales. Su uso incrementa su capacidad de lectura y reduce la probabilidad de errores lógicos que son introducidos cuando líneas nuevas se agregan. (24)

Para estructuras switch:

```php
<?php
switch (condition) {
 case 1:
 action1;
 break;
 case 2:
 action2;
 break;
 default:
 defaultaction;
 break;
}
?> (24)
```

Llamadas de Función

Las funciones deberá de ser llamadas sin espacios entre el nombre de la función, el abrir paréntesis y el primer parámetro; espacios entre comas y cada parámetro, y sin espacios entre el ultimo parámetro, el cierra paréntesis y punto y coma. (24)

Ejemplo:

```php
<?php
$var = foo($bar, $baz, $quux);
?>
```

Como se muestra aquí arriba, deberá de haber un espacio en ambos lados del signo de igual que se usa para asignar el valor de regreso de una función a una variable. En el caso de un bloque de asignaturas relacionadas, mas espacio puede ser agregado para así, promover su facilidad de lectura. (24)

```
<?php
$short = foo($bar);
$long_variable = foo($baz);
?>
```

Definiciones de Función

En las declaraciones de función, el bracket de apertura empieza al final de la linea de definición despues de un espacio. (24)

```
<?php
function foo_function($arg1, $arg2 = 'x') {
 if (condition) {
 statement;
 }
 return $val;
}
?>
```

Los argumentos con valores por default van al final de la lista de argumentos. Siempre tratando de regresar un valor significativo de una función, si una es la apropiada. Un ejemplo más largo: (24)

```
<?php
function connect(&$dsn, $persistent = false) {
 if (is_array($dsn)) {
 $dsninfo = &$dsn;
 } else {
 $dsninfo = DB::parseDSN($dsn);
 }
 if (!$dsninfo || !$dsninfo['phptype']) {
 return $this->raiseError();
 }
 return true;
}
?> (24)
```

Inclusión de Código

En cualquier parte que estés incondicionalmente incluyendo una archivo de una clase, usa required_once. En cualquier parte donde estés condicionalmente incluyendo una archivo de clase (por ejemplo, métodos de fabrica) usa include_once. Cualquiera de esas se asegurará que la class file sea incluida únicamente una vez. Ellas comparten la misma lista de archivo (file list) así que no hay necesidad de preocuparse por mezclarlas, un archivo incluido con required_once no será nuevamente incluido por include_once. (24)

Convención de Nombres de Variables, Funciones, Clases

Variables:

Estas deberán nombrarse con un prefijo de tres letras el cual define el tipo de dato de la misma, seguido de un guión bajo y el nombre descriptivo de la variable. Ejemplo: (24)

```
$txt_descripcion: Campo tipo text
$chr_nombres: Campo tipo char o varchar
$int_cantidad: Campo tipo integer
$flt_total: Campo tipo float
$dbl_precio : Campo tipo double
$dtm_fecha: Campo tipo fecha
$bol_activo: Campo tipo boolean
```

Funciones:

Estas deberán nombrarse con palabras minúsculas seguido de un guión bajo, para separar palabras en el nombre de la misma. Ejemplo: (24)

```
<?php
function conectar_db($arg1, $arg2 = 'x') {
 if (condition) {
 statement;
 }
 return $val;
}
?>
```

Clases:

Estas deberán nombrarse con la primera palabra iniciando con minúscula y separando las demás palabras con letra mayúscula. Ejemplo: (24)

```php
<?php
class vistaRolSistema {
 function foo($arg1) {
 statement;
 }
}
?>
```

2.6 Conclusiones.

En este capítulo se expone las características a seguir por el sistema en cuanto a funcionalidades y estándares de código a utilizar además de la vista de despliegue.

CAPÍTULO 3: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA.

3.1 Introducción.

Para la implementación de un sistema, es importante la descripción visual de las partes físicas que compondrán al software. El presente capítulo contiene diagramas que muestran la distribución de los diferentes componentes de SegRED.

3.2 Valoración crítica del diseño propuesto por el analista.

A continuación se muestran los diagramas de clases del diseño propuestos por el analista y la versión final después de realizarle los cambios acorde a los requerimientos de la aplicación durante el flujo de trabajo de implementación. Los cambios obedecen fundamentalmente a la utilización del framework para PHP CodeIgniter, que utiliza el patrón arquitectónico MVC, que repercute en la estructuración de las clases para adecuarse al patrón aplicado.

Pueden citarse entre los principales cambios a la propuesta del analista, una modificación y estandarización de la Base de Datos, agregándole una nueva tabla para garantizar la seguridad del sistema.

3.3 Diseño.

Durante el modelo de diseño se adquiere una comprensión en profundidad de los aspectos relacionados con los requisitos no funcionales del sistema. El propósito del diseño es modelar el sistema y encontrar su forma para que soporte todos los requisitos. Es un modelo físico que crea una entrada apropiada y un punto de partida para la implementación.

3.3.1 Diagrama de Clases del Diseño.

En un diagrama de clases se representan las relaciones entre clases que mantienen la información manipulada por el sistema, todo el código que irán creando las páginas, así como el contenido dinámico de estas una vez que estén en el navegador del cliente.

En el caso de las aplicaciones Web, modelar las páginas, sus enlaces y todo el contenido dinámico tanto del lado del servidor como del cliente, es muy importante, pues éstos son los artefactos que necesitan conocer los desarrolladores para implementar un producto con calidad

A continuación se muestran los diagramas de clases del diseño, correspondientes al SegRED:

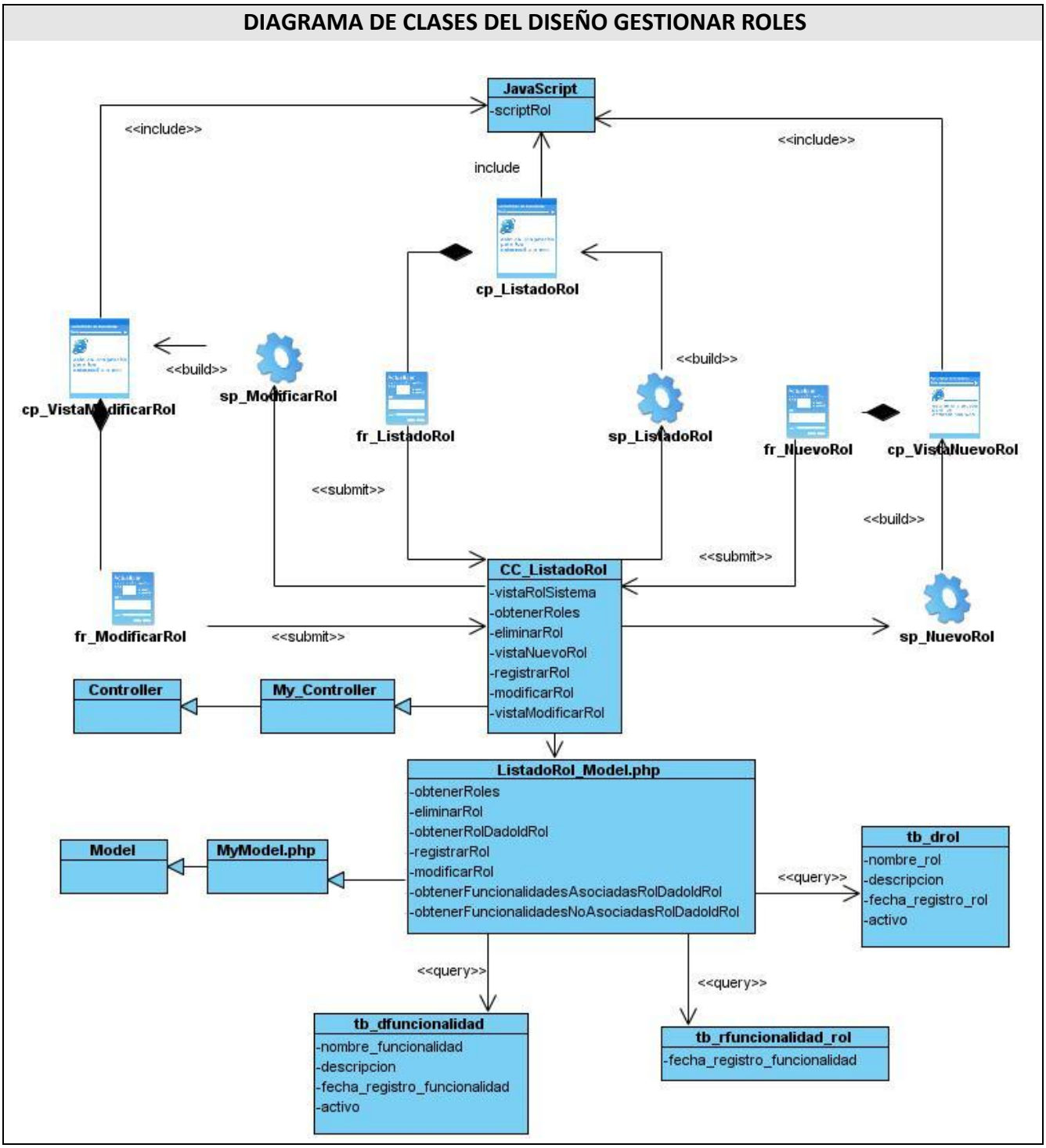


Fig. 2 Diagrama de Clases del Diseño Gestionar Roles de la aplicación SegRED.

3.4 Descripción de las nuevas clases u operaciones necesarias.

Nombre: CC_ListadoRol	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	vistaRolSistema()
Descripción:	Es utilizado para que el Administrador de la solución de software vea todos los roles que están asociados a un sistema.
Nombre:	obtenerRoles()
Descripción:	El administrador del sistema obtiene todos los roles que se le pueden añadir a una aplicación.
Nombre:	eliminarRol()
Descripción:	Elimina un rol del sistema.
Nombre:	vistaNuevoRol()
Descripción:	Permite ver los nuevos roles que se han creado.
Nombre:	registrarRol()
Descripción:	Permite al administrador del sistema registrar un nuevo Rol dentro del sistema.
Nombre:	modificarRol()
Descripción:	Permite dando un Rol dado, modificar las funcionalidades de este.
Nombre:	vistaModificarRol()
Descripción:	Permite ver todos los roles que existen para su modificación.

Descripción de las entidades.

Nombre de la Clase: tb_drol	
Tipo de Clase: Entidad	
Atributo	**Tipo**
Nombre_rol	String
Descripción	string
Fecha_registro_rol	DATE
Activo	INT

Nombre de la Clase: tb_dfuncionalidad	
Tipo de Clase: Entidad	
Atributo	**Tipo**
Nombre_funcionalidad	String
Descripción	string
Fecha_registro_funcionalidad	DATE
Activo	INT

Nombre de la Clase: tb_rfuncionalidad_rol	
Tipo de Clase: Entidad	
Atributo	**Tipo**

Fecha_registro_funcionalidad	DATE

Diseño de la base de datos.

Las bases de datos necesitan una definición de su estructura que le permitan almacenar datos, reconocer el contenido y recuperar la información. La estructura tiene que ser desarrollada para la necesidad de las aplicaciones que la usarán.

La puesta en práctica de la base de datos es el paso final en el desarrollo de aplicaciones de soporte del negocio. Se conforman con los requisitos del proceso del negocio, que es la primera abstracción de la vista de la base de datos.

En este epígrafe se muestra el diseño de la base de datos del sistema propuesto a través del modelo lógico de datos y el modelo físico de datos.

3.4.1 Modelo físico de datos (modelo de datos).

El modelo físico de los datos describe la representación lógica y física de datos persistentes en el sistema, dicho modelo se muestra a continuación.

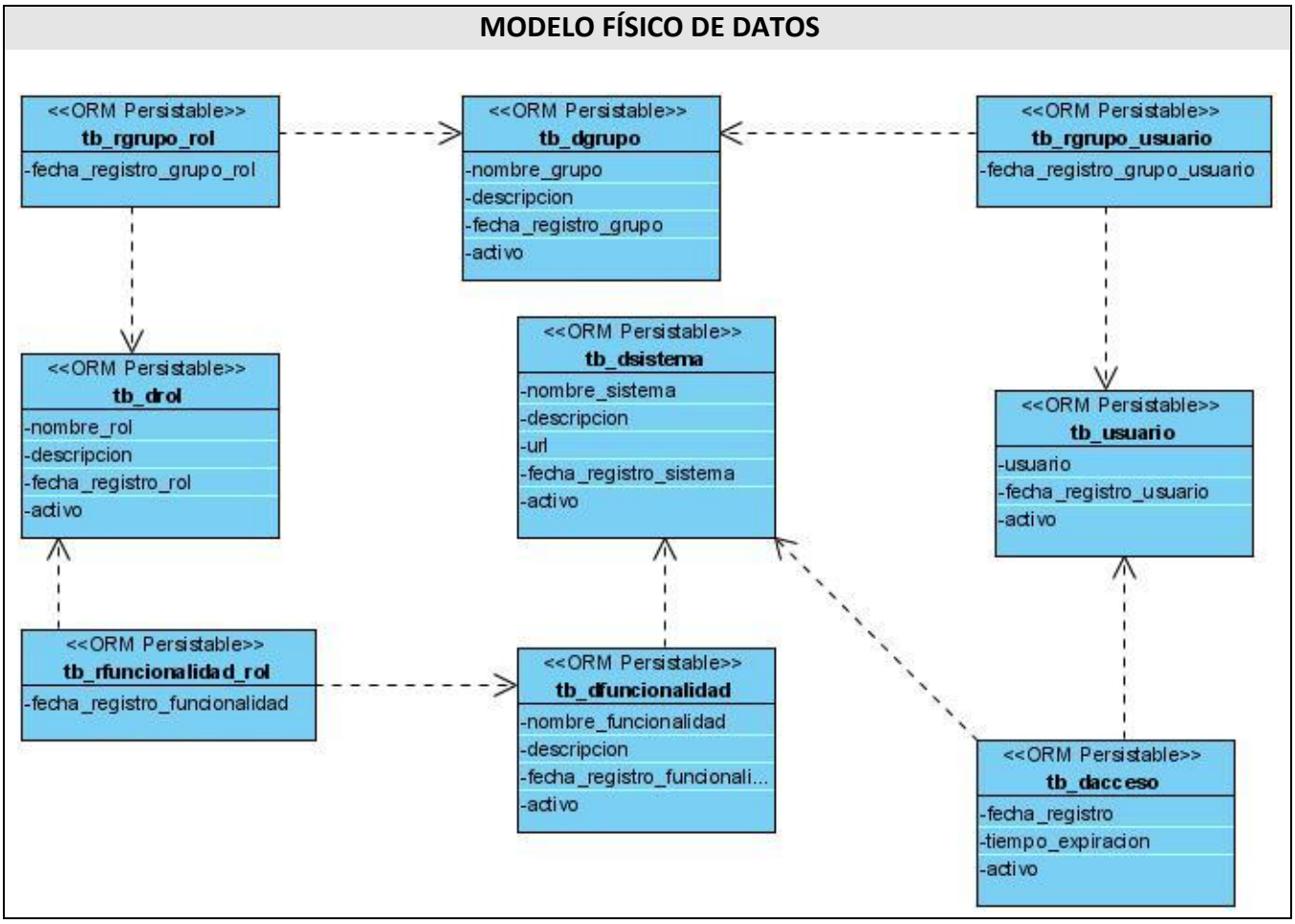


Fig. 3: Modelo Físico de Datos.

Descripción de las Tablas

Nombre de la Clase: tb_dsistema		
Descripción: en esta tabla se guarda todo lo referente a la información de los sistemas que se encuentran dentro de la solución de software.		
Atributo	**Tipo**	**Descripción**
nombre_sistema	String	Nombre por el que será reconocido el sistema dentro de SegRED.
Descripción	String	Una breve descripción del sistema.

url	string	Responde a la URL de acceso que tenga el sitio web.
Fecha_registro	DATE	Es la fecha de registro con que se incluyen los sistemas dentro de SegRED.
Activo	INT	Define si un sistema se encuentra activo en tiempo real.

Nombre de la Clase: tb_dfuncionalidad

Descripción: en esta tabla se guarda todo lo referente a la información de las funcionalidades que se encuentran dentro de la solución de software.

Atributo	Tipo	Descripción
nombre_funcionalidad	String	Nombre que se le darán a las funcionalidades.
Descripción	String	Una breve descripción de la funcionalidad.
Fecha_registro_funcionalidad	DATE	Es la fecha de registro en la que se crea una nueva funcionalidad.
Activo	INT	Define si una funcionalidad se encuentra activa en tiempo real.

Nombre de la Clase: tb_dacceso

Descripción: en esta tabla se guarda todo lo referente a los accesos que se realizan a la solución de software.

Atributo	Tipo	Descripción

Fecha_registro	DATE	Fecha en que se crea una nueva entrada en algún sistema hecha por un usuario.
Tiempo_expiracion	Timestamp	Tiempo de expiración de la sesión o vida del token.
Activo	INT	Define si la sesión o token se encuentra activo.

Nombre de la Clase: tb_rfuncionalidad_rol

Descripción: en esta tabla se guarda todo lo referente a la fecha en que se registra una nueva funcionalidad.

Atributo	Tipo	Descripción
Fecha_registro_funcionalidad	String	Guarda la fecha donde se define una funcionalidad.

Nombre de la Clase: tb_drol

Descripción: en esta tabla se guarda todo lo referente a los roles a los que tendrán acceso los usuarios en la solución de software.

Atributo	Tipo	Descripción
Nombre_rol	String	Nombre que se le da al rol.
Descripción	String	Pequeña descripción que se le da al rol para ayuda del usuario o administrador.
Fecha_registro_rol	DATE	Fecha en la que se registra un nuevo rol.
Activo	INT	Verifica si un rol se encuentra activo o no

		para ser otorgado.

Nombre de la Clase: tb_rgrupo_rol		
Descripción: en esta tabla se guarda todo lo referente a los roles que son asignados a grupos de usuarios.		
Atributo	**Tipo**	**Descripción**
Fecha_registro_grupo_rol	DATE	Guarda la fecha en la que se registra un grupo y se le asigna un rol.

Nombre de la Clase: tb_dgrupo		
Descripción: en esta tabla se guarda todo lo referente a los grupos de usuarios que se registran en la solución de software.		
Atributo	**Tipo**	**Descripción**
Nombre_grupo	String	Nombre de los grupos.
Descripción	String	Descripción de los grupos para saber a qué y donde pertenecen.
Fecha_registro_grupo	DATE	Fecha en que se registra un grupo nuevo.
Activo	INT	Establece si un grupo está activo o no.

Nombre de la Clase: tb_rgrupo_usuario
Descripción: en esta tabla se guarda todo lo referente a las relaciones de grupos de usuarios definidos para la solución de software.

Atributo	Tipo	Descripción
Fecha_registro_grupo_usuario	DATE	Fecha en que se registra un grupo de usuarios.

Nombre de la Clase: tb_usuario		
Descripción: en esta tabla se guarda todo lo referente a los usuarios en la solución de software.		
Atributo	**Tipo**	**Descripción**
Usuario	String	Nombre de usuario.
Fecha_registro_usuario	DATE	Guarda la fecha en que se registra un usuario en el sistema.
Activo	INT	Define si un usuario se encuentra activo en el sistema.

Vista de Implementación (Diagrama de Componentes)

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes de software, sean éstos de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo.

El sistema cuenta con tres subsistemas de implementación: Vistas, Controladoras y Modelo, estos están estructurados según el patrón arquitectónico MVC. En el subsistema Vistas se encuentran encapsulados los componentes que permiten la interacción directa con los usuarios del sistema. El subsistema Controladoras contiene todas las clases que manipulan los eventos del usuario y realizan peticiones al modelo para mostrarlas en las vistas. El subsistema Modelo, agrupa las clases que interactúan con la base de datos y velan por el cumplimento de las reglas del negocio.

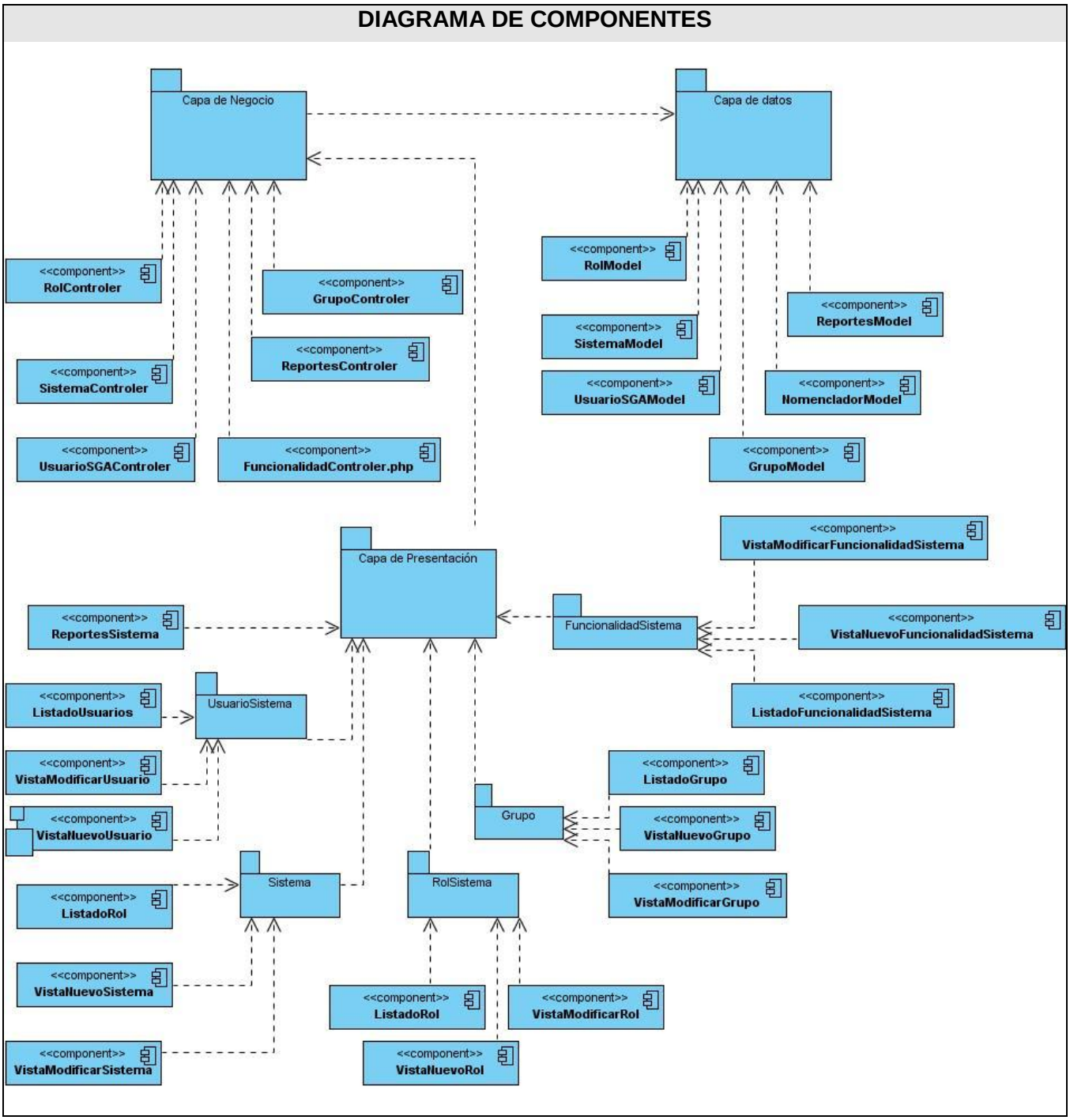


Fig. 4 Diagrama de Componentes.

3.5 Conclusiones.

En este capítulo se llevó a cabo la descripción de las clases y demás elementos necesarios para la implementación. Se obtuvo el diagrama de clases del sistema separado por paquetes atendiendo a su funcionalidad. Se definieron, a partir del mismo, cuáles son las clases persistentes y se construyó el modelo de datos.

CONCLUSIONES GENERALES.

- ✓ Se realizó un estudio de las propuesta de solución de software ya existente el SGA y el SGS obteniendo sus potencialidades.
- ✓ Se efectuó una investigación de los sistemas existentes a nivel nacional e internacional.
- ✓ El sistema propuesto permite la gestión centralizada, de los permisos de usuarios y accesos a las aplicaciones.
- ✓ La aplicación Web desarrollada funciona como interfaz del sistema y permite la configuración del mismo.
- ✓ El servicio Web implementado ofrece la funcionalidad del sistema a las distintas aplicaciones.
- ✓ La utilización de este sistema posibilita una mayor seguridad a las aplicaciones en uso.

RECOMENDACIONES.

- ✓ Poner a prueba el sistema durante un período de tiempo significativo., para comprobar que sus funcionalidades están en correspondencia con lo que se necesita y que todas las aplicaciones de la UCI puedan integrarse a él satisfactoriamente.
- ✓ Profundizar en el tema de la seguridad informática, para realizar una segunda versión del software que reduzca al máximo las vulnerabilidades del sistema.
- ✓ Continuar el desarrollo de este sistema, adicionándole nuevas funcionalidades y servicios, para adecuarlo más a las demandas de la creciente y dinámica intranet de la Universidad.
- ✓ Extender aún más el sistema de manera que pueda ser utilizado no sólo en la UCI, sino en cualquier empresa que requiera de este tipo de servicios.

BIBLIOGRAFÍA.

1. Booch, G., Rumbaugh, J., Jacobson, El Lenguaje Unificado de Modelado. Manual de usuario. Addison-Wesley ed. B.J. Rumbaugh. 1999.
2. Booch, G., Rumbaugh, J., Jacobson, El Lenguaje Unificado de Modelado. Manual de Referencia. Addison -Wesley, ed. B.J. Rumbaugh. 2000.
3. Clases de Ingienería del Software I, curso 2006-2007, UCI.
4. La Oficina del W3C en España. *Guía Breve de Servicios Web.* [Online] enero 2008.
5. Larman, Craig. UML y Patrones. Introducción al análisis y diseño orientado a objetos. Primera Edición por Prentice Hall, Hispanoamericana S.A. 1999.
6. Larman, Craig. UML y Patrones. Introducción al análisis y diseño orientado a objetos y al proceso unificado. Segunda Edición por Prentice Hall.
7. Matos, Rosa María. Introducción al trabajo con Base de Datos. Asignatura de Sistemas de Gestión de Base de Datos.
8. Ortega Ruiz, Idelvis, Bello Batista, Lester O. Sistema de Autenticación de Aplicaciones en la Intranet, 2006, UCI: Ciudad de la Habana.
9. Pablo Martínez, Palo Ruíz Díaz, Sebastián Waisbrot. CodeIgniter Spanish UserGuide. 2008. http://www.scribd.com/doc/3965892/CodeIgniter-Spanish-UserGuide.
10. University, Y. Central Authentication Service (CAS). 2006 [cited; Available from: http://www.yale.edu/tp/auth/

REFERENCIAS BIBLIOGRÁFICAS.

1. **JA-SIG.** jasig. [En línea] 2005-2006. http://www.ja-sig.org.

2. Cams by cafesoft. [En línea] 1996-2009. www.cafesoft.com.

3. La Oficina del W3C en España. *Guía Breve de Servicios Web.* [En línea] enero de 2008.

4. MONTEJAVA Soluciones empresariales basadas en web. [En línea] 2006.

5. @onocimientosWeb. La divisa del nuevo milenio. [En línea]

6. **Valdés, Damián Pérez.** Maestros del Web. *Qué son las bases de datos?* [En línea] octubre de 2007.

7. Encarta. *Aplicación (informática).* [En línea] Microsoft Corporation. http://mx.encarta.msn.com.

8. Lenguajes de programación. [En línea] 2006. http://www.lenguajes-de-programacion.com.

9. **colectivo de mononeurona.org.** Despabilando la mononeurona. *¿PHP, Python, ASP, Perl o JSP?* [En línea] MonoNeurona Commons, 2008-2009. http://www.mononeurona.org.

10. **Gutiérrez., Javier J.** Lenguajes y sistemas informáticos. *¿Qué es un framework web?* [En línea] 2008. http://www.lsi.us.es.

11. **Pablo Martínez, Palo Ruíz Díaz, Sebastián Waisbrot.** CodeIgniter Spanish UserGuide. *www.conocimientovirtual.co.cu.* [En línea] 2008. http://www.scribd.com/doc/3965892/CodeIgniter-Spanish-UserGuide.

12. **Valdés, Damián Pérez.** Noticias de Informática. *Editores que facilitan tu trabajo.* [En línea] 2007. http://www.radiocaribe.co.cu.

13. DiarioLinux. *Beta de Zend Studio para Eclipse.* [En línea] Octubre de 20007. http://diariolinux.com.

14. **M. Fowler, K. Scott.** *Scott. UML gota a gota.* s.l. : Prentice Hall, 1997.

15. **Orallo, Enrique Hernández.** El Lenguaje Unificado de Modelado (UML). [En línea] http://www.disca.upv.es/enheror/pdf/ActaUML.PDF.

16. **Sanchez, María A. Mendoza.** Informatizate. *Metodologías De Desarrollo De Software.* [En línea] Junio de 2007. http://www.informatizate.net.

17. **B. Jacobson, Rumbaugh y otros,.** *El Proceso Unificado de desarrollo de Software.* s.l. : Addison-Wesley, 2000.

18. Software y programas. *Uso de una Herramienta de Modelado .* [En línea] Mayo de 2007. http://softwareyprogramas.blogspot.com.

19. DICCIONARIO INFORMÁTICO. [En línea] ALEGSA, 2008. http://www.alegsa.com.ar/Dic.

20. **Gastón Cortois.** *Modelo de Sistema para la Gestión de Pacientes en Hospitales Públicos bajo.* [En línea]

21. *Conferencia 7 de Ingeniería del Software Patrones de Diseño.* **UCI, Departamento Central de Ingeniería de Software.** Ciudad de La Habana. : s.n., 2005.

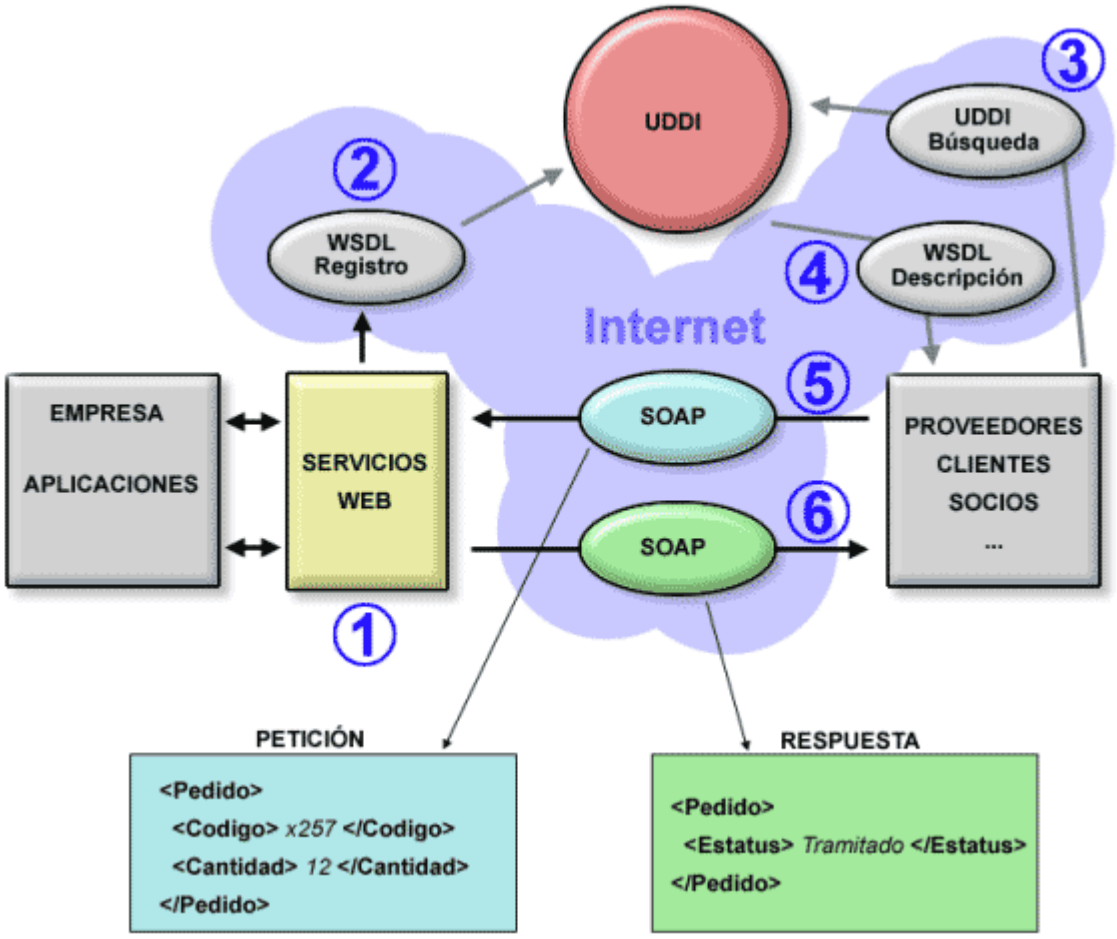
22. **B, Julio César Alcubilla.** Tecnoligí@HechaPalabra. *ABC...SOA...Primera Parte .* [En línea] mayo de 2007.

23. **Sevilla, Iván Ruiz.** Una de codigo. *El paradigma Modelo Vista Controlador (Tutorial ROR II) .* [En línea] mayo de 2007. http://www.unadecodigo.com.

24. Muestra de estándares de codificación utilizando la documentación de la librería PEAR. [En línea] http://www.alterbrain.net/2008/04/03/estandares-de-codificacion-para-php/.

ANEXOS.

Anexo 1: El ciclo de vida de un Servicio Web.



1. El ciclo se origina cuando las empresas se deciden a desarrollar y exponer la funcionalidad de sus aplicaciones en forma de servicio Web.
2. Una vez que los servicios Web se han desarrollado, deben ser registrados en un nodo UDDI para poder ser localizado por los potenciales usuarios. En dicho registro se aportaran datos sobre la empresa, los servicios Web que se ofrecen etc. y también la descripción de las interfaces de uso de cada servicio Web (WSDL). Cuando algún consumidor solicite dicho servicio Web, el servidor UDDI le redirigirá a la URI proporcionada por el fabricante.
3. Los posibles consumidores (proveedores, clientes, socios...) se conectan al servidor UDDI para buscar los servicios Web que les interesan.

4. Una vez que encuentran el servicio Web que desean, obtienen la descripción de sus interfaces de uso (WSDL).
5. Gracias a la descripción de las interfaces de uso, los consumidores son capaces de elaborar paquetes SOAP para comunicarse con el proveedor del servicio Web.
6. El proveedor del servicio Web elabora un paquete SOAP como respuesta a la petición del consumidor del servicio Web.

ANEXO 2: DIAGRAMAS DE CLASES DEL DISEÑO.

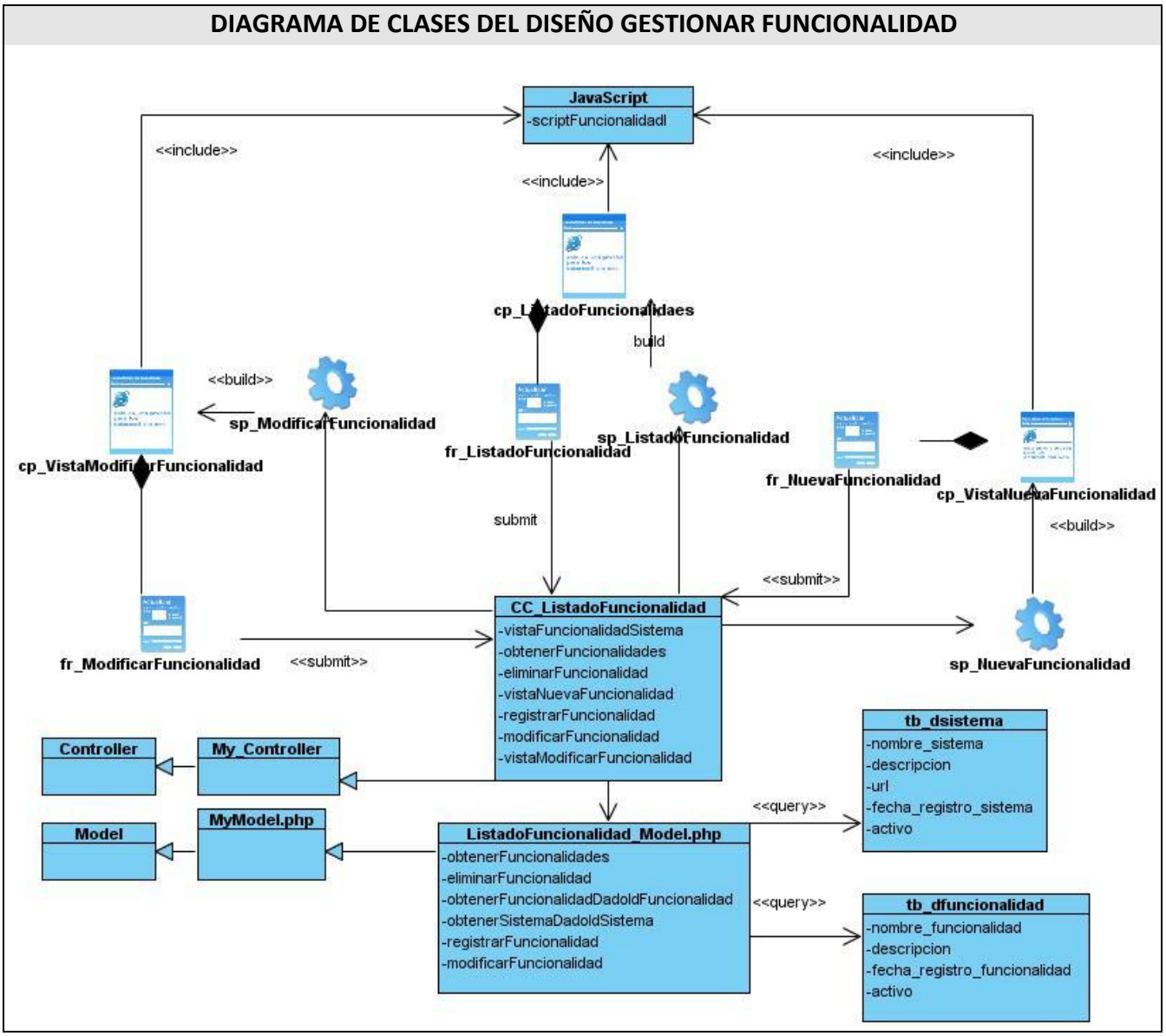


Fig. 5 Diagrama de Clases del Diseño Gestionar Funcionalidad de SegRED.

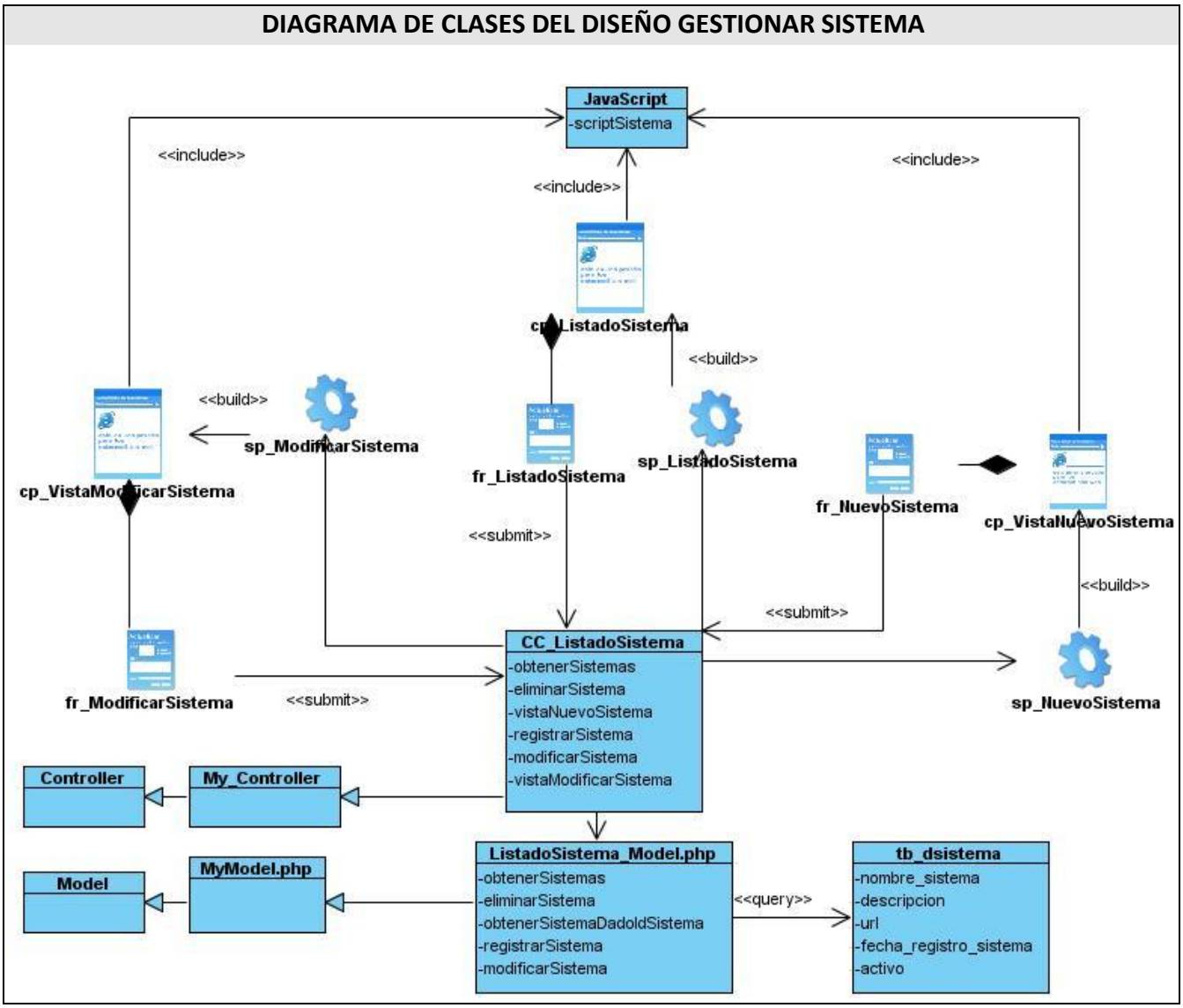


Fig. 6 Diagrama de Clases del Diseño Gestionar Sistema de SegRED.

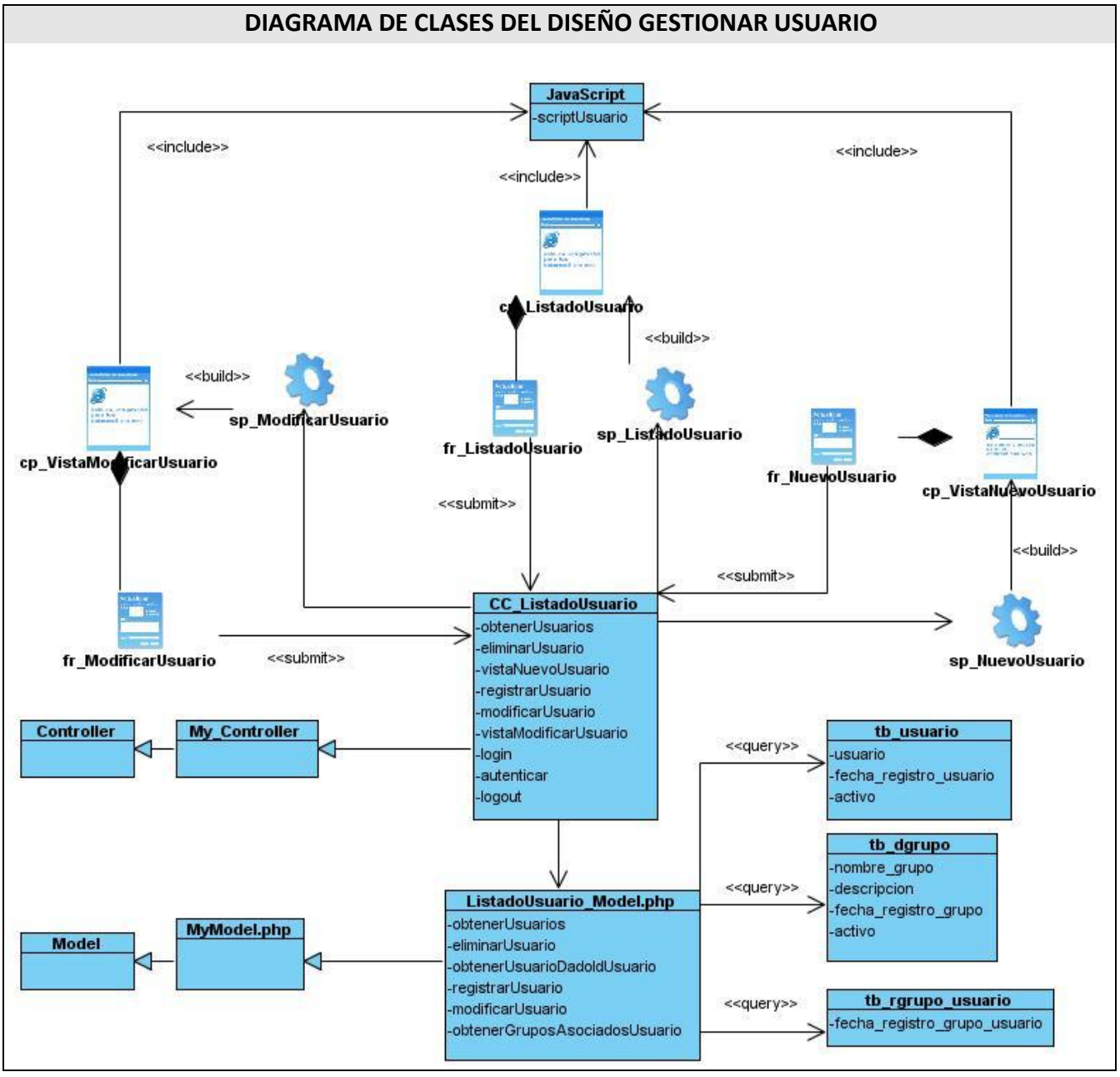


Fig. 7 Diagrama de Clases del Diseño Gestionar Usuario de SegRED.

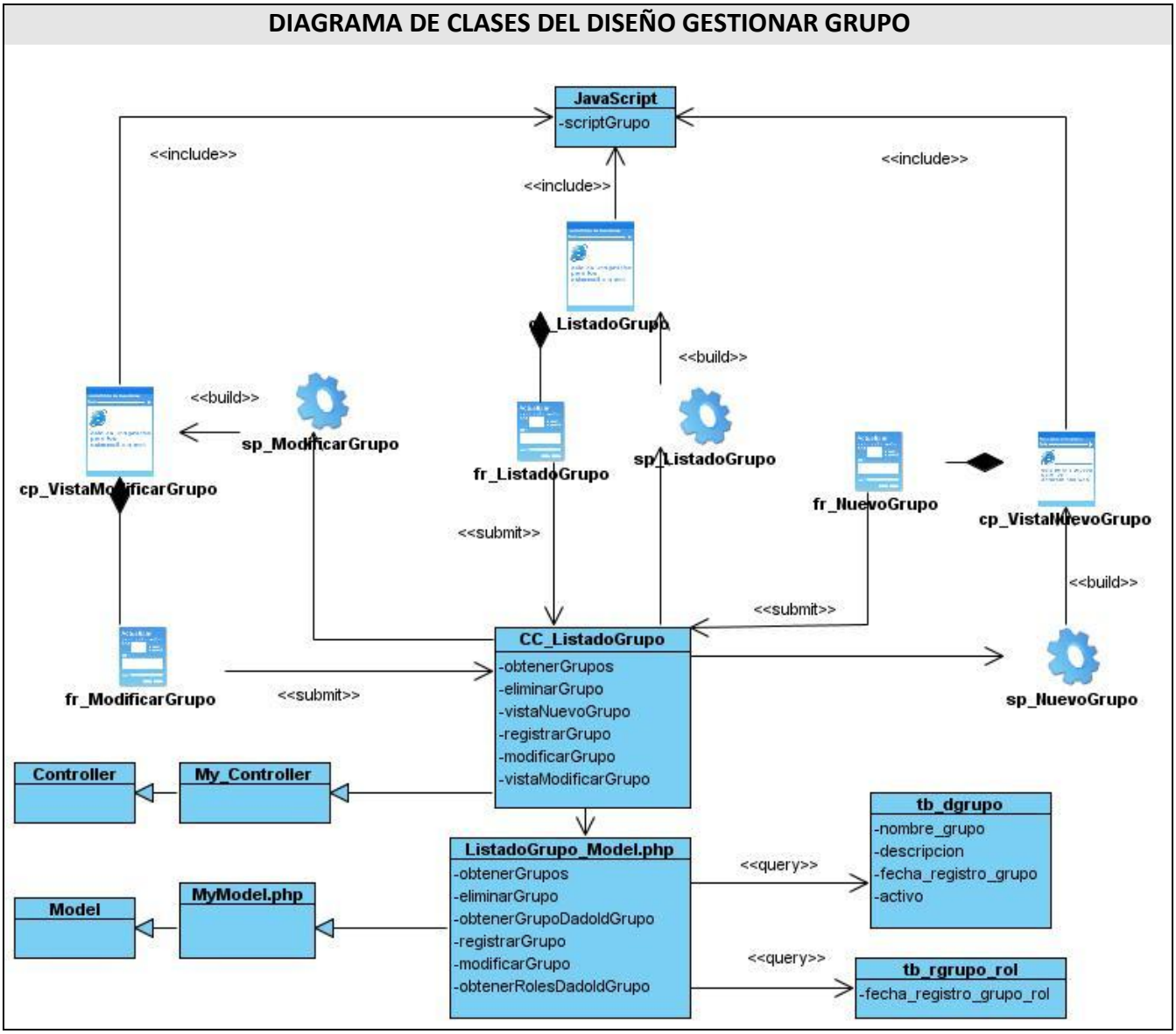


Fig. 8 Diagrama de Clases del Diseño Gestionar Grupo de SegRED.

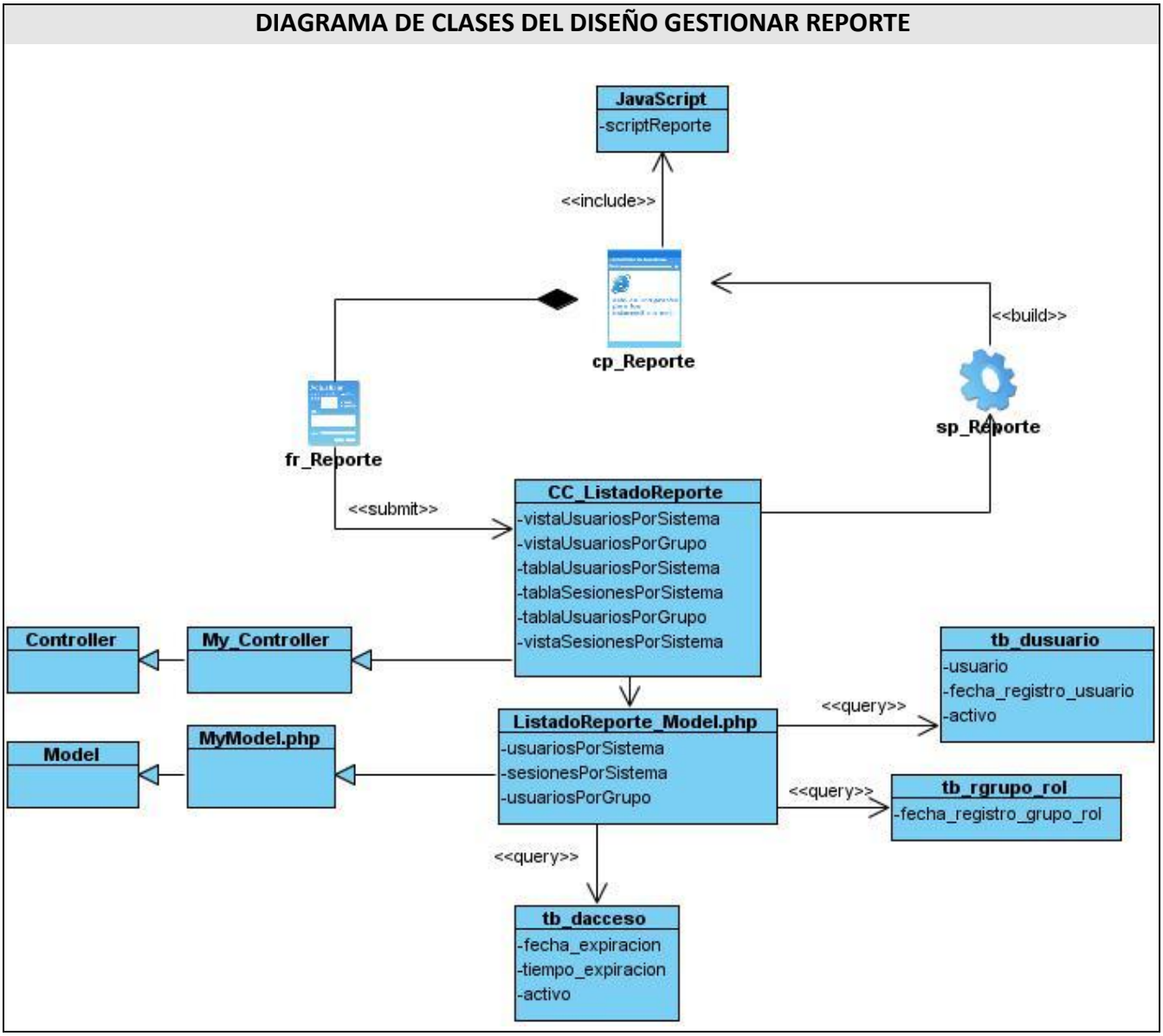


Fig. 9 Diagrama de Clases del Diseño Gestionar Reporte de SegRED.

GLOSARIO

Autenticación: capacidad de identificar al usuario de un sistema de información.

Autorización: proceso por el cual la red de datos faculta al usuario identificado a acceder a determinados recursos de la misma.

Credenciales: par usuario - contraseña. Es utilizado comúnmente en cualquier aplicación que requiera autenticar. En dependencia del contexto, también puede interpretarse como la unión del par anteriormente mencionado, con la suma de permisos que tiene un usuario determinado en un sitio específico, es decir, como un pasaporte o visa para transitar en la red.

Hash o algoritmo Hash: función o método para generar claves o llaves que representen de manera unívoca a un, registro, etc. La función hash, también se utiliza para los procesos de encriptación y decrepitación (utilizadas para autentificar).

Escalabilidad: capacidad de un sistema de aumentar el número de sus usuarios, aumentando sus recursos y sin perder ninguna de sus ventajas.

WebMaster: profesional capacitado para desarrollar un sitio Web. Es la persona de confianza que maneja información privada de la empresa, como base de datos, contraseñas y llaves de acceso a claves para modificar, actualizar o reparar.

Single Sign-On (SSO): mecanismo que permite la autentificación única por parte del usuario para acceder a sus recursos. La idea es introducir una única vez el nombre de usuario y contraseña, sin necesidad de volver introducirlo a la hora de acceder a nuevos recursos en los que aún no se había autentificado.

Solución de Software Implantada: producto informático desarrollado e instalado, para la informatización de tareas que facilitan al usuario la realización de un determinado tipo de trabajo. Es un término que incluye aplicaciones informáticas, servicios Web, bases de datos, etc. Se refiere al sistema en su totalidad sin especificar los componentes que lo conforman.

Open Source: Termino con que se conocen las aplicaciones que son desarrolladas y distribuidas de forma libre.