# Universidad de las Ciencias Informáticas

# Facultad 7


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

## Sistema para la Cooperación Médica: Módulo de Administración

**Autores:**

Alianny Valdivia Castro

Nuria Isabel Betanco Alvarez

**Tutores:**

Ing. Diana Rosa Alfonso Espinosa

Ing. Alfredo Manuel Guzmán Martínez

**La Habana, junio de 2011**

**"Año 53 de la Revolución"**

# Datos de Contacto

**Ing. Diana Rosa Alfonso Espinosa *(dralfonso@uci.cu)***: Graduada de Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el año 2008. Profesora de la Universidad de las Ciencias Informáticas, pertenece al Departamento de Práctica Profesional de la Facultad 7, donde imparte actualmente la asignatura Aplicaciones Informáticas en el sector de la Salud. Es miembro además del Departamento de Sistemas de Apoyo a la Salud del Centro de Informática Médica (CESIM) el cual desarrolla soluciones informáticas en la rama de la salud. Actualmente se desempeña como Analista principal del proyecto de Colaboración Médica.

**Ing. Alfredo Manuel Guzmán Martínez *(amguzman@uci.cu)***: Graduado de Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el año 2008. Profesor de la Universidad de las Ciencias Informáticas, pertenece al Departamento de Ingeniería de Software y Programación de la Facultad 7 y es profesor adjunto al Departamento de Sistemas de Apoyo a la Salud del CESIM. Ha impartido las asignaturas de Preparación para la Prueba de Nivel de Programación, Gráfico por Computadoras y Programación 3. Actualmente se desempeña como Líder de desarrollo del proyecto de Colaboración Médica.

## Agradecimientos

***Nuria:***

*Sin dudas la parte más difícil es la de los agradecimientos, porque más que agradecer por la tesis es necesario agradecer por estos cinco años transcurridos que han sido importantísimos en mi vida y me han convertido en la persona que actualmente soy.*

*Primeramente el agradecimiento más grande es para mi familia. Para mi mamá por su apoyo incondicional, por sus consejos, por sus palabras de aliento y por su ejemplo. A mis hermanas Zoila y Adita, porque son las que me inspiran a ser mejor cada día, por ese cariño que me dan incluso a pesar de que me ven poco y por sus palabras de consuelo en los momentos tristes. Al resto de mi familia que siempre han confiado en mí, por preocuparse, por su amor, por todo, muchas gracias.*

*A mi novio Anniel por su paciencia, por aguantar a diario mis irritables momentos, por su amor y su cariño.*

*A los hermanos que me ha regalado esta Universidad: a Marichú por siempre creer en mí, por sus consejos, por su amistad, a Boris por siempre escucharme, por compartir momentos lindísimos conmigo, por su actitud ante la vida.*

*Le agradezco también a todos mis amigos que desde primer año han estado conmigo por siempre: a Yadia por sus locuras, por sus momentos, por su apoyo, a Danito por sus canciones, por la perseverancia que tiene con todo lo que se propone, a Yasmany por su cariño y su facilidad para escuchar mis problemas. A los que se fueron pero se quedaron en mi corazón: Anisley y Ricardo, la UCI no es lo mismo sin ustedes.*

*También a las muchachitas del apartamento: a Indira, Yurlenis, Dainiri, Daneisy, Magdeline, Amailis, Leydanis y Adriana; por vivir todo este tiempo conmigo y aguantarme los buenos y malos momentos, por sus consejos también.*

*A Shirly, Dayi, Yisel e Hilda, por sus horas de conversación y por aguantar mis quejas, por darme las fuerzas para enfrentar los problemas.*

*A todos los de mi grupo que han marcado sin dudas momentos importantes en estos años: José Carlos, Luismel, Raysel, René, Edgar, Franklin, Arrebato, Javier, Bryan, Guillermo, Meme, Renán, Mojena, Yuyu, Carlos, Eddy, los jimaguas, Idayana, Ismaray, Zenia, Laimerys y Randy.*

*A los muchachos de cuarto año que me ayudaron con la tesis, por sus horas de dedicación prestadas y por su apoyo: a Tony, Ricardo, Milián y Juan Manuel. A los demás que se han sumado a mi grupo de amigos: Noslen, Alpí, Fabelo, Guillermo, Castelvi.*

*A mi compañera de tesis, Alianny, por su optimismo, por su profesionalidad, por su entrega y responsabilidad.*

*A los profesores que han confiado en mí y en mi trabajo: Alexis y Luis Mariano.*

*A Annia por toda su ayuda y por las veces que la molestamos, a Pompa por su paciencia y por aclarar todas mis dudas.*

*A mis tutores por sus exigencias, que me han enseñado a ser una persona más fuerte.*

*A la oponente Yisel por sus ayuda, sus sugerencias y su tiempo.*

*A todos los del tribunal por sus consejos.*

*A todos los que están presentes y no mencioné por ser muchos.* ***Gracias por todo.***

*Alianny:*

*Le agradezco a mi Madre porque sin su apoyo no hubiera sido posible este sueño, gracias a ella he logrado todo lo que soy en este mundo y sobre todo por los sacrificios que ha hecho por mí.*

*A mi Padrastro por ser la persona que me guía cuando tengo un problema, siempre supo darme los mejores consejos y su incondicional apoyo aunque no estuviera de acuerdo con la decisión que tomara.*

*A mi Papá por apoyarme siempre y ser el padre maravilloso que es.*

*A mis Abuelos, Tíos, Tías y Primos por ayudarme durante todo este camino lleno de buenos y malos momentos.*

*A Dayana por tener tanta paciencia conmigo a pesar de mis malcriadeces, darme todo su apoyo y amor.*

*A mis amistades Roselys, Yanelis, Jacqueline y Henry por estar ahí cuando más los necesité, y ser mis mejores consejeros.*

*A mi compañera de tesis Nuria por su paciencia y exigencia, gracias a ti este sueño se hizo realidad.*

*A Annia Arencibia porque aunque no fuera mi tutora me apoyó en todo momento, me brindó su ayuda cuando la necesité.*

*A mi oponente Yisel por darnos los mejores consejos y sus mejores recomendaciones.*

*A mis tutores que gracias a sus exigencias me he convertido en una persona más madura.*

*Y por último pero no menos importante a todos mis compañeros de aula durante estos 5 años, a mis profesores y todas las personas que de una forma u otra me han apoyado.*

## Dedicatoria

***Nuria:***

*Dedico mi tesis, mi vida y mis acciones de cada día a mi mamá que es mi razón de ser, mi luz, mi guía, mi apoyo, mi mundo y mi TODO.*

*A mis hermanas Zoilyta y Adyta, que son mis pedacitos de luz, por ustedes lucho para ser mejor cada día. Este esfuerzo va dedicado a ustedes por todo el amor que les tengo.*

*Para mi familia en general, por su orgullo y confianza.*

***Alianny:***

*Dedico este trabajo a mi Abuelo y a mi Abuela por ser las personas más importantes en mi vida, por apoyarme en todo y ser mis ejemplos a seguir.*

*A mi Madre y a mi tía Carmen por significarlo todo para mí e impulsarme a ser mejor cada día, por ser mi sostén en los tiempos difíciles, por ser más que madres, amigas.*

*A mi padrastro y a mi papá por guiarme cuando más lo necesitaba, por su comprensión aunque tomara decisiones erróneas, por ser el faro que ilumina mi camino.*

*A mis tíos Israel, Juan, Ernesto y Tito por su apoyo incondicional y siempre ayudarme a levantar cuando caía.*

*A mis tías Barbará, Irma, China, Ilia, María Rosa, Zoila e Isabel por saber darme el consejo exacto cuando lo necesité.*

*A Dayana por su amor incondicional, paciencia y comprensión durante todo este año.*

# Resumen

El Sistema para la Cooperación Médica (COLABEN) surge a partir de la necesidad que tiene el estado de conocer la información real de las misiones médicas en el exterior, así como la de sus colaboradores. La información relacionada con la colaboración médica cubana en el exterior, es una tarea llevada a cabo por la Unidad Central de Cooperación Médica (UCCM), que no cuenta con un registro único y actualizado de estos datos. Con el objetivo de gestionar la información de los codificadores y los usuarios, se realiza el Módulo de Administración del Sistema para la Cooperación Médica.

El módulo se desarrolló utilizando el lenguaje de programación PHP v5.3. Para una mejor estructuración del código fuente se utilizó el Framework Symfony 1.4. Como Sistema Gestor de Base de Datos se utilizó PostgreSQL v8.3, se eligió Apache 2.2 como el servidor Web sobre el que correrá la aplicación. Se trabajó con la herramienta Enterprise Architect 7.1 para realizar el modelado haciendo uso de la notación UML v2.1 (Lenguaje Unificado de Modelado). Se desarrolló la aplicación en plataformas de software libre, utilizando el Sistema Operativo Linux Distribución Ubuntu v10.04.

El módulo desarrollado permitirá la gestión de la información de los codificadores y los usuarios de COLABEN; garantizando la seguridad y confiabilidad de la aplicación, además de la facilidad en la gestión de la información con la que trabajan los restantes módulos del sistema.

**Palabras claves:** Administración, Colaboradores, Cooperación médica.

# Índice

# Introducción

Desde hace varios años, la informática se ha fundamentado como una de las herramientas de apoyo al proceso sanitario; evolucionando como ciencia en la informática médica. El objetivo principal de esta ciencia es mejorar la gestión de la información de la salud, para garantizar el aumento en la calidad de la atención sanitaria. Para lograr dicho objetivo, no solo se utilizan métodos novedosos, sencillos y eficaces de gestión administrativa en instituciones médicas, sino que también, se dispone de recursos informáticos de valor que apoyan los procesos de gestión de la información en estas entidades. (1)

En la rama de la salud, las Tecnologías de la Información y las Comunicaciones (TIC) tienen un gran impacto, pues constituyen un apoyo para el fortalecimiento de sistemas orientados a este sector. (2)

La sociedad cubana se encuentra inmersa en un proceso de informatización. Con el uso de las TIC se desea lograr una mayor eficiencia en los procesos de la sociedad, así como un aumento sistemático de la calidad de vida de los cubanos.

Cuba, siempre se ha caracterizado por prestar ayuda médica a otros países desde 1963, año que se considera el comienzo de la Colaboración Médica Internacional Cubana. Entre sus funciones se destacan: la asistencia médica clínico-quirúrgica a la población de los países en los lugares más apartados e intrincados, el desarrollo conjunto de las campañas de educación sanitaria y de vacunación masiva de la población, el control higiénico-epidemiológico y la prestación de servicios de brigadas médicas a damnificados por huracanes, sismos y otras contingencias en diferentes países. (3)

A partir de esto, y debido al alto número de colaboradores médicos cubanos que se encuentran cumpliendo misión internacionalista, surge la necesidad de crear la Unidad Central de Cooperación Médica (UCCM), con el fin de llevar un control sobre la información real de las misiones médicas cubanas en el exterior, así como la de sus colaboradores. Actualmente es la institución rectora donde se gestiona toda esta información.

Con la aplicación de las TIC, la Universidad de las Ciencias Informáticas (UCI), en colaboración con la Empresa de Soluciones Informáticas (SOFTEL), y a petición de la UCCM, en el año 2007 desarrolló el Sistema para la Gestión de Información de los Colaboradores de la Salud (SGCM), un sistema Web que contenía 3 módulos nombrados Gestión, Pago y Administración. Luego de liberarse este sistema por el Centro de Calidad para Soluciones Tecnológicas (CALISOFT), cambió la forma de pago en la UCCM, por lo que el sistema desarrollado ya no cumplía con las necesidades de la entidad; además, el Módulo de Administración presentaba como deficiencia que solamente creaba y actualizaba los codificadores.

En el año 2010 surge la necesidad de realizar una nueva aplicación, a la que se le otorgó el nombre de Sistema para la Cooperación Médica (COLABEN), que se basaría en los nuevos requerimientos de la institución. En sus inicios el desarrollo de este sistema solo contemplaba dos módulos, Economía y Recursos Humanos, y fue a medida que este desarrollo avanzaba que se identificó la ausencia de un mecanismo que llevara a cabo los siguientes procesos:

- ✓ La autenticación de usuarios centralizada y por roles.
- ✓ La gestión y personalización de las cuentas de usuarios.
- ✓ La creación y visualización de las acciones ejecutadas por un usuario sobre los elementos del sistema.
- ✓ La gestión de los codificadores que serán utilizados en los restantes módulos del sistema.

Debido a los elementos planteados anteriormente, surge como **problema científico**: el Sistema para la Cooperación Médica no gestiona la información de codificadores y usuarios.

El **objeto de estudio** se centra en el proceso de gestión de codificadores y usuarios.

El **campo de acción** se enmarca en el proceso de gestión de los codificadores y los usuarios en el Sistema para la Cooperación Médica.

Se propone como **objetivo general** desarrollar el Módulo de Administración que gestione la información de los codificadores y los usuarios del Sistema para la Cooperación Médica.

Las **tareas de la investigación** que se llevarán a cabo para darle cumplimiento al objetivo trazado son:

- ✓ Realizar un análisis de los sistemas informáticos existentes a nivel nacional e internacional, relacionados con la gestión de codificadores y usuarios, que sirvan como punto de partida en la propuesta del sistema.
- ✓ Revisar los procesos implementados en el Módulo de Administración del Sistema para la Gestión de Información de los Colaboradores de la Salud.
- ✓ Aplicar las tecnologías y arquitectura definidas por el Departamento Sistemas de Apoyo a la Salud (SAS), para el desarrollo de la solución.
- ✓ Generar los artefactos que propone el Proceso Unificado de Desarrollo (RUP) en los flujos de trabajo: Negocio, Requerimientos, Análisis y Diseño e Implementación, que sirvan de base para los desarrolladores del sistema.
- ✓ Implementar el módulo aplicando las pautas de diseño definidas por el Departamento SAS, siguiendo lo establecido en la Especificación de Requisitos de Software.

Para dar cumplimiento a estas tareas se emplearon los siguientes **métodos de la investigación**:

El método **Analítico – sintético** para organizar la información de varias fuentes diferentes. El método **Inductivo – deductivo** para relacionar de lo general a lo particular en esta materia y generalizar hallazgos. El **Análisis histórico – lógico** para estudiar la evolución de los sistemas nacionales e internacionales relacionados con la gestión de los codificadores y los usuarios, así como el método **Empírico** de la **observación** para comprender todas las características, ventajas y desventajas de los mismos.

El desarrollo del Módulo de Administración provee los siguientes beneficios:

- ✓ Se garantizará la centralización y seguridad de los usuarios que se autentican en el Sistema para la Cooperación Médica.
- ✓ Permitirá la trazabilidad de los elementos que se gestionan en el sistema, lo que garantizará

una mayor seguridad e integridad de la información que se maneja.

- ✓ Proporcionará la gestión de la información de los codificadores de manera uniforme para los restantes módulos del sistema.

El Sistema para la Cooperación Médica tiene un gran impacto para la sociedad ya que está conformado por una serie de módulos que están vinculados directamente a distintas esferas del país. El Módulo de Administración tiene un aporte indirecto para la sociedad, pues con la gestión de trazas se realiza un seguimiento de las acciones del usuario en el sistema y en caso de producirse algún error en la introducción de los datos permite la detección de la hora, lugar, acción y usuario que realiza el hecho, para tomar medidas con la persona implicada. De este modo puede evitarse cualquier forma de desvío de recursos, como por ejemplo: asignación de montos indebidos, datos de supuestos designados de las cuentas para que reciban ayuda familiar, información no verídica de colaboradores médicos, entre otros.

Lo anterior evidencia un impacto positivo para la economía del país, los trámites legales de pasaportes, la UCCM, el Ministerio de Relaciones Exteriores (MINREX), el Ministerio de Salud Pública (MINSAP) y por consiguiente el Consejo de Estado.

El contenido de este documento cuenta con la siguiente estructura:

**Capítulo I Fundamentación Teórica:** Se realiza un análisis sobre los diferentes aspectos teóricos relacionados con el desarrollo del tema tratado. Incluye un estado del arte en el mundo, en Cuba y en la UCI. Se hace una descripción de las tendencias, técnicas, tecnologías, metodologías y herramientas usadas para dar solución al problema.

**Capítulo II Características del Sistema:** Se refiere a las características del sistema. Se presenta la propuesta del sistema. A través del Modelo de Dominio se describen los conceptos fundamentales manejados en el negocio. Se especifican los requisitos funcionales y no funcionales de la aplicación y se modela el diagrama de casos de uso del sistema.

**Capítulo III Análisis y Diseño del Sistema:** En este capítulo se realizan la descripción de la arquitectura y el modelado de los diagramas de clases de análisis y diseño del sistema. Como apoyo

a la fase de implementación se modelan además los diagramas de interacción por cada realización de caso de uso.

**Capítulo IV Implementación:** Se describen como componentes, las clases y subsistemas desarrollados durante el diseño, indicando cómo estos elementos se implementan y se organizan en el diagrama de despliegue. Además, se realiza el diagrama de componentes general, el cual organiza las dependencias entre estos.

# Capítulo 1: Fundamentación Teórica

El objetivo fundamental de este capítulo es hacer un análisis sobre los diferentes aspectos teóricos relacionados con el tema tratado. Se abordan detalladamente los conceptos fundamentales, el estado del arte del sistema y se describen las herramientas y metodologías que se utilizarán para el desarrollo del mismo. Se realiza un estudio de las tecnologías propuestas por el Centro de Informática Médica que puedan ser utilizadas para el desarrollo del sistema.

## 1.1 Conceptos Fundamentales

**Unidad Central de Cooperación Médica (UCCM):** La UCCM es la institución encargada de gestionar y almacenar la información referente a los colaboradores cubanos de la salud.

**Empresa de Soluciones Informáticas (SOFTEL):** SOFTEL es una empresa de software cubana perteneciente al Ministerio de la Informática y las Comunicaciones (MIC), cuyo objeto social principal es ofrecer soluciones informáticas para el Sistema de Salud. (4)

**Centro de Informática Médica (CESIM):** CESIM surge como parte de la informatización de la salud cubana, que tiene como misión ser un centro de excelencia dedicado al desarrollo de productos, sistemas, servicios y soluciones de alta calidad y competitividad, para la optimización del trabajo y mejoramiento de la atención médica. (5)

**Cooperación:** La cooperación es el aporte de origen externo que se solicita a países u organismos internacionales, con el fin de apoyar el desarrollo nacional mediante acciones, proyectos y programas específicos. Además, se refiere abstractamente a todo proceso en donde se involucre el trabajo de varias personas en conjunto.

**Codificador:** Representa información común y poco variable en el tiempo, que tiene que ver directamente con el negocio. Cada módulo trabaja con esta información para dar cumplimiento a sus funcionalidades.

## 1.2 Estudio del Arte

### 1.2.1 Nivel Internacional

En el mundo existen varias aplicaciones enfocadas a la administración, que brindan la gestión de su información de acuerdo con sus necesidades y características. Algunos ejemplos relevantes son:

Integral Software es una empresa Argentina dedicada al diseño, desarrollo, integración e implementación de soluciones informáticas y tecnológicas. De acuerdo con las necesidades actuales de la gestión de organizaciones de salud se realizó el **Sistema de Gestión Administrativa Prestacional (SGAP),** el cual permite la configuración de los codificadores del sistema.

Este solamente funciona en plataformas del Sistema Operativo Windows (W98, NT, 2000 a 2003, XP). Utiliza las facilidades de ayuda en línea ofrecida por Windows para la correcta utilización de toda la potencialidad y características del sistema. Está desarrollado con la herramienta de última generación Power Builder, que permite diseñar sistemas robustos utilizando arquitectura Cliente-Servidor. Presenta diferentes opciones de motor de base de datos como: SYBASE SQL AnyWhere, SQL SERVER, ORACLE, MAX-DB. Configuración modular que permite habilitar o deshabilitar módulos de acuerdo a las necesidades de cada prestador o usuario. (6)

Una vez concluido el estudio del sistema se define que no puede ser utilizado, pues no funciona en plataformas de software libre, tiene una alta dependencia con el Sistema Operativo Windows, además, utiliza software propietario como plataforma de desarrollo y sistema gestor de base de datos, los cuales resultan muy costosos.

El **Sistema de Gestión de Usuarios (SGU),** fue creado por el Centro Latinoamericano y del Caribe de Información en Ciencia de la Salud, más conocido como la Biblioteca Regional de la Medicina (BIREME), se hizo con el objetivo de que todos los productos y servicios de BIREME compartieran la misma base de datos. Permite el registro de usuarios, la actualización de sus datos, así como la generación de nuevas claves. Esta aplicación no proporciona la posibilidad de listar todos los usuarios, ni de informar si un usuario se ha autenticado o no y no administra los roles de los usuarios. (7)

Luego de analizar el sistema se determinó que no puede ser utilizado, pues las funcionalidades para la gestión de usuarios que realiza no cumplen con los requerimientos de la UCCM. Además no cuenta con una seguridad de usuarios confiable.

La empresa española IDAT o más conocida como Desarrollo Avanzado de Tecnologías de Identificación, tiene como estrategia el desarrollo de tecnologías propietarias a nivel de hardware, software y sistemas, para lo que invierte, de manera permanente, elevados recursos para el desarrollo de sistemas automáticos, soluciones informáticas y dispositivos electrónicos de captura de datos mediante tecnologías RFID. (8)

**TRACE-IDAT**, software para el control y gestión de trazabilidad, permite definir la seguridad de usuarios controlando el acceso a los diferentes módulos y puntos de menú de la aplicación, añadiendo o restringiendo los niveles de acceso permitidos a cada usuario individual o grupo de usuarios de una forma ágil y cómoda. El Visor de Trazabilidad permite la navegación a través de los distintos puntos de análisis de la traza para la búsqueda de registros o de información de cualquier tipo. El diseño del sistema es modular, con lo cual se puede expandir fácilmente, incorporando nuevas interfaces de administración según los servicios y productos que la empresa requiera. (9)

Después de realizar un estudio del sistema se concluyó que no puede ser utilizado pues es un software propietario y su licencia de uso resulta muy costosa.

### 1.2.2 Nivel Nacional

En Cuba, diversas instituciones han desarrollado sistemas informáticos con el objetivo de gestionar los codificadores y los usuarios, ejemplo de esto son:

El **Sistema Automatizado para la Gestión de la Información de Ciencia e Innovación Tecnológica de una Universidad,** fue creado en el año 2008 en la Universidad Camilo Cienfuegos de Matanzas. Cuenta con un grupo de codificadores que son toda la información relacionada con la actividad de Ciencia e Innovación Tecnológica, lo que resulta de gran utilidad para el personal encargado de gestionar datos de este tipo y para todos los profesores e investigadores en general. Además gestiona usuarios que poseen diferentes tipos de acceso dentro del sistema.

Para el análisis y diseño del sistema se utilizó la metodología Proceso Unificado de Desarrollo (RUP), para obtener la aplicación Web se empleó el lenguaje de programación Java, con tecnologías Java Server Pages y Java Server Faces bajo el ambiente de desarrollo integrado NetBeans. Para la generación de reportes se usó el paquete JasperReport y el iReport para diseño visual. Se eligieron GlassFish V2 como servidor de aplicaciones y contenedor web, y el sistema gestor de base de datos PostgreSQL. (10)

Después de realizar un estudio del sistema se define que no puede ser utilizado pues no es configurable para ser utilizado en instituciones de salud. Además las herramientas de trabajo utilizadas no se ajustan a las pautas definidas por el CESIM.

El **Sistema AvilaLink** fue creado por la empresa DESOFT de Ciego de Ávila para la empresa ETECSA con el objetivo de administrar el tiempo de uso de las computadoras y sus niveles de acceso. Es una aplicación Cliente-Servidor que permite controlar el acceso a los servicios locales o de red que se desee ofrecer a los clientes desde una o varias estaciones de trabajo. Además puede controlar las trazas de navegación y la trazabilidad de las operaciones de los usuarios administrativos. Está diseñado para funcionar sobre la plataforma Microsoft Windows NT, particularmente sobre Windows 2000 o XP. Su sistema gestor de base de datos es SQL Server 2000. Requiere una velocidad de procesamiento superior a 1 Gigahertz y capacidad de memoria superior a 1 Gigabyte. (11)

Una vez concluido el estudio de este sistema se define que no puede ser utilizado pues está diseñado para funcionar sobre plataforma Windows y utiliza como gestor de base de datos un software propietario. Además requiere una arquitectura de hardware muy potente y ante un gran volumen de información a manejar, la aplicación exigirá mayores prestaciones.

### 1.2.3 Universidad de las Ciencias Informáticas (UCI)

En la UCI existen algunos sistemas que realizan parcialmente las funcionalidades de gestión de codificadores y usuarios:

**Sistema para la Gestión de los Colaboradores de la Salud**, uno de sus módulos es el de Administración, que tiene como objetivo la gestión de los codificadores usados por el sistema, este permite solamente insertar y modificar los codificadores. Para la implementación se utilizó el lenguaje de programación web PHP versión 5.0, el sistema gestor de base de datos MySql Server versión 5.0 y se utilizó servidor web Apache versión 2.5 sobre el sistema operativo Linux distribución Debian versión 4.0. (12)

Una vez concluido el estudio de este sistema se define que no puede ser utilizado pues gestiona de forma incompleta los codificadores del sistema. Además no gestiona usuarios ni la autenticación de los mismos. Las herramientas utilizadas no se ajustan a las que están definidas actualmente por el CESIM.

El **Sistema Cedrux** fue creado en la UCI por el programa ERP-CUBA, dentro de este se encuentra un subsistema para el control de los activos fijos tangibles (AFT) creado por el Departamento CEIGE de la facultad 3. Este presenta entre sus principales funcionalidades la gestión de los codificadores de las Áreas Físicas, Grupos, AFT, Operaciones, Marcas y Modelos que serán utilizados para la configuración del subsistema, así como en su gestión documental.

El Subsistema de Activos Fijos Tangibles (SAFT) está orientado a la web con protocolo de comunicación HTTP entre las páginas clientes y el servidor Web, se realiza sobre una plataforma libre con lenguaje PHP 5.2 y se utilizan herramientas como el PostgreSQL 8.3 que es el sistema gestor de base de datos y Visual Paradigm 6.1 para el modelado de los procesos. El sistema es orientado a componentes, la arquitectura se basa en un estilo en capas, aplica también el patrón arquitectónico Modelo - Vista - Controlador, se utiliza como patrón de integración entre componentes el inversor de control. (13)

Luego de realizar un estudio de este subsistema se define que no puede ser utilizado pues está restringido al sector de contabilidad por lo que no es factible su uso en las entidades de salud. Además tiene gran dependencia del resto de sus módulos.

El **Sistema de Autenticación, Autorización y Auditoría (AAA),** fue creado en la UCI en el año 2009 por el Departamento SAS, permite un control de la acumulación de las trazas generadas por los

usuarios en las tablas de la base de datos relacionadas con las mismas, además presenta un excelente proceso de los reportes generados en los servicios de auditoría. También cuenta con un servicio web utilizando el patrón proxy que sirve de interfaz a las peticiones de los sistemas clientes y garantiza el acceso seguro a los mismos, con un mecanismo automatizado que permite balancear las trazas almacenadas en las tablas de la base de datos que están relacionadas con este tipo de información.

Su desarrollo está basado sobre una arquitectura en capas, utilizando PHP 5 como lenguaje de programación, el patrón de arquitectura Modelo - Vista - Controlador, PostgreSQL 8.3 como sistema gestor de base de datos, tecnología AJAX para realizar de forma más eficiente 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. (14)

Una vez finalizado el estudio de este sistema se puede concluir que no puede ser utilizado pues solamente brinda la gestión de usuarios y no gestiona codificadores.

## 1.3 Tecnologías, Metodologías y Lenguajes utilizados

**Proceso Unificado de Desarrollo (RUP, por sus siglas en inglés)**

RUP, no es más que un proceso de desarrollo de software, es decir, un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software. RUP, además de ser un proceso de desarrollo es también 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.

Este proceso está compuesto por cuatro fases: Inicio, Elaboración, Construcción y Transición. Además, contiene nueve flujos de trabajo: Modelo de Negocio, Requerimiento, Análisis y Diseño, Implementación, Prueba, Instalación, Ambiente, Administración de Proyecto, Administración de Configuración y Cambios. Utiliza el Lenguaje Unificado de Modelado (UML) para modelar los artefactos de sus flujos de trabajo.

Las características que definen a RUP son:

- ✓ "Dirigido por casos: Se utiliza para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. Los casos de uso son el instrumento para validar la arquitectura del software y extraer los casos de prueba.
- ✓ Centrado en la arquitectura: Incluye los aspectos estáticos y dinámicos más significativos del sistema. La arquitectura es una vista del diseño completo con las características más importantes resaltadas, lo cual ayuda al arquitecto a centrarse en los objetivos adecuados, ya sea la comprensibilidad, la capacidad de adaptación al cambio y la reutilización. Los modelos son proyecciones del análisis, y el diseño constituye la arquitectura del producto a desarrollar.
- ✓ Iterativo e incremental: Las iteraciones ofrecen como resultado un incremento del producto desarrollado, que mejora las funcionalidades del sistema en desarrollo, es decir, las iteraciones hacen referencias a pasos en el flujo de trabajo, y los incrementos, al incremento del producto." (15)

En la actualidad constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Propone cómo deben ser ejecutadas las actividades para la obtención de los productos típicos de trabajo y la forma de documentar todas las acciones que se llevan a cabo en las mismas, soportando el ciclo completo de desarrollo de la aplicación. Es por esto que se decide utilizar RUP como metodología para el desarrollo del sistema.

**Lenguaje Unificado de Modelado (UML, por sus siglas en inglés)**

UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. UML no es una guía para realizar el análisis y diseño orientado a objetos, es decir, no es un proceso. UML es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos. (16)

UML es un lenguaje claro y uniforme definido para el diseño Orientado a Objetos (OO), que no garantiza el éxito de los proyectos, pero sí mejora sustancialmente el desarrollo de los mismos, al permitir una nueva y fuerte integración entre las herramientas, los procesos y los dominios, razón por la cual se escoge para el modelado del sistema a desarrollar.

**Servidor Web Apache**

El servidor Web Apache 2.2 fue creado para proveer un alto grado de calidad y fortaleza para las implementaciones que utilizan el protocolo HTTP. Es un software libre y multiplataforma que permite a clientes o instituciones construir sistemas confiables con fines experimentales o para resolver un problema específico de la organización. Es uno de los servidores web más populares del mercado, y el más utilizado actualmente, de código abierto y gratuito, disponible para Windows y GNU/Linux. Por lo anteriormente planteado se decide utilizar el servidor Web Apache 2.2.

**Sistema Gestor de Base de Datos (SGBD)**

Un Sistema Gestor de Base de datos (SGBD) o DBMA (DataBase Management System) es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos. (17)

**PostgreSQL**

PostgreSQL 8.3 es un SGBD objeto-relacional, presenta varias características, entre ellas:

- ✓ Posee una gran escalabilidad. Es capaz de ajustarse al número de computadoras y a la cantidad de memoria que posee el sistema de forma óptima, pudiendo soportar una mayor cantidad de peticiones simultáneas de manera correcta (algunas comparaciones sugieren que soporta el triple de carga de lo que soporta MySQL).
- ✓ Implementa el uso de sub-consultas y transacciones, haciendo su funcionamiento mucho más eficaz, y ofreciendo soluciones en campos en los que MySQL no podría.
- ✓ Tiene la capacidad de comprobar la integridad referencial, así como también, la de almacenar procedimientos en la propia base de datos.
- ✓ Es multiplataforma, disponible en Linux, *nix, Mac Os X y Windows, entre otros sistemas operativos.

Se seleccionó PostgreSQL v8.3 teniendo en cuenta la necesidad de utilizar herramientas libres para el desarrollo, además de que es un gestor multiplataforma, confiable, estable, con gran escalabilidad, control de concurrencia y funcionalidades que lo destacan como uno de los SGBD más potentes en la actualidad.

**Ajax**

Ajax (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma, es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones. JavaScript es el lenguaje interpretado en el que normalmente se efectúan las funciones de llamada de Ajax. Es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores, dado que está basado en estándares abiertos como JavaScript. (18)

No es un lenguaje de programación sino un conjunto de tecnologías (HTML-JavaScript-CSS-DHTML-PHP/ASP.NET/JSP-XML) que permiten hacer páginas de internet más interactivas. La característica fundamental de Ajax es permitir actualizar parte de una página con información que se encuentra en el servidor sin tener que refrescar completamente la página. De modo similar se puede enviar información al servidor. (19)

Se decide la utilización de Ajax para las validaciones y el envío de datos a través de las páginas.

**Framework Symfony**

Symfony 1.4 es un framework diseñado para optimizar el desarrollo de las aplicaciones web por las características que presenta. Este framework separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación Web, proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación Web compleja. Automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Symfony está desarrollado completamente con PHP 5, es compatible con la mayoría de los gestores de base de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux) como en plataforma Windows. (20)

Se decide usar Symfony 1.4 como framework para el desarrollo del sistema ya que su licencia es de tipo software libre. Además, es uno de los framework PHP más populares entre los usuarios y las empresas, ya que permite que los programadores sean mucho más productivos y a la vez se crea código con calidad, fácil de mantener y de actualizar. Symfony es maduro, estable, profesional y está muy bien documentado.

**ExtJS**

ExtJS 3.1 es una librería JavaScript ligera de alto rendimiento que se utiliza para crear interfaces de usuarios, más agradables e interactivas. Se caracteriza por sus componentes de interfaz de usuario personalizables, con buen diseño y documentación. (21)

Para la realización del sistema se utilizará ExtJS 3.1 como framework de diseño visual, puesto que posee código reutilizable, es un framework de código limpio, que utiliza tecnología Ajax, escrito en JavaScript con soporte para la mayoría de los navegadores, presenta cientos de funcionalidades que hacen de él una herramienta indispensable, ya que junto con el ahorro de tiempo y de líneas de código hace que el usuario se sienta a gusto, pues presenta una interfaz amigable. Contiene Hojas de Estilo en Cascada (CSS) predefinidas aunque es totalmente configurable. Posee 2 licencias, una comercial y otra Open Source (Código Abierto).

**Lenguaje de programación web Hypertext Pre-processor (PHP)**

PHP 5.3 es un lenguaje de programación del lado del servidor, es decir, se ejecuta en el servidor Web justo antes de que se envíe la página al cliente. Es gratuito e independiente de plataforma, rápido, con una gran librería de funciones y una amplia documentación. Permite la conexión a diferentes gestores de base de datos, incluyendo PostgreSQL; lo cual permite la creación de aplicaciones web muy robustas. (22)

**JavaScript**

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Es un lenguaje script u orientado a documento. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. (23)

Es por esto que se decide la utilización de JavaScript como lenguaje del lado del cliente.

## 1.4 Herramientas

**Enterprise Architect**

Enterprise Architect 7.1 es una potente herramienta de análisis y diseño para el desarrollo de software robusto y de fácil mantenimiento. Abarca desde la recogida de requisitos, pasando por el análisis, el modelado, la implementación y las pruebas, hasta el despliegue y mantenimiento. Enterprise Architect es una herramienta versátil y multiusuario, que ayuda a que los proyectos de desarrollo de software tengan éxito a largo plazo. (24)

Es una herramienta flexible y completa. Fue construida en base al excepcional éxito de las versiones previas con un completo soporte para UML 2.1. Puede generar código fuente en PHP. Permite la ingeniería inversa para muchos sistemas que contengan diferentes SGBD como PostgreSQL.

**NetBeans**

NetBeans 6.9 es un proyecto de código abierto con una gran base de usuarios y una comunidad en constante crecimiento. El soporte para PHP 5.3 ha sido extendido para incluir el framework Symfony.

Es un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) gratuito para desarrolladores de software. Ofrece todas las herramientas necesarias para crear aplicaciones web y de escritorio con el lenguaje Java, C/C++ y lenguajes dinámicos como PHP y JavaScript. Es fácil de instalar y se puede ejecutar tanto en Windows como en Linux; también tiene detección de errores de sintaxis en tiempo real. (25)

Esta herramienta fácil de instalar y usar, será utilizada para el desarrollo del presente sistema, ya que es un producto libre y gratuito sin restricciones de uso. Permite crear aplicaciones Web con PHP 5 y además incluye soporte para Symfony y Ajax, lo que lo hace aún más popular entre los desarrolladores.

## 1.5 Arquitectura del Sistema

**Modelo Vista Controlador (MVC)**

"La arquitectura Modelo Vista Controlador (MVC) separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

- ✓ Modelo: Es la representación específica de la información que utiliza el sistema. Representa la lógica del negocio, es decir, encapsula los datos y las funcionalidades.
- ✓ Vista: Presenta el modelo en un formato adecuado para interactuar, usualmente se conoce como la interfaz de usuario.
- ✓ Controlador: Responde a eventos, usualmente acciones del usuario, e invoca peticiones al modelo y, probablemente, a la vista." (26)

"La principal ventaja de esta separación reside en la facilidad para realizar cambios en la aplicación puesto que:

- ✓ Cuando se realiza un cambio de base de datos, programación o interfaz de usuario, solo se toca uno de los componentes.
- ✓ Se puede modificar uno de los componentes sin conocer cómo funcionan los otros.
- ✓ Incrementa la reutilización y flexibilidad." (27)

Se decide la utilización del MVC por la facilidad que brinda para la creación de sistemas web, además de la organización que provee en el código fuente, por lo que se consigue un mantenimiento más sencillo de las aplicaciones.

**Arquitectura Cliente-Servidor**

La arquitectura Cliente-Servidor se divide en dos partes claramente diferenciadas, la primera es la parte del servidor y la segunda la de un conjunto de clientes. Normalmente el servidor es una máquina bastante potente que actúa de depósito de datos y funciona como un sistema gestor de base de datos. Por otro lado los clientes suelen ser estaciones de trabajo que solicitan varios servicios al servidor. Ambas partes deben estar conectadas entre sí mediante una red. (28)

Las principales ventajas de su uso son:

- ✓ "Recursos centralizados: debido a que el servidor es el centro de la red, puede administrar los recursos que son comunes a todos los usuarios, por ejemplo: una base de datos centralizada se utilizaría para evitar problemas provocados por datos contradictorios y redundantes.
- ✓ Seguridad mejorada: ya que la cantidad de puntos de entrada que permite el acceso a los datos no es importante.
- ✓ Administración al nivel del servidor: ya que los clientes no juegan un papel importante en este modelo, requieren menos administración.
- ✓ Red escalable: gracias a esta arquitectura, es posible quitar o agregar clientes sin afectar el funcionamiento de la red y sin la necesidad de realizar mayores modificaciones." (29)

Entre las principales características de la arquitectura Cliente-Servidor, se pueden destacar que:

- ✓ "El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.
- ✓ El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- ✓ Los cambios en el servidor implican pocos o ningún cambio en el cliente." (30)

En este capítulo, después de haber realizado un estudio teórico, referente a las tecnologías y tendencias del mundo actual, se arribó a la conclusión de que ninguna de las aplicaciones existentes, pueden darle solución a la situación problémica planteada. Las tecnologías, metodologías, arquitectura y lenguajes seleccionados para el desarrollo de este sistema se ajustan a las pautas definidas por el CESIM.

# Capítulo 2: Características del Sistema

El presente capítulo tiene como objetivo la descripción de las características del sistema además de la propuesta de solución para el Módulo de Administración. Se desarrolla un Modelo de Dominio y se analizan los requerimientos funcionales y no funcionales, lo cual permite un mejor entendimiento del sistema.

## 2.1 Modelo de Dominio

El Modelo de Dominio es la representación de los conceptos más importantes y significativos en el desarrollo de un sistema. Este representa clases conceptuales del dominio del problema y conceptos del mundo real, no de los componentes de software.

El objetivo fundamental del mismo es definir las interrelaciones de los objetos más importantes representados mediante clases. Desempeña un papel central en la comprensión del entorno actual y en la planificación futura de la posible aplicación. Ayuda a comprender los conceptos que utilizan los usuarios y los conceptos con los que deberá trabajar el sistema. (31)

Durante el modelado del negocio se decide realizar un Modelo de Dominio con las principales clases identificadas, con el objetivo de proveer una representación gráfica del sistema más fácil de comprender para el cliente.

### 2.1.1 Conceptos Fundamentales

Para obtener una mejor visión del diagrama Modelo de Dominio, a continuación se proporciona un marco conceptual con las principales definiciones identificadas.

**Usuario**: El usuario es el encargado de interactuar directamente con el sistema y realizar todas las funcionalidades permitidas según el rol que desempeña.

**Editor**: Es un tipo de Usuario que tiene los privilegios para modificar los datos que se gestionan en el sistema.

**Visualizador:** Es un tipo de Usuario que solo puede ver y buscar la información, pero no modificarla.

**Colaboradores**: Esta clase representa a una persona dentro de la entidad cliente a la cual se le asignan misiones, se le realizan movimientos y se le asigna un monto como resultado de su labor en la misión.

**Trazas**: Representan una descripción detallada de las acciones del Usuario en el sistema.

**Razas:** Esta clase contiene la información de todas las razas que existen. El sistema permite que el Usuario gestione todos estos datos.

**Categorías Docentes:** Contiene los datos de cada una de las categorías docentes que tienen los colaboradores. El sistema permite que el Usuario gestione todos estos datos.

**Categorías Ocupacionales:** En esta clase se encuentran los datos de las diferentes ocupaciones que tienen los colaboradores en las misiones. El sistema permite que el Usuario gestione todos estos datos.

**Diplomados:** Esta clase guarda los datos de los diplomados que han cursado cada uno de los colaboradores. El sistema permite que el Usuario gestione todos estos datos.

**Países**: Esta clase contiene los datos de los países donde los colaboradores pueden cumplir misión. El sistema permite que el Usuario gestione todos estos datos.

**Cargos:** Esta clase contiene los datos de los cargos que ocupan los colaboradores en sus puestos de trabajo. El sistema permite que el Usuario gestione todos estos datos.

**Expediente**: Esta clase contiene los datos personales, laborales y de residencia de un determinado colaborador. El sistema permite que el Usuario gestione todos estos datos.

**Grado Científico**: Esta clase contiene los datos de todos los grados científicos que pueden tener cualquiera de los colaboradores. El sistema permite que el Usuario gestione estos datos.

**Especialidades:** En esta clase se guardan todas las especialidades que pueden tener los colaboradores. El sistema permite que el Usuario gestione todos estos datos.

**Niveles técnicos:** Contiene la información referente a los distintos niveles técnicos que tienen las especialidades. El sistema permite que el Usuario gestione todos estos datos.

**Militancia**: Esta clase contiene los datos de las organizaciones políticas a las que puede pertenecer cualquiera de los colaboradores. El sistema permite que el Usuario gestione estos datos.

**Provincias:** Esta clase contiene los datos de las provincias que tiene Cuba en las cuales residen o trabajan los colaboradores. El sistema permite que el Usuario gestione todos estos datos.

**Dedicaciones**: Esta clase contiene los datos de las dedicaciones que pueden tener los colaboradores en la misión asignada. El sistema permite que el Usuario gestione estos datos.

**Municipios:** Esta clase contiene los datos de los municipios que tienen las provincias de Cuba en los cuales pueden residir o trabajar los colaboradores. El sistema permite que el Usuario gestione todos estos datos.

**Organismos:** Contiene los datos de los diferentes organismos de trabajo que existen en el país y a los cuales pertenecen los colaboradores. El sistema permite que el Usuario gestione todos estos datos.

**Unidades de Salud:** Esta clase contiene los datos de las unidades de salud que existen en el país y en las cuales trabajan los colaboradores médicos. El sistema permite que el Usuario gestione todos estos datos.

**Cuenta:** Esta clase contiene los datos del Usuario que se encuentra autenticado en la aplicación. El sistema debe permitir modificar su contraseña cada vez que el Usuario lo desee, así como la foto del perfil.

### 2.1.2 Diagrama del Modelo de Dominio

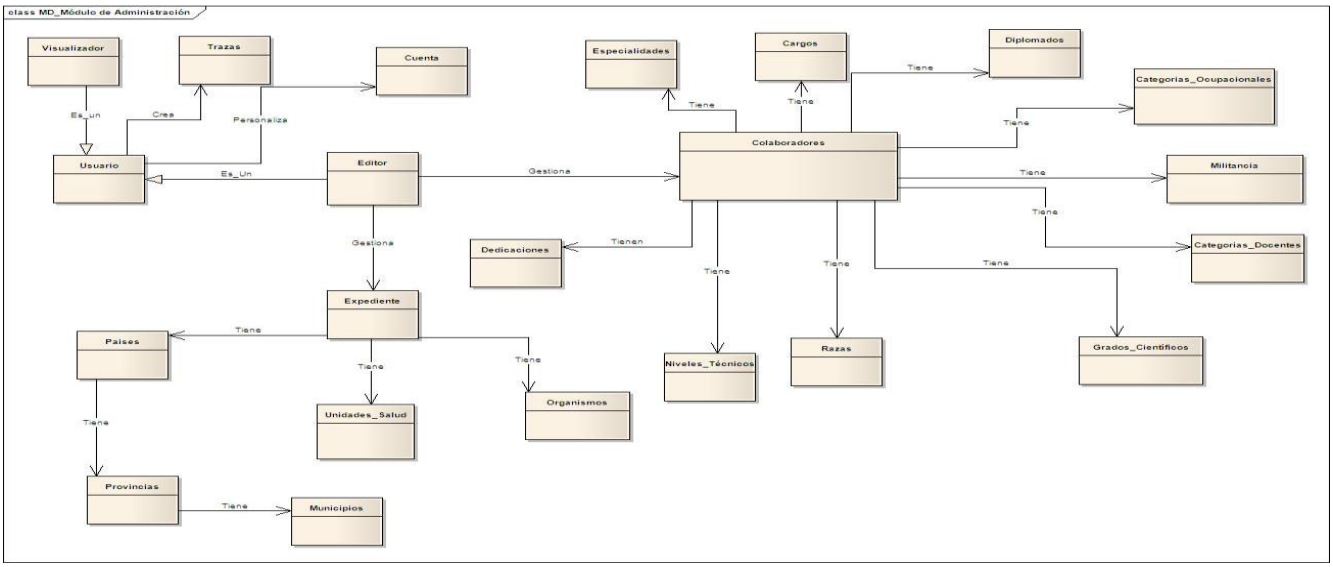


*Figura 1: Diagrama del Modelo de Dominio.*

## 2.2 Propuesta del Sistema

Se propone el desarrollo de un Módulo de Administración en el Sistema para la Cooperación Médica que permita:

- ✓ La autenticación de usuarios centralizada y por roles. De esta manera se garantiza la seguridad y confiabilidad de la aplicación, permitiendo solamente el acceso a cada módulo al personal autorizado previamente por el administrador.

- ✓ La gestión y personalización de las cuentas de usuarios, mediante la cual se le asignarán los permisos y roles necesarios a cada usuario del sistema. Además, el sistema posibilitará a los editores modificar los datos referentes a los usuarios, como por ejemplo usuario, contraseña, teléfono, correo electrónico, nombre y apellidos. Será posible también que cada usuario, sin importar su rol en la aplicación, pueda cambiar su contraseña y además personalizar la foto de usuario que identificará su sesión.

- ✓ La creación y visualización de las acciones ejecutadas por un usuario sobre los elementos del sistema. A partir de estas trazas generadas por el usuario, el sistema brindará la posibilidad de tener un seguimiento sobre las acciones que realizan los usuarios, permitiendo la búsqueda y visualización de las mismas.

- ✓ La gestión de los codificadores que serán utilizados en los restantes módulos del sistema, posibilitando crear, modificar, eliminar y buscar cada uno en caso de que se necesite.

## 2.3 Especificación de los Requisitos de Software

Los requisitos son cualidades o funcionalidades que el sistema debe cumplir para que tenga un correcto funcionamiento. Una buena captura de requisitos es fundamental para el desarrollo exitoso de la aplicación, pues ellos proveen al equipo de desarrollo de un entendimiento específico y común de las cualidades del sistema.

El objetivo fundamental de la captura de los requerimientos del software es traducir de forma sencilla el problema tal y como lo ve el cliente, logrando un mejor entendimiento entre el usuario final y los desarrolladores. Este entendimiento sobre lo que debe y no debe hacer el sistema permite identificar las funcionalidades requeridas y las restricciones que se imponen.

### 2.3.1 Requisitos Funcionales (RF)

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir para darle solución al problema identificado. (32) Ellos no alteran la funcionalidad del producto, esto quiere decir que los requerimientos funcionales se mantienen invariables sin importar con qué propiedades o cualidades se relacionen.

A continuación se muestra el listado de requisitos funcionales obtenidos luego de ser analizados los principales conceptos del Modelo de Dominio. Para una mejor comprensión, las descripciones de cada uno se encuentran en el Expediente del Proyecto en la plantilla de Especificación de Requisitos de Software.

| Listado de Requerimientos Funcionales | |
|---|---|
| RF1 Autenticar | RF2 Adicionar Usuario |
| RF3 Buscar Usuario | RF4 Editar Usuario |
| RF5 Eliminar Usuario | RF6 Listar Usuario |
| RF7 Personalizar Cuentas de Usuarios | RF8 Generar Trazas |
| RF9 Buscar Trazas | RF10 Listar Trazas |
| RF11 Adicionar Raza | RF12 Buscar Raza |
| RF13 Editar Raza | RF14 Eliminar Raza |
| RF15 Listar Raza | RF16 Adicionar Grados Científicos |
| RF17 Buscar Grados Científicos | RF18 Editar Grados Científicos |
| RF19 Eliminar Grados Científicos | RF20 Listar Grados Científicos |
| RF21 Adicionar Categoría Docente | RF22 Buscar Categoría Docente |
| RF23 Editar Categoría Docente | RF24 Eliminar Categoría Docente |
| RF25 Listar Categoría Docente | RF26 Adicionar Categorías Ocupacionales |
| RF27 Buscar Categorías Ocupacionales | RF28 Editar Categorías Ocupacionales |
| RF29 Eliminar Categorías Ocupacionales | RF30 Listar Categorías Ocupacionales |
| RF31 Adicionar Diplomados | RF32 Buscar Diplomados |
| RF33 Editar Diplomados | RF34 Eliminar Diplomados |
| RF35 Listar Diplomados | RF36 Adicionar Cargos |

| | |
|---|---|
| RF37 Buscar Cargos | RF38 Editar Cargos |
| RF39 Eliminar Cargos | RF40 Listar Cargos |
| RF41 Adicionar Especialidades | RF42 Buscar Especialidades |
| RF43 Editar Especialidades | RF44 Eliminar Especialidades |
| RF45 Listar Especialidades | RF46 Adicionar Niveles Técnicos de Especialidades |
| RF47 Buscar Niveles Técnicos de Especialidades | RF48 Editar Niveles Técnicos de Especialidades |
| RF49 Eliminar Niveles Técnicos de Especialidades | RF50 Listar Niveles Técnicos de Especialidades |
| RF51 Adicionar Militancia | RF52 Buscar Militancia |
| RF53 Editar Militancia | RF54 Eliminar Militancia |
| RF55 Listar Militancia | RF56 Adicionar Países |
| RF57 Buscar Países | RF58 Editar Países |
| RF59 Eliminar Países | RF60 Listar Países |
| RF61 Adicionar Provincias | RF62 Buscar Provincias |
| RF63 Editar Provincias | RF64 Eliminar Provincias |
| RF65 Listar Provincias | RF66 Adicionar Municipios |
| RF67 Buscar Municipios | RF68 Editar Municipios |
| RF69 Eliminar Municipios | RF70 Listar Municipios |
| RF71 Adicionar Organismos | RF72 Buscar Organismos |

| | |
|---|---|
| RF73 Editar Organismos | RF74 Eliminar Organismos |
| RF75 Listar Organismos | RF76 Adicionar Dedicaciones |
| RF77 Buscar Dedicaciones | RF78 Editar Dedicaciones |
| RF79 Eliminar Dedicaciones | RF80 Listar Dedicaciones |
| RF81 Adicionar Unidades de salud | RF82 Buscar Unidades de salud |
| RF83 Editar Unidades de salud | RF84 Eliminar Unidades de salud |
| RF85 Listar Unidades de salud | RF86 Imprimir |

*Tabla 1: Listado de Requerimientos Funcionales.*

### 2.3.2 Diagrama de Requisitos Funcionales

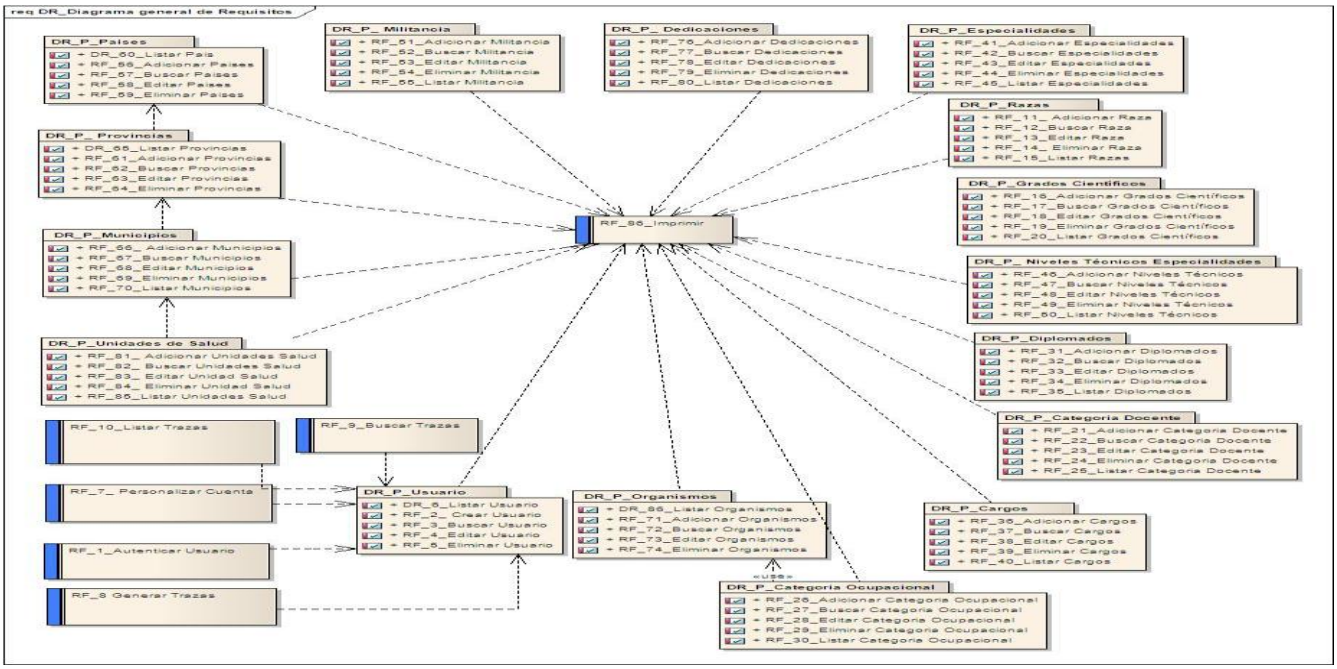


*Figura 2: Diagrama de Requisitos Funcionales.*

### 2.3.3 Requisitos No Funcionales (RNF)

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. (33)

Los requerimientos no funcionales son fundamentales en el éxito del producto y normalmente están vinculados a requisitos funcionales, es decir, una vez que se conozca lo que el sistema debe hacer se puede determinar cómo ha de comportarse, qué cualidades debe tener, cuán rápido o grande debe ser. Existen múltiples categorías para clasificar a los requisitos no funcionales, a continuación se muestran los más destacados en este sistema:

**Usabilidad**

**RNF1** El sistema solo podrá ser utilizado por los usuarios creados por la administración. Cada usuario que se autentique en el sistema solo tendrá acceso a la información que le corresponde según su rol y tipo de usuario.

**RNF2** El sistema debe garantizar un acceso fácil y rápido, podrá ser usado por usuarios con pocos conocimientos informáticos.

**Fiabilidad**

**RNF3** Una vez terminado el Sistema para la Cooperación Médica se realizarán procesos de despliegue, capacitación y mantenimiento de software. El personal que trabaja con el software debe contar con el nivel técnico requerido mediante adiestramiento de servicio.

*Confiabilidad:* La información manejada por el sistema estará protegida de acceso no autorizado.

*Integridad:* La información no podrá ser divulgada. Se harán copias de respaldo que puedan restaurar el sistema en caso de pérdida de información.

*Disponibilidad:* Solo los usuarios autorizados tendrán acceso a la información en todo momento.

**Eficiencia**

**RNF4** El sistema deberá responder rápido ante las solicitudes de los usuarios, por lo que el procesamiento de la información y el tiempo de respuesta deberán ser en el menor tiempo posible.

La eficiencia de la aplicación estará determinada en gran medida por el aprovechamiento de los recursos que se disponen en el modelo Cliente-Servidor, y la velocidad de las consultas a la base de datos. Se realizará la validación de los datos en el cliente y en el servidor, a aquellas que por cuestiones de seguridad o de acceso a los datos lo requieran, lográndose así un tiempo de respuesta más rápido, con mayor velocidad de procesamiento y aprovechamiento de los recursos.

**Soporte**

**RNF5** Se debe acceder al sistema desde cualquier plataforma.

**RNF6** Se debe garantizar que el sistema sea compatible con el resto de los módulos que se realizan.

**RNF7** El sistema debe estar bien documentado de forma tal que el tiempo de mantenimiento sea mínimo en caso de necesitarse.

**Restricciones de diseño**

**RNF8** El sistema informático estará desarrollado sobre una plataforma Web y podrá ser utilizado desde cualquier sistema operativo, recomendándose para su uso Linux.

**RNF9** Se definen además, estándares de diseño y codificación en el departamento de SAS por los cuales debe regirse tanto la implementación como los modelos generados como parte del proceso de desarrollo.

**Requisitos para la documentación de usuarios en línea y ayuda del sistema**

**RNF10** La aplicación estará documentada con una ayuda dirigida a los diferentes tipos de usuarios con que cuenta el sistema para garantizar la utilización del mismo.

**RNF11** Se dispondrá de un Manual de Usuario que indicará cómo interactuar con las funcionalidades del sistema.

**Interfaz**

**RNF12** La interfaz será sencilla, amigable e intuitiva, de fácil navegación por parte del usuario. Estará diseñada para una óptima visualización siendo adaptable a cualquier resolución.

**Interfaces de usuario**

**RNF13** Deben tener claridad y buena organización de la información, permitiendo la interpretación correcta e inequívoca de la misma.

**RNF14** Deben permitir la ejecución de acciones de manera rápida.

**RNF15** Se debe hacer uso de elementos visuales para la selección de información siempre que sea posible para minimizar los posibles errores.

**RNF16** Deben enviar los avisos y mensajes al usuario.

**RNF17** Los errores serán visibles al usuario informando el tipo de error ocurrido.

**Interfaces Hardware**

*Para servidor de aplicación y servidor de base de datos.*

Servidor de aplicación: Equipo en el que se instala el software servidor web, se encargará de atender las peticiones de los usuarios del sistema.

Servidor de base de datos: Equipo en el que se instala el software gestor de base de datos.

**RNF18** Deben tener los siguientes requerimientos de hardware:

Tipo de procesador: Intel Pentium IV o superior.

Velocidad del procesador: 3.00 GHz.

Memoria RAM: 2 GB o superior.

Disco Duro: 80 GB para servidor de aplicación y 160 GB para el de base de datos.

Se requiere tarjeta de red.

*Para estaciones de trabajo.*

**RNF19** Los ordenadores que serán utilizados por los usuarios del sistema para acceder a la aplicación y operar la misma deben tener los siguientes requerimientos de hardware:

Tipo de procesador: Intel Pentium III o superior.

Velocidad del procesador: 2.0 GHz o superior.

Memoria RAM: 512 MB o superior.

Disco Duro: 40 GB.

Se requiere tarjeta de red.

**Interfaz de Software**

**RNF20** Las tecnologías que serán utilizadas en el desarrollo y despliegue de la aplicación serán:

El servidor de aplicaciones debe llevar como sistema operativo Ubuntu Server 10.04.

Sistema gestor de base de datos PostgreSQL 8.3.

Servidor de procesos Supervisor 3.0.

Servidor de caché Memcached 1.4.

Servidor web Apache 2.2.

Lenguaje de programación PHP 5.3.

Flash Player 10.0 o superior para la visualización de las gráficas.

**Interfaces de Comunicación**

**RNF21** Se debe garantizar por parte de la entidad con una infraestructura de comunicación entre el cliente y el servidor.

**Requisitos de Licencia**

No se utilizan licencias en el proyecto.

**Estándares Aplicables**

**RNF22** Para las descripciones de casos de uso, mensajes y avisos que debe emitir el sistema, se deben seguir las pautas de análisis definidas en el CESIM.

**RNF23** Para la implementación del sistema se deberán seguir los estándares de codificación y diseño definidos por el CESIM, fundamentalmente en el caso de los mensajes emitidos por la aplicación.

## 2.4 Modelo del Sistema

El Modelo del Sistema permite que los usuarios y los desarrolladores del software alcancen un entendimiento común relacionado con las capacidades y cualidades que debe poseer el sistema. Cada usuario del sistema será representado mediante un actor. Estos interactúan con el sistema mediante los casos de uso que representan una especificación de secuencia de acciones lógicas, incluyendo variantes que el sistema puede desarrollar y que producen un resultado observable de valor para un actor concreto.

Este modelo es considerado la base fundamental para el análisis, diseño e implementación, por lo que mientras mejor estructurado se encuentre, mucho más sencillo será el proceso de construcción del sistema de software.

### 2.4.1 Actores del Sistema

Los actores del sistema son terceros que no forman parte del sistema sino que interactúan con él, los cuales pueden ser personas o sistemas externos.

| Actor | Descripción |
|---|---|
| Editor | Es el encargado de gestionar todos los codificadores y usuarios del sistema. Además, puede visualizar las trazas de las acciones realizadas por el usuario en la aplicación. |

| | |
|---|---|
| Visualizador | Este usuario solo puede visualizar y buscar los codificadores del sistema, no tiene permisos para modificar la información. |

*Tabla 2: Actores del sistema.*

### 2.4.2 Diagrama de Casos de Uso del Sistema

A continuación se muestra el Diagrama de Casos de Uso del Sistema donde se pueden apreciar las relaciones que se establecen entre el Editor y los casos de uso que realiza.

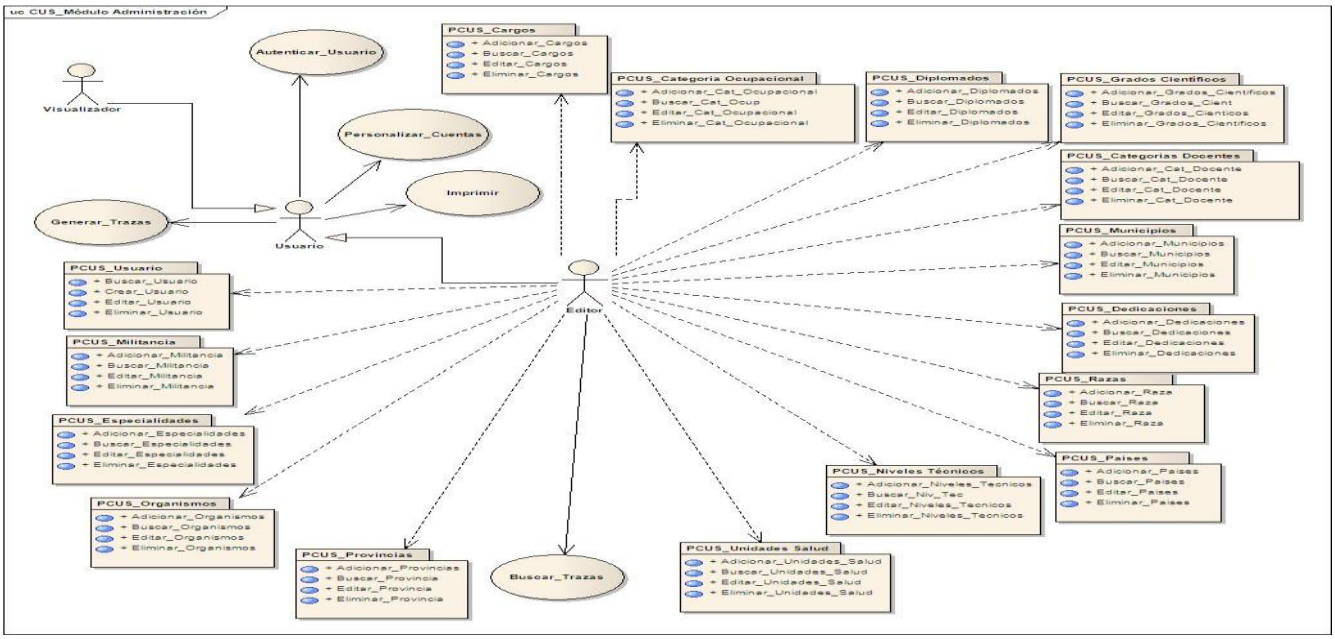


*Figura 3: Diagrama de Casos de Uso del Sistema.*

A continuación se muestra el Diagrama de Casos de Uso del Sistema donde se pueden apreciar las relaciones que se establecen entre el Visualizador y los casos de uso que realiza.

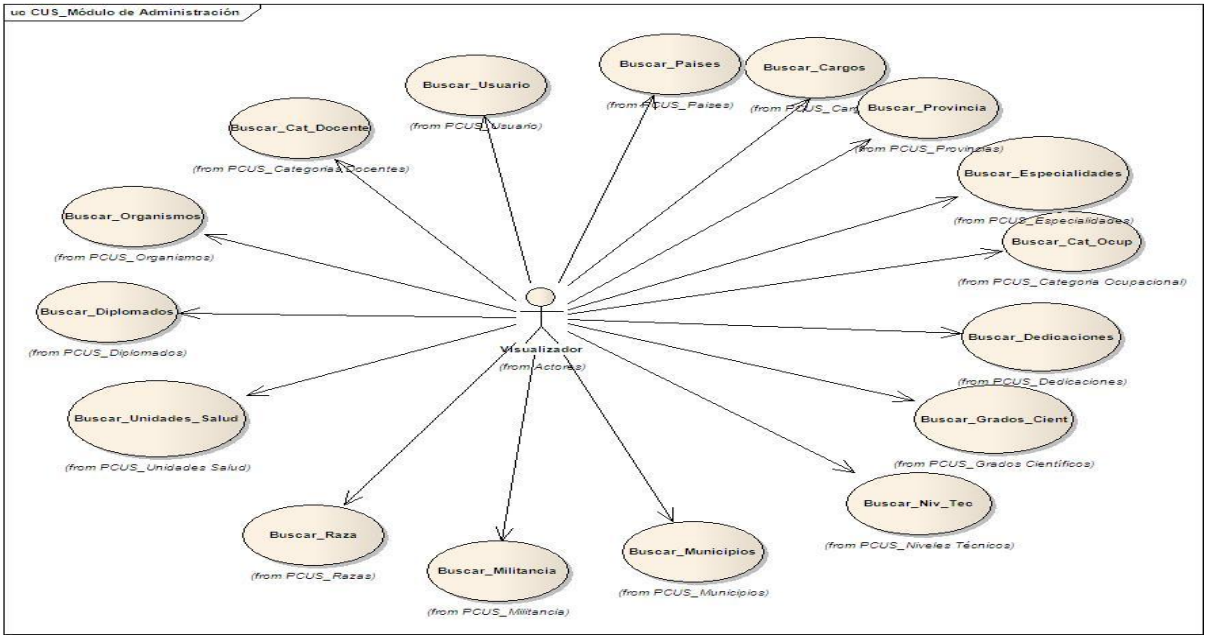


*Figura 4: Diagrama de Casos de Uso del Sistema.*

### 2.4.3 Descripción de los Casos de Uso del Sistema

Un caso de uso es una descripción de la secuencia de interacciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica. (34)

A continuación se realiza la descripción de algunos de los casos de usos arquitectónicamente significativos. Las descripciones restantes serán mostradas en el Expediente de Proyecto en la plantilla de Modelo del Sistema.

| | |
|---|---|
| **Caso de Uso:** | Autenticar Usuario |
| **Actores:** | Usuario |
| **Resumen:** | El caso de uso se inicia cuando el usuario requiere de acceso al sistema, permite que solamente puedan acceder a la aplicación todos aquellos que trabajarán directamente con la misma. Dándole acceso a operar en esta. |
| **Precondiciones:** | El usuario debe tener los permisos necesarios para poder acceder a la aplicación. |

| | |
|---|---|
| **Referencias** | RF1 |
| **Prioridad** | Crítico |
| **Pos condiciones** | El usuario ha sido autenticado correctamente. |

*Tabla 3: Descripción CUS Autenticar Usuario.*

| | |
|---|---|
| **Caso de Uso:** | Crear Usuario |
| **Actores:** | Editor |
| **Resumen:** | El caso de uso inicia cuando el actor accede a la opción Crear un usuario, el sistema brinda la posibilidad de introducir y/o seleccionar los datos para crear el usuario, el actor introduce los datos del usuario, el sistema crea el usuario y el caso de uso termina. |
| **Precondiciones:** | La información debe ser correcta. |
| **Referencias** | RF2 |
| **Prioridad** | Crítico |
| **Pos condiciones** | El usuario ha sido creado correctamente. |

*Tabla 4: Descripción CUS Crear Usuario.*

| | |
|---|---|
| **Caso de Uso:** | Buscar Usuario |
| **Actores:** | Editor |
| **Resumen:** | El caso de uso inicia cuando el actor accede a la opción Buscar usuario, el sistema brinda la posibilidad de introducir criterios de búsqueda para localizar el usuario. El actor introduce los datos para realizar una búsqueda, el sistema busca y muestra el usuario que cumple con los criterios de búsqueda, el caso de uso termina. |
| **Precondiciones:** | La información debe ser correcta. |
| **Referencias** | RF3 |
| **Prioridad** | Crítico |
| **Pos condiciones** | La búsqueda se realizó correctamente. |

*Tabla 5: Descripción CUS Buscar Usuario.*

| | |
|---|---|
| **Caso de Uso:** | Editar Usuario |

| | |
|---|---|
| **Actores:** | Editor |
| **Resumen:** | El caso de uso inicia cuando el actor selecciona un usuario y accede a la opción Editar, el sistema muestra los datos del usuario y brinda la posibilidad de cambiar sus valores ya sea introduciendo nuevos o seleccionando diferentes, el actor modifica los datos que necesita, el sistema actualiza los datos del usuario y el caso de uso termina. |
| **Precondiciones:** | La información debe ser correcta. |
| **Referencias** | RF4 |
| **Prioridad** | Crítico |
| **Pos condiciones** | El usuario ha sido editado. |

*Tabla 6: Descripción CUS Editar Usuario.*

| | |
|---|---|
| **Caso de Uso:** | Eliminar Usuario |
| **Actores:** | Editor |
| **Resumen:** | El caso de uso inicia cuando el actor selecciona un usuario y accede a la opción Eliminar, el sistema elimina el usuario y el caso de uso termina. |
| **Precondiciones:** | La información debe ser correcta. |
| **Referencias** | RF5 |
| **Prioridad** | Crítico |
| **Pos condiciones** | Se elimina el usuario de la lista y la información restante mantiene su integridad y disponibilidad. |

*Tabla 7: Descripción CUS Eliminar Usuario.*

| | |
|---|---|
| **Caso de Uso:** | Generar Trazas |
| **Actores:** | Usuario |
| **Resumen:** | El caso de uso se inicia desde que el usuario se autentica en el sistema y cada vez que realice alguna acción se genera una traza con la información referente al usuario. |
| **Precondiciones:** | La información debe ser correcta. |
| **Referencias** | RF8 |
| **Prioridad** | Crítico |

| | |
|---|---|
| **Pos condiciones** | La traza ha sido generada correctamente. |

*Tabla 8: Descripción CUS Generar Trazas.*

| | |
|---|---|
| **Caso de Uso:** | Buscar Trazas |
| **Actores:** | Editor |
| **Resumen:** | El caso de uso se inicia cuando el editor desea buscar algún dato en específico de alguna acción realizada por el usuario en el sistema. Se introduce el usuario o la fecha para realizar la búsqueda. |
| **Precondiciones:** | La información debe ser correcta. |
| **Referencias** | RF9 |
| **Prioridad** | Crítico |
| **Pos condiciones** | La búsqueda se realizó correctamente. |

*Tabla 9: Descripción CUS Buscar Trazas.*

En este capítulo se realiza la propuesta de solución para llevar a cabo el desarrollo de la aplicación. A partir del análisis detallado de cada uno de los requerimientos funcionales fueron definidas las principales funcionalidades del sistema. La aplicación propuesta contará con varios tipos de usuarios que asumirán diferentes roles, identificados en el diagrama de actores del sistema. Para que el sistema funcione adecuadamente debe cumplir con los requerimientos de software y hardware planteados.

# Capítulo 3: Análisis y Diseño del Sistema

Con el desarrollo de este capítulo se profundiza en el análisis y diseño del sistema, se tienen como objetivos fundamentales: transformar los requerimientos definidos a una propuesta de diseño que será la guía a seguir para la implementación del sistema, evolucionar hacia una arquitectura del software robusta y flexible ante la aplicación de nuevos cambios, y realizar la propuesta de interfaz de usuario del software adaptando al diseño para que coincida con el ambiente de implementación.

Durante el diseño, se toman decisiones estratégicas y tácticas para cumplir los requerimientos funcionales y no funcionales del sistema en cuestión. Es la parte del proceso de desarrollo de software cuyo propósito primario es decidir cómo la implementación del sistema se llevará a cabo. Para lograr lo antes expuesto se realizan artefactos, tales como: los Diagramas de Interacción y de Clases del Diseño, y la Descripción de la Arquitectura del Sistema. Se construye además el Modelo de Despliegue, donde se representa la estructura física del sistema.

## 3.1 Descripción de la arquitectura

La arquitectura de software es la organización fundamental de un sistema. Se representa a través de sus componentes, las relaciones entre ellos, el ambiente y los principios que orientan su diseño y evolución. Se concentra en requerimientos no funcionales, que son satisfechos mediante los modelos y diseños de la aplicación. Un diseño correcto de la arquitectura del sistema es esencial para el éxito o fracaso del proyecto. La arquitectura asocia las capacidades del sistema especificadas en el requerimiento con los componentes del sistema que habrán de implementarla.

La base arquitectónica de COLABEN se ha modelado haciendo uso del patrón MVC, fundamentado en el capítulo 1, con el objetivo de utilizar la separación de las responsabilidades de cada una de las capas que lo conforman y así lograr facilidades de desarrollo. Además se aplicarán patrones de diseño, pues conducen hacia una arquitectura más pequeña, simple y comprensible.

La arquitectura pasa por todos los flujos de trabajo, se va perfeccionando y refinando a medida que se vayan realizando tareas particulares en cada flujo; es importante porque representa distintas vistas

que inciden en la implementación futura del sistema. Es el esqueleto o base de una aplicación, en esta se analiza la aplicación desde varios puntos de vista.

El uso del framework Symfony propone construir el software bajo este patrón. Brindándole al sistema la posibilidad de tener una separación clara entre cómo se muestra la información al usuario, cómo se manejan las acciones que el usuario desea hacer sobre el sistema y cómo se realizan estas acciones modificando y validando la información.

Además, se utilizan dos patrones de diseño de gran importancia, los cuales son:

- ✓ El Factory: Consiste en la definición de una clase que realiza una determinada tarea. Symfony utiliza las factorías en su funcionamiento interno, como por ejemplo, para los controladores y las sesiones. Cuando el framework necesita crear un nuevo objeto para una petición, busca en la definición de la factoría el nombre de la clase que se debe utilizar para esta tarea. La principal ventaja de utilizar las definiciones de las factorías es que se hace muy sencillo modificar las características internas. (35)
- ✓ El Decorator: Es la unión del archivo llamado layout.php con las plantillas, donde el primero almacena el código HTML que es común para todas las páginas de la aplicación, y el segundo contiene los formularios y datos específicos de cada interfaz.

En las aplicaciones web, el Controlador, generalmente se encuentra muy cargado, ya que es el encargado de las tareas comunes como el manejo de las peticiones del usuario, el manejo de la seguridad, cargar la configuración de la aplicación y otras similares.

Por tal motivo, el Controlador en Symfony normalmente se divide en un Controlador Frontal, que es único para cada aplicación, cuya principal función es re-direccionar las peticiones realizadas por la Vista hacia las Acciones, que incluyen el código específico del controlador de cada página. Una de las principales ventajas de utilizar un controlador frontal es que ofrece un punto de entrada único para toda la aplicación. Así, en caso de que sea necesario impedir el acceso a la aplicación, solamente es necesario editar el script correspondiente al controlador frontal. Si la aplicación no dispone de controlador frontal, se deberían modificar cada uno de los controladores.

En la Vista también se puede aprovechar la separación del código, por lo que Symfony la separa en capa externa (layout), plantilla (templates) y lógica de la vista. En la capa externa se agrupa la parte de la Vista, que permanece invariable para todas o parte de las páginas de la aplicación, mientras que las plantillas solo se encargan de visualizar los formularios y variables definidas en el controlador. En los archivos de la lógica se definen los elementos de la configuración de la Vista. En esta capa se utiliza además, la librería ExtJS.

La capa del modelo se divide en la capa de Acceso a los Datos y en la Capa de Abstracción de la Base de Datos. La primera contiene todas las funcionalidades que responden a la lógica del negocio y la segunda provee una abstracción de la base de datos que posibilita que la aplicación no dependa de un SGBD determinado. De esta forma, las funciones que acceden a los datos no utilizan sentencias ni consultas que dependen de una base de datos, sino que utilizan otras funciones para realizar las consultas. Así, si se cambia de SGBD que en este caso es el PostgreSQL 8.3, solamente es necesario actualizar la capa de abstracción de la base de datos, ya que el acceso a datos se realiza mediante el uso del Mapeo de Objetos Relacional (ORM) Doctrine, que abstrae al sistema del uso de cualquier SGBD, mediante el uso de la programación orientada a objeto y haciendo posible tratar las tablas como objetos del sistema.

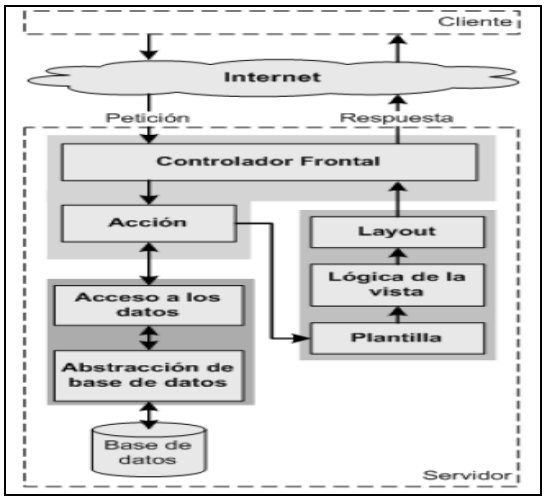


*Figura 5: Implementación del patrón MVC según el Framework Symfony.* (36)

De esta forma cuando el usuario interactúa con la aplicación lo hace mediante la Vista, que envía las peticiones al Controlador Frontal, que las re-direcciona hacia la Acción correspondiente, encargándose esta de llamar a la función implementada en el Modelo, donde se obtiene la respuesta, que es enviada a través del Controlador hacia la Vista.

Otros patrones utilizados fueron los llamados Patrones Generales de Software para Asignar Responsabilidades (GRASP, por sus siglas en inglés), que tuvieron una importante utilidad en el diseño realizado.

A cada clase le fueron asignadas las tareas que podían realizar según la información que poseían, además de crear las instancias de otras clases en correspondencia con la responsabilidad dada, poniéndose de manifiesto los patrones Experto y Creador. De esta manera, los objetos logran valerse de la información que contienen para realizar las tareas que se les pide, además de utilizar el patrón Creador para definir quién será el responsable de crear una nueva instancia de una clase. Este diseño obtenido cumple con los patrones de Bajo Acoplamiento y de Alta Cohesión facilitando así la centralización de actividades de cada elemento, y logrando que estos realicen una única tarea dentro del sistema.

## 3.2 Análisis

Se realiza un análisis de los requerimientos que fueron descritos en la captura de requisitos, refinándolos y estructurándolos. El objetivo de hacerlo es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que ayude a estructurar todo el sistema, incluyendo su arquitectura.

### 3.2.1 Modelo de Análisis

El Modelo de Análisis, que en realidad es una serie de modelos, es la primera representación técnica de un sistema. Utiliza una combinación de formatos en textos y diagramas para representar los requisitos de los datos, las funciones y el comportamiento de una manera que es relativamente fácil de entender y, aún más importante, conduce a una revisión para lograr la corrección, la integridad y la consistencia. (37)

Aunque en el modelo del análisis hay un refinamiento de los requisitos, no se tiene en cuenta el lenguaje de programación a usar en la construcción, la plataforma en la que se ejecutará la aplicación, los componentes prefabricados o reusables de otras aplicaciones, entre otras características que afectan al sistema, ya que el objetivo del análisis es comprender perfectamente los requisitos del software y no precisar cómo se implementará la solución.

### 3.2.2 Clases de Análisis

Se centran en los requisitos funcionales y son evidentes en el dominio del problema porque representan conceptos y relaciones del dominio. Tienen atributos y entre ellas se establecen relaciones de asociación, agregación / composición, generalización / especialización y tipos asociativos. RUP propone clasificar a las clases en:

- ✓ "Entidad: Modelan información que posee larga vida y que es a menudo persistente.
- ✓ Interfaz: Modelan la interacción entre el sistema y sus actores.
- ✓ Control: Coordinan la realización de uno o unos pocos casos de uso coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso." (38)

### 3.2.3 Diagrama de Clases del Análisis

Un Diagrama de Clases del Análisis es un artefacto en el que se representan los conceptos en un dominio del problema. Representa el funcionamiento del mundo real, no de la implementación automatizada del mismo. (39) A continuación se muestran los diagramas de clases del análisis de algunos casos de uso del sistema, los restantes se encuentran en el Expediente de Proyecto.

**CUS Autenticar Usuario**

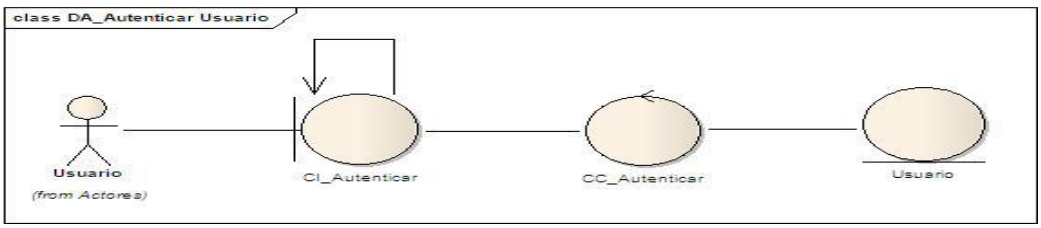


*Figura 6: Diagrama de Clases del Análisis del CUS Autenticar Usuario.*

**CUS Crear Usuario**

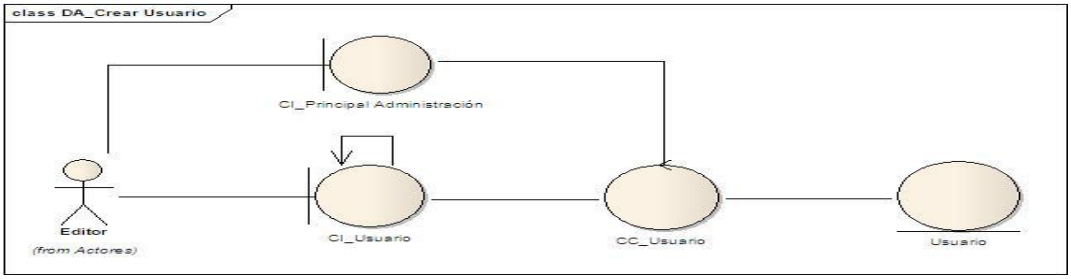


*Figura 7: Diagrama de Clases del Análisis del CUS Crear Usuario.*

**CUS Editar Usuario**

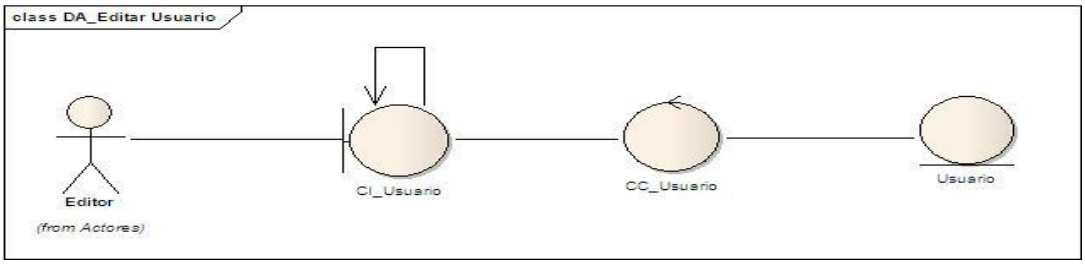


*Figura 8: Diagrama de Clases del Análisis del CUS Editar Usuario.*

**CUS Eliminar Usuario**

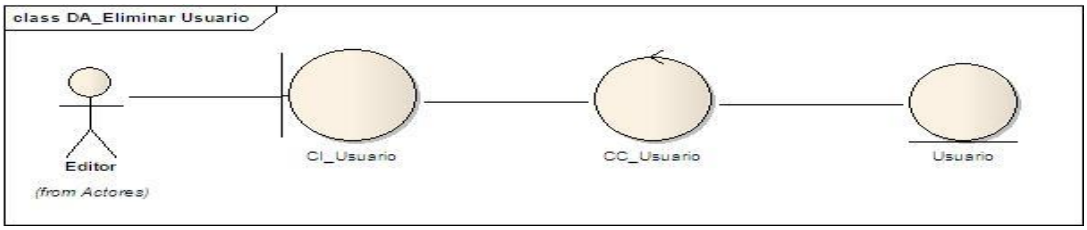


*Figura 9: Diagrama de Clases del Análisis del CUS Eliminar Usuario.*

**CUS Personalizar Cuenta**

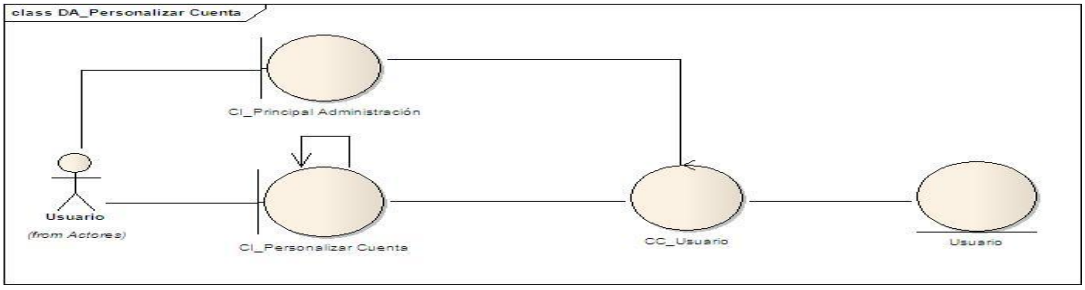


*Figura 10: Diagrama de Clases del Análisis del CUS Personalizar Cuenta.*

**CUS Buscar Usuario**

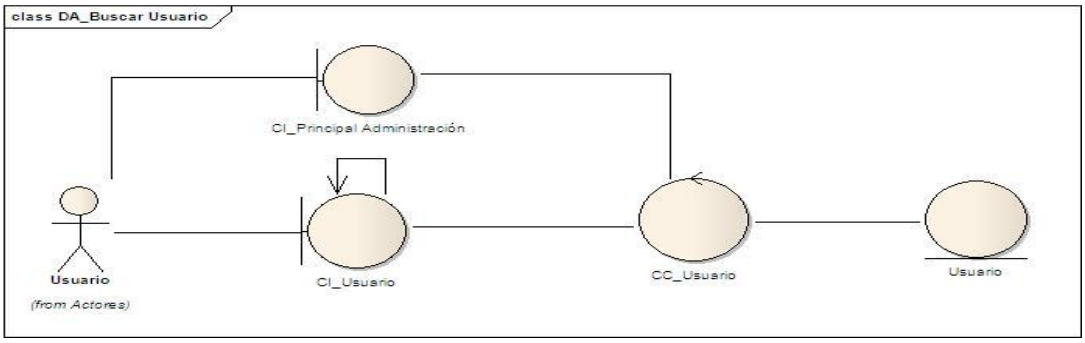


*Figura 11: Diagrama de Clases del Análisis del CUS Buscar Usuario.*

**CUS Buscar Trazas**

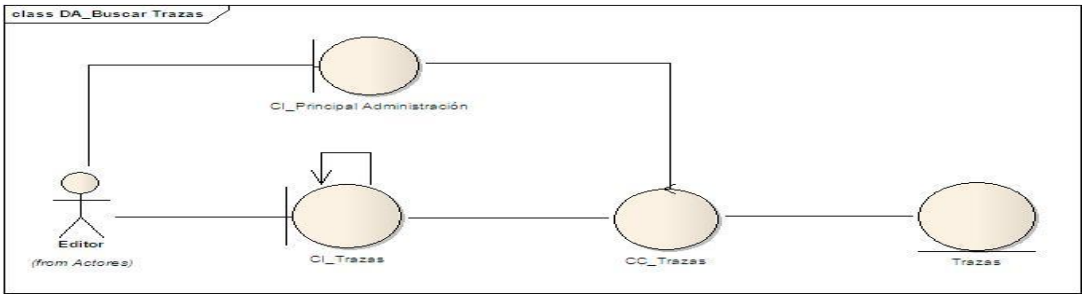


*Figura 12: Diagrama de Clases del Análisis del CUS Buscar Trazas.*

## 3.3 Diseño

El diseño crea una representación o modelo del software, pero a diferencia del modelo de análisis, proporciona detalles acerca de las estructuras de los datos, las arquitecturas, las interfaces y los componentes del software que son necesarios para implementar el sistema. El diseño permite al ingeniero de software modelar el sistema o producto que se va a construir.

En el diseño se modela el sistema y se encuentra la forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. Una entrada esencial en el diseño es el resultado del análisis, o sea el modelo de análisis, que proporciona una comprensión detallada de los requisitos. Además impone una estructura del sistema que se debe conservar lo más fielmente posible cuando se le dé forma al sistema. (40)

### 3.3.1 Modelo de Diseño

Es el principal artefacto utilizado como entrada para el correcto desarrollo de la fase de implementación. Como parte del mismo se construyen los Diagramas de Interacción y Diagramas de Clases del Diseño.

### 3.3.2 Diagramas de Clases del Diseño

Describen gráficamente las especificaciones de las clases del software, los elementos básicos que se pueden encontrar son las clases y sus relaciones. (41)

A continuación se muestran los Diagramas de Clases del Diseño de algunos CUS. Los diagramas restantes serán mostrados en el Expediente de Proyecto en la plantilla de Modelo de Diseño.

**CUS Autenticar Usuario**

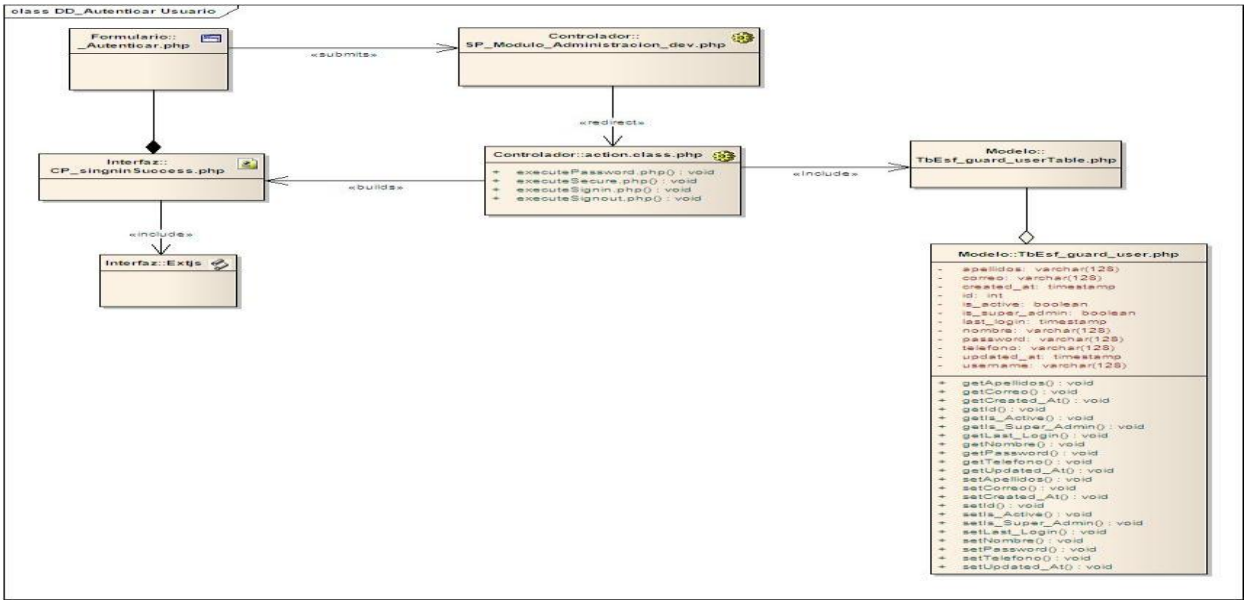


*Figura 13: Diagrama de Clases del Diseño del CUS Autenticar Usuario.*

**CUS Crear Usuario**

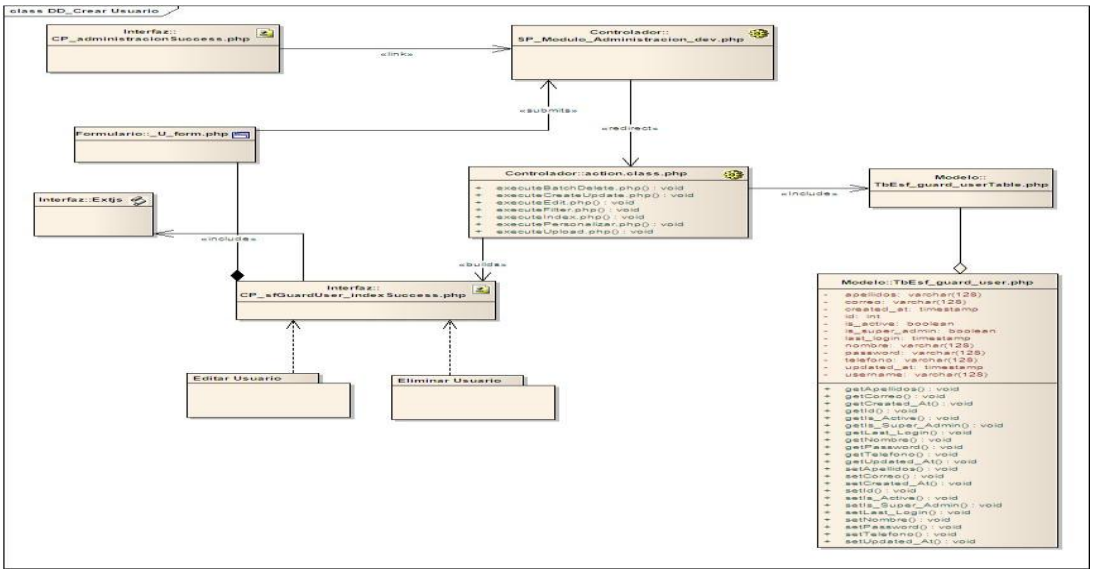


*Figura 14: Diagrama de Clases del Diseño del CUS Crear Usuario.*

**CUS Editar Usuario**

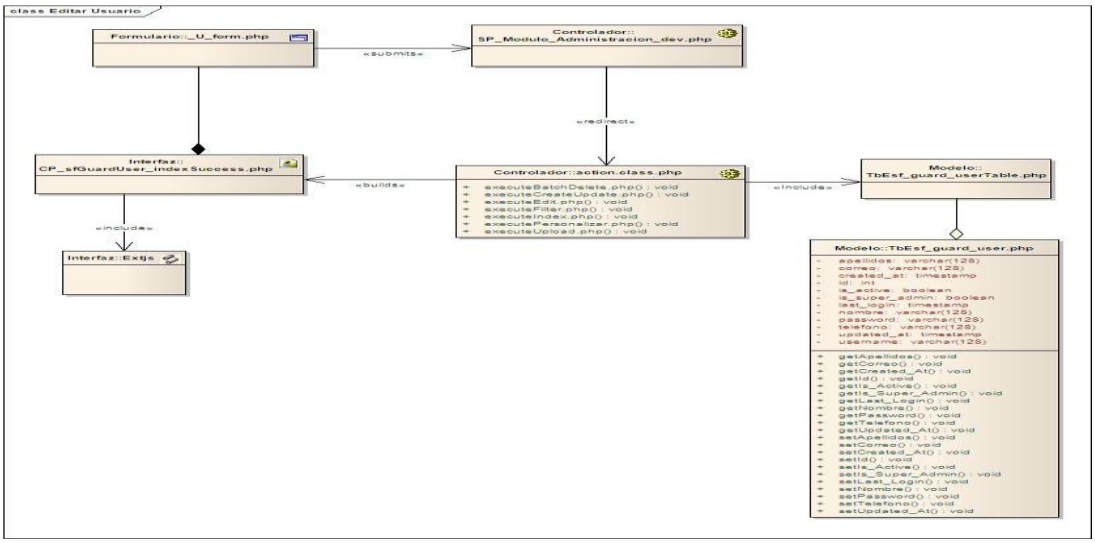


*Figura 15: Diagrama de Clases del Diseño del CUS Editar Usuario.*

**CUS Buscar Usuario**

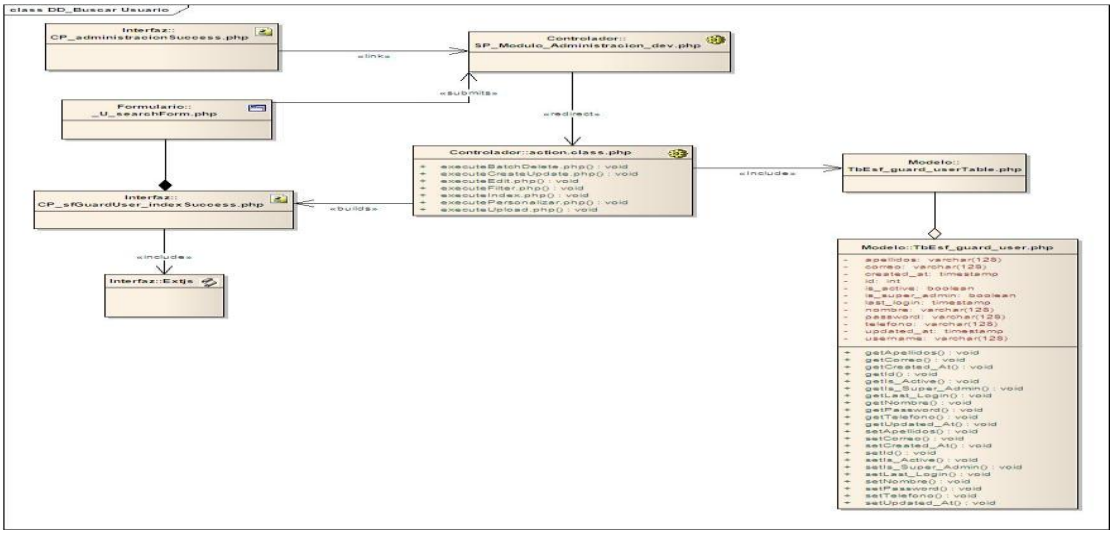


*Figura 16: Diagrama de Clases del Diseño del CUS Buscar Usuario.*

**CUS Eliminar Usuario**

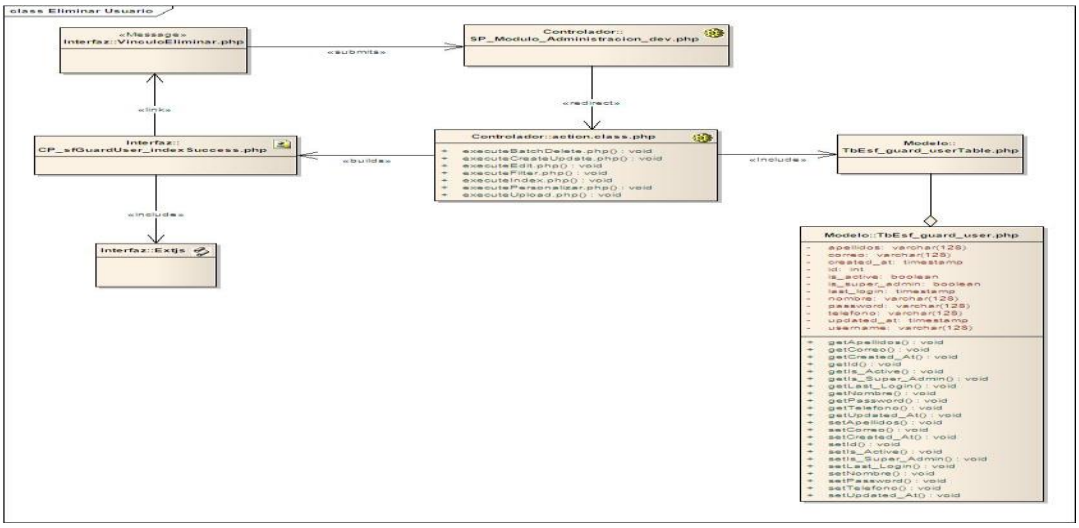


*Figura 17: Diagrama de Clases del Diseño del CUS Eliminar Usuario.*

**CUS Buscar Trazas**

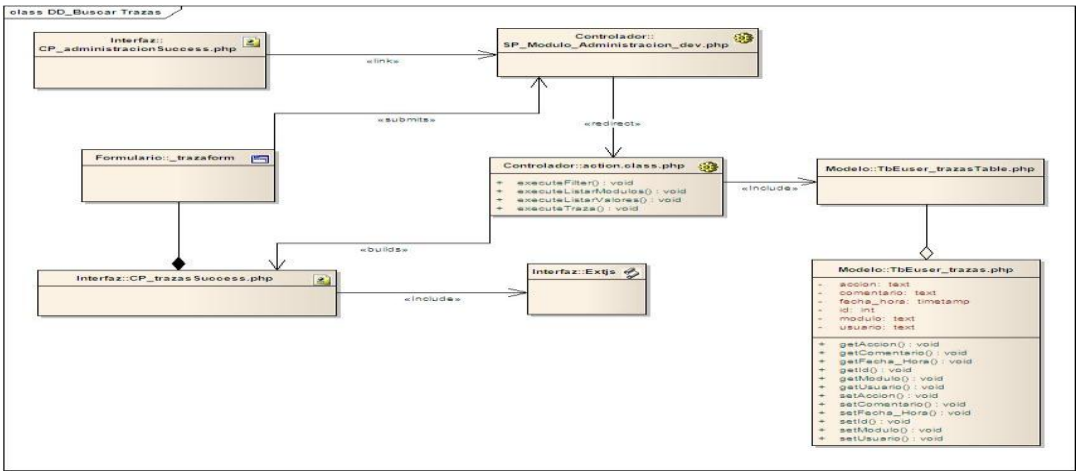


*Figura 18: Diagrama de Clases del Diseño del CUS Buscar Trazas.*

### 3.3.3 Descripción de las Clases del Diseño

**Clase CP_client page**

El propósito de estas clases es representar el acceso del usuario al sistema, se encargan de mostrar todos los formularios, permiten al usuario realizar las peticiones e interactuar con la aplicación. A continuación se muestra un ejemplo de una de ellas ya que todas se comportan de igual manera.

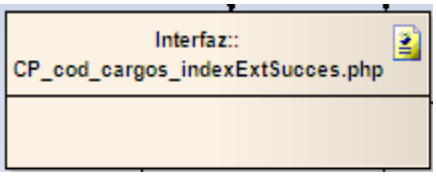


*Figura 19: Módulo de Administración. CP_cod_cargos_indexExtSucces.php*

Descripción

Esta clase tiene como principal objetivo representar el acceso del usuario a la interfaz del codificador cargo. Se encarga de mostrar los formularios correspondientes y permite al usuario interactuar con las funcionalidades referentes a la gestión de cargos.

**Formularios**

Las clases formularios representan la entrada y salida de datos en el sistema. Seguidamente se muestra un ejemplo de la descripción y el propósito de un formulario, debido a que todos los formularios se comportan de manera similar.


*Figura 20: Módulo de Administración. _Autenticar.php*

Descripción

El propósito de esta clase es representar cómo se lleva a cabo la inserción de los datos para autenticarse en el sistema. Estos datos son primeramente validados y luego enviados a la clase controladora correspondiente.

**SP Actions Class**

Las clases actions son las encargadas de procesar las peticiones de los usuarios y de realizar los cambios apropiados en el modelo o en la vista. A continuación se muestra un ejemplo de una de ellas, debido a que todas se comportan de manera similar.

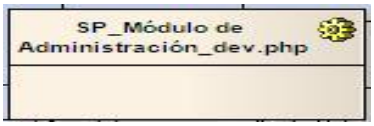


*Figura 21: Módulo de Administración. SP_Módulo de Administración_dev.php.*

Descripción

El propósito de esta clase es representar la página controladora del sistema, está vinculada a la gestión de todo el código y re-direcciona las peticiones a las clases controladoras de cada página.

### 3.3.4 Diagramas de Interacción

Un Diagrama de Interacción muestra una interacción, que consiste en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos. Los Diagramas de Interacción se utilizan para modelar los aspectos dinámicos de un sistema. (42)

**Diagramas de Comunicación**

Un Diagrama de Comunicación es un Diagrama de Interacción que destaca la organización estructural de los objetos que envían y reciben mensajes. (43)

A continuación se muestran los Diagramas de Comunicación de algunos casos de uso del sistema. Los diagramas restantes se encuentran en el Expediente de Proyecto.

**CUS Autenticar Usuario**

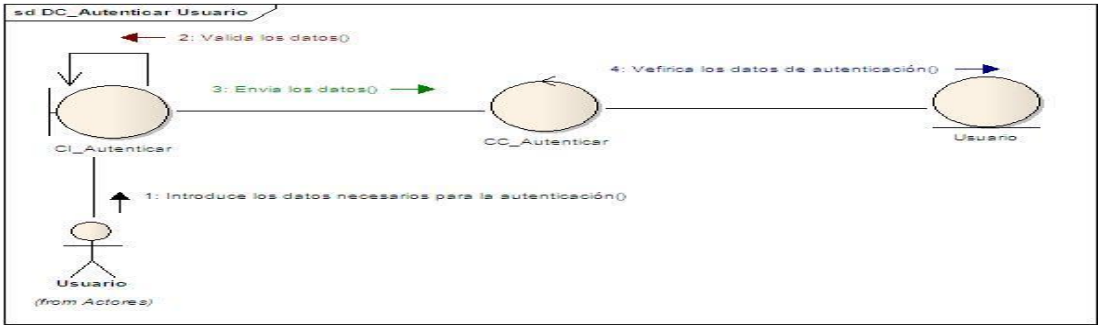


*Figura 22: Diagrama de Comunicación del CUS Autenticar Usuario.*

**CUS Crear Usuario**

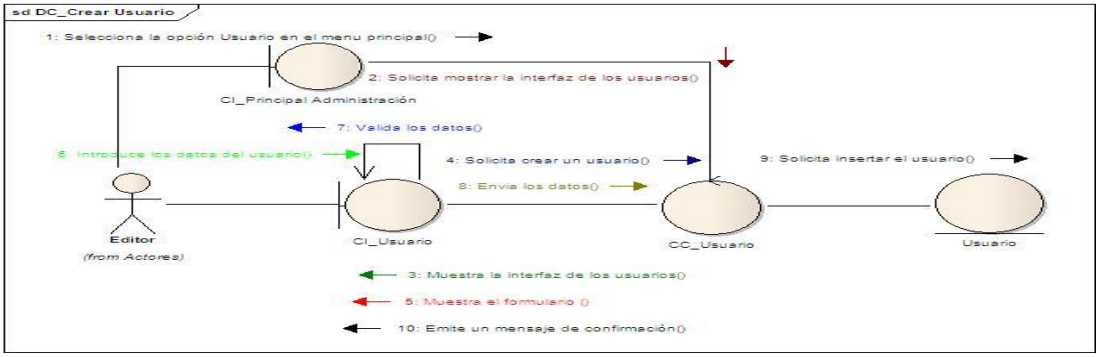


*Figura 23: Diagrama de Comunicación del CUS Crear Usuario.*

**CUS Editar Usuario**

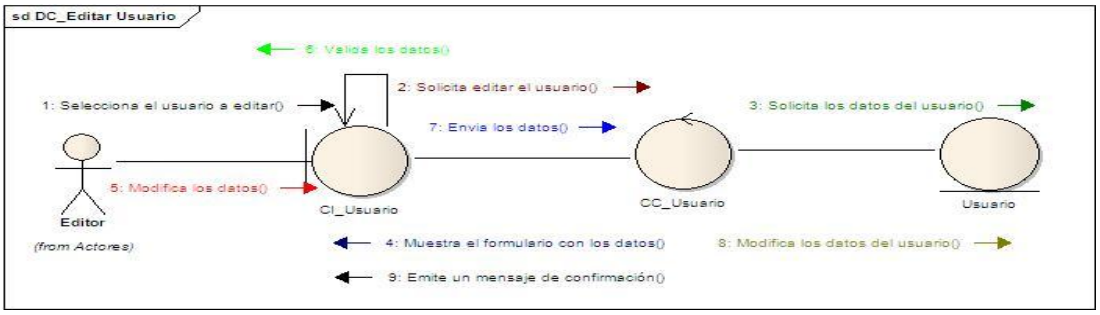


*Figura 24: Diagrama de Comunicación del CUS Editar Usuario.*

**CUS Buscar Usuario**

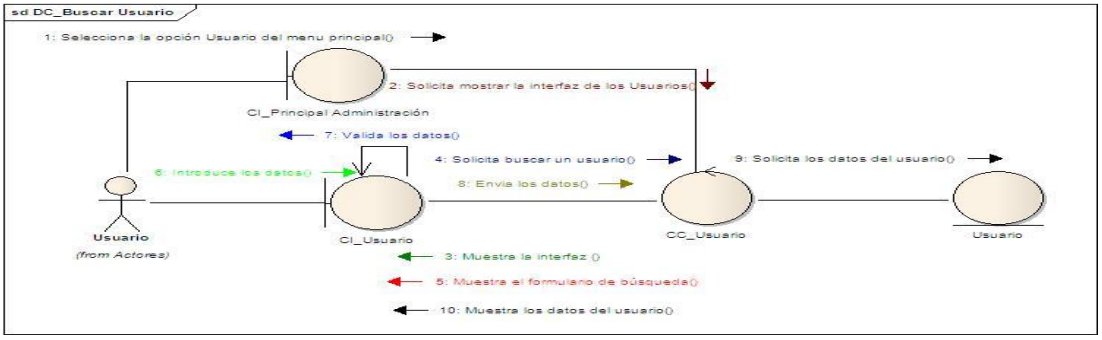


*Figura 25: Diagrama de Comunicación del CUS Buscar Usuario.*

**CUS Eliminar Usuario**

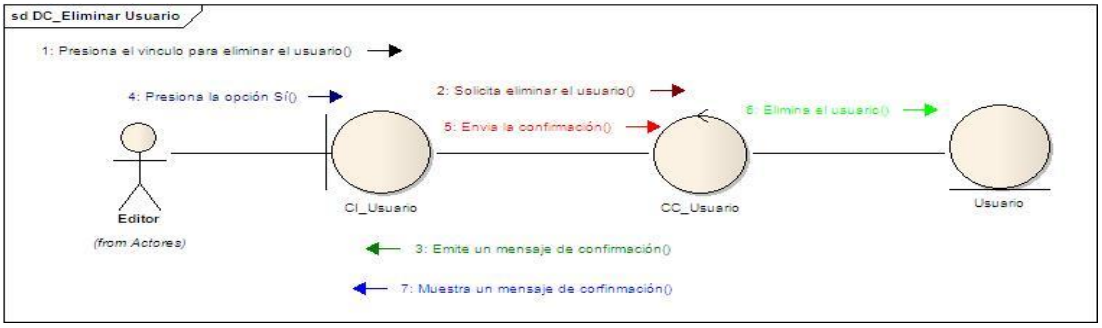


*Figura 26: Diagrama de Comunicación del CUS Eliminar Usuario.*

**CUS Buscar Trazas**

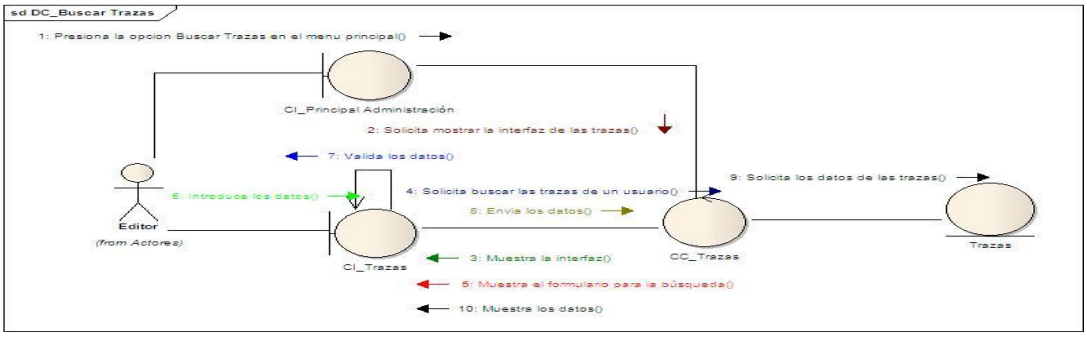


*Figura 27: Diagrama de Comunicación del CUS Buscar Trazas.*

### 3.3.5 Diagrama de Despliegue

Es un diagrama que muestra la configuración de los nodos que participan en la ejecución y de los componentes que residen en ellos. Es utilizado para capturar los elementos de configuración del procesamiento y las conexiones entre esos elementos. También se utiliza para visualizar la distribución de los componentes de software en los nodos físicos. (44)

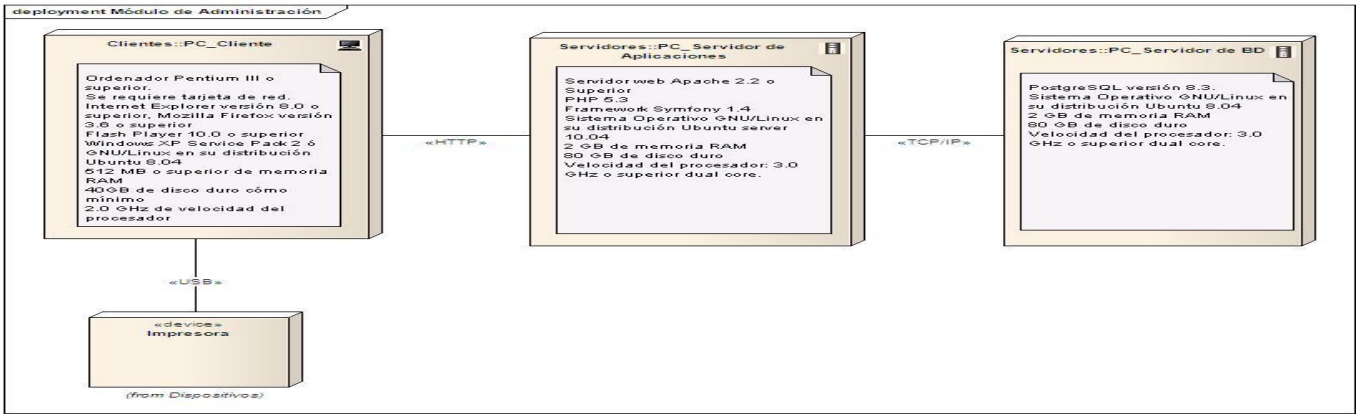


*Figura 28: Diagrama de Despliegue.*

Con el desarrollo del este capítulo se logró modelar el sistema de manera que soporte los requisitos funcionales y no funcionales. A partir del modelado de los diagramas de análisis, diseño e interacción, se muestra la estructura del funcionamiento del sistema internamente. Se representa la estructura física a partir del Modelo de Despliegue. Se realizó la descripción de la arquitectura del software de acuerdo a las características y estructura propuestas por el Framework Symfony.

# Capítulo 4: Implementación

El trabajo en este capítulo parte del resultado del análisis y diseño obtenidos en el capítulo anterior y se procederá a implementar las clases y subsistemas en términos de componentes (ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares). Además, se explicarán las estrategias de codificación, tratamiento de errores y seguridad. Se describe el Modelo de Datos sobre el cual se va a realizar la implementación.

## 4.1 Modelo de Datos

Un Modelo de Datos es básicamente, una descripción de un contenedor de datos, así como de los métodos para almacenar y recuperar información de esos contenedores. Los modelos de datos no son elementos físicos, son abstracciones que permiten la implementación de un sistema eficiente de base de datos. (45)

Para el funcionamiento del sistema se diseñó un Modelo de Datos de clases persistentes que describe la información que se manejará en el software. A continuación se muestra el Modelo de Datos definido para describir la representación lógica y física de la información persistente que se maneja en el Módulo de Administración.

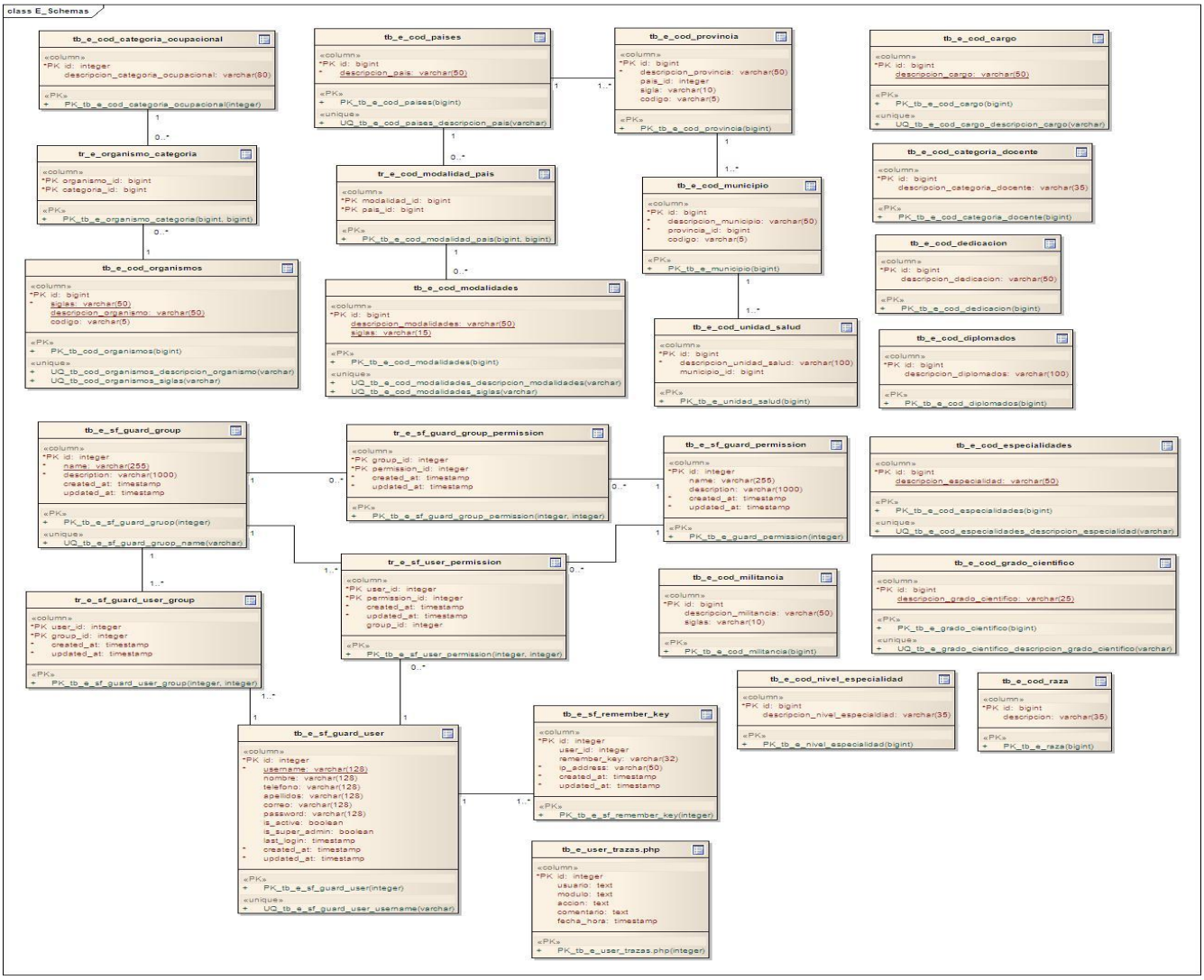


*Figura 29: Modelo de Datos.*

## 4.2 Modelo de Implementación

El Modelo de Implementación describe cómo los elementos del modelo de diseño, como las clases, se implementan en términos de componentes, como ficheros de código de fuentes y ejecutables. El Modelo de Implementación describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros. (46)

### 4.2.1 Diagrama de Componentes

El Diagrama de Componentes muestra la relación entre componentes de software, sus dependencias, su comunicación, su ubicación y otras condiciones. Teniendo en cuenta la utilización del MVC y la división de los archivos y funcionalidades que implementa el Symfony, se han definido tres paquetes de componentes generales: la Vista, el Controlador y el Modelo, donde se agrupan otros componentes más pequeños en dependencia de las funciones que realizan.

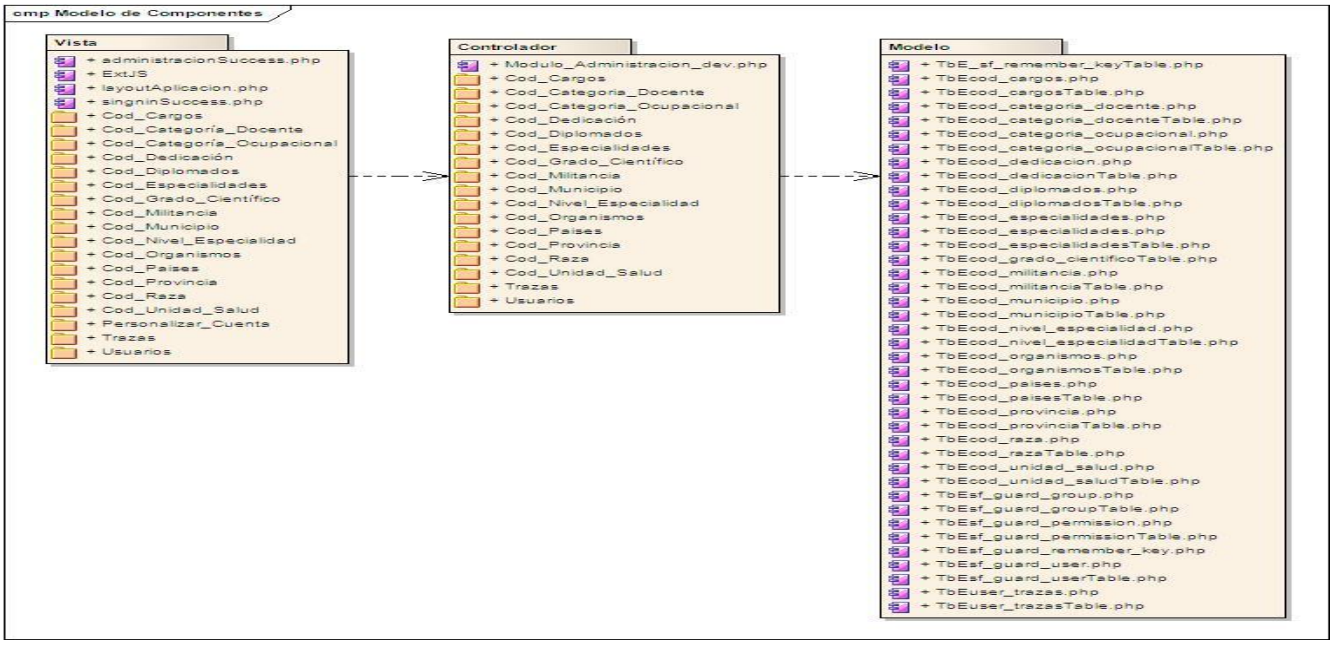


*Figura 30: Diagrama General de Paquetes de Componentes.*

Para una mejor visualización de los componentes, se muestran a continuación los diagramas de componentes referentes a la Vista y al Controlador. El resto de los diagramas, así como cada uno de los paquetes en detalle, se pueden encontrar en el Expediente del Proyecto.

En la Vista se encuentran agrupados todos los formularios y las clases Success que representan la forma en que el usuario puede interactuar con la aplicación.

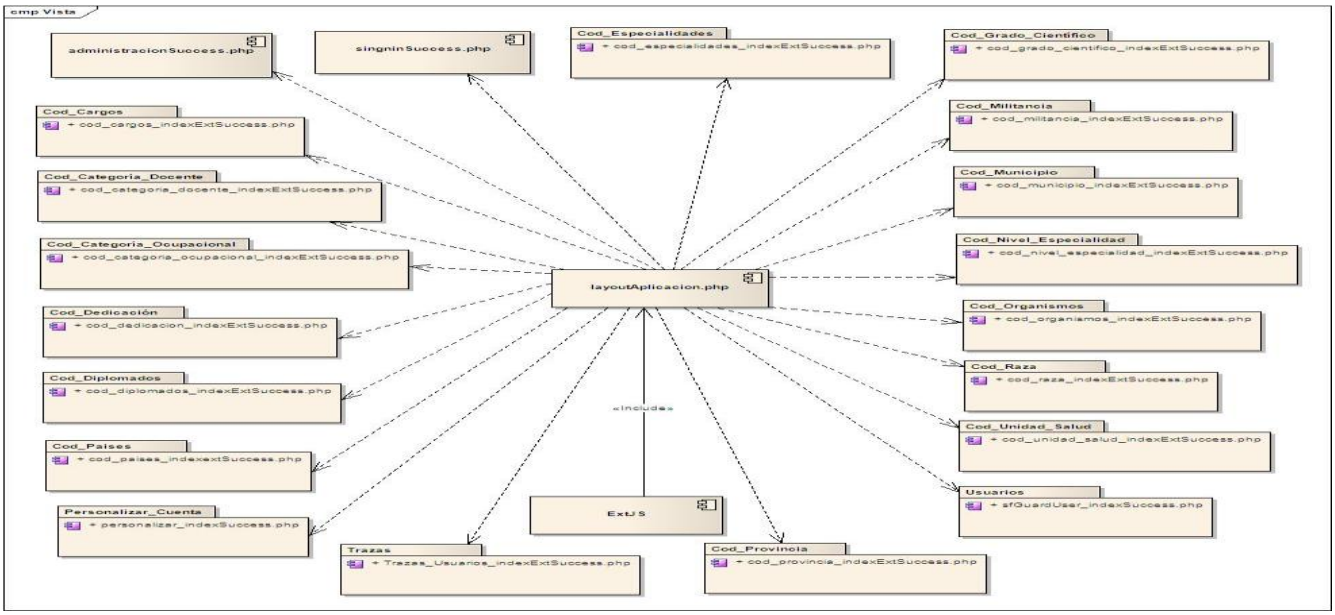


*Figura 31: Diagrama General de Componentes de la Vista.*

En el Controlador se encuentran agrupados el Controlador Frontal y las clases Action de cada uno de los módulos de la aplicación, en los cuales se gestionan las peticiones y se realizan los cambios apropiados en el Modelo o en la Vista.

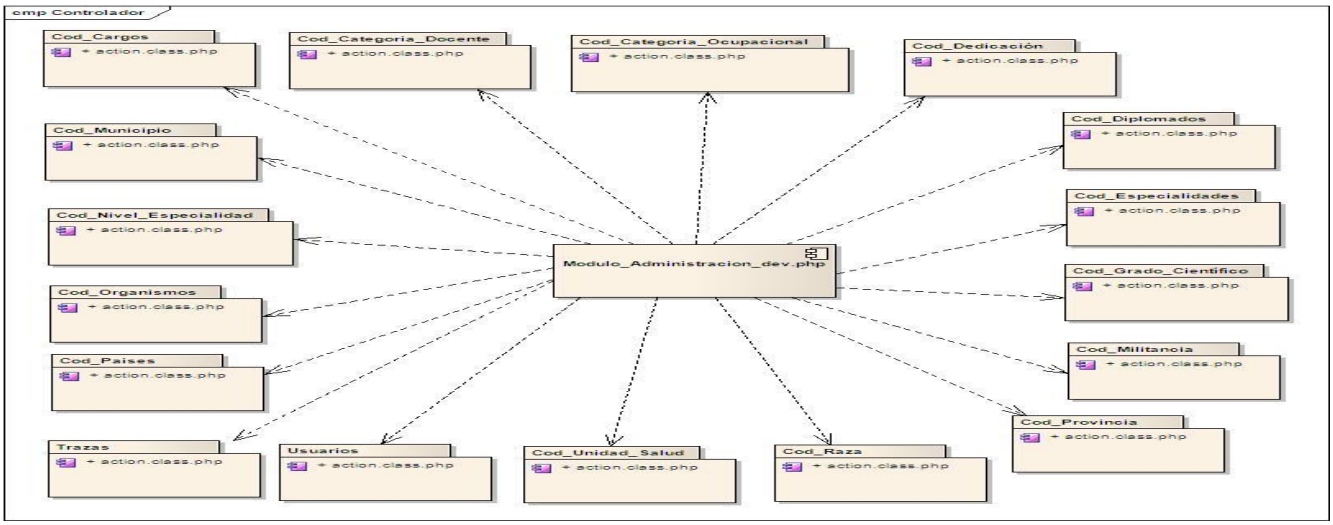


*Figura 32: Diagrama General de Componentes del Controlador.*

## 4.3 Tratamiento de errores

El tratamiento de errores es un paso fundamental para obtener un sistema óptimo, debido a que garantiza la integridad y confidencialidad de la información que se maneja en él. Para minimizar estos errores uno de los métodos utilizados fue el trabajo con el lenguaje de programación JavaScript, ya que mediante el mismo se le informan al usuario muchos de los errores ocurridos del lado del cliente.

En este módulo serán emitidos los mensajes de errores en el momento en que el usuario cometa alguna equivocación, por ejemplo: que no llene todos los campos, que introduzca los datos erróneamente, entre otros.

Además se realizan validaciones de los datos del lado del servidor.

## 4.4 Seguridad

La seguridad e integridad del sistema se garantizan una vez que se hayan creado los usuarios con los permisos necesarios para la gestión de la información en la base de datos del sistema. A partir de los roles definidos para el sistema, solo acceden a cada módulo de la aplicación aquellos usuarios definidos previamente por un administrador. Además, el sistema gestor de base datos debe mantenerse en un lugar restringido y asegurado, se deben realizar salvas continuas para evitar las pérdidas de datos que hayan sido insertados en el sistema con anterioridad.

Para añadir funciones de autorización, seguridad y autenticación a las funciones de seguridad estándar de Symfony se utiliza el plugin sfGuardPlugin. Este brinda el modelo (usuario, grupo y objetos de permisos) y los módulos (backend y frontend) para asegurar la aplicación en un minuto de configuración. (47)

En Symfony el acceso a las action puede ser restringido en el archivo security.yml de tal manera que si un usuario pide realizar una action que está asegurada, este tiene que estar autenticado y tener las credenciales correctas. El manejo de sesiones se hace a través del objeto sfUser este es un objeto singlenton y puede ser accedido desde cualquier parte de la aplicación. Las clases secure y login action, serán redefinidas para usar el sfGuardPlugin. (48)

Una vez instalado el plugin se agregan 7 tablas a la base de datos:

- ✓ **sf_guard_user:** Es la tabla con todos los usuarios, tiene username, password, etc.
- ✓ **sf_guard_permission:** Es la tabla con todos los permisos, incluye permissionname y description.
- ✓ **sf_guard_user_permission:** Es la tabla que relaciona de muchos a muchos 'user' y 'permission', es decir un usuario puede tener múltiples permisos y un permiso puede ser compartido por múltiples usuario.
- ✓ **sf_guard_group:** Es la tabla que define grupos de usuarios.
- ✓ **sf_guard_user_group:** Es la tabla que lista los usuarios de un grupo, un usuario puede ser parte de múltiples grupos.
- ✓ **sf_guard_group_permission:** Es la tabla que relaciona de muchos a muchos 'group' y 'permission'.
- ✓ **sf_guard_remember:** almacena la ip-address y remember-key de los users.

En este capítulo se reflejó la estructura organizativa que tienen todas las clases y componentes del sistema, lo que permite un mejor entendimiento del módulo desarrollado. Luego de realizar un análisis de la importancia que representa el tratamiento de errores y la seguridad, para la construcción de la aplicación, se logró obtener un módulo con todas las funcionalidades previstas que satisface las principales necesidades de los clientes.

# Conclusiones

Con la elaboración del presente trabajo de diploma, se ha cumplido con el objetivo general propuesto, por lo que se puede concluir que:

- ✓ Con la asimilación de las tecnologías y arquitectura definidas por el Departamento SAS y a partir del desarrollo enfocado al trabajo con el Framework Symfony, se potencia la reutilización de código y un fácil mantenimiento del sistema en el futuro.
- ✓ Se desarrolló un producto funcional que cumple con los requerimientos de seguridad necesarios para garantizar la confidencialidad de la información que se manipula en todo el sistema.

# Recomendaciones

Con el objetivo de enriquecer la investigación realizada se recomienda para el desarrollo de futuras versiones del sistema:

- ✓ Permitir dentro de la gestión de trazas que se puedan deshacer los cambios realizados por un usuario determinado.

# Referencias Bibliográficas

1. Informática en la medicina. [En línea] [Citado el: 20 de Mayo de 2010.] http://informaticaenlamedicina.blogspot.com.

2. Las TIC en la salud (inglés). [En línea] [Citado el: 20 de Mayo de 2010.] http://www.infodev.org/en/Project.38.html.

3. Historia Colaboración Médica en Cuba. [En línea] [Citado el: 20 de Mayo de 2010.] http://www.sld.cu/sitios/bmn/temas.php?idv=12389.

4. Concepto de SOFTEL. [En línea] [Citado el: 22 de Noviembre de 2010.] http://www.ecured.cu/index.php/Softel.

5. Concepto de CESIM. [En línea] [Citado el: 22 de Noviembre de 2010.] http://www.fec.uh.cu/CUGIO/1%20acciones/Proyectos-Protocolos/Velmour%20Munos %20Casals.pdf.

6. Software Integral SGAP. [En línea] [Citado el: 30 de Octubre de 2010.] http://www.integralsoftware.com.ar/?s=sgap.

7. Sistema de Gestión de Usuarios. [En línea] [Citado el: 30 de Octubre de 2010.] http://www.eventos.bvsalud.org/EDRedes1/public/documents/SGU%20e.

8. Sistema IDAT. [En línea] [Citado el: 30 de Octubre de 2010.] http://www.logismarket.es/idat/1218962939-1177339-c.html.

9. Sistema TRACE-IDAT. [En línea] [Citado el: 30 de Octubre de 2010.] http://www.idat.es/index.php?option=com_content&task=view&id=15&Itemid=61.

10. Sistema en Cuba para la gestión de la información. [En línea] [Citado el: 30 de Octubre de 2010.] www.gestec.disaic.cu/PONENCIAS2009/gestec/Cuba/P57.doc.

11. Sistema Nacional Ávila Link. [En línea] [Citado el: 25 de Noviembre de 2010.] http://www.desoft.cu/Portals/0/Ficha%20del%20Producto%20AvilaLink%20v2.5.doc.

12. Sistema para la gestión de los colaboradores de la salud. [En línea] [Citado el: 30 de Octubre de 2010.] http://informatica2009.sld.cu/Members/raglz/sistema-para-la-gestion-de-los-colaboradores-de-la-salud-2/trabajo.

13. Gestión del subsistema Activos Fijos Tangibles del sistema Cedrux. [En línea] [Citado el: 31 de Octubre de 2010.] http://www.magon.cu/infociencia/Art%C3%ADculos/2009/Art.%20Vol.%2013-%284%292009/activos%20fijos%20573.pdf.

14. Sistema de Autenticación, Autorización y Auditoría (AAA). [En línea] [Citado el: 31 de Octubre de 2010.] http://10.36.7.201:5800/sas/Area%20Tematica%20Sistema_Apoyo_Salud/Branches/Tesis/Tesis%202008-2009/alas_Generales%20Componente%20Seguridad/.

15. Proceso unificado de desarrollo de software. [En línea] [Citado el: 28 de Octubre de 2010.] http://prug.solucionesracionales.com/node/12.

16. Lenguaje Unificado de Modelado. [En línea] [Citado el: 28 de Noviembre de 2010.] http://www.dcc.uchile.cl/~psalinas/uml/introduccion.html.

17. Sistema Gestor de BD. [En línea] [Citado el: 24 de Enero de 2011.] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.

18. Definición de Ajax. [En línea] [Citado el: 1 de Noviembre de 2010.] http://www.ajaxfacil.com/ique-es-ajax.

19. Definición de Ajax. [En línea] [Citado el: 1 de Noviembre de 2010.] http://www.ajaxya.com.ar/temarios/descripcion.php?cod=8&punto=1.

20. Symfony. [En línea] [Citado el: 1 de Noviembre de 2010.] http://www.librosweb.es/symfony_1_0/capitulo1/symfony_en_pocas_palabras.html.

21. WordPress y Green Park. ExtJS. [En línea] [Citado el: 1 de Noviembre de 2010.] http://blogs.antartec.com/desarrolloweb/2008/10/extjs-lo-bueno-lo-malo-y-lo-feo/.

22. Concepto de PHP. [En línea] [Citado el: 1 de Noviembre de 2010.] http://www.desarrolloweb.com/articulos/392.php.

23. JavaScript. [En línea] [Citado el: 1 de Noviembre de 2010.] http://www.librosweb.es/javascript/capitulo1.html.

24. Enterprise Architect. [En línea] [Citado el: 2 de Noviembre de 2010.] http://www.sparxsystems.es/New/products/index.html#EA.

25. NetBeans 6.9. [En línea] [Citado el: 2 de Noviembre de 2010.] http://blogultura.com/java/.

26. Modelo-Vista-Controlador. [En línea] [Citado el: 2 de Noviembre de 2010.] http://www.neleste.com/modelo-vista-controlador/.

27. Modelo-Vista-Controlador. [En línea] [Citado el: 22 de Noviembre de 2010.] http://www.programacionweb.net/articulos/articulo/?num=505.

28. Arquitectura Cliente-Servidor. [En línea] [Citado el: 2 de Noviembre de 2010.] http://www.desarrolloweb.com/articulos/arquitectura-cliente-servidor.html.

29. Arquitectura Cliente-Servidor. [En línea] [Citado el: 2 de Noviembre de 2010.] http://es.kioskea.net/contents/cs/csintro.php3.

30. Arquitectura Cliente-Servidor. [En línea] [Citado el: 22 de Noviembre de 2010.] http://www.csae.map.es/csi/silice/Global71.html.

31. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El proceso unificado de desarrollo de software.* España : Addison Wesley, 2004. ISBN.

32. **Sommerville, Ian.** *Ingeniería del software. Séptima edición.* Madrid, España : Pearson Educación, 2005. ISBN.

33. Ídem 32.

34. **Rumbaugh, James, Jacobson, Ivar y Grady, Booch.** *El lenguaje unificado de modelado. Manual de referencia.* España : Addison Wesley, 2000. ISBN.

35. **Alfonso Hernández, Alejandro y Mariño Valdés, Arlenys.** *Módulo para el control del pago a los colaboradores de la salud.* Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba : s.n., 2010.

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

37. **Pressman, Roger S.** *Ingeniería del Software. Un enfoque práctico.* Sexta edición. s.l. : McGraw-Hill Interamericana. ISBN.

38. Ídem 31.

39. Ídem 37.

40. Ídem 31.

41. Ídem 31.

42. Ídem 34.

43. Ídem 34.

44. Ídem 31.

45. Ídem 34.

46. Ídem 31.

47. Documentación SfGuardDoctrinePlugin. [En línea] [Citado el: 28 de Enero de 2011.] http://www.symfony-project.org/plugins/sfDoctrineGuardPlugin.

48. Documentación Extra de sfGuardPlugin. [En línea] [Citado el: 28 de Enero de 2011.] http://trac.symfony-project.org/wiki/sfGuardPluginExtraDocumentation.

# Bibliografía

**Alfonso Hernández, Alejandro y Mariño Valdés, Arlenys.** *Módulo para el control del pago a los colaboradores de la salud.* Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba : s.n., 2010.

Arquitectura Cliente-Servidor. [En línea] [Citado el: 2 de Noviembre de 2010.] http://www.desarrolloweb.com/articulos/arquitectura-cliente-servidor.html.

Arquitectura Cliente-Servidor. [En línea] [Citado el: 2 de Noviembre de 2010.] http://es.kioskea.net/contents/cs/csintro.php3.

Arquitectura Cliente-Servidor. [En línea] [Citado el: 22 de Noviembre de 2010.] http://www.csae.map.es/csi/silice/Global71.html.

Concepto de CESIM. [En línea] [Citado el: 22 de Noviembre de 2010.] http://www.fec.uh.cu/CUGIO/1%20acciones/Proyectos-Protocolos/Velmour%20Munos %20Casals.pdf.

Concepto de PHP. [En línea] [Citado el: 1 de Noviembre de 2010.] http://www.desarrolloweb.com/articulos/392.php.

Concepto de SOFTEL. [En línea] [Citado el: 22 de Noviembre de 2010.] http://www.ecured.cu/index.php/Softel.

Definición de Ajax. [En línea] [Citado el: 1 de Noviembre de 2010.] http://www.ajaxfacil.com/ique-es-ajax.

Definición de Ajax. [En línea] [Citado el: 1 de Noviembre de 2010.] http://www.ajaxya.com.ar/temarios/descripcion.php?cod=8&punto=1.

**Doctrine Core Team.** *Doctrine Manual.* 2008.

Documentación Extra de sfGuardPlugin. [En línea] [Citado el: 28 de Enero de 2011.] http://trac.symfony-project.org/wiki/sfGuardPluginExtraDocumentation.

Documentación SfGuardDoctrinePlugin. [En línea] [Citado el: 28 de Enero de 2011.] http://www.symfony-project.org/plugins/sfDoctrineGuardPlugin.

Enterprise Architect. [En línea] [Citado el: 2 de Noviembre de 2010.] http://www.sparxsystems.es/New/products/index.html#EA.

**Frederick, Shea, Ramsay, Colin y Blades, Steve 'Cutter'.** *Learning Ext JS.* Birmingham-Mumbai : Packt Publishing Ltd., Noviembre 2008. ISBN.

Gestión del subsistema Activos Fijos Tangibles del sistema Cedrux. [En línea] [Citado el: 31 de Octubre de 2010.] http://www.magon.cu/infociencia/Art%C3%ADculos/2009/Art.%20Vol.%2013-%284%292009/activos%20fijos%20573.pdf.

Historia Colaboración Médica en Cuba. [En línea] [Citado el: 20 de Mayo de 2010.] http://www.sld.cu/sitios/bmn/temas.php?idv=12389.

Informática en la medicina. [En línea] [Citado el: 20 de Mayo de 2010.] http://informaticaenlamedicina.blogspot.com.

**Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El proceso unificado de desarrollo de software.* España : Addison Wesley, 2004. ISBN.

JavaScript. [En línea] [Citado el: 1 de Noviembre de 2010.] http://www.librosweb.es/javascript/capitulo1.html.

Lenguaje Unificado de Modelado. [En línea] [Citado el: 28 de Noviembre de 2010.] http://www.dcc.uchile.cl/~psalinas/uml/introduccion.html.

Modelo-Vista-Controlador. [En línea] [Citado el: 2 de Noviembre de 2010.] http://www.neleste.com/modelo-vista-controlador/.

Modelo-Vista-Controlador. [En línea] [Citado el: 22 de Noviembre de 2010.] http://www.programacionweb.net/articulos/articulo/?num=505.

NetBeans 6.9. [En línea] [Citado el: 2 de Noviembre de 2010.] http://blogultura.com/java/.

**Pompa González, Rolando y Flores Ramos, Dannier.** *Balance y Planificación de Insumos Médicos Módulo alasBAPAlmacén v1.0.* Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba : s.n., 2009.

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

**Potencier, Fabien.** *symfony Forms in Action.* 2009.

**Pressman, Roger S.** *Ingeniería del Software. Un enfoque práctico.* Sexta edición. s.l. : McGraw-Hill Interamericana. ISBN.

Proceso unificado de desarrollo de software. [En línea] [Citado el: 28 de Octubre de 2010.] http://prug.solucionesracionales.com/node/12.

**Riverón González, Yuandy y González Benítez, Diainys.** *Módulo para la gestión de la información de los colaboradores de la salud.* Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba : s.n., 2010.

**Rumbaugh, James, Jacobson, Ivar y Grady, Booch.** *El lenguaje unificado de modelado. Manual de referencia.* España : Addison Wesley, 2000. ISBN.

Sistema de Autenticación, Autorización y Auditoría (AAA). [En línea] [Citado el: 31 de Octubre de 2010.] http://10.36.7.201:5800/sas/Area%20Tematica%20Sistema_Apoyo_Salud/Branches/Tesis/Tesis%202008-2009/alas_Generales%20Componente%20Seguridad/.

Sistema de Gestión de Usuarios. [En línea] [Citado el: 30 de Octubre de 2010.] http://www.eventos.bvsalud.org/EDRedes1/public/documents/SGU%20e.

Sistema en Cuba para la gestión de la información. [En línea] [Citado el: 30 de Octubre de 2010.] www.gestec.disaic.cu/PONENCIAS2009/gestec/Cuba/P57.doc.

Sistema Gestor de BD. [En línea] [Citado el: 24 de Enero de 2011.] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.

Sistema IDAT. [En línea] [Citado el: 30 de Octubre de 2010.] http://www.logismarket.es/idat/1218962939-1177339-c.html.

Sistema Nacional Ávila Link. [En línea] [Citado el: 25 de Noviembre de 2010.] http://www.desoft.cu/Portals/0/Ficha%20del%20Producto%20AvilaLink%20v2.5.doc.

Sistema para la gestión de los colaboradores de la salud. [En línea] [Citado el: 30 de Octubre de 2010.] http://informatica2009.sld.cu/Members/raglz/sistema-para-la-gestion-de-los-colaboradores-de-la-salud-2/trabajo.

Sistema TRACE-IDAT. [En línea] [Citado el: 30 de Octubre de 2010.] http://www.idat.es/index.php?option=com_content&task=view&id=15&Itemid=61.

Sistemas de Trazabilidad. [En línea] [Citado el: 28 de Octubre de 2010.] http://www.informatica-hoy.com.ar/software-erp/Que-son-los-sistemas-de-trazabilidad.php.

Software Integral SGAP. [En línea] [Citado el: 30 de Octubre de 2010.] http://www.integralsoftware.com.ar/?s=sgap.

**Sommerville, Ian.** *Ingeniería del software. Séptima edición.* Madrid, España : Pearson Educación, 2005. ISBN.

Symfony. [En línea] [Citado el: 1 de Noviembre de 2010.] http://www.librosweb.es/symfony_1_0/capitulo1/symfony_en_pocas_palabras.html.

TIC en la salud (inglés). [En línea] [Citado el: 20 de Mayo de 2010.] http://www.infodev.org/en/Project.38.html.

WordPress y Green Park. ExtJS. [En línea] [Citado el: 1 de Noviembre de 2010.] http://blogs.antartec.com/desarrolloweb/2008/10/extjs-lo-bueno-lo-malo-y-lo-feo/.

# Glosario de Términos

**Autenticar:** Efectuar un procedimiento que garantice la autenticidad y, por lo tanto, la legalidad de un documento, de un procedimiento o de un hecho, en este caso de una persona a un sistema informático.

**Casos de uso arquitectónicamente significativos:** Casos de uso que son los más importantes para los usuarios del sistema y aquellos que ayudan a cubrir todas las funcionalidades significativas.

**Clases:** Conjunto de objetos que comparten atributos, operaciones, relaciones y semántica, las mismas representan los conceptos fundamentales del sistema.

**CSS:** Acrónimo en inglés de Cascading Style Sheets (hojas de estilo en cascada), es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML).

**Código abierto:** Es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales y/o filosóficas, las cuales se destacan en el llamado software libre.

**Expediente de Proyecto:** Conjunto de documentos y plantillas estructuradas siguiendo una jerarquía, que constituyen referencia para la documentación de los proyectos de la Universidad de las Ciencias Informáticas.

**Framework:** Es una estructura de soporte definida en la cual un proyecto de software puede ser organizado y desarrollado.

**HTML:** Acrónimo en inglés de HyperText Markup Language (lenguaje de marcado de hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

**HTTP:** Acrónimo en inglés de Hypertext Transfer Protocol (protocolo de transferencia de hipertexto) es el protocolo usado en cada transacción de la World Wide Web.

**Software Libre:** Es la denominación del software que respeta la libertad de los usuarios sobre su producto adquirido y, por tanto, una vez obtenido puede ser usado, copiado, estudiado, modificado y redistribuido libremente.

**XHTML:** Acrónimo en inglés de eXtensible Hypertext Markup Language (lenguaje extensible de marcado de hipertexto), es el lenguaje pensado para sustituir a HTML como estándar para las páginas web.

**XML:** Acrónimo en inglés de Extensible Markup Language (lenguaje de marcado extensible), puede usarse para almacenar datos en un formato estructurado, basado en texto y definido por el usuario.

**TCP/IP:** Conjunto de protocolos de red en los que se basa Internet y que permiten la transmisión de datos entre redes de computadoras.

**USB:** Acrónimo en inglés de Universal Serial Bus (bus universal en serie), es un puerto que sirve para conectar periféricos a un ordenador. Es un estándar que permite la transferencia de información desde o hacia otro periférico.