Universidad de las Ciencias Informáticas
Facultad 5

Título: Sistema para el control de reportes de las violaciones en el uso de la electricidad en la Universidad de las Ciencias Informáticas.

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

Autor: **Yulieska Hernández Escobar**

Tutor: **Ing. Mailyn Medero Ruíz**

Cotutor: Ing. **Aldo Lara González**

Junio 2009

“La verdadera grandeza de la ciencia acaba valorándose por su utilidad."

Gregorio Marañon.

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmo la presente a los 27 días del mes de Junio del año 2009.

Yulieska Hernández Escobar

_____

Firma del Autor

Ing. Mailyn Medero Ruíz

_____

Firma del Tutor

Ing. Aldo Lara González

_____

Firma del Co-Tutor

DATOS DE CONTACTO

Ing. Mailyn Medero Ruíz
Email: mmedero@uci.cu

Ingeniera en Ciencias Informáticas. Graduada en Julio de 2008. Profesora del Departamento de Ciencias Básicas de la Facultad 5 desde septiembre de 2008; Profesor Adiestrado. Líder de Calidad de la Facultad 5. Ha cursado varios postgrados durante su etapa de adiestramiento. Participó en UCIENCIA 2008.

Ing. Aldo Lara González
Email: algonzalez@uci.cu

Ingeniero en Ciencias Informáticas. Graduado en Julio de 2008. Informático de la Dirección de Gestión Energética; Profesor Adiestrado. Ha cursado varios postgrados durante su etapa de adiestramiento.

AGRADECIMIENTOS

A mi tutora por guiarme y apoyarme durante todo el desarrollo del trabajo.

A mi cotutor Aldo Lara Gonzales por su incansable apoyo durante el seguimiento de todo el proceso de desarrollo de la aplicación, su esfuerzo y colaboración fueron muy importantes para mí.

A todas las personas que de una forma u otra me ayudaron en todos estos años en la Universidad.

A todos mis amigos y compañeros que siempre estuvieron con migo en los buenos y malos momentos.

DEDICATORIA

A mis padres por el incansable esfuerzo y sacrificio durante todos estos años, para que yo hiciera posible este sueño, su apoyo y compresión fueron de vital importancia para mí.

A mi hermanita linda por ayudarme siempre, en los buenos y malos momentos.

A esa gran familia que me aconsejan y me guían día a día.

A Elí, a mi niño y a todos esos amigos que me quieren y me apoyan siempre.

RESUMEN

El control de la información de forma automatizada, es una solución técnica para agilizar todo el proceso de gestión de información almacenada en empresas, centros de enseñanzas o cualquier otra institución en general que guarde grandes volúmenes de información. Después del estudio realizado en la Universidad, debido a las problemáticas existentes para el control de las violaciones del reglamento para el uso de la energía eléctrica, surge la necesidad de desarrollar un nuevo sistema para la elaboración de los reportes que muestran el cumplimiento que se le da en la UCI[1] al reglamento para el ahorro de la energía eléctrica. El nuevo sistema que se desarrolló está diseñado para que corra sobre un servidor Web, cuenta con una base de datos donde se guarda toda la información que se va a gestionar para su funcionamiento. En aras de lograr dicho objetivo se utilizó el framework de Symfony y se eligió RUP[2] para guiar el proceso a través de sus fases y flujos de desarrollo, las facilidades que brinda el framework escogido hace posible obtener una aplicación web amigable, funcional y eficiente que permita realizar el control de reportes a la Dirección Energética de la UCI de forma fácil y sencilla.

Palabras Claves: electricidad, violaciones, reporte.

[1] Universidad de Ciencias Informáticas.
[2] Proceso Unificado de Software.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

INDICE DE FÍGURAS

INTRODUCCIÓN

Con el impulso de las nuevas Tecnologías de la Información y las Comunicaciones (TIC) y su impacto para el desarrollo de aplicaciones informáticas con un amplio nivel de utilización en las operaciones relacionadas con la gestión de información, la sociedad actual está más digitalizada, lo que provoca la necesidad de un mayor equipamiento electrónico en las diferentes áreas y además supone un incremento en su intensidad de uso, lo que se traduce en una mayor demanda energética.

En Cuba, se desarrolla la Revolución Energética, la misma está conformada por varios programas, desde inicios de 2004 ha permitido a la nación ahorrar cuantiosos volúmenes de combustible y de recursos en general. Desde distintas entidades se está trabajando para controlar el uso eficiente de la energía eléctrica, debido a la ignorancia de personas que no perciben los esfuerzos que realiza el país por mejorar el nivel de vida de la población, y poner fin a los molestos apagones.

La Universidad de Ciencias Informáticas (UCI), según datos estadísticos de las entidades consumidoras de energía en la Ciudad de La Habana, ocupa el 2do lugar dentro de los mayores consumidores energéticos. Por lo que se crea por parte de la Dirección Energética en la UCI el reglamento del uso eficiente de la electricidad, por el cual velan un conjunto de auditores energéticos en todas las áreas de la Universidad.

El control del reglamento por parte de los auditores energéticos se lleva de forma manual lo que dificulta la rapidez, e incrementa el tiempo de búsqueda de informaciones relevantes provocando un atraso en las medidas a aplicar para los incumplimientos.

Como parte del proceso de informatización de la UCI, se propone automatizar el proceso de control del reglamento para el uso eficiente de la electricidad, lo que facilitará gestionar la información de forma rápida y la salida eficaz de informes o reportes de incumplidores.

Por lo antes expuesto se propone el siguiente **Problema Científico**: ¿Cómo lograr el control de las violaciones que se cometen a diario en la universidad con el incorrecto uso de la energía eléctrica?

La presente investigación, tiene como **objeto de estudio** el proceso de adquisición, auditoría y control de datos del uso de la energía eléctrica en la UCI.

Campo de acción: proceso de control de las violaciones del reglamento para el correcto uso de la energía eléctrica en la UCI

Partiendo de la **Hipótesis** de que si se diseña e implementa el sistema para el control de reportes de las violaciones del reglamento en el uso de la energía eléctrica en la UCI incrementará la eficiencia en el desarrollo del proceso de control de reportes de violaciones y gestión de la información en la Dirección Energética de la UCI.

Variables independientes: Sistema para el control de reportes de las violaciones del reglamento en el uso de la energía eléctrica en la UCI.

Variable dependiente: eficiencia en el desarrollo del proceso de control de reportes de violaciones y gestión de la información en la Dirección Energética.

Para ello se planteó como **objetivo general:** Desarrollar un sistema que permita realizar el control de reportes de las violaciones del reglamento para el uso eficiente de la energía eléctrica de forma fácil y sencilla, en la Dirección Energética de la UCI.

Objetivos específicos:

1. Caracterizar las aplicaciones Web a partir de la fundamentación teórica asociada a ellas.
2. Analizar y diseñar todos los módulos del Sistema para el control de reportes.
3. Implementar todos los módulos del sistema para el control de reportes.

Para dar cumplimiento estos objetivos específicos se trazaron las siguientes **Tareas Investigativas:**

- ✓ Entrevistas a personas implicadas en el proceso de control de violaciones para identificar correctamente todas las funcionalidades necesarias y llevar a cabo una adecuada automatización de cada uno de los módulos.
- ✓ Estudio del proceso actual de control y tratamiento de las violaciones del reglamento del uso eficiente de la energía eléctrica en la UCI para realizar la captura de requisitos.
- ✓ Estudio y selección de las tecnologías y herramientas adecuadas para el desarrollo de la aplicación, teniendo en cuenta lenguaje de programación, framework a utilizar, gestor de base datos y metodología de desarrollo.
- ✓ Estudio e investigación de las funcionalidades e importancia de la arquitectura MVC (Modelo Vista Controlador) que propone el framework de Symfony.

La realización de las tareas estuvo sustentada en un conjunto de **métodos de investigación** que ahorran esfuerzos y tiempo. Estos métodos se dividen en dos grandes grupos: los métodos teóricos y los empíricos.

En el grupo de los **métodos teóricos** utilizados se encuentra el analítico-sintético, el cual plantea que el análisis y la síntesis son dos procesos inherentes al pensamiento, operaciones lógicas importantes que permiten, como métodos teóricos, buscar la esencia de los fenómenos, los rasgos que los caracterizan y los distinguen. Este método se utilizó para realizar el análisis y la síntesis a las bibliografías consultadas cuyas temáticas esenciales fueran: sistemas informáticos para reportes, lenguajes de programación, herramientas de modelado, así como otros temas de interés relacionados con la energía eléctrica.

También se empleó el método inductivo–deductivo que consiste en aplicar formas de razonamiento que permiten llegar a un grupo de conocimientos generalizadores, tanto desde el análisis de lo particular a lo general, como desde el análisis de elementos generalizadores a uno de menor nivel de generalización, siendo éste uno de los métodos esenciales para lograr la generalización de un conjunto de entidades en el diseño de la base de datos para la herramienta informática que facilite el control y tratamiento de las violaciones del reglamento del uso eficiente de la electricidad en la UCI.

Por su parte, del grupo de los **métodos empíricos** se emplea la observación a través de un registro visual de lo que ocurre en las inspecciones de la residencia, docentes y otras entidades laborales, clasificando y consignando los hechos y acontecimientos pertinentes. Se utilizó además la entrevista a auditores energéticos, con el fin de obtener información relacionada con el tratamiento que se le da a las infracciones del reglamento cometidos por el personal que labora y reside en la UCI.

Aportes prácticos del trabajo:
El desarrollo de una herramienta informática automatizada, que permita facilitar el control y tratamiento de las violaciones del reglamento del uso eficiente de la energía eléctrica en la UCI.

El **documento está estructurado** en cuatro capítulos, permitiendo dar solución al problema que guió la investigación:

- ✓ **Capítulo No 1-** Fundamentación Teórica: Este capítulo constituye la base teórica del presente trabajo. En él se describen tendencias actuales de sistemas de control de reportes existentes, además se describen y justifican las tecnologías, herramientas, metodologías y lenguajes a utilizar para la implementación del sistema.

- ✓ **Capítulo No 2-** Propuesta de solución al problema existente: Contiene una propuesta de solución al problema existente. Se describen los artefactos generados en la etapa de Modelo de Negocio como diagrama de caso de uso del negocio, diagrama de actividades, el modelo de datos. Se realiza el levantamiento de requerimientos funcionales y no funcionales que deberá cumplir el sistema, la construcción del diagrama de casos de usos del sistema, realizándosele a cada caso de uso del sistema la descripción textual.

- ✓ **Capítulo No 3-** Análisis y diseño del sistema: Representa el análisis y diseño del sistema que se propone mostrándose los diagramas de clases del análisis para cada caso de uso del sistema, la realización de los diagramas de iteración del sistema en este caso los diagramas de secuencia y los diagramas de clases del diseño donde se presentan los estereotipos Web y el patrón de diseño a usar debido al tipo de aplicación que se va a construir, así como la descripción de la arquitectura usada.

- ✓ **Capítulo No 4-** Implementación y Resultados: En este capítulo se detalla el diagrama de despliegue y los diagramas de componentes para la base de datos, así como el código fuente y los resultados.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

El siglo XXI llegó acompañado de grandes transformaciones tecnológicas, que se ampliaron notablemente en el conocimiento del hombre acerca del mundo que le rodeaba y que, a su vez, condujeron a cambios en la forma de interactuar con él. Poco a poco, ocurrieron cambios en los soportes de la información y se aplicaron las nuevas tecnologías para la organización, almacenamiento y recuperación de la información. El surgimiento de control de reportes ha sido uno de estos cambios significativos, pues constituyen el punto de apoyo para controlar la información que se maneja en cualquier empresa que lo use. El objetivo en este capítulo, es realizar el estudio del estado del arte de la situación actual para la elaboración de aplicaciones web que faciliten el control de reportes, desarrollar un estudio de las aplicaciones web en general teniendo en cuenta características, ventajas y desventajas desde el surgimiento de las mismas y hacer un estudio detallado de las tecnologías actuales para la selección de las posibles herramientas y lenguajes a utilizar en el desarrollo de una herramienta informática automatizada que facilite el control y tratamiento de las violaciones del reglamento del uso eficiente de la energía eléctrica en la UCI.

1.2 Trabajos similares

El control de reportes, es una forma más de gestionar información almacenada en empresas, instituciones y demás entidades. El control de reportes de violaciones para el uso eficiente de energía es un sistema de este tipo de gestión, por lo que se hizo necesario un estudio de los sistemas similares que pudieran existir.

1.2.1 Ámbito Nacional e Internacional

En el marco particular de los sistemas para el control de reportes a partir de grandes volúmenes de información almacenada. La construcción de aplicaciones web es la solución más utilizada para el control de reportes, ya que son sistemas fáciles de diseñar e implementar, teniendo en cuenta la amplia comunidad informática existente, los mismos están diseñados en disímiles formas, en dependencia de las necesidades de los clientes. Después de un amplio estudio realizado no se encontró ningún sistema específicamente para el control de violaciones en el uso de la electricidad. En el mundo existen leyes y normativas que rigen el uso de la energía eléctrica pero ningún sistema automatizado para el control de violaciones a partir de un reglamento específico.

1.2.1.1 En la UCI[3]

Centrados en la construcción de este tipo de sistemas diseñados para dar solución al control de reportes para el uso eficiente de la energía eléctrica en la Universidad se encontró una propuesta de sistema, pero la misma no da cumplimiento a los requisitos funcionales que el cliente solicita, por lo que el sistema que se propone constituirá la única solución funcional al problema en cuestión.

1.3 Aplicaciones Web.

La Web tiene una influencia diaria y permanente en la vida de los seres humanos; es tal que se ha extendido a sectores como la economía, industria, educación, salud, administración pública y entretenimiento. Actualmente es difícil encontrar un área que no haya sido infiltrada por la misma. Uno de los factores existentes para esta omnipresencia descansa en la naturaleza de la Web, que es caracterizada por su disponibilidad global y permanente, así como por el acceso cómodo y uniforme a la gran cantidad de información distribuida y producida por cualquiera en forma de páginas Web[4]. Los criterios anteriores muestran el alcance e impacto de la Web en la sociedad humana y permiten vislumbrar cual es y será su importancia para la vida. Uno de los factores que han impulsado esto es la evolución y desarrollo de las aplicaciones Web.

1.3.1 ¿Que son las aplicaciones Web?

La Web fue diseñada originalmente como un medio puramente informativo pero ha evolucionado hasta convertirse en un medio de aplicación, pero antes de analizar cuál ha sido su historia y evolución, es importante definir qué se entiende como tal. El primer pasó para diferenciar las aplicaciones Web de los tradicionales sitios Web centrados en el contenido consiste, según Bob Baxley[5], al enfocarse en el significado de la palabra aplicación. La mayoría de las definiciones expresan que una aplicación es un software o programa de computadora que tiene como objetivo realizar una tarea específica. A partir de este concepto una aplicación Web puede definirse de forma sencilla como un software al que se accede mediante un navegador Web y cuyo propósito es la realización de una o varias tareas solicitadas por un usuario. La siguiente definición aporta otros elementos importantes a tener en cuenta: Una aplicación Web es un software basado en tecnologías y estándares de W3C

[3] Universidad de Ciencias Informáticas

[5] Bob Baxley (2003). What is a Web Application? [Internet]. Disponible en: <http://www. boxesandarrows.com/view/what_is_a_web_application_> [Citado 13 Mayo 2008].

que provee recursos específicos tales como contenidos y servicios a través de una interfaz de usuario a la que puede accederse utilizando un navegador Web.[6]

Ventajas:

Son independientes de las tecnologías de hardware y software (sistema operativo) que se utilicen. Una empresa podría realizar un proceso de migración y todo quedaría funcionando correctamente. La centralización del código implica que los cambios en el software no tengan riesgos ni incompatibilidades, así como la actualización inmediata a todos los clientes. No es necesario descargarlas, instalarlas y configurarlas. Simplemente se acceden a través de un navegador. Poseen menos errores ya que todos los clientes utilizan la misma versión y una vez que un error es detectado se corrige rápidamente y todos se benefician. Sé pueden utilizar diferentes lenguajes de programación del lado del servidor y quedar de modo transparente para los usuarios.

Desventajas:

Se precisa de una conexión de red constante, elemento que no está al alcance de todos. Los usuarios continúan desconfiando de la seguridad de sus datos personales, relacionados con las compras en línea, cuentas bancarias, entre otras. Aún quedan algunas deficiencias frente aplicaciones de escritorio.

1.3.2. Historia o evolución de las aplicaciones Web

La evolución de las aplicaciones Web ha transformado los sistemas informáticos, rompiendo barreras físicas (debido a la distancia), económicas y lógicas (debido a los diferentes sistemas operativos) y ha abierto un abanico de nuevas posibilidades[7]. En este proceso ha existido una retroalimentación entre el desarrollo de la Web y las aplicaciones que en la misma se pueden construir. Se puede enmarcar su desarrollo en cuatro grandes etapas o generaciones [8] y aunque en la actualidad las cuatro conviven, ya casi nadie utiliza las dos primeras. La primera generación abarca desde el 1992 (nacimiento de la Web) hasta mediados de 1994. En esta etapa se emplea la Web como un medio de comunicación tradicional (libros, revistas, etc.), una simple colección de documentos estáticos. Puede definirse como la Web 1.0 donde las páginas tienen las siguientes características:

- ✓ Tiempo de carga rápido. Uso de textos, pocas imágenes y ningún recurso multimedia.
- ✓ Navegación poco estructurada.

[6] Gerti Kappel et al. Web Engineering: The Discipline of Systematic Development of Web Applications,chapter An Introduction to Web Engineering, pages 1–22. John Wiley & Sons Ltd., 2006.
[7] Sergio Luján Mora. Programación de aplicaciones Web: historia, principios básicos y clientes Web, Pages III–IV. Editorial Club Universitario, 2002.

- ✓ Páginas largas que parece que nunca se acaban.
- ✓ Poco uso de enlaces entre páginas.
- ✓ Páginas que poseen un contenido educativo o científico. Empleo de listas para organizar la información.
- ✓ Se pueden visualizar prácticamente en cualquier navegador.

A finales de este período aparece la tecnología Common Gateway Interface (CGI), que permite la creación de páginas dinámicas dando lugar a la segunda generación que se extiende desde 1995 hasta la actualidad. Su diferencia principal con la primera generación está dada por la incorporación de elementos gráficos:

- ✓ Los iconos sustituyen las palabras.
- ✓ El color de fondo se sustituye por una imagen.
- ✓ Los banners sustituyen los encabezados de las páginas.

Aunque en esta etapa la mayoría de las páginas continúan siendo estáticas se emplea cada vez más la tecnología CGI, mostrándose mucho más interés en su uso que en el propósito final de la aplicación. La aparición de esta tecnología revolucionó el mundo de las aplicaciones Web y a partir de ese momento comenzó el proceso de interacción con base de datos para la creación de aplicaciones de entretenimiento y educación, entre otras. Se había evolucionado a una etapa superior, ya se hablaba de la Web 1.5. Las principales características de esta etapa son:

- ✓ Tiempo de carga más lento. Empleo excesivo de imágenes y animaciones debido a la novedad de su uso.
- ✓ Empleo de tablas, aunque no con el propósito del diseño, sino para mostrar datos tabulados.
- ✓ Las páginas poseen una estructura de arriba hacia abajo.
- ✓ Prima el uso de tecnologías multimedia (imágenes y sonido), aunque no puedan ser visualizadas por el público correctamente.
- ✓ Aún no existe una filosofía de navegación, aunque existe una navegación jerárquica a partir de la página principal.

La tercera generación aparece a mediados de 1996. Durante esta etapa ocurre una explosión en el número de herramientas relacionadas con la Web. En cuanto a la generación de páginas

dinámicas se logró un notable avance por el uso extendido de CGI[8], pero debido a sus limitantes aparecen nuevas tecnologías. La primera solución fue Internet Database Conector (IDC) de Microsoft, le siguió Active Server Page (ASP) que resultó ser la verdadera revolución de las páginas dinámicas. Después de esto aparecieron otras como: ColdFusion, PHP y Java Server Page (JSP) basada en Java. En esta etapa aún la Web se encuentra en su versión 1.5 pero tuvo a su favor una serie de herramientas que hacían de las aplicaciones Web un fuerte rival frente a las aplicaciones de escritorio. Se utilizaban en la construcción de sistemas empresariales y prácticamente cualquier otro tipo. Las páginas pertenecientes a esta generación son las más comunes en la actualidad. Se caracterizan por:

- ✓ Tiempo de carga rápido. Uso de Cascading Style Sheets (CSS) y optimización del código Hipertexto Markup Lenguaje (HTML).
- ✓ Las páginas se limitan para que se puedan visualizar completamente en una pantalla sin tener que realizar desplazamiento (scroll).
- ✓ Se crean pensando en el usuario final, teniendo como objetivos ofrecer servicios, informar, vender, etc.
- ✓ Se tienen en cuenta principios tipo gráfico y de organización visual de la información.
- ✓ Se incorporan los principios de usabilidad y accesibilidad.
- ✓ Se comprueba con usuarios reales su funcionamiento.
- ✓ Se emplean de forma coherente los colores, las imágenes, tipos de letras, los símbolos e iconos, etc.

Esta etapa sentó bases muy sólidas para el desarrollo de la cuarta generación que comenzó en 1999 y llega hasta nuestros días, donde la mayoría de las páginas Web se crean a partir de información almacenada en una base de datos. Representa un salto en la forma de ver Internet con la aparición del concepto Web 2.0. Las aplicaciones Web alcanzan su máximo esplendor y son utilizadas tanto en la gestión de la información de las empresas como en sistemas de transferencia de voz y video. Las características de las páginas de esta etapa son:

- ✓ Se emplean nuevamente los recursos gráficos.
- ✓ HTML evoluciona: se extiende el uso de tecnologías poco empleadas hasta ese momento, como CSS y aparecen nuevas como Dynamic HTML (DHTML). Estas tecnologías generan incompatibilidad entre distintos navegadores, a pesar de introducir considerables ventajas.

[8] Common Gateway Interface. Estándar que permite el intercambio de información entre el servidor y un programa externo al servidor. Un programa CGI es un programa preparado para recibir y enviar datos desde y hacia un servidor Web según este estándar. Normalmente de programan en C o en Perl, aunque se puede usar cualquier lenguaje de propósito general.

- ✓ Uso de nuevas tecnologías multimedia: Se puede crear un sitio Web sin emplear HTML.
- ✓ Crean a los usuarios una experiencia agradable desde que visita la primera página hasta que abandona el sitio.

Aunque el desarrollo actual llega hasta la Web 2.0, se tienen aspiraciones en las cuales ya se incursiona. La Web 3.0 pretende añadirle significado a la Web y promete transformar la experiencia de los usuarios desde todos los puntos de vista, contribuyendo al desarrollo de aplicaciones Web; se asocia en gran medida al término de Web Semántica y aunque no es lo mismo, será la principal característica que se incorpore. Mecanismos con inteligencia artificial serán los encargados de sentar las bases de esta etapa de desarrollo. Hablar de la Web 4.0 (WebOS) puede que suene a ciencia ficción, pero Raymond Kurzweil9, afirma que .Para el 2029 las computadoras tendrán la potencia de proceso equivalente al de un cerebro humano. La WebOS como se le atribuye, pretende convertir la Web en un sistema operativo virtual, los usuarios podrán conectarse usando un navegador y trabajar en línea del mismo modo que lo hacen hoy en sus puestos de trabajo.

1.3.3. Características

Las aplicaciones Web poseen una serie de características que las diferencian de las aplicaciones tradicionales. Las que se mencionan a continuación están basadas en las que presenta Roger S. Pressman. A partir de su propia naturaleza son intensivas de red, dado que residen en una red (Intranet, Extranet, Internet) y brindan servicios a los usuarios.

En muchos casos una aplicación Web brinda a los usuarios contenidos como textos, gráficos, sonido y vídeo, haciendo que sean controladas por el contenido. Junto a esta característica puede identificarse la estética y no es de extrañar que el éxito o fracaso de este tipo de aplicaciones este estrechamente ligado a la apariencia que poseen y la facilidad con que los usuarios puedan interactuar con ellas. A diferencia del software convencional, que evoluciona con una serie de versiones planificadas y espaciadas en el tiempo, las aplicaciones Web están en constante evolución. Existen otras dos propiedades importantes, la inmediatez y la seguridad. La primera responde a que el proceso para desarrollar una aplicación Web es generalmente más rápido que el seguido para el software tradicional. La segunda se debe a la necesidad de tener en cuenta los controles de seguridad apropiados para evitar accesos indebidos a la información o funcionalidades que se brinden por parte de los usuarios que acceden a estas aplicaciones. En sentido general las aplicaciones Web siguen las características anteriores, pero no todas las presentan en el mismo grado. Seguidamente se mencionan cuales son las categorías que existen para este tipo de aplicaciones.

1.3.4. Categorías o tipos

Las aplicaciones Web se pueden categorizar de varias maneras y en este aspecto, al igual que en muchos otros, no existe un criterio ampliamente aceptado. En esta sección se presentarán dos nuevos criterios que se consideran adecuados para lograr una mayor comprensión del tema. El primero, basado en la funcionalidad, es resaltado por algunos autores (Murugesan y Ginige) argumentando su utilidad en la compresión de los requerimientos necesarios para el desarrollo y despliegue de las mismas. El segundo, está basado en el grado de complejidad o tamaño. Existen además enfoques donde se mezclan los criterios anteriores con la evolución que han tenido este tipo de aplicaciones para realizar una categorización que abarca desde aquellas centradas en el contenido, hasta las que siguen tendencias más actuales en el marco de la Web semántica.

Atendiendo a las funcionalidades que brindan, las aplicaciones Web pueden ser agrupadas en las categorías que se describen a continuación, aunque es importante señalar que algunas pueden clasificarse en varias de ellas.

Las aplicaciones Web informacionales o centradas en el contenido son precursoras de las actuales. Generalmente las páginas Web que las componen son estáticas y actualizadas a mano mediante herramientas específicas, lo cual puede considerarse un factor de costo cuando son numerosas, esto trae como resultado que usualmente estén desactualizadas. Los principales beneficios son la simplicidad y estabilidad así como los cortos tiempos de respuesta que presentan. Los catálogos de productos y la prensa digital se enmarcan en esta categoría.

El uso de formularios que permiten la realización de peticiones con diversos parámetros y la generación dinámica de enlaces y páginas nuevas de acuerdo con dichas solicitudes, es una de las características de las aplicaciones Web interactivas.

Los sistemas bancarios, los de reservación aérea y de los pago, son algunos ejemplos de las aplicaciones orientadas a transacciones que permiten a los usuarios una mayor interactividad, teniendo como requisito el uso de bases de datos para el almacenamiento de información y la posibilidad de realizar consultas sobre las mismas de forma eficiente y consistente.

EL manejo de operaciones y procesos son objetivos de las aplicaciones Web orientadas a flujos de trabajo. El uso de servicios Web es necesario para poder manejar la interoperabilidad,

siendo la necesidad de flexibilidad y robustez el reto más importante para este tipo de aplicaciones, ejemplificadas en los sistemas B2B[11] de comercio electrónico.

Las aplicaciones Web colaborativas están orientadas a grupos de usuarios que colaboran entre sí en la realización de diferentes tareas y que por lo general residen en lugares distantes. Los sistemas de planificación de tareas y las plataformas de aprendizaje electrónico son ejemplos de esta categoría.

En correspondencia con las funcionalidades y objetivos que posea una aplicación Web, será el tamaño y complejidad de la misma. Esto da lugar al segundo criterio antes mencionado, según el cual muchas aplicaciones pertenecen a la categoría de pequeñas a mediana escala[9] . Estas utilizan lenguajes scripts en los que se mezclan etiquetas HTML, funciones y consultas a las bases de datos que utilizan, además el tiempo de desarrollo es corto y sus requerimientos son bastante sencillos. Sin embargo, si las funcionalidades que se ofrecen son numerosas, se necesita un alto rendimiento y disponibilidad y se trabaja con grandes volúmenes de datos teniendo en cuenta la consistencia de los mismos, entonces puede hablarse de una aplicación a gran escala. Los conocimientos y actividades ingenieriles que se requieren son mucho mayores y elementos como la seguridad y usabilidad adquieren más importancia.

1.4 Metodologías de desarrollo

La tendencia actual de la producción de software requiere cada día la construcción de sistemas más grandes, complejos y rápidos, que se ajusten a las necesidades de los usuarios. Para esto se necesita lograr la productividad del software, que integre las múltiples facetas del desarrollo. Por razones como estas, se hizo necesario realizar un estudio de las metodologías de ingeniería de software que guiarán el proceso de automatización de la herramienta informática que se propone en este trabajo.

1.4.1 Proceso Unificado de Desarrollo (RUP)

RUP es un proceso de desarrollo de software que constituye una de las metodologías más utilizadas para el análisis, implementación y documentación de sistemas orientados a objetos.

Es un marco de desarrollo de software (conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software), está basado en componentes, lo cual quiere decir que el sistema de software en construcción está formado por componentes de software interconectados a través de interfaces bien definidas.

[9] Andrew van der Stock et al. What are web applications? [Internet]. Disponible en: <http://www.owasp.org/index.php/What_are_web_applications%3F> [Citado 13 Mayo 2008]

Esta metodología de desarrollo se basa en casos de uso para describir lo que se espera del software y para guiar el proceso además de prestar atención especial al establecimiento temprano de una buena arquitectura la cual se basa en componentes, se caracteriza por su forma disciplinada de asignar tareas y responsabilidades (quien hace qué, cuándo y cómo), RUP pretende implementar las mejores prácticas de Ingeniería de Software, acepta los cambios que se realiza a lo largo del desarrollo del software llevando para el mismo el control. RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al concluir cada ciclo; estos se dividen en 4 fases:

- ✓ Inicio: El objetivo en esta etapa es determinar la visión del proyecto.
- ✓ Elaboración: En esta etapa el objetivo es determinar la arquitectura óptima.
- ✓ Construcción: En esta etapa el objetivo es llegar a obtener la capacidad operacional inicial.
- ✓ Transmisión: El objetivo es llegar a obtener el reléase del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, consisten en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

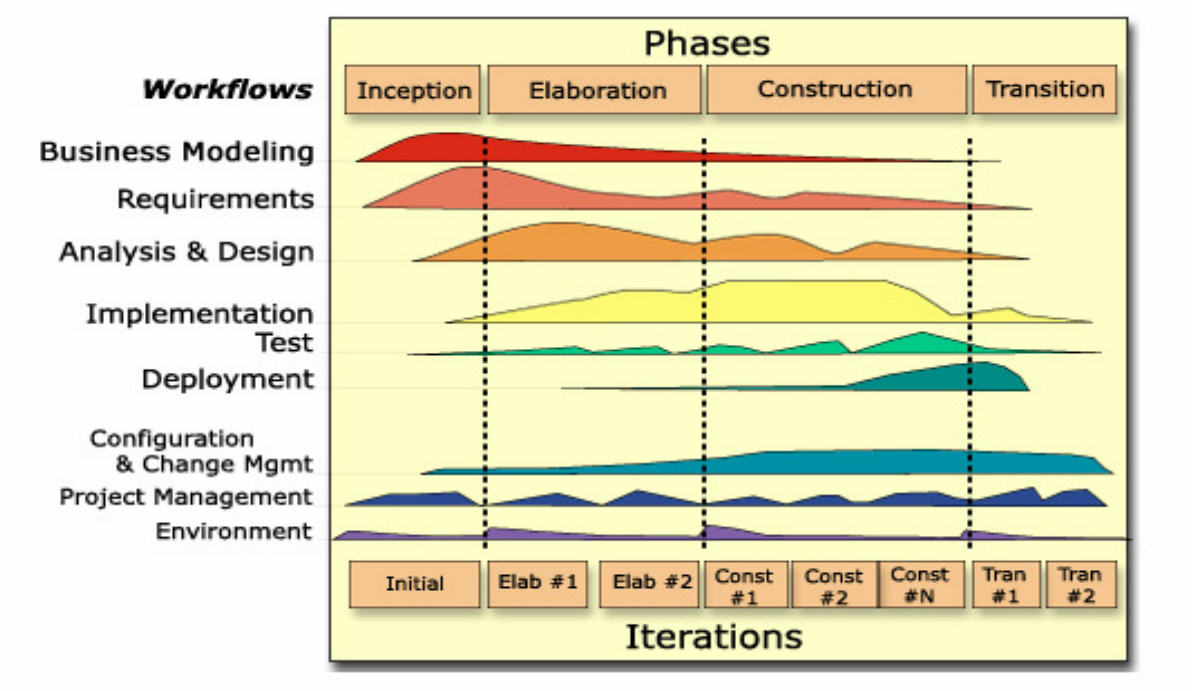
El ciclo de vida que se desarrolla por cada iteración, es llevado bajo dos disciplinas:

1. Disciplina de Desarrollo
   - ✓ Ingeniería de Negocios: Entendiendo las necesidades del negocio.
   - ✓ Requerimientos: Trasladando las necesidades del negocio a un sistema automatizado.
   - ✓ Análisis y Diseño: Trasladando los requerimientos dentro de la arquitectura de software.
   - ✓ Implementación: Creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.
   - ✓ Pruebas: Asegurándose que el comportamiento requerido es el correcto y que todo los solicitado está presente.

2. Disciplina de Soporte
   - ✓ Configuración y administración del cambio: Guardando todas las versiones del proyecto.
   - ✓ Administrando el proyecto: Administrando horarios y recursos.
   - ✓ Ambiente: Administrando el ambiente de desarrollo.
   - ✓ Distribución: Hacer todo lo necesario para la salida del proyecto.

Es recomendable que a cada una de estas iteraciones se les clasifique y ordene según su prioridad, y que cada una se convierta luego en un entregable al cliente. Esto trae como beneficio la retroalimentación que se tendría en cada entregable o en cada iteración.

En la figura 1.1 Se muestra la distribución de las fases del proceso unificado de desarrollo.



En la figura 1.1 proceso unificado de desarrollo.

1.4.2 Programación Externa (XP)

Mientras que RUP intenta reducir la complejidad del software por medio de estructura y la preparación de las tareas pendientes en función de los objetivos de la fase y la actividad actual, XP, como toda metodología ágil, lo intenta por medio de un trabajo orientado al objetivo basado en las relaciones interpersonales. XP intenta minimizar el riesgo de fallo del proceso de la disposición permanente de un representante competente del cliente a disposición del equipo de desarrollo. Este representante debería estar en condiciones de contestar rápido y correctamente a cualquier pregunta del equipo de desarrollo de forma que no se retrase la toma de decisiones, de ahí lo competente. Es una de las metodologías de desarrollo de software más exitosas usadas en la actualidad para proyectos de corto plazo y equipos de pequeño desarrollo.

La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

La metodología se basa en:

- ✓ Pruebas unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, podamos hacer pruebas de las fallas que pudieran ocurrir. Es como si nos adelantáramos a obtener los posibles errores.
- ✓ Refabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- ✓ Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.

XP como metodología:

- ✓ Empieza en pequeño y añade funcionalidad con retroalimentación continua.
- ✓ El manejo del cambio se convierte en parte sustantiva del proceso.
- ✓ El costo del cambio no depende de la fase o etapa.
- ✓ No introduce funcionalidades antes que sean necesarias.
- ✓ El cliente o el usuario se convierten en miembro del equipo.

Esta metodología garantiza los derechos tantos del cliente como del desarrollador que lo utiliza, en el caso del cliente están dadas de la siguiente manera:

- ✓ Decidir que se implementa.
- ✓ Saber el estado real y el progreso del proyecto.
- ✓ Añadir, cambiar o quitar requerimientos en cualquier momento.
- ✓ Obtener lo máximo de cada semana de trabajo.
- ✓ Obtener un sistema funcionando cada 3 o 4 meses.

Esto garantiza que el cliente obtenga en momentos determinados cuando lo solicite el estado del software de cómo marcha hasta que punto se han cumplido los requisitos, mientras para el desarrollador las siguientes:

- ✓ Decidir cómo se implementan los procesos.
- ✓ Crear el sistema con la mejor calidad posible.
- ✓ Pedir al cliente en cualquier momento aclaraciones de los requerimientos.
- ✓ Estimar el esfuerzo para implementar el sistema.
- ✓ Cambiar los requerimientos en base a nuevos descubrimientos.

Con el análisis de los derechos de clientes y desarrolladores la metodología XP se fundamenta.

Lo fundamental en este tipo de metodología es:

- ✓ La comunicación, entre los usuarios y los desarrolladores.
- ✓ La simplicidad, al desarrollar y codificar los módulos del sistema.
- ✓ La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

Se concluye que la metodología a usar para el desarrollo del sistema, es RUP por ser la metodología más adaptable para proyectos de largo plazo. Caracterizado además por ser iterativo e incremental, elemento que permitirá el refinamiento constante del sistema así como la adición de nuevas funcionalidades a cada una de las iteraciones. RUP incluye artefactos[10] y roles[11]. Esta metodología utiliza además el Lenguaje Unificado de Modelado (UML) para modelar cada una de las fases del software, por lo que se propone el siguiente epígrafe para cararterizar UML.

1.5 Lenguaje Unificado de Modelado (UML).

El Lenguaje Unificado de Modelado es un lenguaje gráfico de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos, pueden ser artefactos: un modelo, una descripción que comprende el desarrollo de software que se basen en el enfoque orientado a objetos, utilizándose también en el diseño Web, de un sistema de software orientado a objetos (OO).

Se usa para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir. No es un lenguaje de programación pero las herramientas pueden ofrecer generadores de código de UML para una gran variedad de lenguajes de programación, así como construir modelos por ingeniería inversa a partir de programas existentes. Es lo suficientemente expresivo para manejar todos los conceptos que se originan en un sistema moderno, tales como la concurrencia y distribución, así como también los mecanismos de la ingeniería de software, como son la encapsulación y componentes.

UML es el lenguaje de modelado orientado a objetos estándar predominante ahora y en los próximos años, ya que cuenta con la participación de metodólogos influyentes, de importantes empresas y un estándar del Object Management Group (OMG), siendo utilizado diariamente por grandes organizaciones como: Microsoft, Oracle y Rational.

[10] Son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc.
[11] Papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso.

1.6 Herramienta CASE

Las herramientas CASE (Computer Aided Software Engineering) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, calculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

1.6.1 Rational Rose

Es la herramienta CASE que cubre todo el ciclo de ayuda de un proyecto: concepción y formalización del modelo, construcción de los componentes y certificación de las distintas fases. Permite trazabilidad real entre modelo (análisis y diseño) y el código ejecutable.

Sus principales características son:

- ✓ Mantiene la consistencia de los modelos del sistema software.
- ✓ Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.
- ✓ Chequeo de la sintaxis UML.
- ✓ Generación documentación automáticamente.
- ✓ Generación de código a partir de los modelos.
- ✓ Disponibilidad en múltiples plataformas.
- ✓ Capacidades de Ingeniería Inversa[12].

Rational Rose es una herramienta que se puede encuadrar dentro del grupo de herramientas más técnicas debido a que se encarga de llevar a cabo la automatización de los sistemas para la posterior generación de código. Entre las diferentes ventajas que ofrece el Rose, la más importante es que utiliza la notación estándar en la arquitectura de software UML, permitiendo esta a los desarrolladores y arquitectos de software utilizar un lenguaje común para poder visualizar el sistema completo. También pueden los diseñadores modelar sus componentes e interfaces en forma individual y luego unirlos con otros componentes del proyecto.

[12] Crear modelos a partir de sistemas existentes.

1.6.2 Visual Paradigm

Visual Paradigm es una herramienta CASE que utiliza UML como lenguaje de modelado que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, importación desde Rational Rose exportación/importación XMI, generador de informes, editor de figuras, etc.

Soporta un conjunto de lenguajes, tanto en generación de código e ingeniería inversa en: Java, C++, CORBA IDL, PHP, y Python.

En fin, Visual Paradigm ofrece:

- ✓ Entorno de creación de diagramas para UMP 2.0.
- ✓ Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- ✓ Uso de lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- ✓ Capacidades de ingeniería directa (versión profesional) e inversa.
- ✓ Disponibilidad en múltiples plataforma.
- ✓ Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- ✓ Disponibilidad de múltiples versiones, para cada necesidad.

1.7 Sistemas Gestores de Bases de Datos

Un servidor de Base de Datos, es el encargado de garantizar el almacenamiento, integridad, protección y manipulación de la información de sistema.

La Base de Datos es un conjunto de datos interrelacionados entre sí, almacenados con carácter más o menos permanente en la computadora. O sea, que una base de datos puede considerarse una colección de datos variables en el tiempo.

Un Sistema de Gestión de Base de Datos (SGDB[13]) es un software específico, que permite crear y mantener una (o varias) Base (s) de Datos, asegurando mantener su integridad, confidencialidad y seguridad. Un SGDB está compuesto por un lenguaje de definición de datos, un lenguaje de manipulación de datos y un lenguaje de consulta.

Los SGDB deben cumplir algunos objetivos específicos como:

- ✓ Abstracción de la información.
- ✓ Independencia.

[13] Sistemas gestores de bases de datos.

- ✓ Redundancia mínima.
- ✓ Consistencia.
- ✓ Seguridad.
- ✓ Integridad.
- ✓ Respaldo y recuperación.
- ✓ Control de la concurrencia.
- ➢ Tiempo de respuesta.

1.7.1 Lenguaje de Consulta Estructurado (SQL).

El Lenguaje de Consulta Estructurado (Structured Query Language), es un lenguaje de base de datos normalizado, utilizado por el motor de base de datos de Microsoft Jet, es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar -de una forma sencilla- información de interés de una base de datos, así como también hacer cambios sobre la misma.

En la actualidad diversos SGDB utilizan el SQL para realizar el tratamiento de los datos almacenado como son Sistema de Administración de Bases de Datos (MySQL) y SQL Server.

1.7.2 My SQL

MySQL es un sistema de administración de bases de datos (Database Management System, DBMS) para bases de datos relacionales. MySQL utiliza una arquitectura cliente/servidor compuesta por un servidor SQL multihilo, varios programas clientes y bibliotecas, herramientas administrativas, y una gran variedad de interfaces de programación, (APIs[14]). MySQL, como base de datos relacional, utiliza múltiples tablas para almacenar y organizar la información.

El servidor de bases de datos MySQL es muy rápido, seguro, y fácil de usar. Aunque se encuentra en desarrollo constante, este servidor ofrece hoy un conjunto rico y útil de funciones. Su conectividad, velocidad, y seguridad hacen de él un servidor bastante apropiado para acceder a bases de datos en Internet.

1.7.3 PostGreSQL

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos de software libre, publicado bajo la licencia BSD[15], lo cual permite el uso y distribución sin costo, tanto para aplicaciones comerciales como no comerciales. Entre sus características más sobresalientes está la del soporte variado de tipos de datos como tipos de fecha, gráfico y

[14] Interfaz de Programación de Aplicaciones (APIs).
[15] Gestor de base de datos (BSD).

permite la creación de tipos propios. PostgreSQL es un descendiente “open-source” de este código original de Berkeley.

Es considerado uno de los mejores gestores de base de datos en open source, es por eso que un gigante como google aloja sus datos en este gestor.

Ventajas

- ✓ Instalación Ilimitada.
- ✓ Mejor soporte que los proveedores comerciales.
- ✓ Ahorros considerables en costos de operación.
- ✓ Estabilidad y confiabilidad legendarias.
- ✓ Extensible.
- ✓ Multiplataforma.
- ✓ Diseñado para ambientes de alto volumen.
- ✓ Herramientas gráficas de diseño y administración de bases de datos.

Desventajas

- ✓ Sobre la plataforma de Windows en ocasiones se vuelve inestable.
- ✓ Sobre plataformas Windows no tiene el beneficio de años de uso en ambientes de producción que tiene sobre plataformas Unix.
- ✓ Para aplicaciones OLAP, como Data Warehouses y Data Mining, PostgreSQL no es la mejor alternativa.

1.8 Lenguajes de programación.

Para la construcción del sistema, se necesita realizar un estudio de los posibles lenguajes de programación que permita crear con mayor facilidad y rapidez dicho sistema. A continuación la descripción de algunos lenguajes para ver características, usos y ventajas para tomar la decisión del que será utilizado para la implementación del sistema.

1.8.1 C#

C# es un lenguaje de programación simple pero eficaz, diseñado para escribir aplicaciones empresariales. Este toma las mejores características de lenguajes preexistentes como Visual Basic, Java o C++ y las combina en uno solo.

Es el único lenguaje que ha sido diseñado específicamente para ser utilizado en la Plataforma .NET. Programarla usando C# es mucho más sencillo e intuitivo que hacerlo con cualquiera de

los otros lenguajes soportados. Por esta razón, se suele decir que es el lenguaje nativo de .NET.

1.8.2 PHP

PHP (PHP Hypertext Preprocessor) es un lenguaje multiplataforma, multiparadigma, script (no se compila para conseguir códigos máquina sino que existe un intérprete que lee el código y se encarga de ejecutar las instrucciones que contiene éste código) para el desarrollo de páginas Web dinámicas del lado del servidor, cuyos fragmentos de código se intercalan fácilmente en páginas HTML, debido a esto y a que es de código abierto, es el más popular y extendido en la Web.

Cada día son más los programadores que utilizan PHP. Actualmente la red de Internet compuesta por páginas Web sirve de soporte a una gran cantidad de sistemas de información y comunicaciones que engloban áreas tan importantes como la investigación, el comercio electrónico, la visualización de información y el correo electrónico.

La última versión es PHP5, presenta mejoras significativas y un entorno de programación orientado a objetos mucho más completo que permite que el PHP proporcione un alto rendimiento a las aplicaciones Web empresariales a nivel de las plataformas J2EE y .NET. Una diferencia sensible es que PHP ha sido desarrollado inicialmente para entornos UNIX y es en este sistema operativo donde se aprovechan mejor sus prestaciones y consigue un mayor rendimiento.

Dentro de las principales ventajas que ofrece el lenguaje PHP se encuentran:

- ✓ Es multiplataforma, multiparadigma, libre y gratuito.
- ✓ Cuenta con una buena capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad.
- ✓ Permite leer y manipular datos desde diversas fuentes, incluyendo datos que pueden ingresar los usuarios desde formularios HTML.
- ✓ Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- ✓ Permite crear los formularios para la Web.
- ✓ Presenta una buena integración con Apache y MySQL.
- ✓ Su sintaxis es bastante clara, fácil de aprender, es seguro y popular. (10)

Luego del estudio realizado a los posibles lenguajes de programación para la implementación del sistema, se decide utilizar el lenguaje PHP 5 por varias causas:

1. El cliente solicitó dentro de los requisitos que dicha aplicación se realizara en este lenguaje.
2. Para la licencia no habría problemas pues el lenguaje es libre.
3. Presenta ventajas que los hace totalmente diferente frente a otros lenguajes tales como:
    - ✓ El código se puede tratar íntegramente como un objeto.
    - ✓ Es un lenguaje orientado a objetos y a componentes.
    - ✓ Se ahorra tiempo en la programación pues cuenta con una librería de clases muy completa y bien diseñada.

1.8.3 PHP 5

PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. Esta diseñado específicamente para ser un lenguaje más seguro, para escribir programas y con la selección correcta de configuración en tiempos de compilación y ejecución, y siguiendo algunas prácticas correctas de programación.

Ventajas

- ✓ Muy fácil de aprender.
- ✓ Se caracteriza por ser un simple lenguaje muy rápido.
- ✓ Soporta en cierta medida la orientación a objetos. Clases y Herencias.
- ✓ Es un lenguaje multiplataforma. Windows, Linux, entre otros.
- ✓ Capacidad de conexión con a mayoría de los manejadores de bases de datos: MySQL, PostGreSQL, Oracle, MS SQL Server, entre otros.
- ✓ Capacidad de expandir su potencial utilizando módulos.
- ✓ Posee documentación en su página oficial la cual incluye descripción y ejemplos de cada una de sus funciones.
- ✓ Es libre, por lo que se representa como una alternativa de fácil acceso para todos.
- ✓ Incluye gran cantidad de funciones.
- ✓ No requiere definición de tipos de variables ni manejo detallado del bajo nivel.

Desventajas

- ✓ Se necesita instalar en un servidor.

- ✓ Todo el trabajo lo realiza el servidor y no delega al cliente. Por tanto puede ser más ineficiente a medida que las solicitudes aumenten de número.
- ✓ La legibilidad del código puede verse afectada al mezclar sentencias HTML y PHP.
- ✓ La programación orientada a objetos es aún muy difícil para aplicaciones grandes.
- ✓ Dificulta la modularización.
- ✓ Dificulta la organización por capas de aplicaciones.

1.9 Framework para aplicaciones Web en PHP

1.9.1 Symfony

Es un completo framework diseñado para optimizar el desarrollo de las aplicaciones web, preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo, su código es fácil de leer e incluye comentarios de phpDocumentor que permiten un mantenimiento muy sencillo.

Symfony automatiza la mayoría de elementos comunes de los proyectos web, como son:

- ✓ La capa de internacionalización que incluye Symfony permite la traducción de los datos y de la interfaz, así como la adaptación local de los contenidos.
- ✓ Los formularios incluyen validación automatizada y relleno automático de datos ("repopulation"), lo que asegura la obtención de datos correctos y mejora la experiencia del usuario.
- ✓ Los datos incluyen mecanismos de escape que permiten una mejor protección contra los ataques producidos por datos corruptos.
- ✓ La gestión de la caché reduce el ancho de banda utilizado y la carga del servidor.
- ✓ La autenticación y la gestión de credenciales simplifican la creación de secciones restringidas y la gestión de la seguridad de usuario.
- ✓ El sistema de enrutamiento y las URL limpias permiten considerar a las direcciones de las páginas como parte de la interfaz, además de estar optimizadas para los buscadores.
- ✓ El soporte de e-mail incluido y la gestión de APIs[16] permiten a las aplicaciones web interactuar más allá de los navegadores.
- ✓ Los listados son más fáciles de utilizar debido a la paginación automatizada, el filtrado y la ordenación de datos.

[16] Librerias.

Symfony puede ser completamente personalizado para cumplir con los requisitos de las empresas que disponen de sus propias políticas y reglas para la gestión de proyectos y la programación de aplicaciones. Por defecto incorpora varios entornos de desarrollo diferentes e incluye varias herramientas que permiten automatizar las tareas más comunes de la ingeniería del software:

- ✓ Las herramientas que generan automáticamente código han sido diseñadas para hacer prototipos de aplicaciones y para crear fácilmente la parte de gestión de las aplicaciones.
- ✓ La interfaz de línea de comandos automatiza la instalación de las aplicaciones entre servidores.
- ✓ Es posible realizar cambios "en caliente" de la configuración (sin necesidad de reiniciar el servidor).
- ✓ El completo sistema de log[17] permite a los administradores acceder hasta el último detalle de las actividades que realiza la aplicación.

Ventajas

- ✓ Simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes.
- ✓ Proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener.
- ✓ Facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas.
- ✓ Nos permite reutilizar código entre los proyectos.
- ✓ Extensible a través de multitud de plugins disponibles para su descarga e instalación.

Desventajas

- ✓ Fuerte curva de aprendizaje, aunque la documentación disponible facilita enormemente esta tarea. Este es un factor a tener en cuenta ya que la productividad del equipo de desarrollo no será alta en los primeros meses.
- ✓ Todavía hay muchas aplicaciones y desarrolladores que programan en PHP4. Esto puede plantear un problema ya que para extraer toda la potencia del framework, el programador debe tener un buen conocimiento del paradigma de programación orientado a objetos.

[17] Registro (Log).

- ✓ Aumenta el peso de la aplicación, ya que la implementación del patrón MVC introduce varias capas de objetos que debe procesar el intérprete. La parte que más afecta al rendimiento es Propel, el sistema ORM utilizado por defecto en Symfony. En cualquier caso, este puede ser sustituido por otros sistemas más eficientes (propel 1.3, doctrine) a través de plugins.

1.9.2 CodeIgniter

Es un buen framework, utilizado por una gran comunidad de usuarios. Construido para codificadores PHP que necesitan una herramienta de desarrollo fácil para crear aplicaciones web simples y elegantes.

Ventajas:

- ✓ Amplia documentación disponible.
- ✓ Ligero, y sin instalación (para comenzar a desarrollar una aplicación basta con copiar los archivos, y ponerse a trabajar).
- ✓ Compatibilidad con una amplia variedad de servidores y configuraciones (la aplicación se concebiría para ejecutarse en un hosting compartido con otros clientes, y con relativamente poca posibilidad de configuración).
- ✓ Flexibilidad, ya que no obliga a tener una determinada estructura de tablas, nombres de campos, ni adherirse a una forma de programar concreta.

Desventajas:

- ✓ Curva de aprendizaje: necesidad de aprender nuevas funciones, estructuras y métodos de programación.
- ✓ Dificultad para adaptar el código escrito en PHP tradicional.

1.9.3 CakePHP

Es un framework para php que permite programar más rápido evitando escribir código tedioso de tareas muy comunes.

Ventajas:

- ✓ Tiene una comunidad muy activa, desde la bakery hasta los grupos en google y este grupo en español CakePHP-es.
- ✓ Licencia flexible - CakePHP está distribuido bajo la MIT License.
- ✓ IP limpia - Cada línea de código está escrita por el equipo de desarrollo de CakePHP.
- ✓ Extremadamente simple.
- ✓ Desarrollo rápido.

- ✓ Buenas prácticas - Cake es muy fácil de entender y cumple los estándares en seguridad y autentificación, manejo de sesiones y muchas otras características.
- ✓ Orientado a Objetos - Si te gusta la programación orientada a objetos que bien y si eres principiante te sentirás cómodo.
- ✓ Cero Configuración - Solamente pon la información de la base de datos y la magia comenzará.

Desventajas:

- ✓ Ninguna oferta de formación está disponible para este framework.
- ✓ No ofrecen más apoyo que foros, listas de correo y Google Grupos.
- ✓ Ninguna protección por definición de la licencia MIT.

Nota: Un framework es un conjunto de herramientas que facilita la programación: por ejemplo automatizar las tareas más rutinarias, generar código de forma automática y reutilizable en diferentes proyectos.

1.10 Conclusiones

Teniendo en cuenta la necesidad de realizar un sistema que permita llevar el control de los reportes de las violaciones de la electricidad en la UCI, se realizó un estudio de los sistemas existentes de control de reportes, una caracterización y análisis de las aplicaciones web y por último se estudiaron algunas de las tecnologías actuales para el desarrollo de aplicaciones web con el fin de seleccionar las mas adecuadas para el diseño e implantación del sistema en cuestión. Por lo que se escoge PHP 5 como lenguaje de programación unido al framework Symfony, como entorno compilado para realizar aplicaciones Web y como gestor de bases de datos PostGreSQL, se eligió RUP[18] para guiar todo el proceso a través de sus fases y flujos de desarrollo, las facilidades que brindó el framework escogido hizo posible obtener una aplicación web amigable, funcional y eficiente que permite realizar el control de reportes a la Dirección Energética de la UCI de forma fácil y sencilla.

[18] Proceso Unificado de Software.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN AL PROBLEMA EXISTENTE

2.1 Introducción

Antes de comenzar a desarrollar un sistema es necesario comprender el entorno de trabajo donde se va aplicar la herramienta informática, bajo el estudio de los procesos que en ella se desarrollan. Primeramente se realiza la modelación del negocio, durante esta etapa se definen y se describen los actores y trabajadores del negocio, se realizan los diagramas de actividades, modelo de objetos, se analizan los requerimientos funcionales para dar paso a la solución del problema, pues serían las funcionalidades a implementar en el sistema, es ahí donde entra toda lo referente al sistema, definición de los actores del sistema, diagrama de casos de usos del sistema así como las especificaciones de cada uno de estos casos de usos para tener una idea de cómo interactúan actor-sistema. Son precisamente tales elementos los que se desarrollan en este capítulo.

2.2 Propuesta de sistema

Para dar solución al problema existente, se ha decidido realizar una aplicación Web que facilite el control de las violaciones que se cometen a diario en la Universidad, a fin de lograr una mejor organización de los datos, posibilitando a las personas autorizadas la rapidez necesaria para la generación de reportes. Para ello, se han definido varios módulos: Autenticar usuario, Gestionar usuario, Gestionar violación, Gestionar reportes, Gestionar Auditor y Visualizar datos.

2.3 Modelo de negocio

El modelo del negocio se realiza con el propósito de comprender las características y actividades que se llevan a cabo en el contexto a automatizar. En él, se modelan los clientes y trabajadores del negocio y su interacción con los procesos y objetos que lo componen. Está formado por el modelo de casos de uso del negocio y el modelo de objetos del negocio.

Cada uno de los epígrafes que siguen a continuación describe textualmente y gráficamente todas las etapas del modelo del negocio.

2.3.1 Definición de actor del negocio

Un actor del negocio es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externos que interactúan con el negocio y se benefician de sus resultados. El nombre de un actor debe expresar su rol dentro del negocio.

Nombre del actor	Descripción
Dirección Energética de la UCI	Institución que se encargará de controlar todo lo referente a la electricidad en la UCI.

Tabla 2. 1 Descripción del actor del negocio

2.3.2 Trabajadores del negocio

Un trabajador del negocio representa a personas, o sistemas (software) dentro del negocio que son las que realizan las actividades que están comprendidas dentro de un caso de uso. Son los candidatos a futuros usuarios del sistema que se quiere construir.

Nombre del trabajador	Descripción
Jefe de Auditores energéticos	Es la persona que se encargará del control y el chequeo del trabajo de los auditores energéticos. Además de gestionar los auditores y los usuarios que interactúan con el sistema.
AuditorEncargado	Persona que se encarga de interactuar con el sistema.

Tabla 2. 2 Descripción de los trabajadores del negocio

2.3.3 Diagrama de casos de uso del negocio

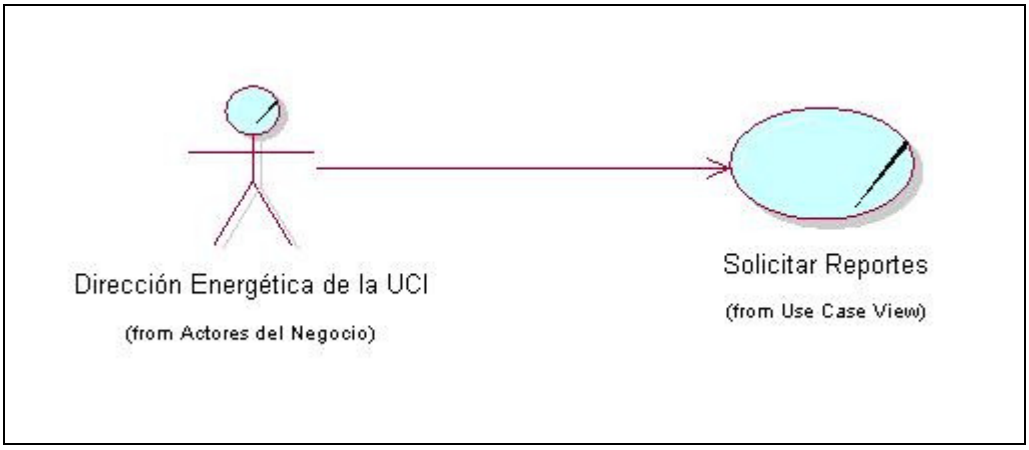


Figura 2.2 Diagrama de Caso de Uso del Negocio

2.3.4 Descripción textual del caso de uso del negocio

La descripción textual de los casos de uso del negocio, ayuda a comprender la lógica del mismo, mostrando además como ocurre la relación actor-negocio.

Caso de uso	Solicitar Reportes
Actores	Dirección Energética de la UCI
Trabajadores del negocio	Jefe de auditores energéticos, auditores energéticos
Resumen	El caso de uso comienza cuando la Dirección Energética de la UCI solicita un reporte completo de todas las sanciones, violaciones cometidas, personas implicadas, etc. El caso de uso termina cuando la Dirección Energética obtiene el reporte.
Precondiciones	Que los reportes estén archivados.
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Negocio
1. La Dirección Energética de la UCI solicita el estado de los reportes de las inspecciones en la UCI.	2. El jefe de los auditores energéticos solicita a los auditores energéticos la información de todo lo referente a las inspecciones y sanciones aplicadas.
	3. Los auditores energéticos entregan un informe de las inspecciones y sanciones aplicadas.
	4. El jefe de los auditores energéticos recibe la información de los auditores

	energéticos y crea un reporte completo.
	5. El jefe de los auditores energéticos entrega el reporte completo a la Dirección Energética de la UCI.
6. La Dirección Energética de la UCI recibe el reporte enviado por el jefe de los auditores energéticos.	
Poscondiciones	Se actualiza la información acerca de la electricidad en la UCI.
Mejoras	El registro de los reportes permitirá tener un mayor control de la electricidad en la UCI, así como de tener el conocimiento de si se mejora o no en cuanto al ahorro de la misma.
Prioridad	Importante para tener el control del uso de la electricidad en la UCI.

Tabla 2.3 Descripción textual CU Solicitar Reportes

2.4 Diagrama de actividades

Los casos de uso del negocio tienen realizaciones mostradas por diagramas de actividades o descripciones textuales. Un diagrama de actividades es una manera de modelar el flujo de eventos internos de un proceso de manera gráfica, mostrando pasos, puntos de decisión y entidades que intervienen.

A continuación se muestra el diagrama de actividades para el CU Solicitar Reportes:

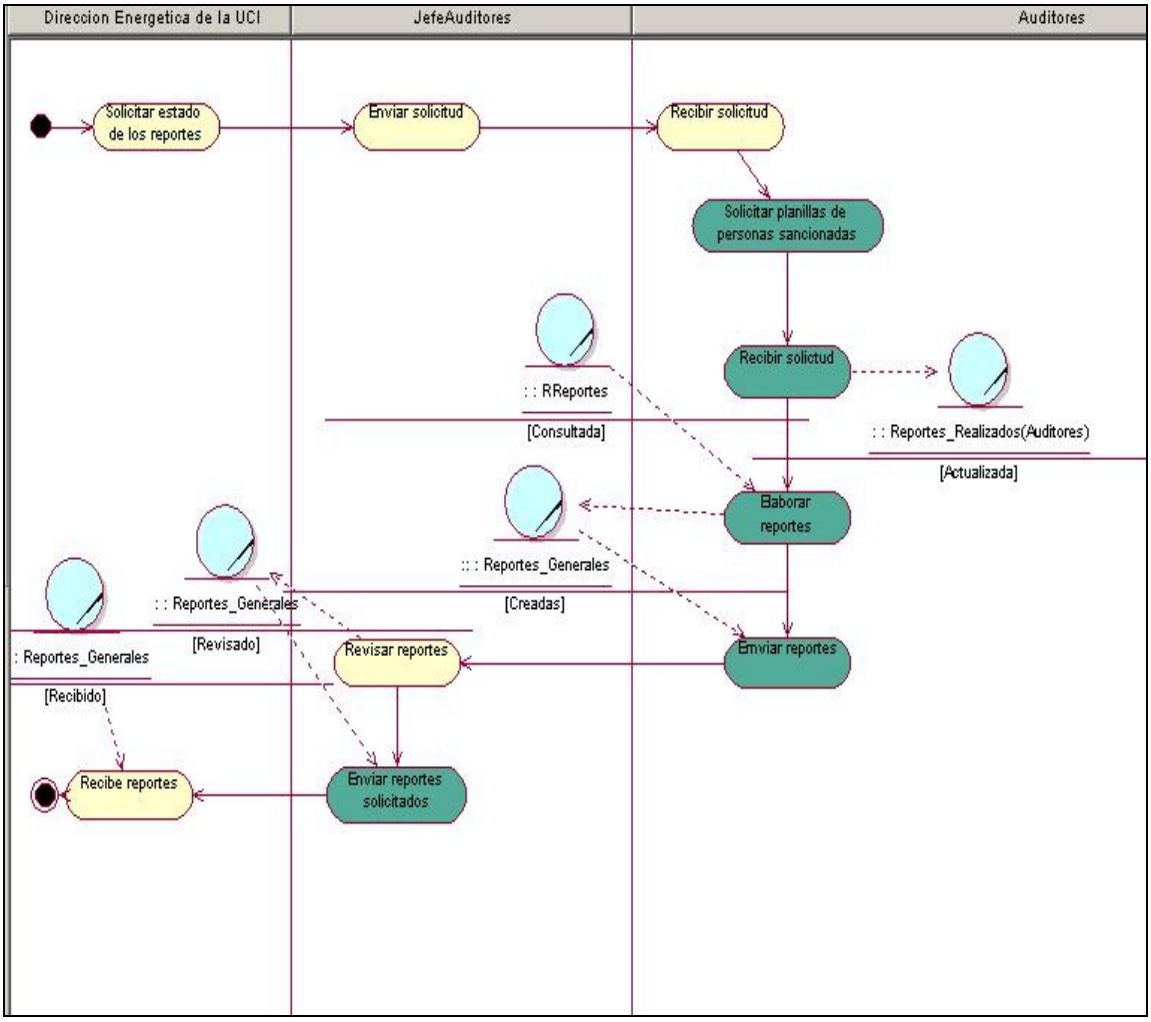


Figura 2. 3 Diagrama de Actividades

2.5 Modelo de objetos del negocio

El modelo de objetos del negocio describe cómo colaboran los trabajadores y entidades del negocio dentro del flujo.

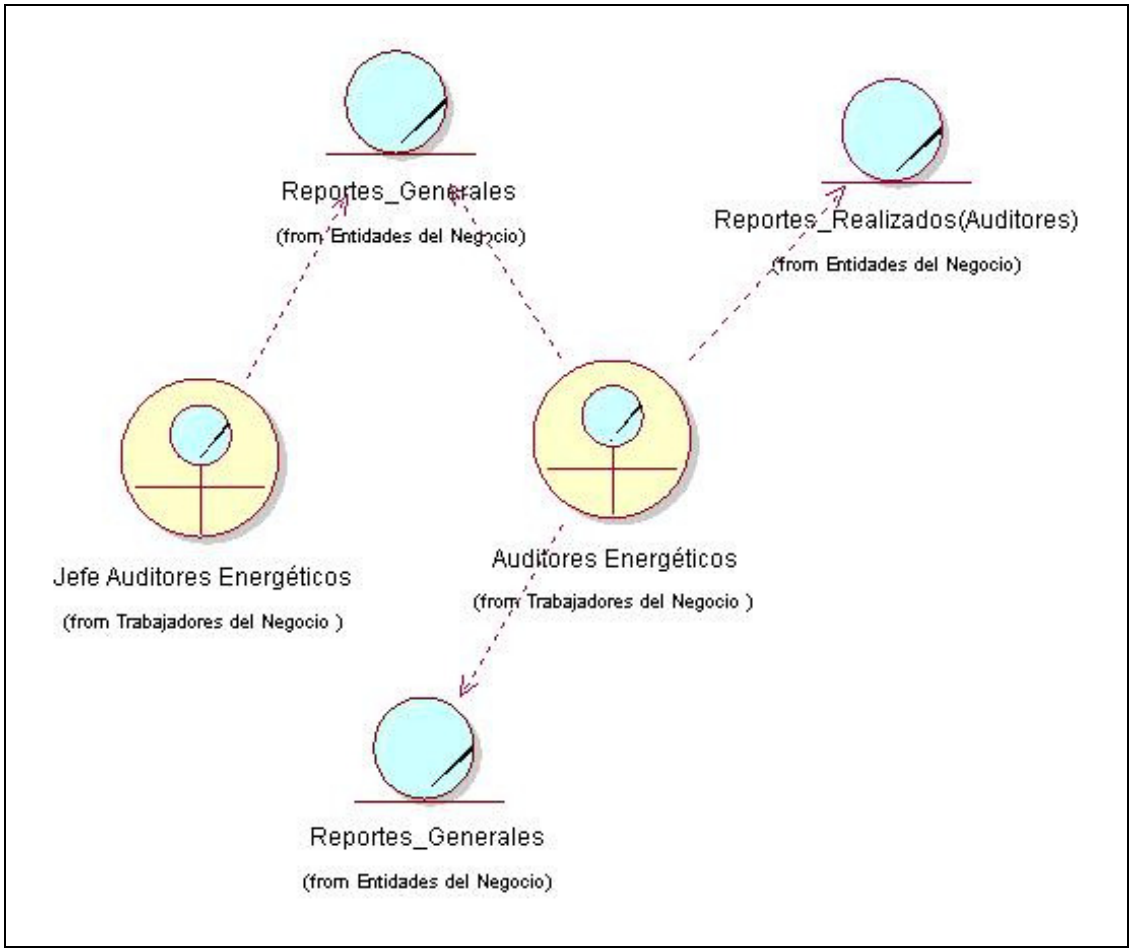


Figura 2. 4 Diagrama de Modelo de Objetos

2.6 Relación de los requerimientos

Para lograr un sistema automatizado con eficiencia y calidad se debe tener en cuenta los requerimientos funcionales y no funcionales, la captura de requisitos constituye el paso fundamental para desarrollar cualquier sistema, a continuación se muestran las especificaciones encontradas durante el proceso.

2.6.1 Requerimientos funcionales

Los Requerimientos funcionales especifican acciones que el sistema debe ser capaz de realizar, sin tomar en consideración ningún tipo de restricción física. Por lo general se describen mejor a través del modelo de Casos de uso. Por lo tanto los requerimientos funcionales especifican el comportamiento de entrada y salida del sistema y surgen de la razón fundamental de la existencia del producto. (13)

R1. Autenticar Usuario

- ✓ Registrar usuario, contraseña (del dominio UCI)

R2. Gestionar usuario (de los usuarios se conoce generalmente: nombre, apellidos, sexo, CI, correo, provincia, municipio).

- ✓ Adicionar Usuario
- ✓ Modificar Usuario
- ✓ Eliminar Usuario

R3. Gestionar reportes según los criterios:

- ✓ Salvar reportes
- ✓ Modificar Reporte
- ✓ Eliminar Reporte

R4. Gestionar violación

- ✓ Adicionar violación
- ✓ Modificar violación
- ✓ Eliminar violación

R5.Gestionar Auditor

- ✓ Adicionar Auditor
- ✓ Modificar Auditor

R6.Visualizar Datos según los criterios especificados:

- ✓ Fecha
- ✓ Facultad
- ✓ Auditores
- ✓ Área
- ✓ Violación

2.6.2 Requerimientos no funcionales

Los requerimientos 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 y confiable.

- ✓ Requerimientos de software: En la computadora de los usuarios solo se necesita un navegador Web, bajo cualquier sistema de Windows. Para su implementación se utilizará como herramienta de desarrollo: PHP5 y como gestor de base de datos: PostGreSQL: para la optimización de código se utilizo el Framework: Symfony.

- ✓ Requerimientos de hardware: El cliente necesita, al menos una computadora que contenga como mínimo 128 MB de RAM, el servidor Web tener 256 MB de RAM, todas las PC implicadas en la funcionalidad de la aplicación deben estar conectadas a la red al menos con una velocidad de 100 Mbps.
- ✓ Restricciones en el diseño y la implementación: Sé debe crear una herramienta que permita dar respuesta lo más rápido posible, garantizando así la calidad del sistema y las conexiones al servidor Web. Para garantizar que la aplicación se desarrolle con la calidad requerida se utilizará la metodología RUP, para obtener la documentación con los artefactos propuestos, y para realizar los modelos del sistema UML se utilizará la herramienta Rational Rose Suite 2003.
- ✓ Requerimientos de apariencia o interfaz externa: La interfaz debe ser amigable y fácil de usar, de manera que no ocasione problemas a los usuarios. La comunicación entre el servidor Web y la Base de Datos será a través del protocolo TCP/IP, entre las máquinas clientes y el servidor Web mediante HTTP.
- ✓ Requerimientos de Seguridad: El sistema debe autenticar al usuario antes de que pueda realizar cualquier actividad en el sistema. Garantizar que la información solo sea vista por aquella persona que tiene privilegios para verla.
- ✓ Requerimientos de Usabilidad: La aplicación solo será usada por aquellas personas que tienen previa autorización para tener acceso a la información que se maneja, en este caso que guarden relación con la Dirección Energética (los auditores energéticos y el jefe de auditores energéticos).
- ✓ Requerimientos de Soporte: El sistema debe de tener una garantía de instalación y prueba del mismo.

2.7 Modelo de casos de uso del sistema

2.7.1 Definición de los actores del sistema a automatizar

Nombre del actor	Descripción
Administrador	Es la persona que controla toda la información de la aplicación.
Usuario	Es toda persona que se autentica en el sistema, independientemente del tipo de usuario que este sea.

Tabla 2. 4 Descripción de los actores del sistema

2.7.2 Diagrama de casos de uso a automatizar

Un diagrama de casos de uso del sistema, describe una parte del modelo de casos de uso y muestra un conjunto de casos de uso y actores con una asociación entre cada par de ellos que interactúa.

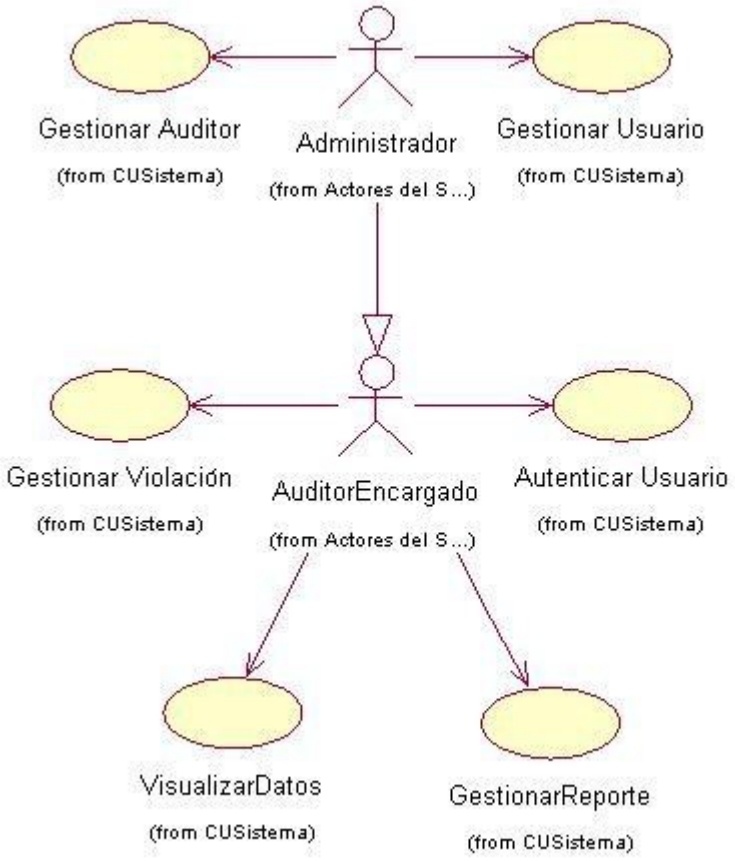


Figura 2. 5 Diagrama de Casos de Uso del Sistema

2.7.3 Descripción textual de los casos de uso del sistema

Mediante la descripción de los casos de uso del sistema se detalla paso a paso la secuencia de eventos que los actores utilizan para completar un proceso a través del sistema. Este seria el último paso para pasar a la construcción de la solución propuesta.

A continuación se muestran las descripciones de los casos de uso a automatizar en el sistema, donde se muestra de forma especificada la interacción de los actores con el sistema.

2.7.3.1 CU Autenticar Usuario

Caso de uso	CU Autenticar Usuario
Actores	Administrador , AuditorEncargado
Resumen	El caso de uso comienza cuando el usuario abre el navegador para entrar al sistema, y este pide que se autentique con usuario y contraseña del dominio UCI
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1. El actor abre la aplicación.	1.1 La aplicación muestra una interfaz de inicio de sesión.
2. El actor introduce usuario y clave de acceso, y da clic en entrar o presiona la tecla Enter.	2.1 El sistema verifica que estén los campos llenos.
	2.2 El sistema verifica que sea usuario del dominio UCI.
	2.3 El sistema verifica que el usuario esté registrado en la Base de Datos.
	2.4 Se configura el sistema con los privilegios correspondientes al usuario autenticado.

Tabla 2. 5 Descripción: Caso de uso Autenticar Usuario

Flujos Alternos

Acción3 F1: Si los campos no están completos.

Acción3 F2: Si el usuario y la contraseña no son válidos

Acción3 F3: Si el usuario no está registrado

Flujos alternos	
Acción del Actor	**Respuesta del Sistema**
Acción 3 F1	3 F1.1 La aplicación muestra el mensaje de advertencia "Especifique el (usuario o contraseña)".
3 F1.1.2 El actor acepta el mensaje	3 F1.2 La aplicación pasa a la acción 2.
Acción 3 F2	3 F2.1 La aplicación muestra el

	mensaje “usuario o contraseña son inválidos”.
	3 F2.2 La aplicación pasa a la acción 2 automáticamente.
Acción 3 F3	3 F3.1 La aplicación muestra el mensaje de error "El usuario no está registrado”.
	3 F3.2 La aplicación pasa a la acción 2 automáticamente.
Precondiciones	Que el sistema esté disponible.
Poscondiciones	El usuario accede al sistema según sus privilegios o es denegado según las acciones anteriores.

Tabla 2. 6 Flujos Alternos: CU Autenticar Usuario

Interfaz vinculada al caso de uso:

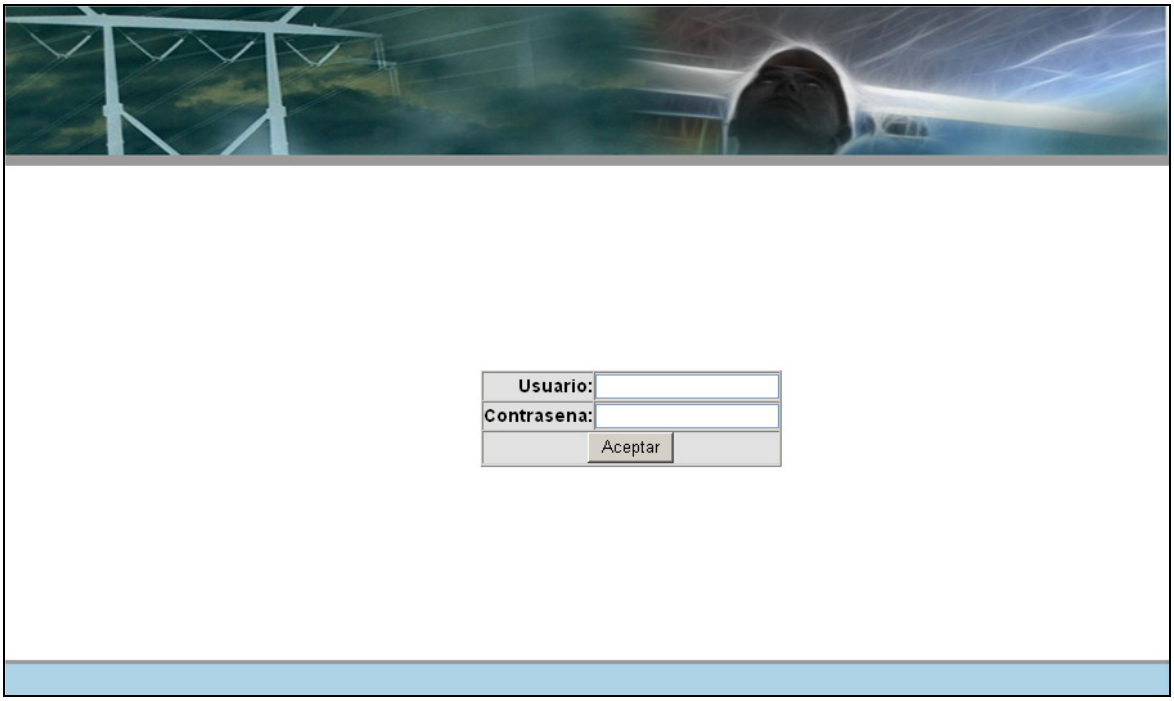


Figura 2. 6 Interfaz Autenticar

2.7.3.2 Caso de Uso Visualizar Datos

Caso de Uso	Visualizar Datos

Actores	Administrador , AuditorEncargado
Resumen	El caso de uso comienza cuando el Usuario selecciona la acción Visualizar Datos, el sistema muestra las opciones para Visualizar los Datos de los reportes según el o los criterios seleccionados por el usuario.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1. El actor escoge la opción Visualizar Datos	1.1 El sistema muestra los criterios de: ➢ Fecha ➢ Facultad ➢ Auditor ➢ Área ➢ Violación
2.El actor selecciona una de las opciones para Visualizar los datos de los reportes: 2.1 Selecciona la opción Fecha Ver Sección Fecha 2.2 Selecciona la opción facultad Ver Sección facultad 2.3 Selecciona la opción Auditor Ver Sección Auditor 2.4 Seleccionar la opción Área Ver sección Área 2.5 Seleccionar la opción Violación Ver sección Violación	
Sección Fecha	

2.1.2 El actor introduce las fechas para obtener el reporte.	2.1.1 El sistema muestra los campos a llenar. 2.1.3 El sistema verifica que estén todos los campos llenos. 2.1.4 El sistema verifica que las fechas estén registradas. 2.1.5 El sistema muestra el reporte. 2.1.6 Se pasa a la acción 2.
Sección Facultad	
2.21 El actor escoge la opción facultad. 2.2.3 El actor selecciona la facultad para obtener el reporte de las personas de la facultad implicada.	2.2.2 El sistema muestra el campo con la lista de facultades. 2.2.4 El sistema muestra el reporte. 2.2.5 Se pasa a la acción 2.
Sección Auditor	
2.3.1 El actor escoge la opción Auditor. 2.3.3 El actor selecciona el Auditor para obtener los reportes realizados por Auditor en cuestión.	2.3.2 El sistema muestra el campo con una lista de auditores. 2.3.5 El sistema muestra el reporte. 2.3.6 Se pasa a la acción 2.
Sección Área	
2.4.1 El actor escoge la opción Área. 2.4.3 El actor selecciona el Área.	2.4.2 El sistema muestra el campo con la lista de Áreas. 2.4.4 El sistema muestra el reporte. 2.4.5 Se pasa a la acción 3.
Sección Violación	
2.5.1 El actor escoge la sección Violación. 2.5.3 El actor selecciona la violación.	2.5.2 El sistema muestra el campo con la lista de violaciones. 2.5.4 el sistema muestra el reporte 2.5.5 se pasa a la acción 3.

Tabla 2. 7 Descripción: CU Visualizar Datos

Flujos Alternos

Acción 4F1: Si los campos no están completos.

Acción 4F2: Si la fecha introducida no existe.

Acción 5F1: Debe seleccionar una facultad.

Acción 5F2: Si los campos no están completos.

Acción 6F1: Debe seleccionar un auditor.

Acción 6F2: Si los campos no están completos.

Flujos Alternos	
Acción del actor	**Respuesta del sistema**
Acción 4 F1	4F1.1 El sistema muestra un mensaje de advertencia.
4 F1.2 El actor acepta el mensaje.	4F1.2 Pasa a la acción 2.1.
Acción 4 F2	4F2.1 El sistema muestra un mensaje: “La fecha introducida no existe.”
4 F2.2 El actor acepta el mensaje.	4 F2.2 Pasa a la acción 2.2.
Acción 5 F1	5 F1.1 El sistema muestra un mensaje: “Debe seleccionar una facultad.”
5F1.2 El actor acepta el mensaje.	5 F1.2 Se pasa a la acción 2.2.
Acción 5 F2	5 F2.1 El sistema muestra un mensaje de advertencia.
5 F2.2 El actor acepta el mensaje.	5 F2.2 Pasa a la acción 3.
Acción 6F1	6F1.1 el sistema muestra un mensaje:” Debe seleccionar un auditor “.
6F1.2 El actor acepta el mensaje.	6F1.2 Se pasa a la acción 2.3.
Acción 6F2	6F2.1 El sistema muestra un mensaje de advertencia.
6F2.2 El actor acepta el mensaje.	6F2.2 Pasa a la acción 3.
Precondiciones	Que esté autenticado en el sistema.

Pos condiciones	Se actualiza la Base de Datos.

Tabla 2. 8 Flujos alternos: CU Visualizar Datos

2.7.3.3 Caso de Uso Gestionar Usuario

Casos de uso	Gestionar Usuario
Actor	Administrador
Resumen	El caso de uso comienza cuando el Administrador selecciona la opción gestionar usuario. El Administrador selecciona una de las opciones para modificar, adicionar o eliminar. El caso de uso termina cuando el Administrador termina la acción seleccionada.
Flujos Alternos	
Acción del actor	**Respuesta del sistema**
1. El Administrador selecciona la opción Gestionar Usuario.	1.1 El sistema muestra el listado de usuarios registrados y las opciones de: adicionar, modificar/Eliminar al usuario.
2. El Administrador selecciona una de las opciones: 2.1 Adicionar usuario Ver Sección Adicionar Usuario 2.2 Modificar Usuario Ver Sección Modificar/Eliminar Usuario	
Sección Adicionar Usuario	
	2.1.2 El sistema muestra una interfaz con los campos a llenar: a) usuario b) tipo
2.1.3 El Administrador introduce los datos del nuevo usuario que vamos a crear y presiona el botón correspondiente para salvar los datos agregados.	2.1.4 El sistema chequea que estén todos los datos.
	2.1.5 El sistema verifica que el usuario no

	esté registrado.
	2.1.6 El sistema muestra un mensaje de confirmación “Usuario adicionado correctamente” y limpia los campos para adicionar un nuevo usuario.
	2.1.7 Se pasa a la Acción 2.
Sección (Modificar Usuario/Eliminar Usuario)	
2.2.1 El Administrador selecciona el usuario que desea modificar.	2.2.2 El sistema muestra una interfaz con los campos a modificar: a) usuario b) tipo
2.2.3 El Administrador modifica los datos del usuario.	2.2.4 El sistema verifica que los campos estén llenos.
	2.2.5 El sistema brinda la posibilidad de Modificar, cancelar, o eliminar los datos del usuario.
2.2.6 El Administrador escoge la opción que desea.	2.2.7 El sistema Actualiza los datos.
2.2.8 El Administrador escoge la opción cancelar para visualizar los datos actualizados.	

Tabla 2. 9 Descripción: CU Gestionar Usuario

Flujos Alternos

Acción 7 F1: Si los datos del usuario que se quiere adicionar están incompletos.

Acción 7 F2: Si el usuario que se quiere adicionar ya existe.

Acción 7 F3: Si los datos del usuario que se desean modificar o eliminar están incompletos.

Flujos Alternos	
Acción del actor	**Respuesta del sistema**
Acción 7 F1	7 F1.1 El sistema muestra un mensaje de advertencia.
7 F1.2 El Administrador acepta el mensaje.	7 F1.3 Pasa a la Acción 3.1
Acción 7 F2	7 F2.1 El sistema muestra un mensaje de error.
7 F2.2 El Administrador acepta el mensaje.	7 F2.3 Pasa a la Acción 3.

Acción 7 F3	7 F3.1 El sistema muestra un mensaje de advertencia.
7 F3.2 El Administrador acepta el mensaje.	7 F3.3 Pasa a la Acción 3.2
Precondiciones	Que esté autenticado en el sistema
Poscondiciones	Que se actualice la Base de Datos

Tabla 2. 10 Flujos Alternos: CU Gestionar Usuario

2.7.3.4 Caso de Uso Gestionar Violación

Caso de uso	Gestionar Violación
Actor	Administrador , AuditorEncargado
Resumen	El caso de uso comienza cuando el actor selecciona una de las opciones para modificar, adicionar o eliminar una violación. El caso de uso termina cuando el usuario termina la acción seleccionada.
Flujos Alternos	
Accion del Actor	**Respuesta del sistema**
1. El actor escoge la opción Gestionar Violación.	1.1 El sistema muestra todas las violaciones registradas y las opciones adicionar, modificar y eliminar.
2.El actor selecciona una de las opciones: 2.1 Selecciona la opción Adicionar Violación. Ver Sección Adicionar Violación. 2.2 Selecciona la opción Modificar Violación. Ver Sección Modificar /Eliminar (Violación).	
Sección Adicionar Violación	
2.1.2 El actor introduce los datos y presiona sobre el botón que brinda la opción de salvar los datos.	2.1.1 El sistema muestra una interfaz con las campos a llenar: ➢ Nombre ➢ Grado de la violación

	2.1.3 El sistema verifica que estén todos los campos llenos. 2.1.4 El sistema verifica que la nueva violación que se va a adicionar no exista. 2.1.5 El sistema limpia los campos para adicionar una nueva Violación. 2.1.6 Se pasa a la acción 2.
Sección (Modificar/Eliminar) Violación	
2.2.1 El actor selecciona la violación que va a modificar o a eliminar. 2.2.3 El actor modifica los datos del reporte y presiona el botón correspondiente a la acción a realizar. 2.2.6 El actor selecciona el botón cancelar para ver los datos actualizados en la página correspondiente.	2.2.2 El sistema muestra la interfaz con los campos a introducir: ➢ nombre ➢ el grado de violación 2.2.4 El sistema verifica que los campos estén llenos. 2.2.5 El sistema guarda los cambios realizados. 2.2.7 Se pasa a la acción 2.

Tabla 2. 11 Descripción: CU Gestionar Violación

Flujos Alternos

Acción 3 F1: Si los campos no están completos.

Acción 3 F2: Si la violación que se va a adicionar ya existe.

Acción 4 F1: Si el Usuario no selecciona la violación que desea modificar o eliminar.

Acción 4 F2: Si los campos no están completos.

Flujos Alternos	
Acción del actor	**Respuesta del sistema**
Acción 3 F1	3 F1.1 El sistema muestra un mensaje de advertencia.
3 F1.2 El actor acepta el mensaje.	3 F1.3 Pasa a la acción 2

Acción 3 F2	3 F2.1 El sistema muestra un mensaje: “ La violación ya existe”
3 F2.2 El actor acepta el mensaje.	3 F2.3 Pasa a la acción 2.
Acción 4 F1	4 F1.1 El sistema muestra un mensaje de advertencia.
4 F1.2 El actor acepta el mensaje.	4 F1.3 Se pasa a la acción 2
Acción 4 F2	4 F2.1 El sistema muestra un mensaje de advertencia.
4 F2.2 El usuario acepta el mensaje.	4 F2.3 Pasa a la acción 2.2
Precondiciones	Que esté autenticado en el sistema.
Poscondiciones	Se actualiza la Base de Datos.

Tabla 2. 12 Flujos Alternos: Gestionar Violación

2.7.3.5 Caso de uso Gestionar Reportes

Caso de uso	Gestionar Reportes
Actor	Administrador , AuditorEncargado
Resumen	El caso de uso comienza cuando el Usuario selecciona la opción para Gestionar Reporte. El caso de uso termina cuando el Usuario termina la acción seleccionada.
Flujos Alternos	
Acción del Actor	**Respuesta del sistema**
1. El actor escoge la opción Gestionar reportes.	1.1 El sistema muestra el formulario a llenar y brinda las opciones de editar, salvar o cancelar.
2. El actor selecciona una de las opciones: 2.1 Selecciona la opción Salvar Reporte Ver Sección Salvar reporte 2.2 Selecciona la opción Editar Reporte Ver sección Editar reporte 2.3 Selecciona la opción de cancelar Ver sección de Cancelar reporte	
Sección Salvar	

2.1.1 El actor pasa a llenar el reporte lo primero que hace es introducir el solapín del implicado o del testigo. 2.1.3 El actor pasa a terminar de llenar los campos restantes del reporte. Y va a la opción de salvar.	2.1.2 El sistema verifica que el solapín exista, si existe el sistema muestra en la parte superior del formulario toda la información referente al solapín introducido. 2.1.4 El sistema verifica que todos los datos sean correctos. 2.1.5 Se pasa a la acción 2.
Editar Reporte	
2.2.1 El actor selecciona la opción de Editar reporte. 2.2.3 El actor escoge una de la opciones: 2.2.4 Modificar Reporte Ver sección Modificar Reporte 2.2.5 Eliminar reporte Ver sección Eliminar Reportes	2.2.2 El sistema muestra los datos de todos los reportes salvados con la fecha actual del sistema y brinda las opciones de eliminar o modificar el reporte.
Sección Modificar Reporte	
2.2.4.1 El actor escoge la opción de Modificar reporte. 2.2.4.3 El actor hace los cambios. 2.2.4.5 El actor escoge la acción de cancelar para pasar a la acción 2.2.	2.2.4.2 El sistema muestra el formulario con el reporte que escogió el usuario. 2.2.4.4 El sistema verifica que los datos sean correctos.
Sección Eliminar reporte	
2.2.5.1 El actor selecciona la opción de Eliminar reporte.	2.2.5.2 El sistema muestra los datos actualizados. 2.2.5.3 Pasa a la acción 2.
Sección Cancelar	

2.3 El actor escoge la acción de cancelar	2.3.1 El sistema muestra la página principal.

Tabla 2.13 Descripción: CU Gestionar Reporte

Flujos Alternos

Acción 3 F1: Si los datos no están completos.

Acción 3 F2: Si los datos son incorrectos.

Acción 4 F1: Si el actor no selecciona la opción correcta.

Flujos Alternos	
Acción del actor	**Respuesta del sistema**
Acción 3 F1	3 F1.1 El sistema muestra un mensaje de advertencia.
3 F1.2 El actor acepta el mensaje.	3 F1.3 Pasa a la acción 2
Acción 4 F1	4 F1.1 El sistema muestra un mensaje de advertencia.
4 F1.2 El actor acepta el mensaje.	4 F1.3 Se pasa a la acción 2.
Acción 4 F2	4 F2.1 El sistema muestra un mensaje de advertencia.
4 F2.2 el actor acepta el mensaje.	4 F2.3 Pasa a la acción 2.2
Precondiciones	Que no existan los datos.
Poscondiciones	Salir de la sección.

Tabla 2. 14 Flujos Alternos: Gestionar Reporte

2.7.3.6 Caso de Uso Gestionar Auditor

Casos de uso	Gestionar Auditor
Actor	Administrador
Resumen	El caso de uso comienza cuando el Administrador selecciona la opción gestionar Auditor. El Administrador selecciona una de las opciones para, Adicionar, Modificar el auditor. El caso de uso termina cuando el Administrador termina la acción seleccionada.
Flujos Alternos	
Acción del actor	**Respuesta del sistema**
1. El Administrador selecciona la opción Gestionar Auditor	1.1 El sistema muestra un listado con todos los auditores.

2. El Administrador selecciona el Auditor que desea gestionar.	2.1 El sistema muestra el listado Auditores registrados y las opciones de: adicionar y modificar.
3.El Administrador selecciona una de las opciones: 3.1 Adicionar Auditor Ver Sección Adicionar Auditor 3.2 Modificar Auditor Ver Sección Modificar	
Sección Adicionar Auditor	
	3.1.2 El sistema muestra una interfaz con los campos a llenar: a) Nombre b) Apellidos
3.1.3 El Administrador introduce los datos del nuevo Auditor y presiona el botón para salvar los datos	3.1.4 El sistema chequea que estén todos los datos.
	3.1.5 El sistema verifica que el Auditor no esté registrado.
	3.1.6 El sistema muestra un mensaje de confirmación “Auditor adicionado correctamente” y limpia los campos para adicionar un nuevo usuario.
3.1.7 El Administrador selecciona el botón cancelar.	3.1.8 Se pasa a la Acción 3.
Sección (Modificar Auditor)	
3.2.1 El Administrador selecciona el auditor que desea modificar.	3.2.2 El sistema muestra una interfaz con los campos a modificar: a)Nombre b)Apellidos c)Habilitado
3.2.3 El Administrador modifica los datos del usuario.	3.2.4 El sistema verifica que los campos estén llenos.

	3.2.5 El sistema brinda la posibilidad de Modificar los datos, cancelar.
3.2.6 El Administrador escoge la opción correspondiente.	3.2.7 El sistema Actualiza los datos.
3.2.8 El Administrador selecciona cancelar para visualizar los datos actualizados.	3.2.9 Se pasa a la acción 3.

Tabla 2. 15 Descripción: CU Gestionar Auditor

Flujos Alternos

Acción 8 F1: Si los datos del usuario que se quiere adicionar están incompletos.

Acción 8 F2: Si el usuario que se quiere adicionar ya existe.

Acción 8 F3: Si el administrador no selecciona el usuario que quiere editar.

Acción 8 F4: Si los datos del usuario que se desea modificar o eliminar están incompletos.

Flujos Alternos	
Acción del actor	**Respuesta del sistema**
Acción 8F1	8 F1.1 El sistema muestra un mensaje de advertencia.
8 F1.2 El Administrador acepta el mensaje.	8 F1.3 Pasa a la Acción 3.1
Acción 8 F2	8 F2.1 El sistema muestra un mensaje de error.
8 F2.2 El Administrador acepta el mensaje.	8 F2.3 Pasa a la Acción 3.
Acción 8 F3	8 F3.1 El sistema muestra un mensaje de advertencia.
8 F3.2 El Administrador acepta el mensaje.	8 F3.3 Pasa a la Acción 3.
Acción 8 F4	8 F4.1 El sistema muestra un mensaje de advertencia.
8 F4.2 El Administrador acepta el mensaje.	8 F4.3 Pasa a la Acción 3.2
Acción 8 F5	8 F5.1 El sistema muestra un mensaje de advertencia.
8 F5.2 El Administrador acepta el mensaje.	8 F5.3 Pasa a la Acción 3
Precondiciones	Que esté autenticado en el sistema.
Poscondiciones	Que se actualice la Base de Datos.

Tabla 2. 16 Flujos Alternos: CU Gestionar Auditor

2.8 Conclusiones

Este capítulo muestra una clara definición de los requisitos que deberá cumplir el sistema, además se hizo una representación de las funcionalidades a construir mediante el diagrama de casos de usos del sistema, donde finalmente se describieron detalladamente todas las acciones del actor del sistema con los casos de uso que interactúan. El uso de los modelos de casos de usos para describir el sistema propuesto permite una adecuada captación y modelación de requerimientos, dejando demostrado su importancia en esta etapa. Gana claridad en cuanto a la concepción del sistema a construir y se establecen las bases para las restantes fases del proceso de análisis, diseño e implementación del sistema. Estos resultados serán utilizados como entrada en las etapas posteriores del trabajo.

CAPÍTULO3: ANÁLISIS Y DISEÑO DEL SISTEMA

3.1 Introducción

En este capítulo, se aborda todo lo referido a los flujos de trabajo, y parte del análisis, diseño para el desarrollo de la aplicación, teniendo en cuenta lo que se obtuvo en las etapas anteriores a estas, se representan los artefactos fundamentales generados; en el análisis se define el modelo de clases de análisis y en el diseño los diagramas de interacción; en el diseño, el diagrama de clases del diseño y el diseño de la base de datos utilizado.

3.2 Análisis

Con el flujo de trabajo análisis se obtiene una visión general del futuro sistema en función de los requisitos funcionales anteriormente definidos. En dicha etapa se analizan cada caso de uso siendo transformados a un lenguaje técnico para obtener la primera aproximación al diseño e incluso la arquitectura. De forma general, en esta etapa se han definido por caso de uso todas las clases de la interfaz externa expresando la interacción del usuario con el sistema en dependencia del nivel jerárquico que ocupe, una clase de control encapsula funciones complejas para el manejo de la información almacenada mostrando el comportamiento de los casos de usos y se definen también todas las clases entidades que modelan la información utilizada por el sistema.

3.2.1 Modelo del análisis

“El modelo de análisis nos ayuda a refinar los requisitos y permite razonar sobre los aspectos internos del sistema, incluido sus recursos compartidos internos, además, el modelo de análisis ofrece un mayor poder expresivo y una mayor formalización, como por ejemplo las que proporcionan los diagramas de interacción que se utilizan para describir los aspectos dinámicos del sistema.

El modelo de análisis también ayuda a estructurar los requisitos y proporciona una arquitectura centrada en el mantenimiento, en aspectos tales como la flexibilidad ante los cambios y la reutilización...

Esta estructura no solo es útil para el mantenimiento de los requisitos como tales, sino que también se utiliza como entrada en las actividades de diseño y de implementación”. (16)

A continuación se muestran los diagramas de clases de análisis:

3.2.1.1 Diagrama de Clases de Análisis: Autenticar Usuario

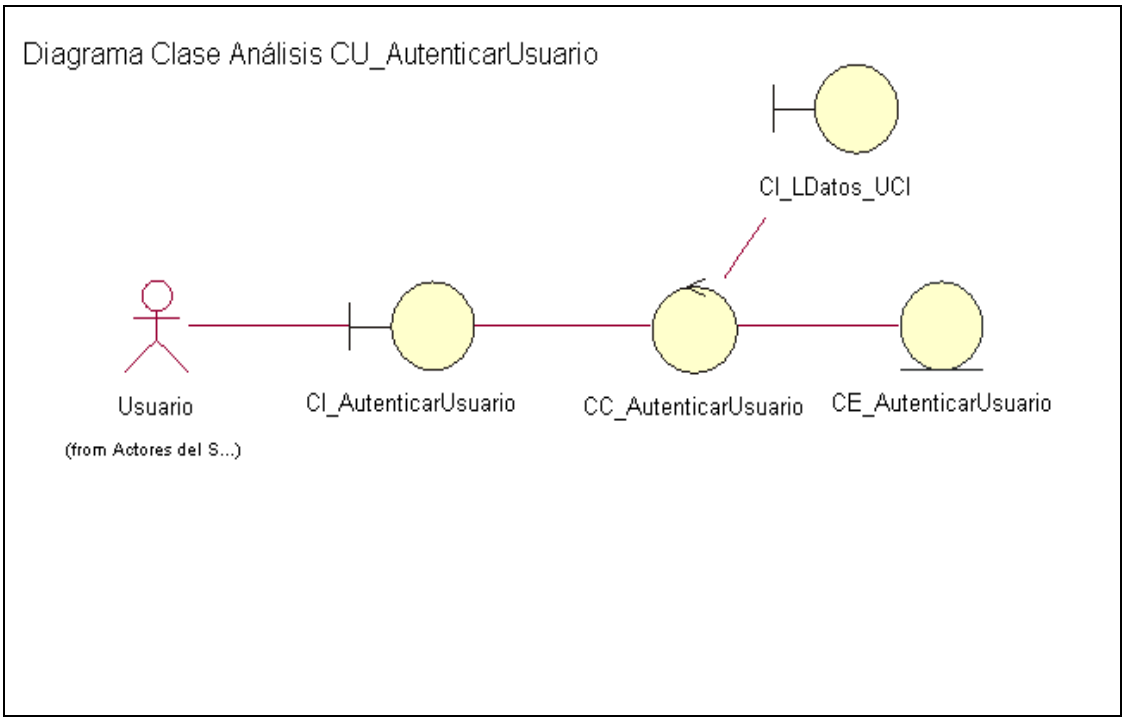


Figura 3. 1 Diagrama de Clases de Análisis: Cu _ Autenticar Usuario

3.2.1.2 Diagrama de Clases de Análisis: Gestionar Usuario

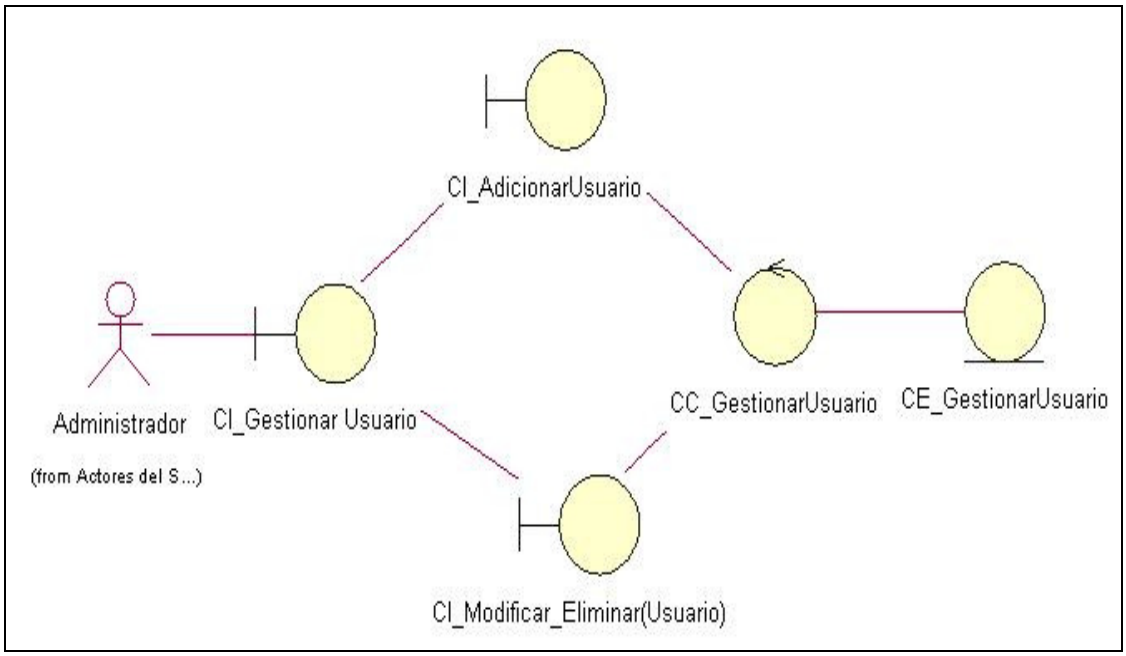


Figura 3. 2 Diagrama de Clases de Análisis: Cu _ Gestionar Usuario

3.2.1.1 Diagrama de Clases de Análisis: Gestionar Reporte

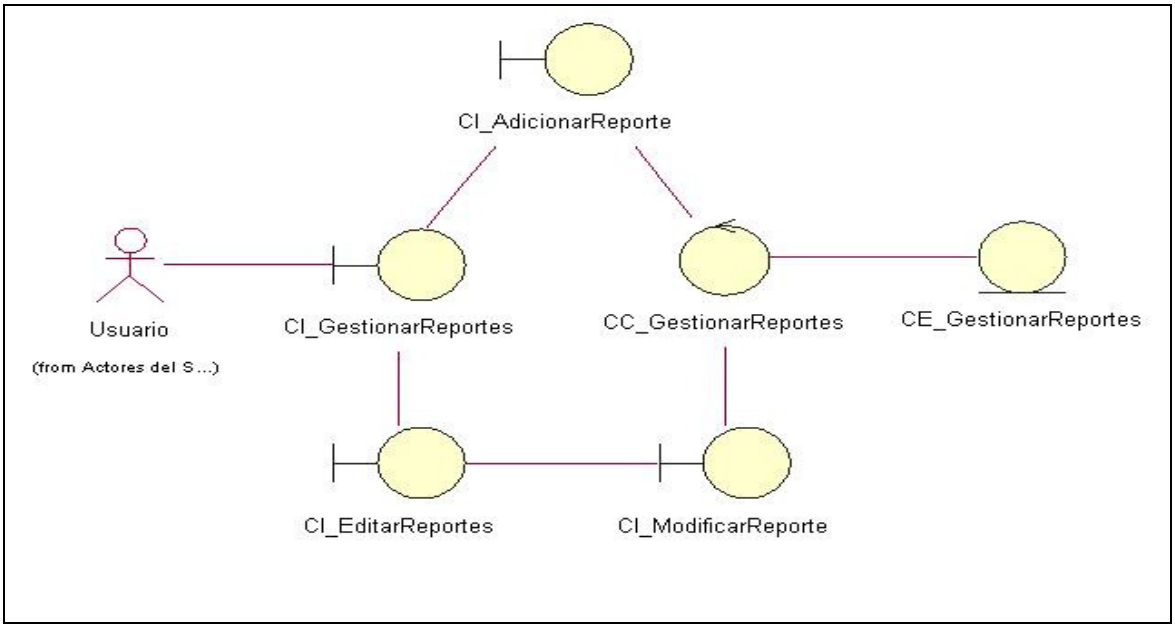


Figura 3. 3 Diagrama de Clases de Análisis: Cu _ Gestionar Reporte

El resto de los diagramas de Clase Análisis en los Anexos A.

3.3 Arquitectura y Patrones utilizados

Todo sistema o aplicación requiere de una visión futura, una arquitectura para tener una idea de como se va a implementar, para ver la forma que va a tener dicha aplicación o sistema, en este caso de software, pues bien, primero partir, ¿Qué es una arquitectura de software?

Definición:
La arquitectura de software (AS) es la parte de la ingeniería de software que se encarga de la descripción de la estructura de un sistema como una serie de componentes, con el objetivo de organizar los distintos subsistemas adecuadamente que conforman el sistema y permitir la integración de diversos grupos de desarrollo en mismo proyecto.

3.3.1 El patrón de acceso a datos ORM (Mapeo Relacional de Objetos).

Librerías de Mapeo Relacional de Objetos (más conocida por sus siglas en inglés como ORM) que se encargan de, como su nombre lo indica, mapear las bases de datos de manera tal que se trabaje con ellas como si fueran más objetos de desarrollo. La mayoría de los framework completos traen por defecto incluido estas librerías. Como es el caso de Symfony que es el framework que se usará durante la implementación del sistema.

3.3.2 Arquitectura en n-capas

El modelo n-tiers (n-capas) de informática ha emergido como la arquitectura predominante para la construcción de aplicaciones multiplataforma en la mayor parte de las empresas debido a sus grandes ventajas.

Esta arquitectura nos permite hacer que tanto la interfaz de usuario, las reglas de negocios y el motor de datos se conviertan en entidades separadas unas de otras, lo importante es mantener bien definidas las interfaces que cada una de estas expongan para comunicarse con la otra. La que más comúnmente tenemos entre nosotros es la de cuatro capas, la capa que se agrega es la que surge de separar definitivamente las reglas de negocio de la de "Acceso a Datos". Esta arquitectura nos brinda la ventaja de aislar definitivamente nuestra lógica de negocios de todo lo que tenga que ver con el origen de datos, ya que desde el manejo de la conexión, hasta la ejecución de una consulta, la manejará la capa de Acceso a Datos. De este modo, ante cualquier eventual cambio, solo se deberá tocar un módulo específico, así como al momento de plantear la escalabilidad del sistema, si se respetan las reglas básicas de diseño no deberíamos afrontar grandes modificaciones.

Principales ventajas:

- ✓ Abstracción total acerca del origen de datos: las distintas capas se especializan absolutamente en la funcionalidad que deben brindar, sin importar cual es el origen de los datos procesados.
- ✓ Bajo costo de desarrollo y mantenimiento de las aplicaciones: es más sencillo cambiar un componente que modificar una aplicación monolítica, además de que brinda un control más cercano de cada componente, así como también la posibilidad de una verdadera reutilización del código.
- ✓ Aplicaciones más robustas: debido al encapsulamiento.
- ✓ Mayor flexibilidad: se pueden añadir nuevos módulos para dotar al sistema de nueva funcionalidad.
- ✓ Alta escalabilidad: la principal ventaja de una aplicación distribuida bien diseñada es su buen escalado, es decir, que puede manejar muchas peticiones con el mismo rendimiento simplemente añadiendo más hardware. El crecimiento es casi lineal y no es necesario añadir más código para conseguir esta escalabilidad.
- ✓ Mejor calidad en las aplicaciones: como las aplicaciones son construidas en unidades separadas, estas pueden ser testeadas independientemente y con mucho más detalle, esto conduce a obtener un producto mucho más sólido.

El Sistema se implementará en la arquitectura de 3 capas (MVC).

3.3.2.1 Arquitectura de 3 capas (MVC):

- ✓ Capa de presentación o interfaz de usuario

  Esta capa está formada por los formularios y los controles que se encuentran en los formularios. Capa que interactúa con el usuario, es decir son las vistas y las clases controladoras relacionadas con cada módulo.
- ✓ Capa de negocio

  Esta capa esta formada por las operaciones lógicas que se realizaran sobre las entidades del negocio, objetos que van a ser manejados o consumidos por toda la aplicación, es decir las clases modelos de cada módulo.
- ✓ Capa de acceso a datos

  Esta capa contiene las clases que interactúan directamente con la base de datos, las operaciones que estas realizan son transparente para la capa de negocio, en este caso son la clase modelo principal, la clase que permite la conexión con la base de datos.

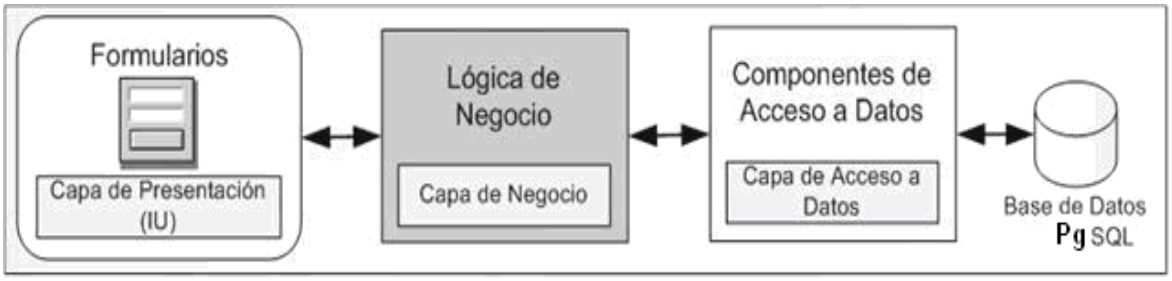


Figura 3. 4 Arquitectura en tres capas

La utilización de este tipo de arquitectura proporciona las siguientes ventajas:

- ✓ Centralización de los aspectos de seguridad y transaccionalidad, que serían responsabilidad del modelo.
- ✓ No replicación de lógica de negocio en los clientes: esto permite que las modificaciones y mejoras sean automáticamente aprovechadas por el conjunto de los usuarios, reduciendo los costes de mantenimiento. Mayor sencillez de los clientes.
- ✓ Desarrollos paralelos (en cada capa).
- ✓ Aplicaciones más robustas debido al encapsulamiento.
- ✓ Mantenimiento y soporte más sencillo (es más sencillo cambiar un componente que modificar una aplicación monolítica).

- ✓ Mayor flexibilidad (se pueden añadir nuevos módulos para dotar al sistema de nueva funcionalidad).
- ✓ Alta escalabilidad. La principal ventaja de una aplicación distribuida bien diseñada es su buen escalado, es decir, que puede manejar muchas peticiones con el mismo rendimiento simplemente añadiendo más hardware. El crecimiento es casi lineal y no es necesario añadir más código para conseguir esta escalabilidad.

Como tecnología, la arquitectura de 3 capas proporciona gran cantidad de beneficios para las empresas que necesitan soluciones flexibles y fiables para resolver complejos problemas inmersos en cambios constantes.

3.3.3 Symfony

Tal como se describe Symfony en el Capítulo 1, Symfony es un framework diseñado para agilizar el desarrollo de aplicaciones Web a partir de sus características claves. Separa la lógica del negocio, la lógica del servidor y la presentación de la aplicación. Contiene numerosas herramientas y clases con el objetivo de acortar el tiempo. De desarrollo de una aplicación compleja y le permite al desarrollador centrarse en los aspectos específicos de la aplicación al automatizar las tareas comunes. El resultado de estas ventajas es que no se necesita reinventar la rueda cada vez que se construye una nueva aplicación.

3.3.4 Implementación del MVC por Symfony

Symfony ha sido escrito completamente en PHP5 con el objetivo de aprovechar todas las ventajas de esta Versión del lenguaje y se basa en el patrón arquitectónico MVC, implementándolo de modo que el desarrollo de aplicaciones sea rápido y sencillo. Además se hace una separación en capas más allá del MVC tal como se muestra en la figura 3.4

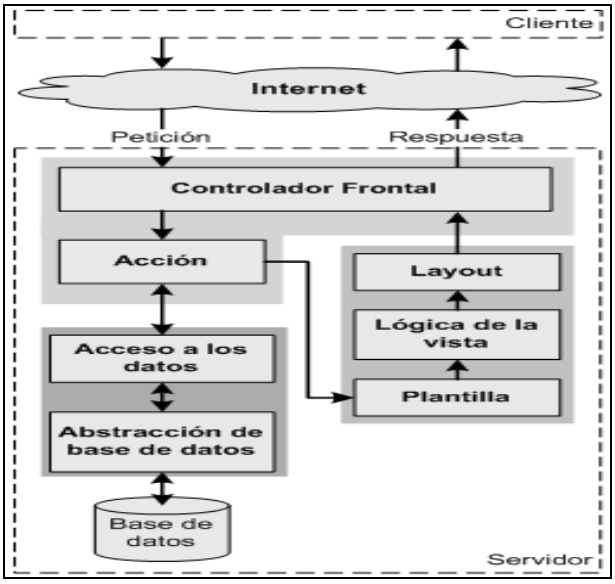


Figura 3.4: Esquema del MVC en Symfony

En esta variante el controlador que contiene el código que liga la lógica de negocio con la presentación, está compuesto por el controlador frontal que es el único punto de entrada a la aplicación y las acciones que se encargan de verificar la validez de las peticiones y preparar los datos para la vista. Esta última está formada por el layout que posee el código común a todas las acciones, las plantillas que contienen el código asociado a cada acción en particular y la lógica que se puede manipular a través de un fichero de configuración sencillo sin necesidad de programarla. Por último, el modelo está formado por la capa de acceso a datos cuyas clases son generadas automáticamente por Propel (ORM) en función de las estructura de datos de la aplicación y por la capa de abstracción de la base de datos Creole, que es completamente transparente al programador; así si se cambia el sistema gestor de bases de datos en cualquier momento, no se debe reescribir ni una línea de código, siendo necesario solamente modificar un parámetro en un archivo de configuración.

3.3.4.1 Tareas del controlador frontal

El controlador frontal es el encargado de despachar las peticiones de los usuarios, pero ello implica algo mas que detectar cual es la acción que se ejecuta. A continuación se muestran cuales son las tareas que realiza:

1. Define las constantes del núcleo.
2. Localiza las librerías de Symfony.
3. Carga e inicializa las clases del núcleo del framework.
4. Carga la configuración.
5. Decodifica la URL de la petición para determinar la acción a ejecutar y los parámetros de la petición.
6. Si la acción no existe, redirecciona a la acción error 404.
7. Activa los filtros.
8. Ejecuta los filtros, primera pasada.
9. Ejecuta la acción y produce la vista.
10. Ejecuta los filtros, segunda pasada.
11. Muestra la respuesta.

3.3.4.2 Filtros

Los filtros son un mecanismo que Symfony usa para realizar las tareas que son comunes a toda la aplicación, tales como la validación y la seguridad. Cada petición se procesa como una cadena de filtros que son ejecutados de forma sucesiva.

1. sfRenderingFilter: encargado de renderizar la vista.
2. sfBasicSecurityFilter: chequea la seguridad de cada petición.
3. sfCacheFilter: controla el mecanismo de cache del framework.
4. sfCommonFilter: añade los ficheros java script y css a la respuesta.
5. sfFlashFilter: elimina las variables flash de la sesión.
6. sfExecutionFilter: se encarga de validar los parámetros de la petición, la ejecución de la acción y de la vista.

Representación gráfica en la figura 3.8

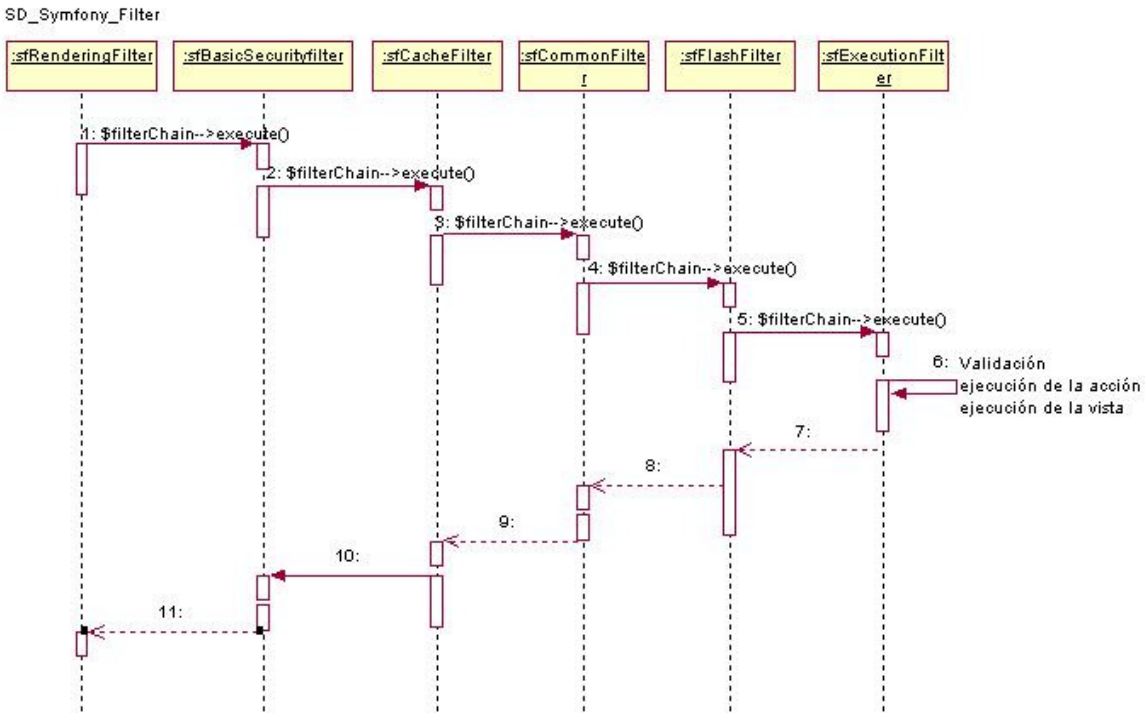


Figura 3.5: Cadena de filtros.

3.3.5 Organización de la aplicación

Symfony organiza el código fuente en una estructura de tipo proyecto y almacena los archivos del proyecto en una estructura estandarizada de tipo árbol.

En Symfony un proyecto se considera como “un conjunto de servicios y operaciones disponibles bajo un determinado nombre de dominio y que comparten el mismo modelo de objetos”. Dentro de un proyecto, las operaciones se agrupan de forma lógica en aplicaciones y cada aplicación está formada por uno o más módulos. Un módulo normalmente representa a una página Web o un grupo de páginas con un propósito relacionado y almacenan las acciones, que representan cada una de las operaciones que

se pueden realizar en un módulo. La IAW seguirá una estructura aproximada a la que se muestra en la figura 3.9

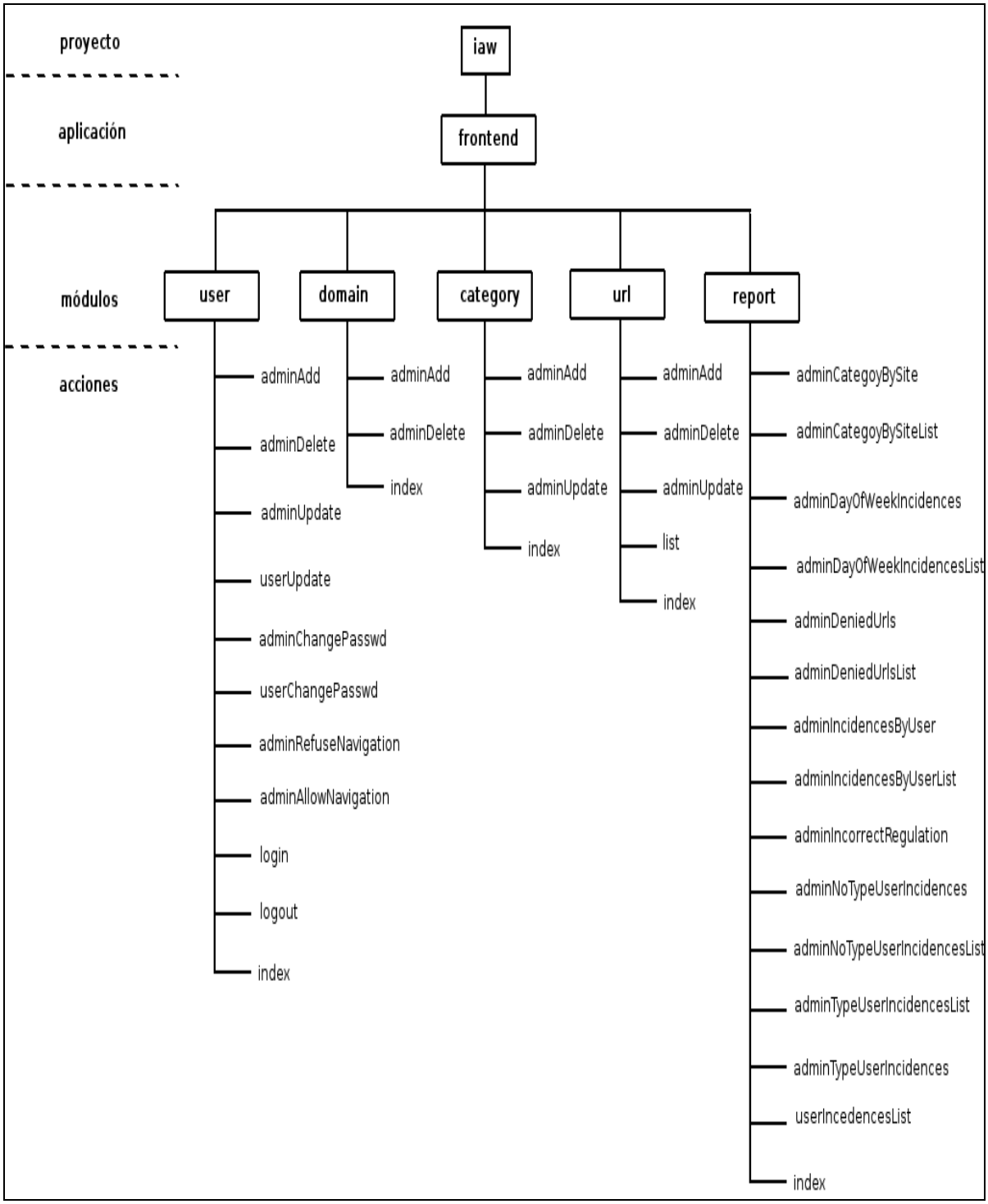


Figura 3.6: Organización de la aplicación

3.3.6 Patrones de diseño que se usan

Symfony sigue las mejores prácticas y patrones de diseño para el desarrollo de aplicaciones Web. Seguidamente se mencionan cuales son lo patrones de diseño más significativos que

se utilizan. Las clases que conforman el núcleo están basadas en el patrón Factory Method permitiendo de la creación fácil de los objetos y su extensión de forma sencilla. El patrón Decorator es usado para decorar el layout de la aplicación con el témplate asociado a cada acción, dando como resultado la vista que es mostrada al usuario. Esto se ilustra en la figura 3.7.

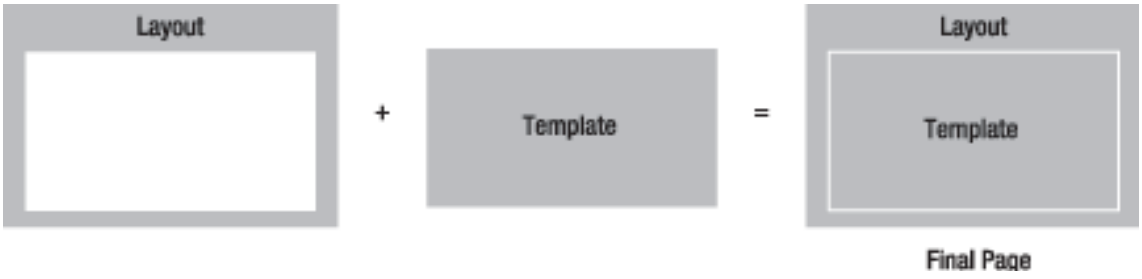


Figura 3.7: Patrón Decorator entre el layout y el templete

El patrón Singleton es usado en Symfony para permitir el acceso desde cualquier lugar de la aplicación a los objetos relacionado con el núcleo del framework.

La ejecución de la secuencia de filtros se basa en el patrón Chain of Responsibility que permite que más de un objeto pueda manejar una petición.

Los patrones anteriores pertenecen a los conocidos como GoF 1. Además de ellos se utilizan otros descritos por Martin Fowler en, entre los que están Front controller, Row Data Gateway y Table Data Gateway.

Los dos últimos son usados en las clases de acceso a datos y permiten realizar operaciones sobre un registro de una tabla y un conjunto de estos respectivamente. Patrones de diseño orientado a objetos.

3.3.7 Validación y tratamiento de errores

La validación y el tratamiento de errores en la aplicación se han diseñado en torno a los mecanismos que Symfony provee. Posee un mecanismo de validación en el servidor a través del uso de ficheros de validación y una serie de validadores que permite hacer este proceso más fácil y extensible, sin la consecuente promoción de errores que puede traer consigo la programación relacionada con este aspecto. No obstante hay casos especiales donde es necesario hacer la validación manualmente o combinar ambos métodos. Una característica importante de Symfony en este sentido es que permite el relleno automático de datos repopulación, lo que asegura la obtención de datos correctos y mejora la

experiencia de los usuarios. Tal como se mencionó anteriormente la validación se realiza como parte de la ejecución del filtro sfExecutionFilter. La figura 3.11 muestra como se realiza este proceso. Para el tratamiento de excepciones se utilizará el mecanismo provisto por PHP y las clases que para este fin Symfony posee.

3.3.8 Seguridad

Autenticación y autorización

Cuando se diseña una aplicación Web es importante definir los aspectos relacionados con la seguridad de la misma. En este contexto es importante prestar atención a la autenticación y autorización de los usuarios. Symfony provee todo lo necesario para que ambas cosas puedan realizarse de forma sencilla, mejorando el mecanismo de sesiones que utiliza PHP y haciéndolo más configurable y fácil de usar. Una vez que un usuario se autentica, se le asignan los privilegios necesarios (credenciales) para que pueda utilizar las funcionalidades que brinda la aplicación según el rol que desempeñe. A partir de los requisitos funcionales y no funcionales, y tal como se había planteado, la IAW poseerá dos tipos de usuarios diferentes, un administrador y un usuario común o básico. De este modo solo será necesario definir la seguridad y la credencial o credenciales que se requieran para la ejecución de cada una de las acciones, pues la responsabilidad de chequear que un usuario esté correctamente autenticado y posea los privilegios necesarios será del filtro sfBasicSecurityFilter.

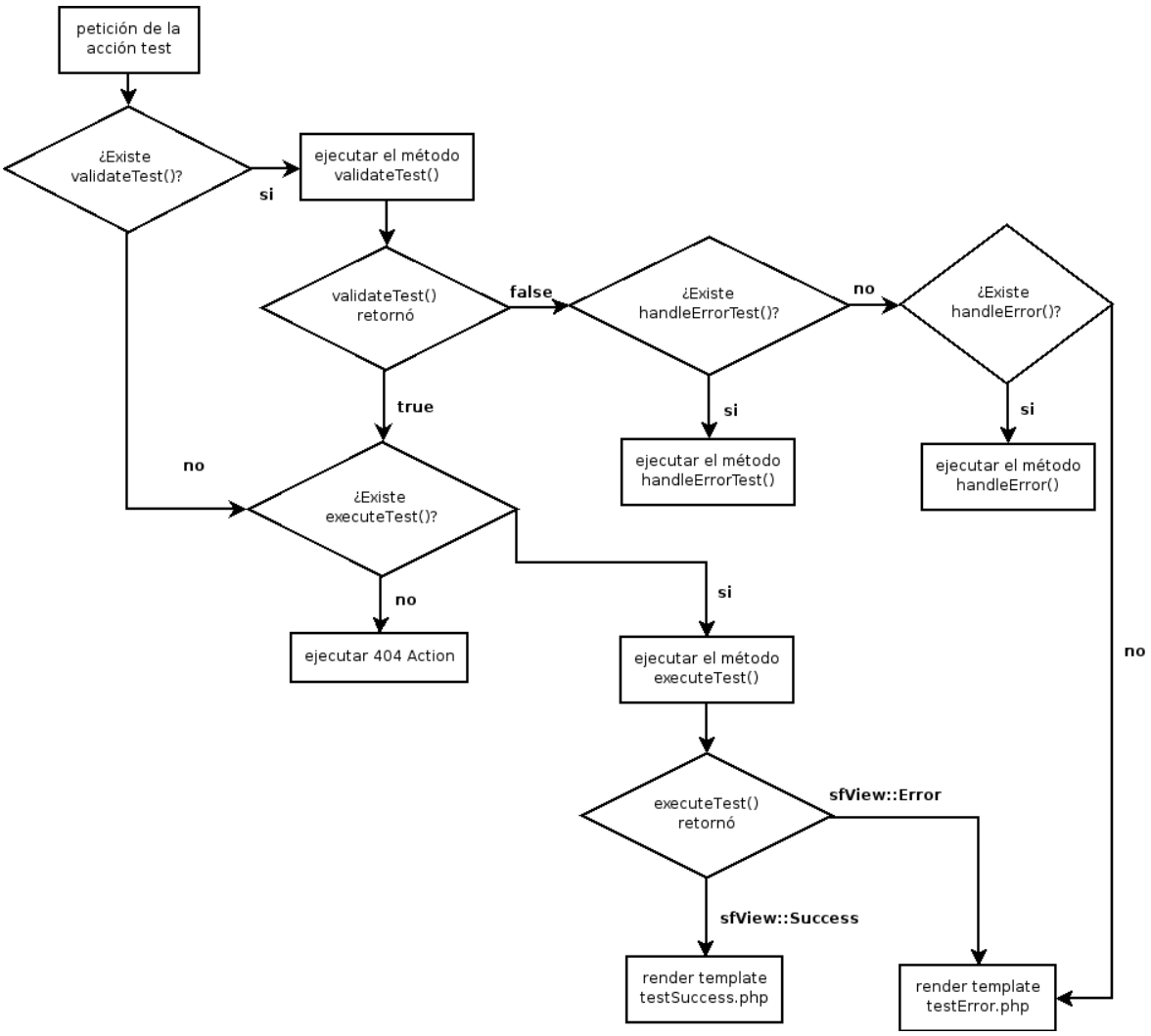


Figura 3.8 Diagrama de flujo del proceso de validación.

3.3.9 Defensa contra ataques

Existen numerosos ataques informáticos que pueden afectar a las aplicaciones Web. Cuando se produce alguno de ellos es porque se explota alguna vulnerabilidad existente relacionada con los principios de diseño e implementación y con los mecanismos de control o defensa con que se cuenten. A continuación se describen brevemente los principales ataques que podrían afectar la seguridad de la aplicación y por tanto la integridad de los datos que se manejan, así como los principios que regirán la prevención de los mismos en el contexto de Symfony. Cross Site Scripting (XSS) es un tipo de ataque que aprovecha la confianza que un usuario posee en un sitio determinado, se basa en la inyección de códigos hechos en java script que pueden impedir el uso normal de una aplicación. Para su prevención Symfony posee un mecanismo de escape que puede ser activado mediante un parámetro en un fichero de configuración.

Cross Site Request Forguery (CSRF) aunque parezca parecido, es contrario al XSS ya que este se basa en aprovechar la confianza que un sitio posee en un usuario para realizar peticiones, al mismo tiempo que puede comprometer su seguridad. Aunque la versión que se utiliza no posee defensa contra el mismo, existe una extensión que permite añadirlo al framework como un filtro, que se insertará en la secuencia de filtros después del sfBasicSecurityFilter. SQL injection es un tipo de ataque que permite realizar acciones con el objetivo de corromper la información de un sistema o ganar acceso al mismo través de la entrada de sentencias SQL maliciosas por medio de formularios o como parámetros de la petición en la URL. EL uso de Propel y Creole ayuda a prevenirlo, ya que ambos poseen los mecanismos de escape necesarios para este tipo de ataque. La mayor preocupación relacionada con el uso de sesiones está relacionada con el identificador de la sesión. Generalmente para obtener un identificador válido, un atacante puede utilizar tres variantes, la pre-dicción, la captura y una tercera conocida en inglés como Sesión Fixation. La primera y la segunda pueden ser mitigadas por el carácter altamente aleatorio que posee la generación de identificadores de sesión en PHP y por el uso de SSL para la transmisión de datos entre cliente y servidor. Sin embargo, en la última el atacante puede lograr que la víctima use un identificador de sesión elegido por él, para así obtener un identificador válido. Como medio para prevenirlo no se utilizará la URL para enviar el identificador de sesión y se regenerará uno nuevo para cada usuario una vez que se autentique en la aplicación. Tal como se mencionó anteriormente el uso de SSL para la transmisión de los datos contribuye a la integridad de los mismos y a la vez fortalece la defensa prevista contra los ataques anteriores. Por ello la aplicación utilizará el protocolo HTTPS para todas sus funcionalidades. Otros ataques que pueden afectar la aplicación es la denegación de servicios (DoS) a partir del envío masivo de peticiones al servidor, pueden ser prevenidos a través de reglas en el cortafuegos que utiliza el sistema Filpacon y con las configuraciones del servidor Web relacionadas con la cantidad de usuarios permitidos de forma concurrente, en relación con el consumo de la aplicación y los recursos que este posea. La protección contra el acceso físico al servidor recae en la institución donde se utilice el Sistema de Filtrado y en los controles de seguridad que ella posea la misma.

3.4 Diseño del sistema

Con el flujo de trabajo Diseño, se modela el futuro sistema que da soporte a los requisitos funcionales planteados en la etapa de la captura de requisitos. También se aterrizan todos los aspectos con las restricciones y características del sistema como lo son el lenguaje de programación a utilizar, el sistema operativo donde se podrá ejecutar la aplicación, las

tecnologías de interfaz de usuario, en fin agrupar en el diseño los requerimientos no funcionales definidos.

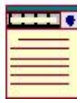
3.4.1 Modelo del diseño

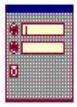
El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Además, el modelo de diseño sirve de abstracción de la implementación del sistema y es, de ese modo, utilizada como una entrada fundamental de las actividades de implementación.

3.4.2 Extensión para el modelado de aplicaciones Web

En el caso del presente trabajo se lleva a cabo el diseño para una aplicación Web siendo la aproximación más concreta del diagrama de clases del análisis y utilizándose los estereotipos para las clases en función de este tipo de aplicación que es una extensión que propone UML.

Los estereotipos son los que se muestran a continuación:

 <<Client Page>>: una instancia de página cliente, es una página Web con formato HTML; mezcla datos, presentación y lógica. Son interpretadas por el browser. Cada página cliente solo puede ser construida por una página servidor.

 <<Form>>: grupo de elementos entrada que son parte de una página cliente. Se relaciona directamente con la etiqueta de igual nombre del HTML. Sus atributos son los elementos de entrada al formulario (Text Field, Text Area, Button, Label, radio Button, radio Group, Select, Check Box y Hidden Fields).

 <<Server Page>> representa la página Web que tiene el código que se ejecuta en el servidor. Este código interactúa con recursos del servidor. Las operaciones representan las funciones del código y los atributos las variables dentro del alcance de la página.

3.4.3 Diagramas de Iteración

Los Diagramas de Interacción no son más que la forma de representar el modo en que interactúan los objetos en una secuencia de tiempo, además de los mensajes que se envían entre ellos ordenadamente. En UML los Diagramas de Interacción pueden representarse a través de los Diagramas de Secuencia y/o Diagramas de Colaboración.

Para la construcción de los Diagramas de Interacción para los casos de uso, se seleccionó el Diagrama de Secuencia porque muestra de forma más detallada la interacción entre los objetos.

A continuación se muestran algunos diagramas:

3.4.4 Diagramas de Secuencia

3.4.4.1 Diagrama de Secuencia: Autenticar Usuario

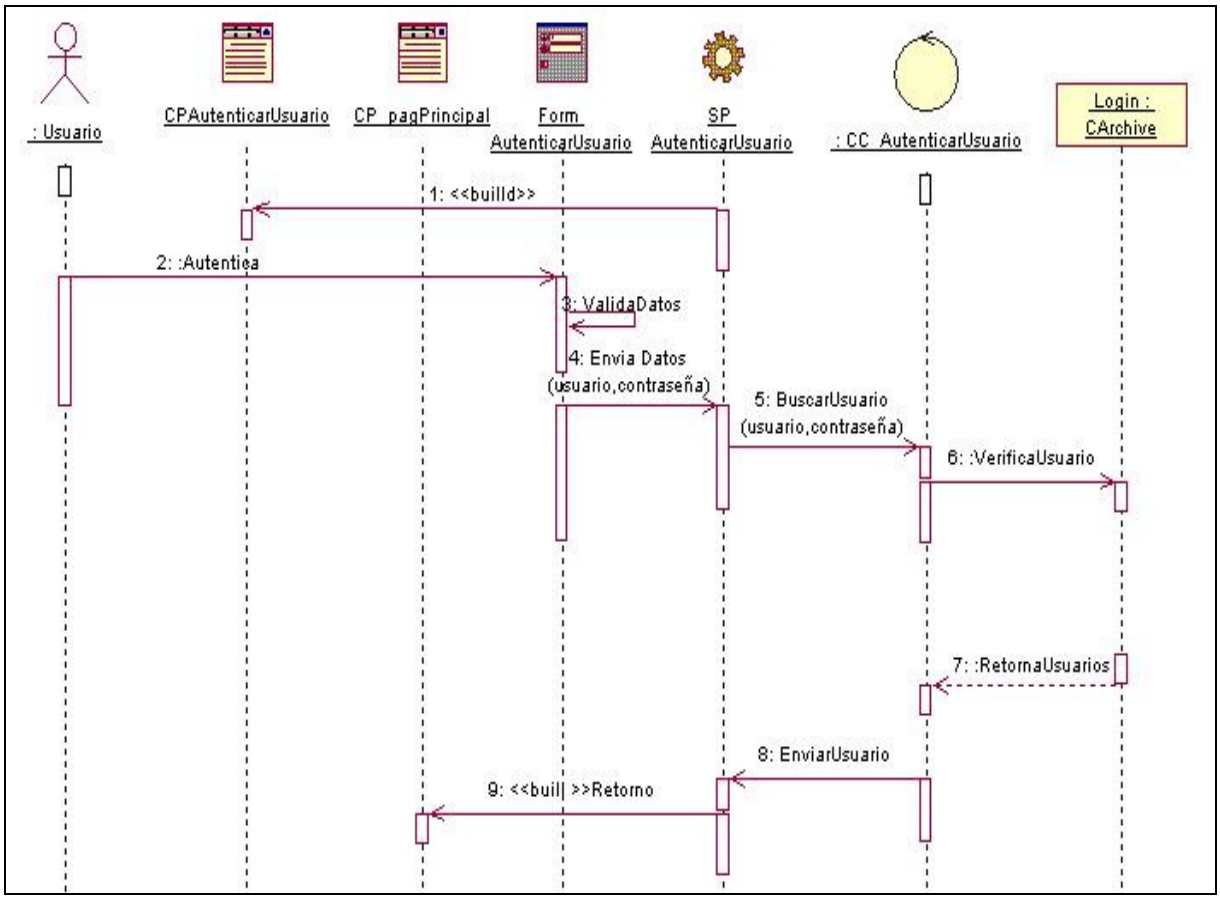


Figura 3. 9 Diagrama de Secuencia: CU Autenticar Usuario

3.4.4.2 Diagrama de Secuencia: Gestionar Usuario (Escenario Adicionar Usuario).

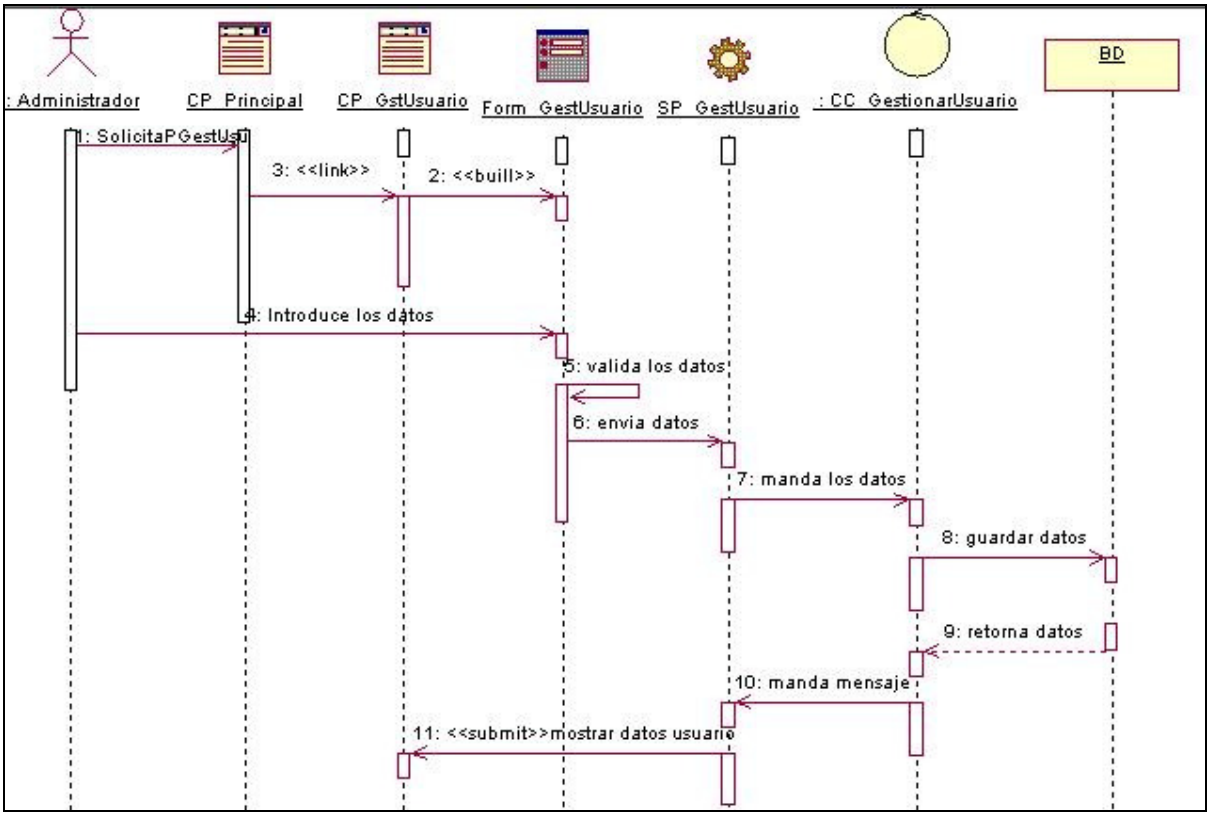


Figura 3.10 Diagrama de Secuencia: Gestionar Usuario (Escenario Adicionar Usuario).

3.4.4.3 Diagrama de Secuencia: Gestionar Usuario (Escenario Modificar Usuario).

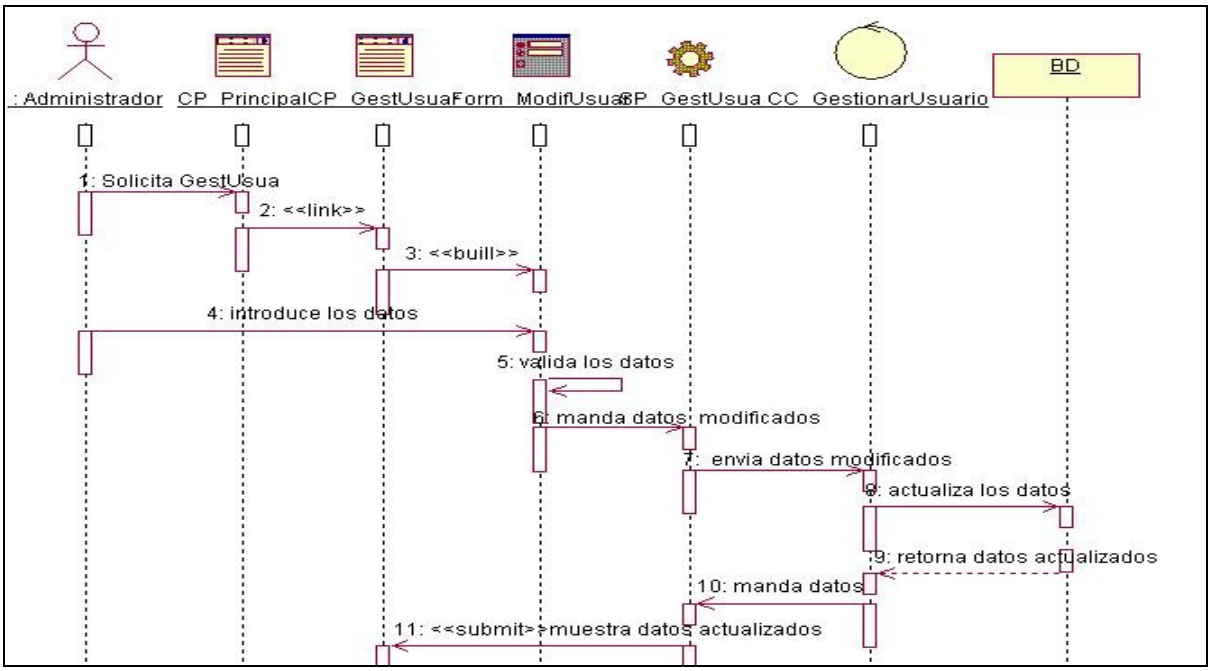


Figura 3. 11 Diagrama de Secuencia: Gestionar Usuario (Escenario Modificar Usuario).

3.4.4.4 Diagrama de Secuencia: Gestionar Usuario (Escenario Eliminar Usuario).

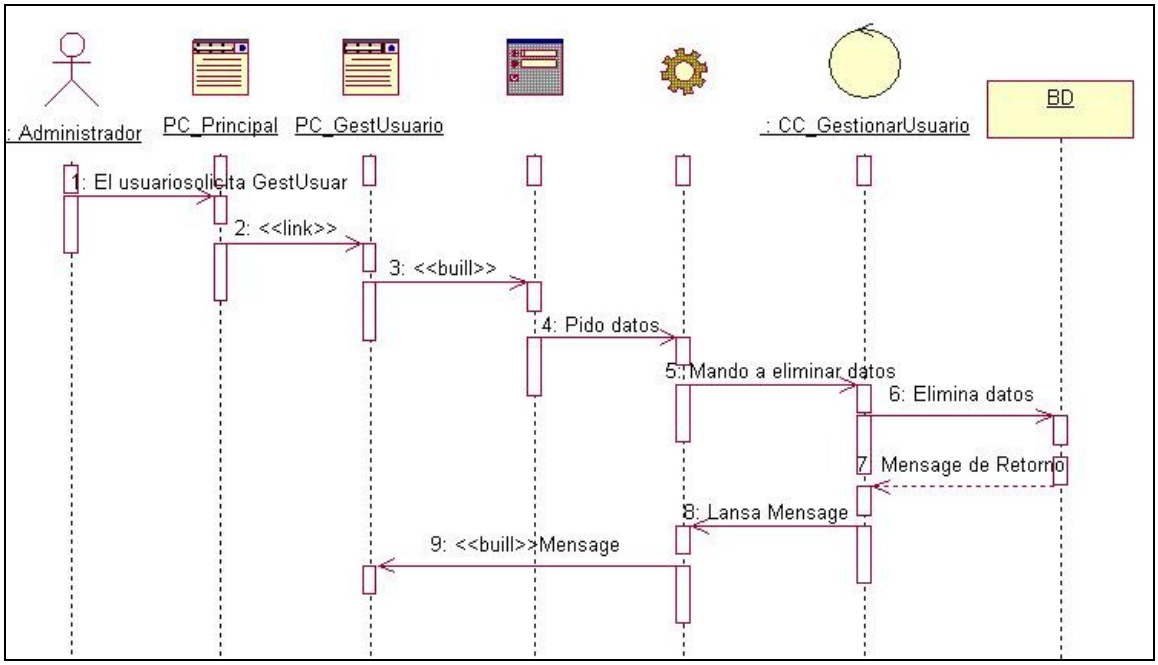


Figura 3. 12 Diagrama de Secuencia: Gestionar Usuario (Escenario Eliminar Usuario).

3.4.4.5 Diagrama de Secuencia: Gestionar Reporte (Escenario Salvar Reporte).

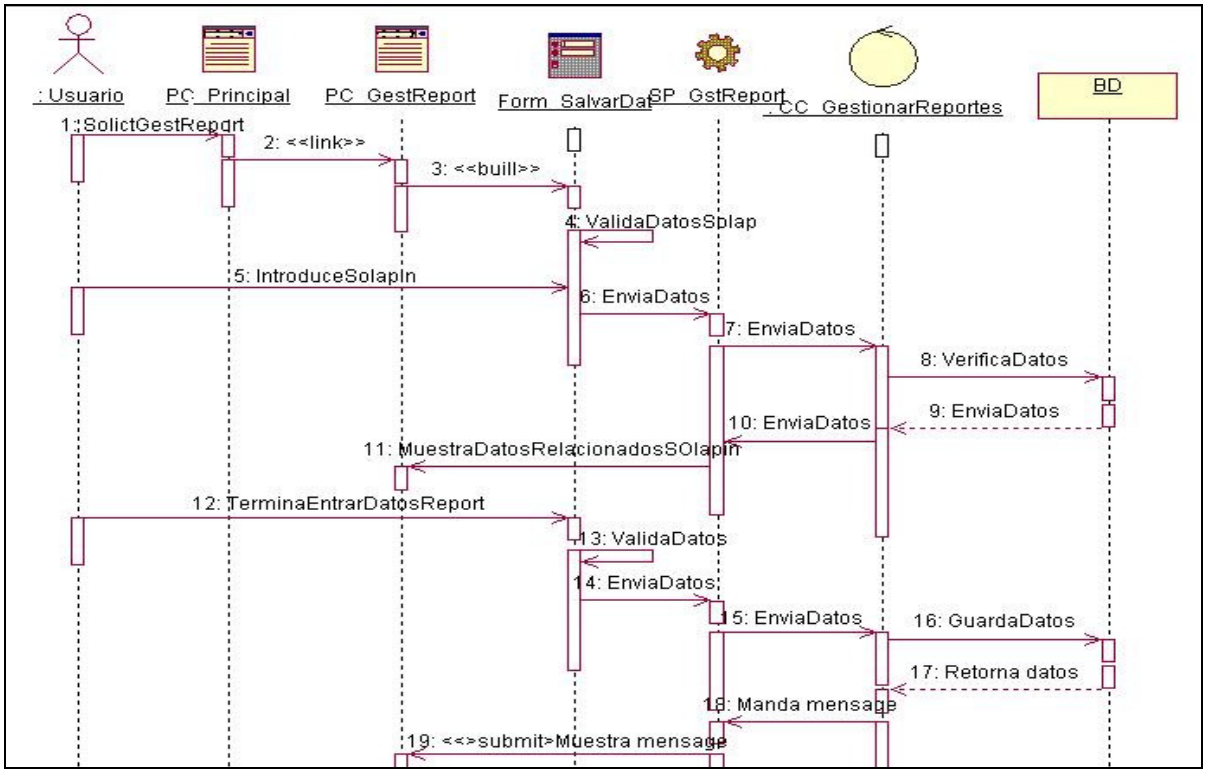


Figura 3.13 Diagrama de Secuencia: Gestionar Reporte (Escenario Salvar Reporte).

3.4.4.6 Diagrama de Secuencia: Gestionar Reporte (Escenario Editar Reporte).

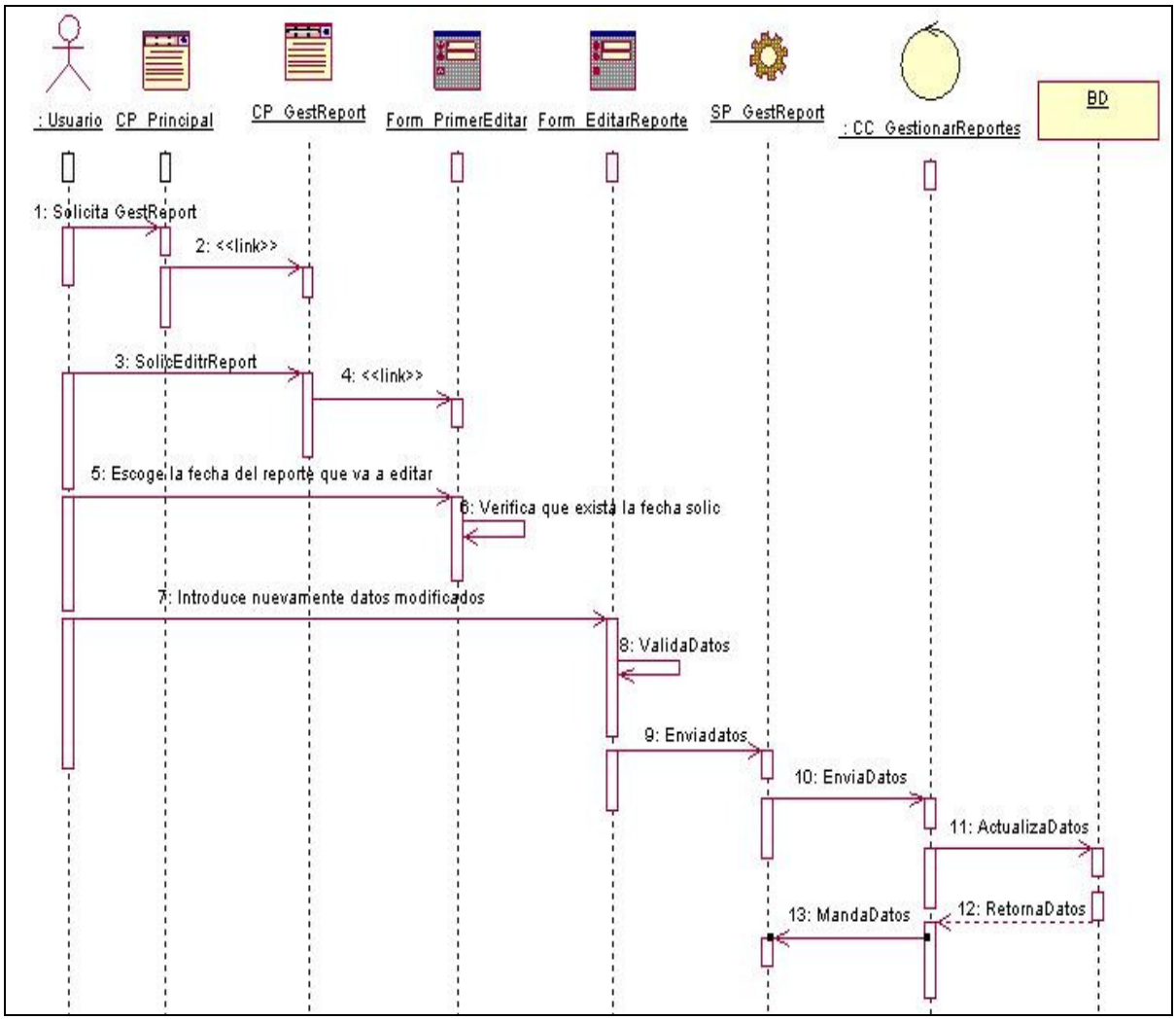


Figura 3.14 Diagrama de Secuencia: Gestionar Reporte (Escenario Editar Reporte).

El resto de los diagramas se encuentran en el Anexo C.

3.4.5 Diagramas de clases del diseño

El diagrama de diseño se obtiene a partir del refinamiento del modelo conceptual y se basa fundamentalmente en los diagramas de interacción. A continuación se muestran tres de los diagramas:

3.4.5.1 Diagrama de clase de diseño Autenticar Usuario

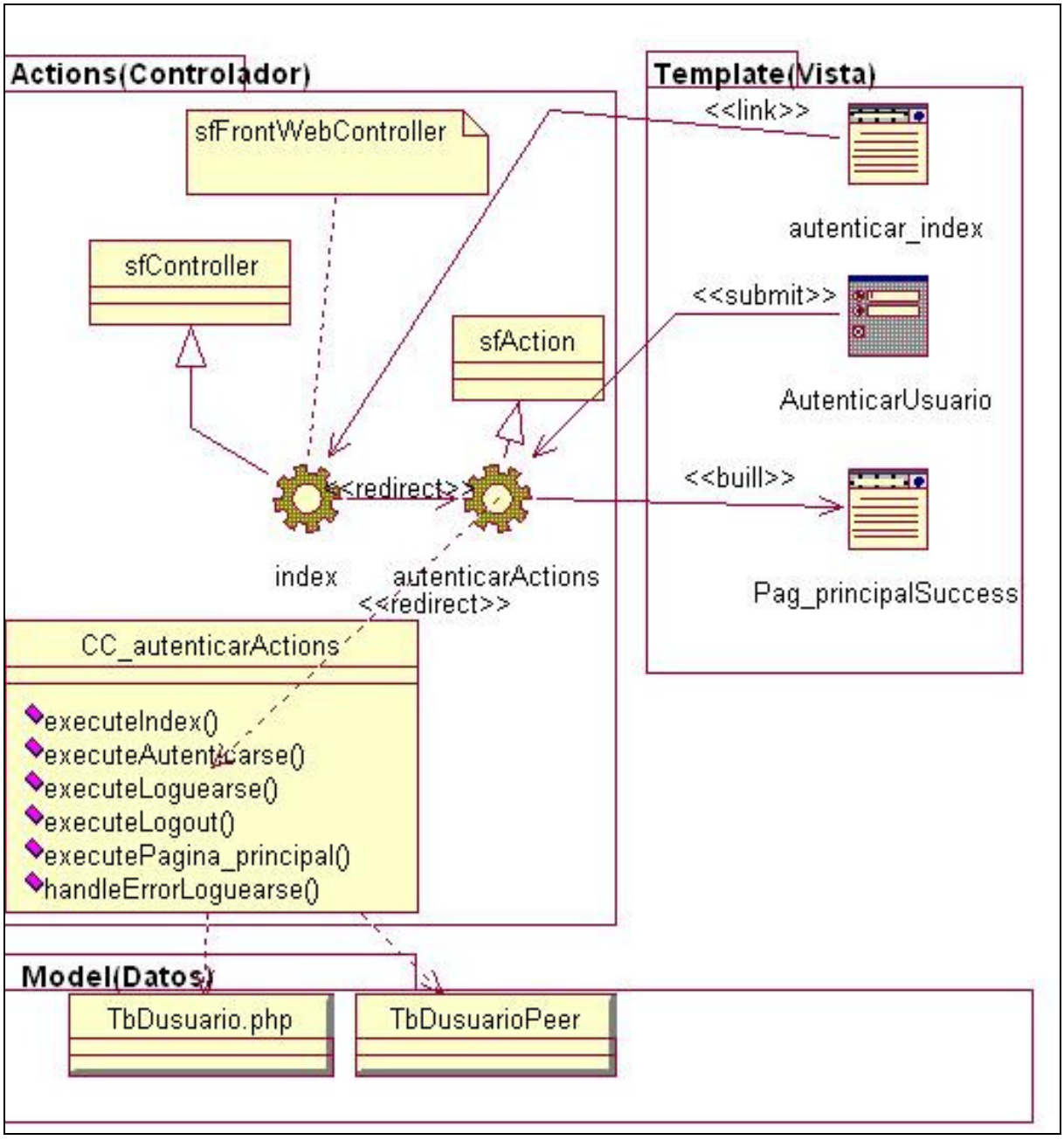


Figura 3.15 Diagrama de clase de diseño Autenticar Usuario

3.4.5.2 Diagrama de clase de diseño Gestionar Usuario

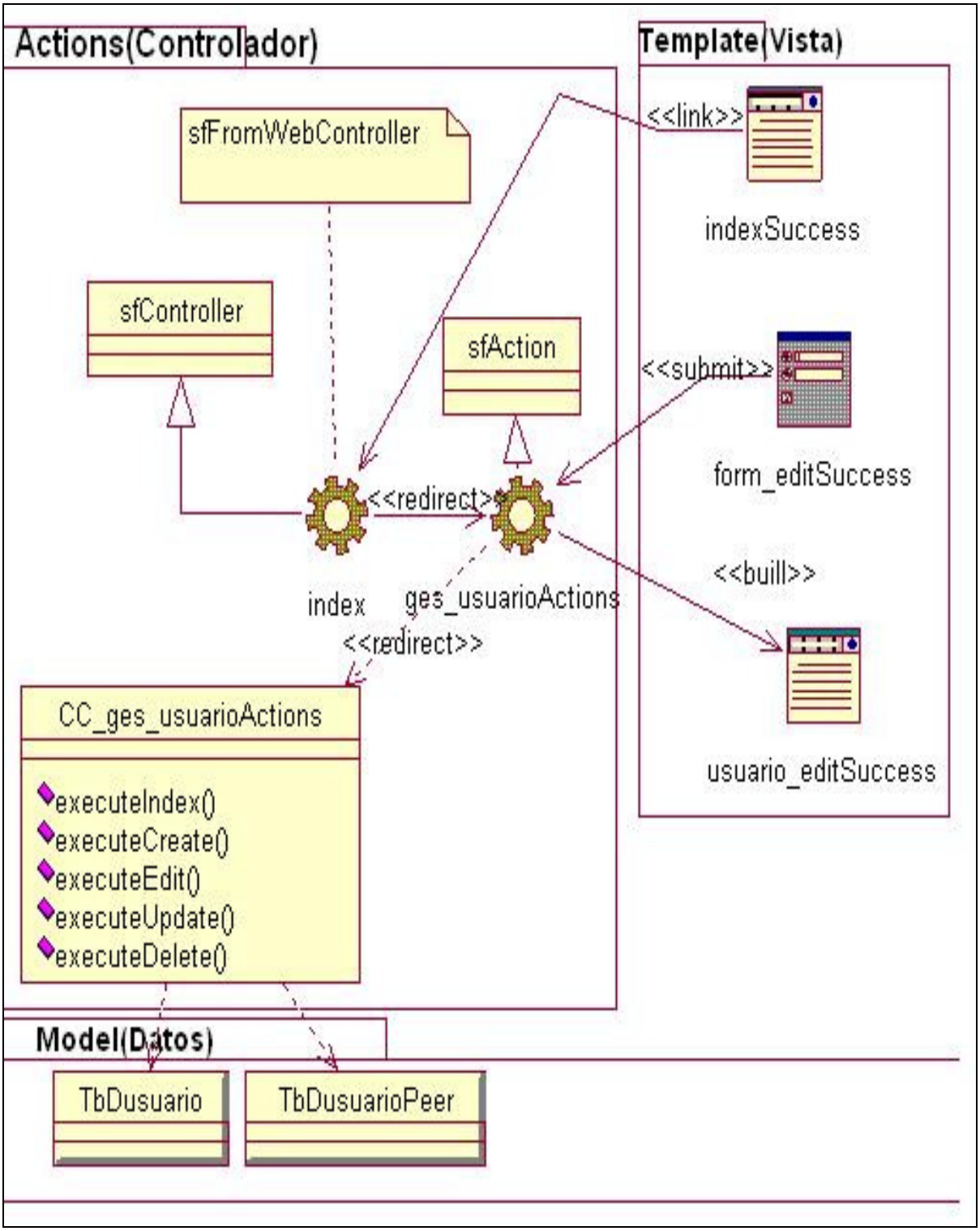


Figura 3.16 Diagrama de clase de diseño Gestionar Usuario

3.4.5.5 Diagrama de clase de diseño Gestionar Reporte

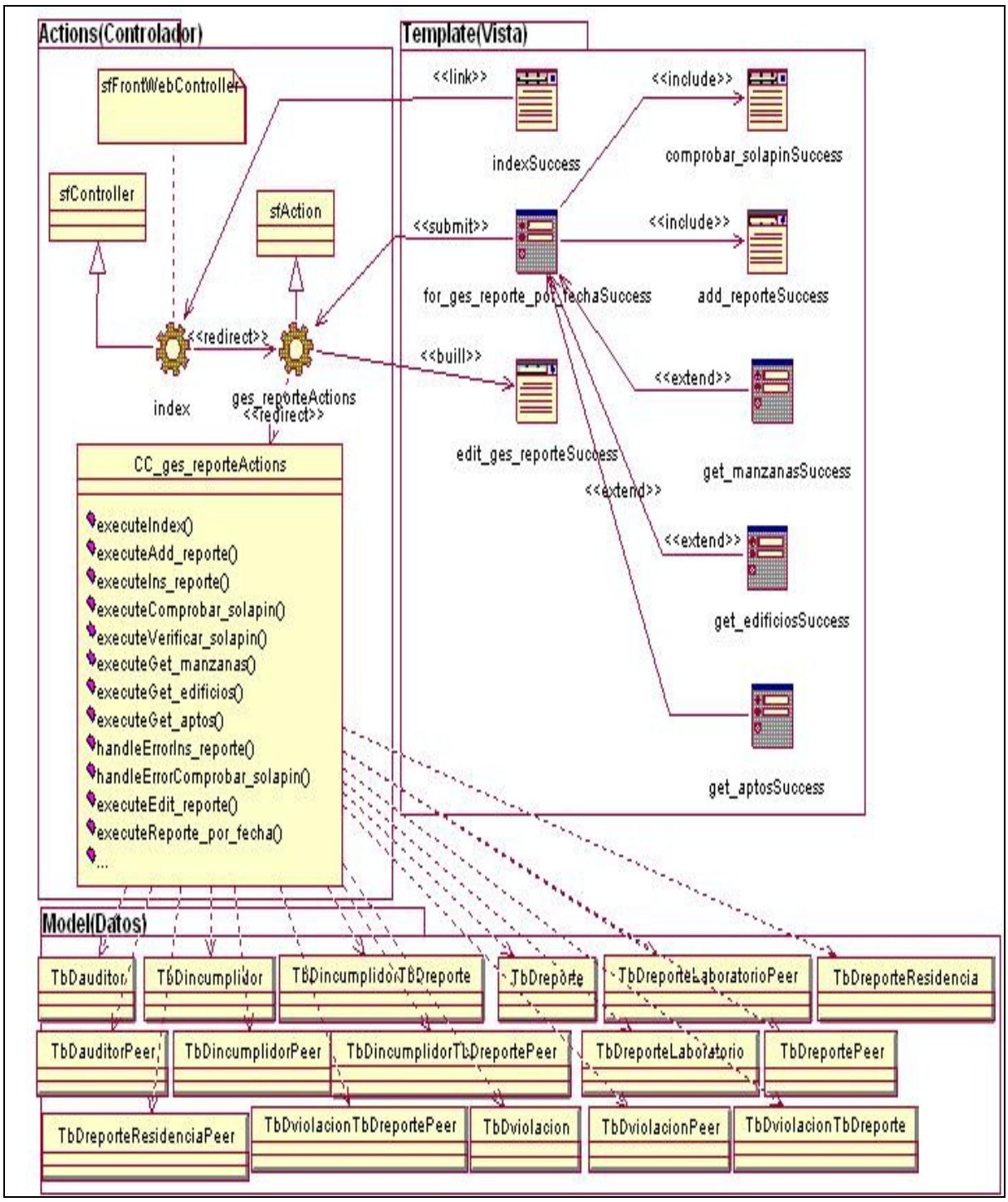


Figura 3.17 Diagrama de clase de diseño Gestionar Reporte

3.5 Diseño de la base de datos

Las tablas se originan a partir del modelo conceptual. A continuación se muestran los diagramas del modelo lógico y físico de datos.

3.5.1 Modelo lógico de datos (diagrama de clases persistentes)

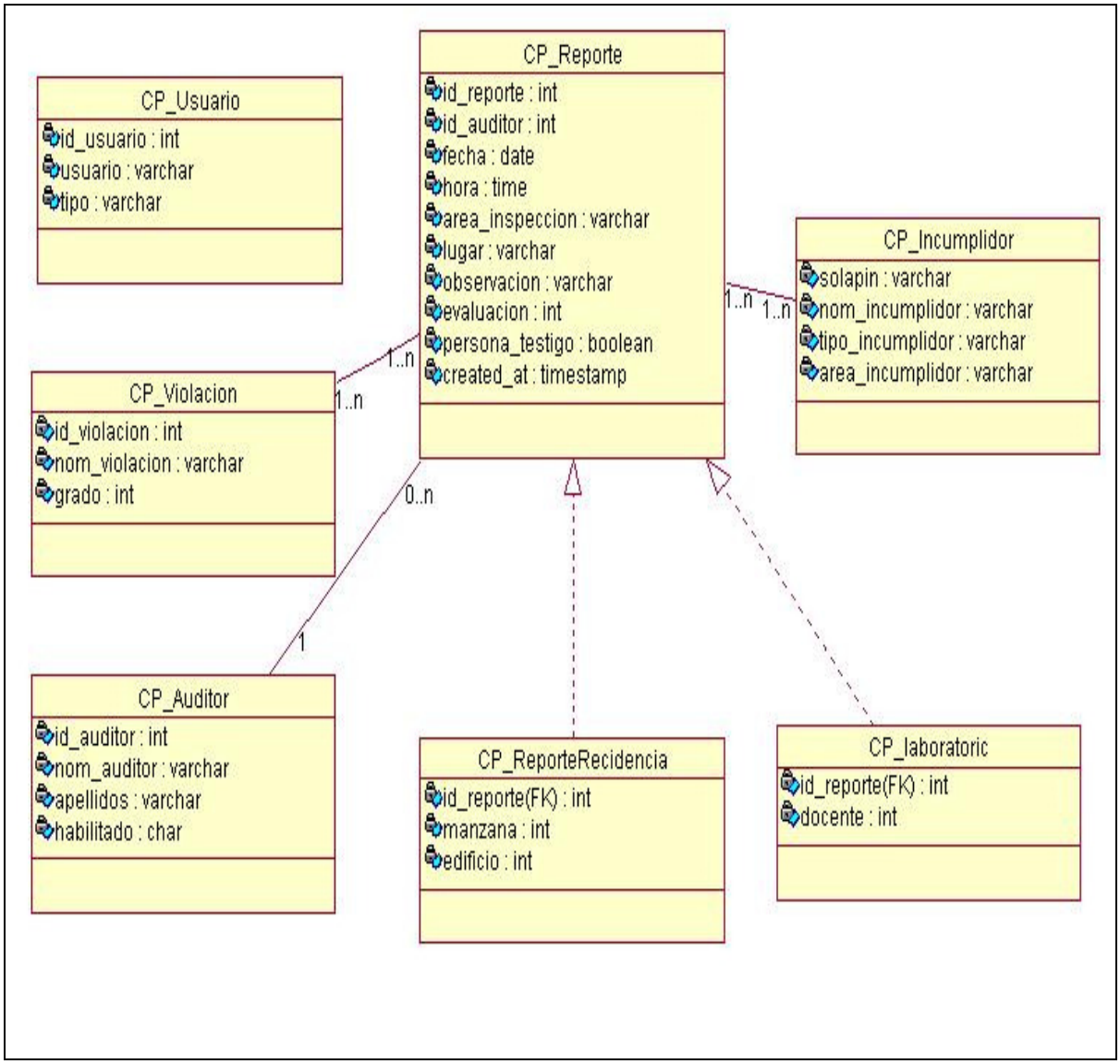


Figura 3.18 Diagrama de Clases Persistentes

3.5.2 Modelo físico de datos (modelo de datos)

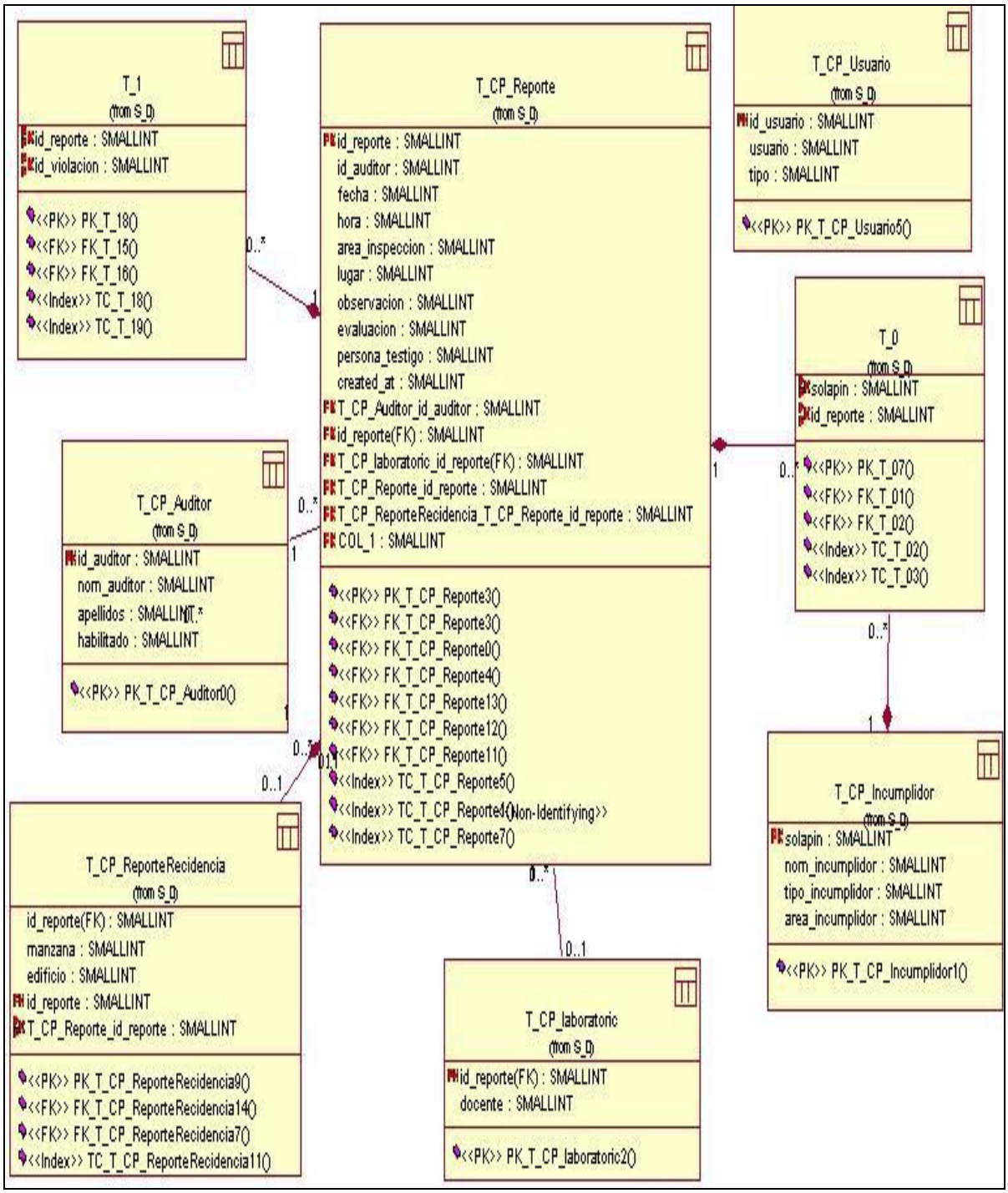


Figura 3.19 Modelo físico de datos

3.5 Conclusiones

La elaboración de algunos artefactos relacionados con el flujo de análisis y diseño teniendo en cuenta el funcionamiento de la arquitectura MVC que Symfony establece, permitió obtener una mejor comprensión del sistema y definir los principios que guiarán la implementación y organización del mismo. Además se analizaron las estrategias a seguir para dar respuesta a los requisitos no funcionales relacionados con la seguridad, se obtuvo un prototipo de interfaz y el diseño para el sistema.

CAPÍTULO 4: IMPLEMENTACIÓN Y RESULTADOS.

4.1 Introducción

El flujo de implementación y prueba es el más eminente de la fase de construcción en RUP. Como resultado del mismo, deben traducirse los modelos generados en las fases previas en componentes que formen el sistema final, reflejando cuales son, su relación y como habrán de distribuirse en los nodos físicos a través de nuevos diagramas. La realización de pruebas también adquiere alta importancia pues permite vislumbrar la existencia de defectos en la implementación realizada.

4.2 Implementación

4.2.1 Diagrama de Despliegue

El diagrama de despliegue muestra las relaciones entre los componentes hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software.

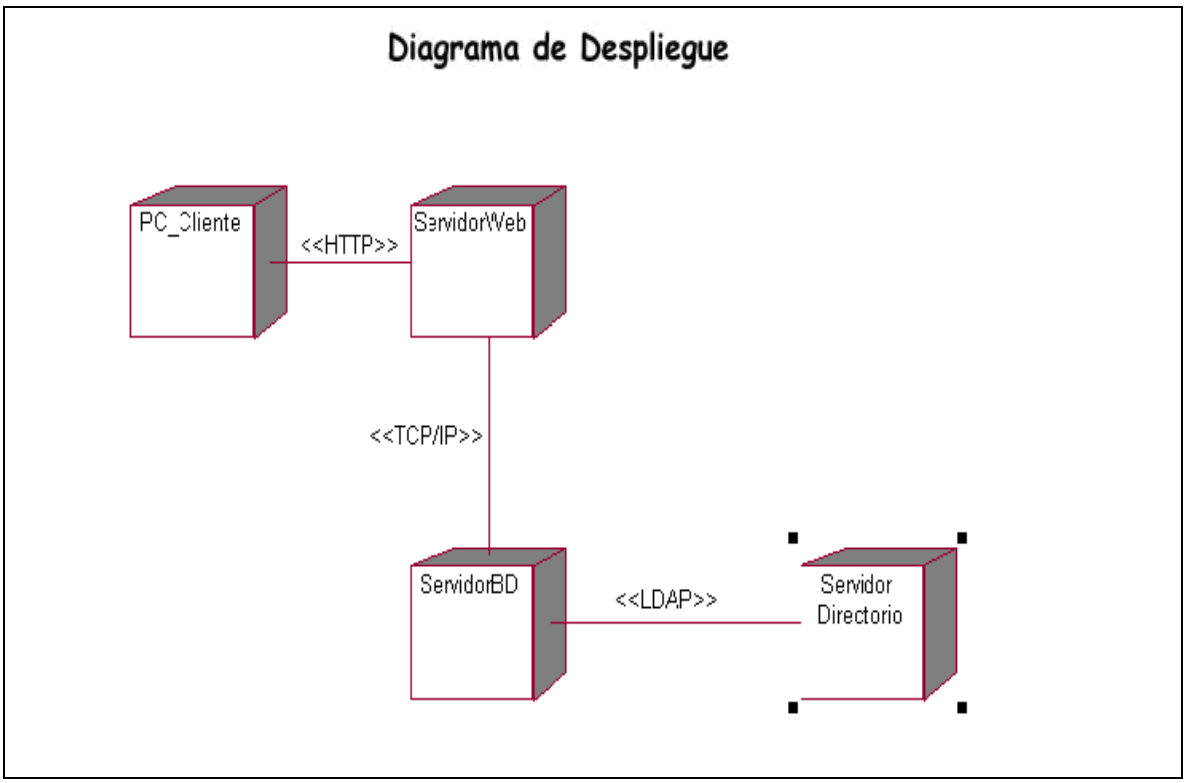


Figura 4. 1 Diagrama de Despliegue

4.2.2 Diagramas de Componentes.

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

4.2.2.1 Diagrama de componentes: Base de Datos

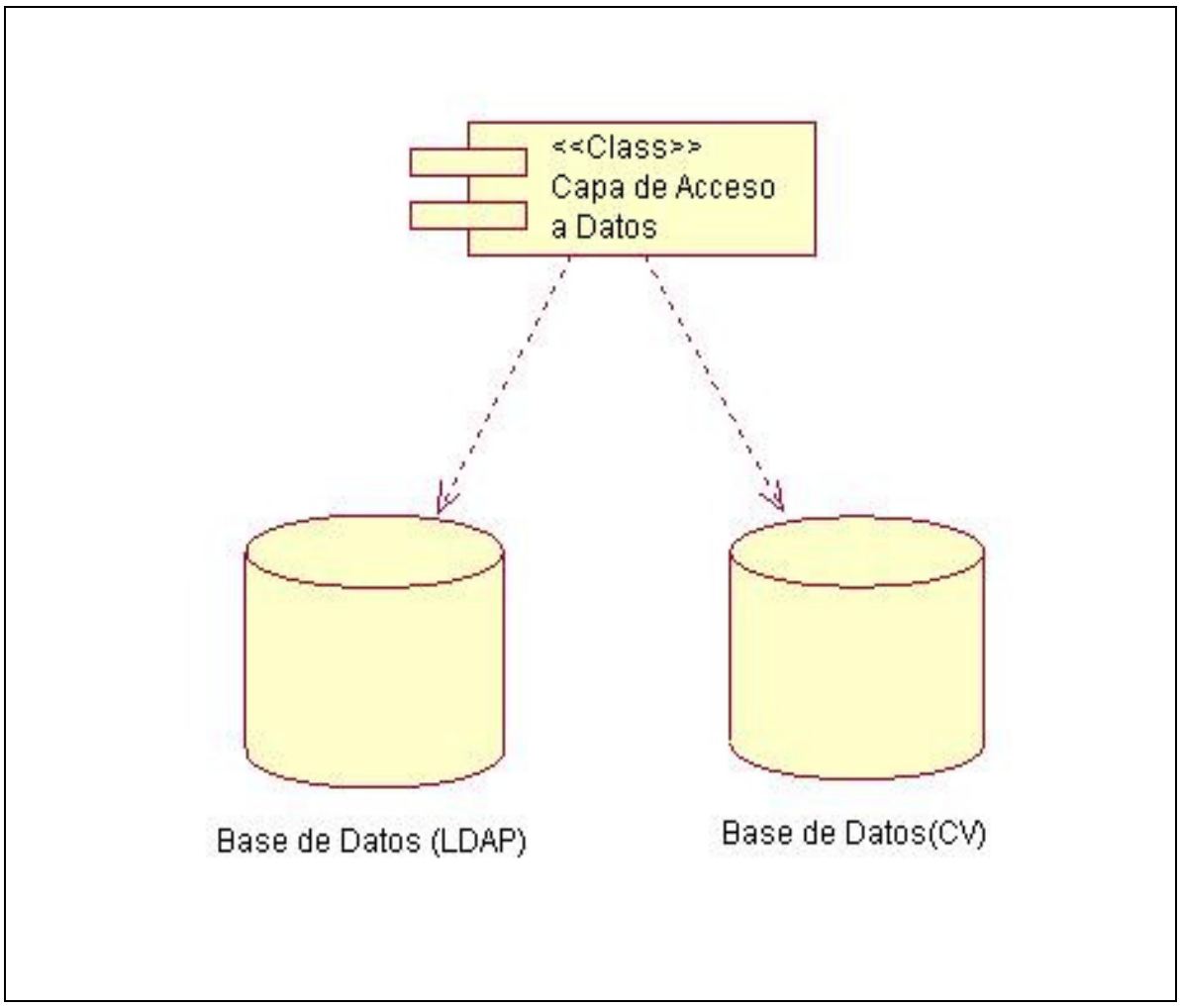


Figura 4. 2 Diagrama de Componentes: Base de Datos

4.2.2.2 Diagrama de Componentes: código fuente

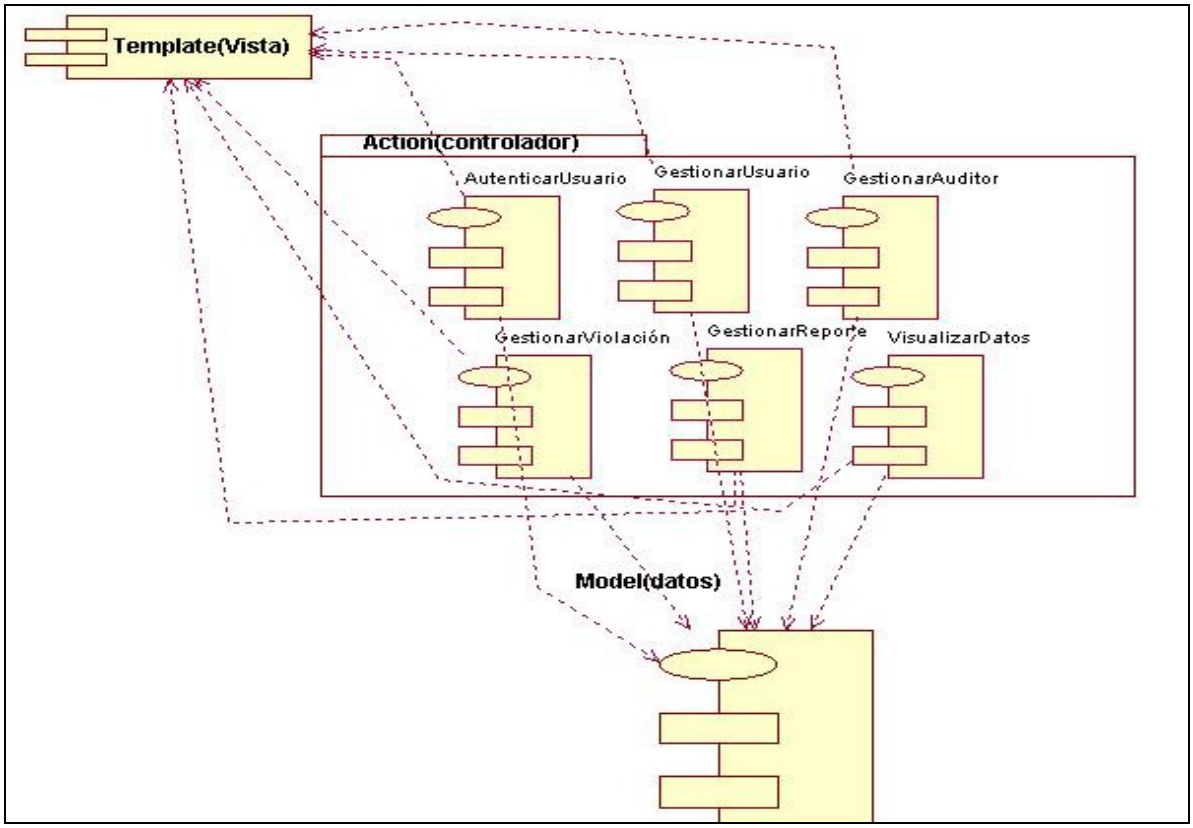


Figura 4.3 Diagrama de Componentes: Código Fuente

4.3 Conclusiones

Tras el flujo de implementación y prueba el sistema quedó desarrollado. En los diagramas generados pudo mostrarse la relación entre lo principales componentes del sistema y distribución del mismo. Las pruebas realizadas arrojaron algunos errores que fueron corregidos durante la implementación, estos permitieron aumentar la calidad final del sistema. El sistema final cultivó las ventajas de Symfony y quedó listo para entrar en funcionamiento.

CONCLUSIONES GENERALES

Después de haber realizado un profundo estudio del proceso a automatizar y haber pasado por las distintas etapas que lleva la construcción de un software; las etapas de Negocio, Requerimientos, Análisis y Diseño e Implementación, se obtuvieron todos los artefactos pertenecientes a las mismas que permitieron dar solución al problema propuesto.

Durante la etapa de negocio, se realizaron entrevistas para conocer la necesidad de la Dirección Energética y además conocer profundamente el proceso de control de violaciones. Se definieron cada uno de los actores y trabajadores del negocio, así como construir el diagrama de actividad, modelo de objetos y diagrama de casos de uso del negocio. Además se describieron cada uno de los requerimientos funcionales y no funcionales.

En la etapa de modelado del sistema, se refinaron y describieron los casos de uso, obteniendo el diagrama de casos de uso del sistema.

En la etapa de análisis, se obtuvieron los artefactos de modelo de análisis, diagrama de clases de análisis donde se definieron cada una de las clases (interfaz, control y entidad), y se construyeron los diagramas por cada caso de uso. En la etapa del diseño, se realizaron los diagramas de clases del diseño y los diagramas de secuencia. Además se obtuvo el diagrama de clases persistentes y el modelo de datos para la base de datos que tendrá la aplicación donde se guardarán los datos necesarios.

Se diseñó e implementó una aplicación para la representación del proceso de control de violaciones de la electricidad.

El sistema cuenta con una interfaz amigable y fácil de usar para el usuario, el cual cumple con los estándares de diseño y los requerimientos de software trazados.

RECOMENDACIONES

Aunque al objetivo de este trabajo se le da cumplimiento con las funcionalidades del sistema, han surgido nuevas ideas durante el desarrollo del mismo, por lo que se recomienda:

1. Agregar un módulo para la gestión del corte eléctrico.
2. Aplicar el sistema si es posible en otras instituciones del país con el fin de contribuir con la revolución energética.

BIBLIOGRAFÍA

1. **Sánchez, Maria A. Mendoza.** [Online] junio 7, 2007. [Cited: febrero 22, 2008.] http://www.informatizate.net/articulos/metodologías_de_desarrollo_de_software_07062004.html. MCP.

2. [Online] [Cited: febrero 25, 2008.] http://gidis.ing.unlpam.edu.ar/personas/glafuente/uml/uml.html..

3. **Rational Software Corporation.** El Lenguaje Unificado de Modelado. [book auth.] WESLEY ED ADDISON and JAMES RUMBAUGH. 2000.

4. [Online] [Cited: febrero 27, 2008.] http://ateam.lsi.upc.es/~es-e/web/documents/lab/0304Q2/lessons/lese-2/LESE-2%20-%20Introduccion%20a%20Rational%20Rose.ppt?PHPSESSID=0ade5aba4d6b566cb794d3b7def7cd08. .

5. **Visual Paradigm**. [Online] [Cited: marzo 1, 2008.] Disponible en: http://alarcos.inf-cr.uclm.es/per/fgarcia/isoftware/doc/LabTr1.pdf.

6. **Sistemas Gestores de Base de Datos**. [Online] [Cited: marzo 2, 20008.] http://es.wikipedia.org/wiki/Sistemas_Gestores_de_Bases_de_Datos.

7. **Casares, Claudio.** *Maestros del Web.* [Online] [Cited: marzo 2, 2008.] http://www.maestrosdelweb.com/editorial/tutsql1/.

8. **Microsoft**. [Online] [Cited: marzo 3, 2008.] http://www.microsoft.com/spain/sql/default.mspx.

9. En Introducción a la Tecnología .NET. 2004-2005 . *Elementos básicos del lenguaje C#.* 2004-2005.

10. **PHP**. [Online] [Cited: marzo 3, 2008.] http://www.php.net/.

11. **Rodríguez, Lazaro Orlando Aneiro.** *ELEMENTO DE ARQUITECTURA Y SEGURIDAD INFORMATICA.* La Habana : Pueblo y Educación, 2001.

12. **PRESSMAN, Roger.** *"Ingeniería del Software. Un enfoque práctico".* 2002. pp. 184-186

13. **CRESPO.** "Anotaciones del Proceso Unificado para Desarrollo de Software (RUP)". 2001.

14. **IVAN JACOBSON and JAMES RUMBAUGH.** *El proceso unificado de desarrollo de software.* 2004. p. 434.

15. *Flujo de Análisis y Diseño.* Curso 2007-2008.

16. **Ivar Jacobson, G. B, James Rumbaugh.** *El proceso unificado de desarrollo del software.* 1999. p. 168.

17. **Peña, Javier Sevilla y García, Abduly Díaz.** Sistema de Gestión de Información de la Facultad 8. Ciudad de la Habana : s.n., 2007.

18. Generator FD. [Online] [Cited: marzo 18, 2008.] http://www.generatorfd.com..

19. **González, Carlos Sánchez.** [Online] [Cited: marzo 22, 2008.] http://oness.sourceforge.net/proyecto/html/ch03s02.html..

20. **Lago, Ramiro.** [Online] abril 2007. [Cited: marzo 25, 2008.] http://www.proactiva-calidad.com/java/patrones/DAO.html.

21. **Vilas, Ana Fernandez.** [Online] marzo 20, 2001. [Cited: marzo 27, 2008.] http://www-gris.det.uvigo.es/~avilas/UML/node49.html.

22. Carl Vondrick. **[Online]** Octubre 8, 2007. [Cited: junio 6, 2009.] http://necudeco.com/index.php/2008/06/24/phporm/

23. **Hermosilla, José Ramón Moreno y Sánchez, Luis Enrique Arce**. Interfaz de Administración Web para el Sistema de Filtrado Filpacon de la Facultad 10. Ciudad de la Habana : s.n., 2008.

24. Gerti Kappel et al. Web Engineering: The Discipline of Systematic Development of Web Applications,
Chapter an Introduction to Web Engineering, pages 1–22. John Wiley & Sons Ltd., 2006.

25. http://wobzip.filetap.com/

26. http://www.rememberthemilk.com/

27. http://www.nozbe.com/

28 http://labelia.net/

29 http://www.myquire.com/

30 http://www.project2manage.com/

Anexos A.

Diagramas de Clase de Análisis

Diagrama de Clases de Análisis: Cu _ Gestionar Auditor

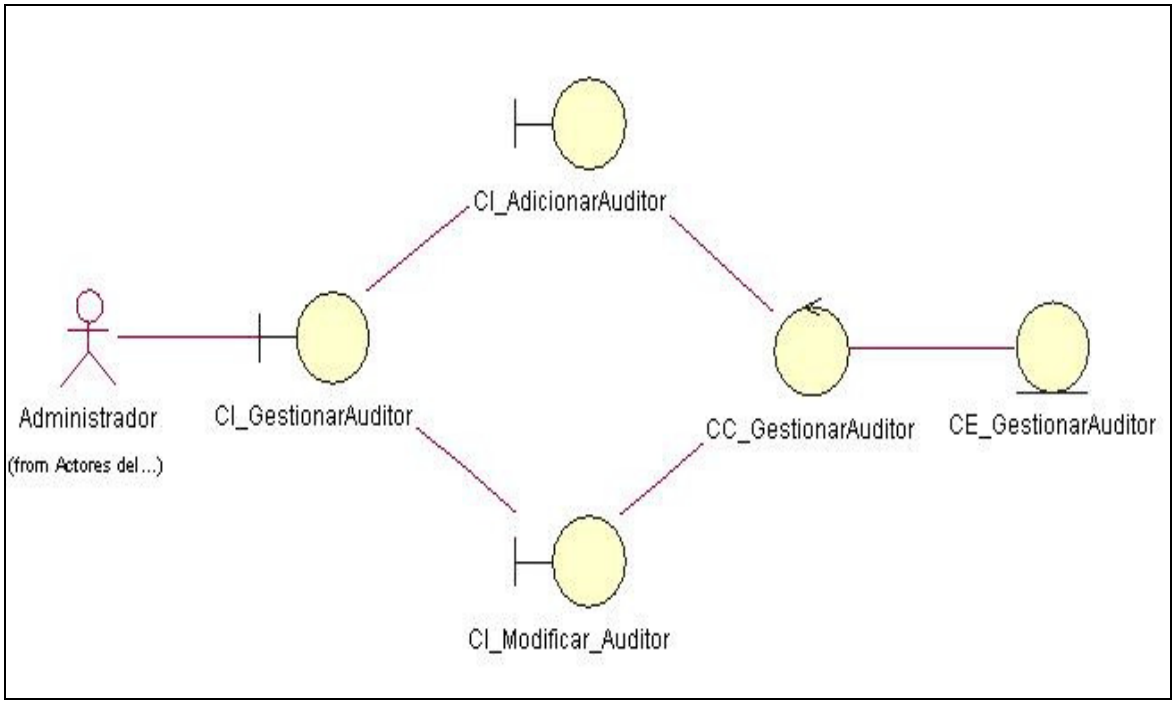


Diagrama de Clases de Análisis: Cu _ Gestionar Violación

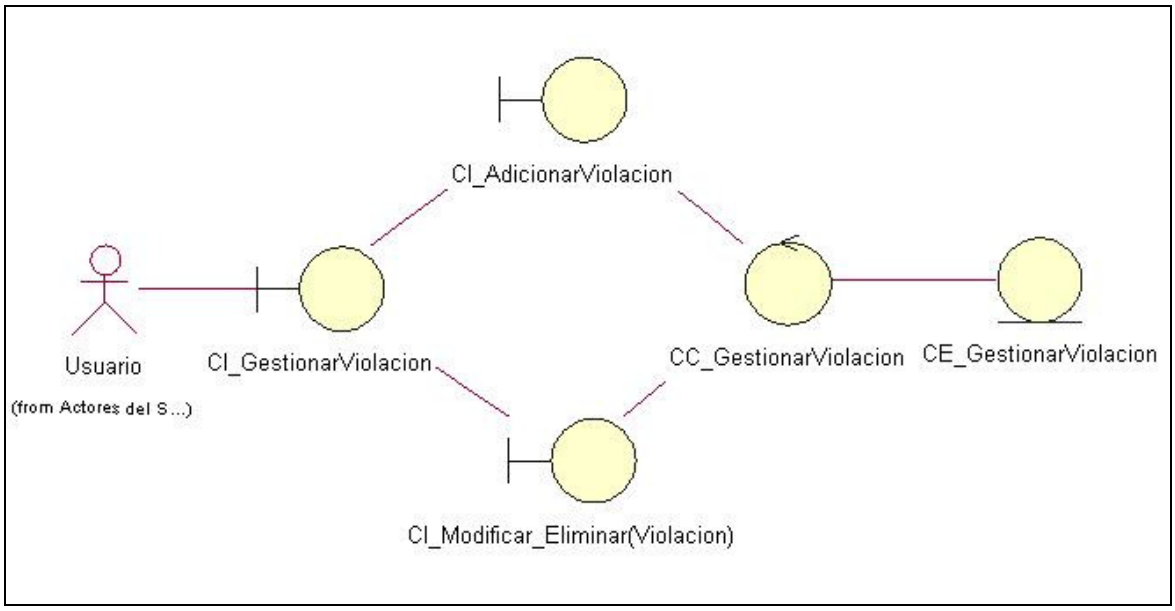
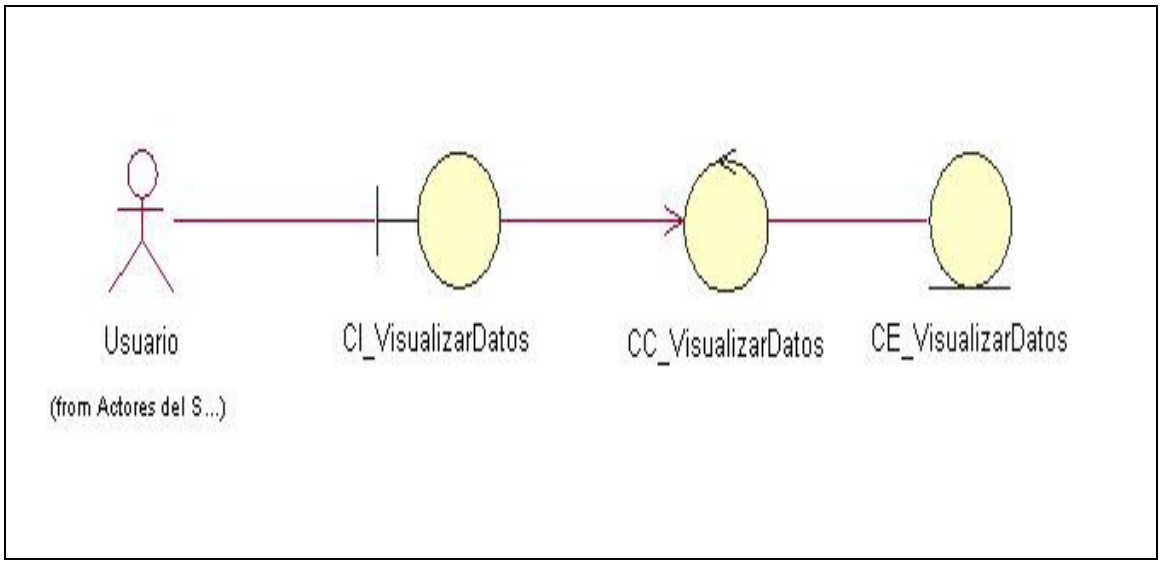


Diagrama de Clases de Análisis: Cu _ Visualizar Dato



Anexos B.

Diagramas de Clase de Diseño

Diagrama de clase de diseño Gestionar Auditor

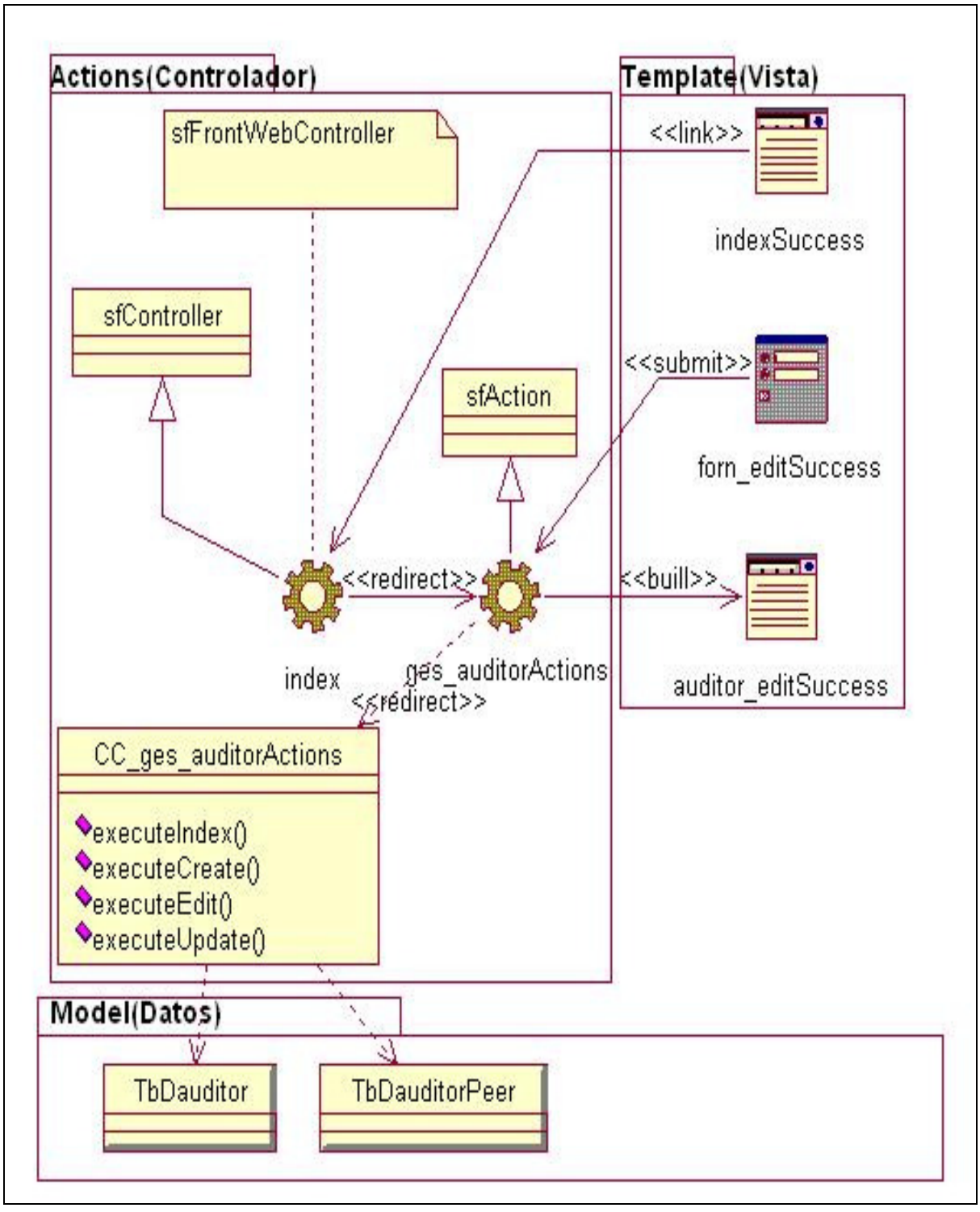


Diagrama de clase de diseño Gestionar Violación

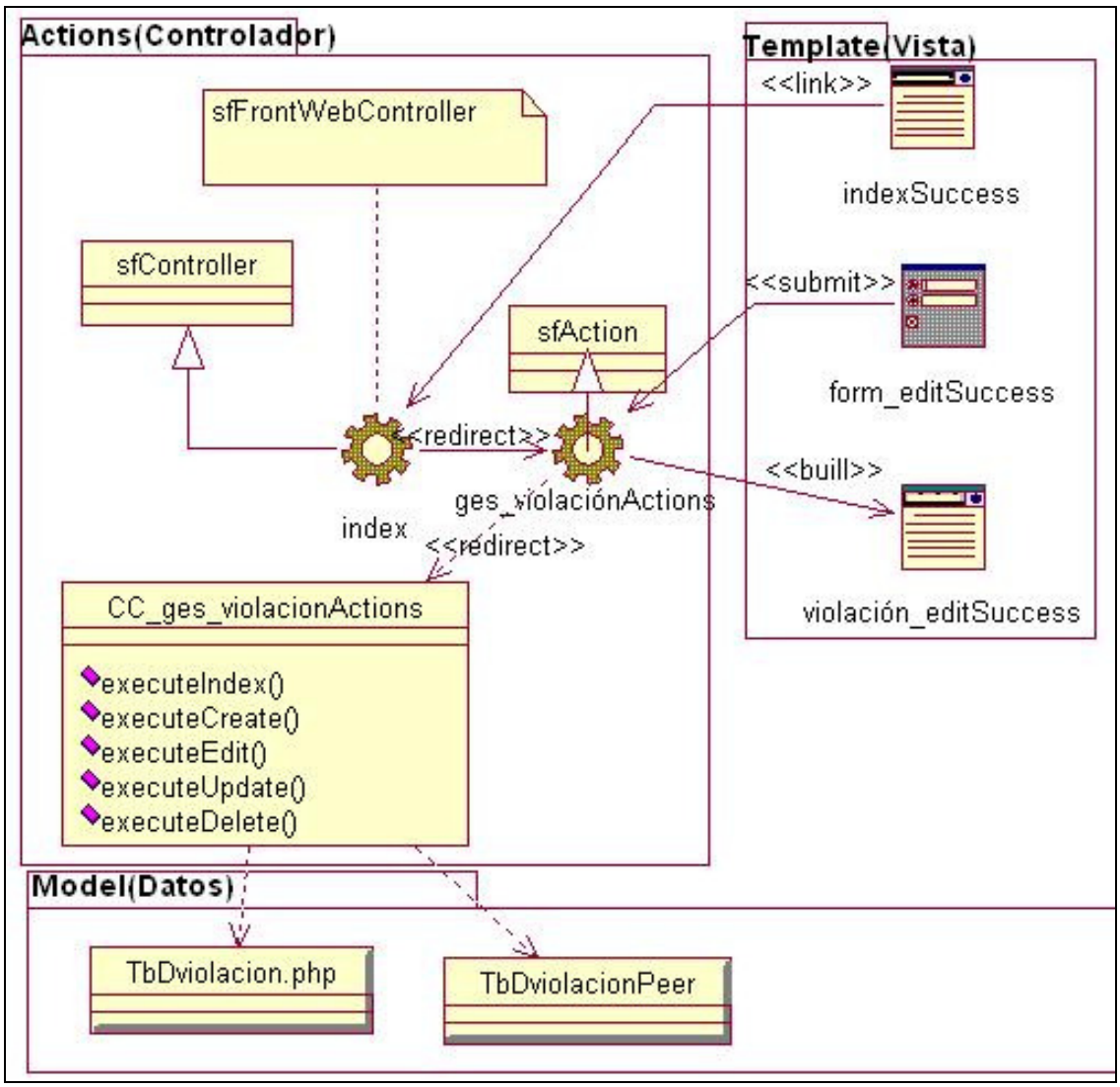
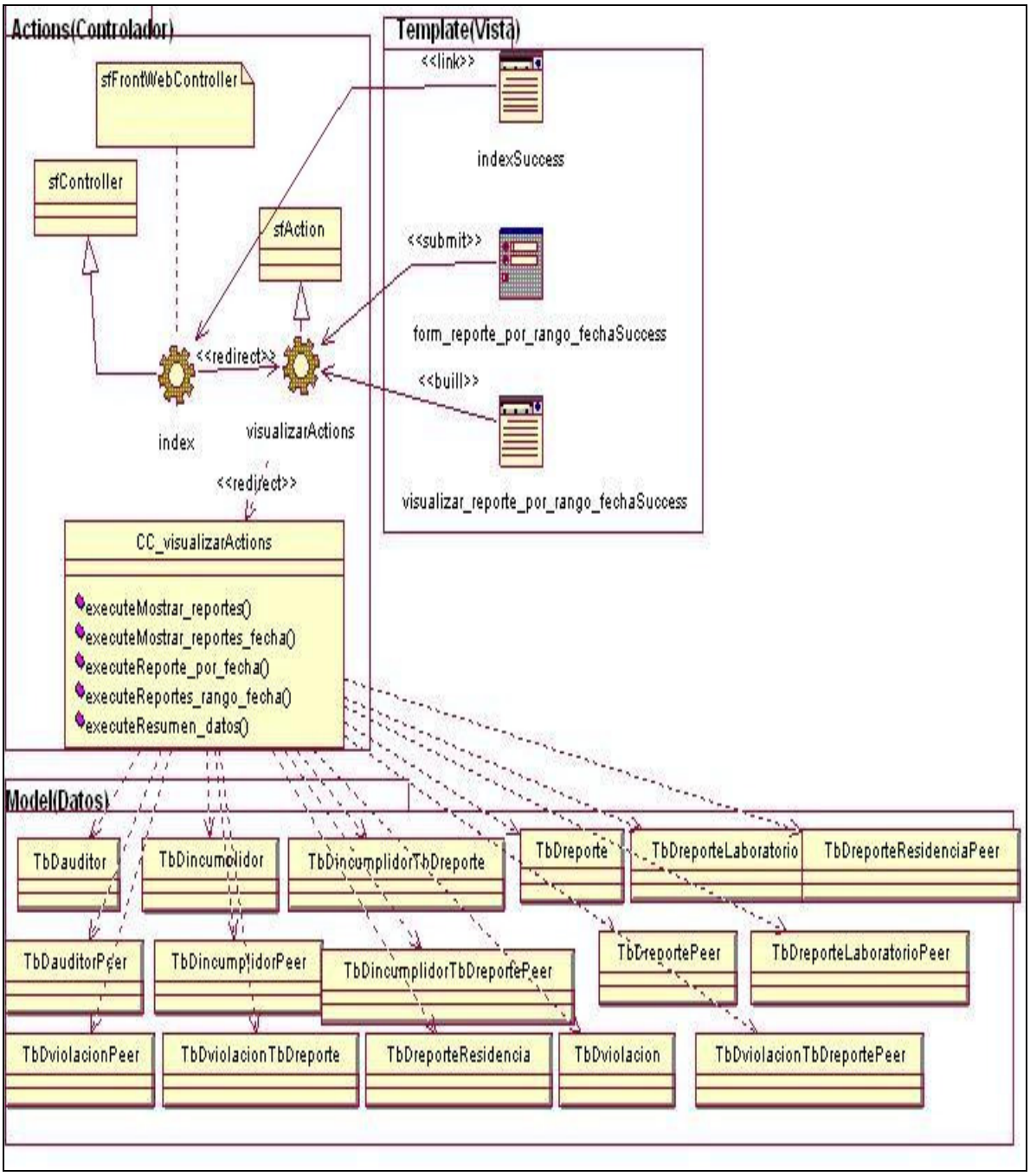


Diagrama de clase de diseño Visualizar Datos



Anexos C.

Diagramas de Secuencia

Diagrama de Secuencia Gestionar Auditor (Escenario Adicionar Auditor).

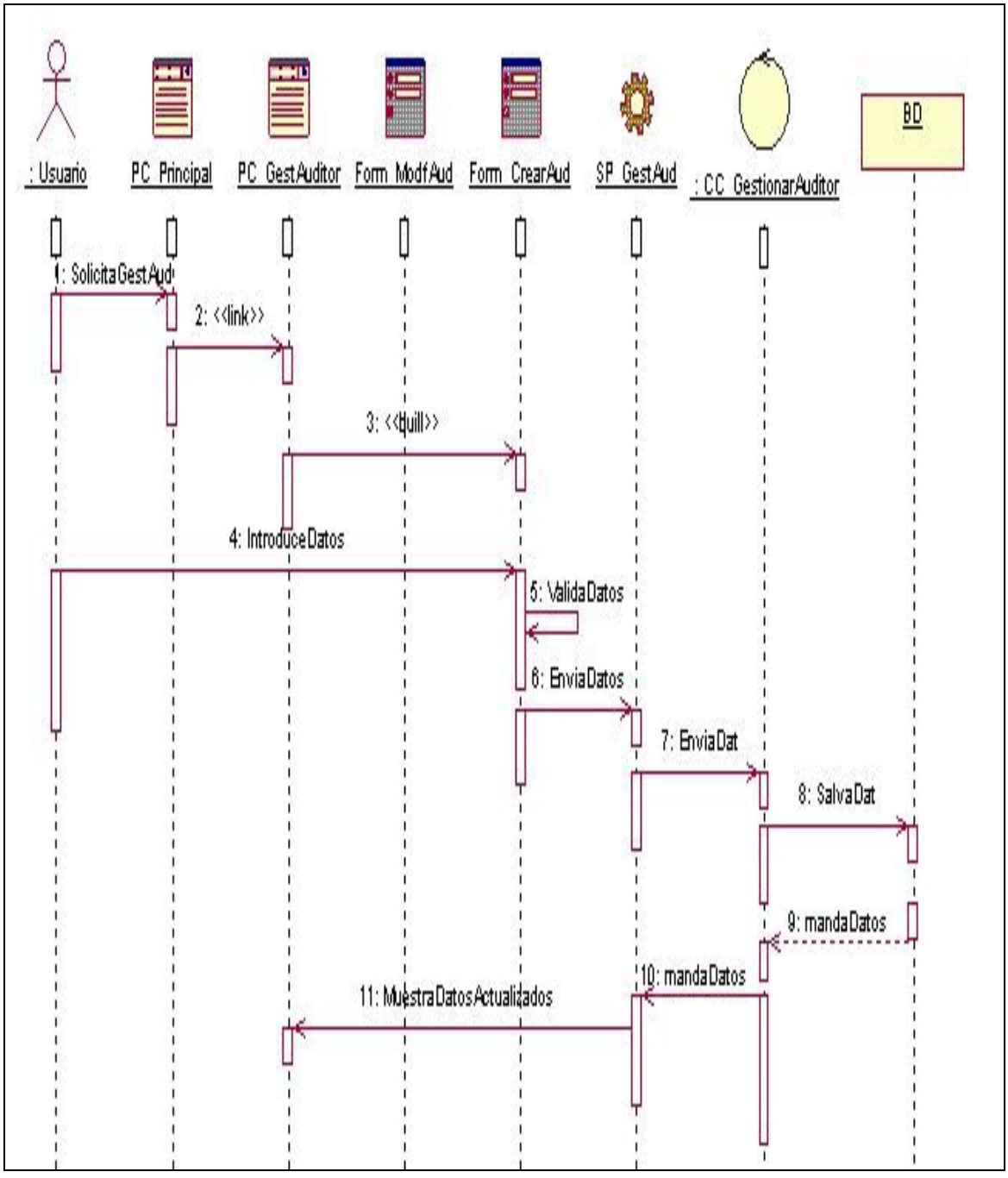


Diagrama de Secuencia Gestionar Auditor (Escenario Modificar Auditor).

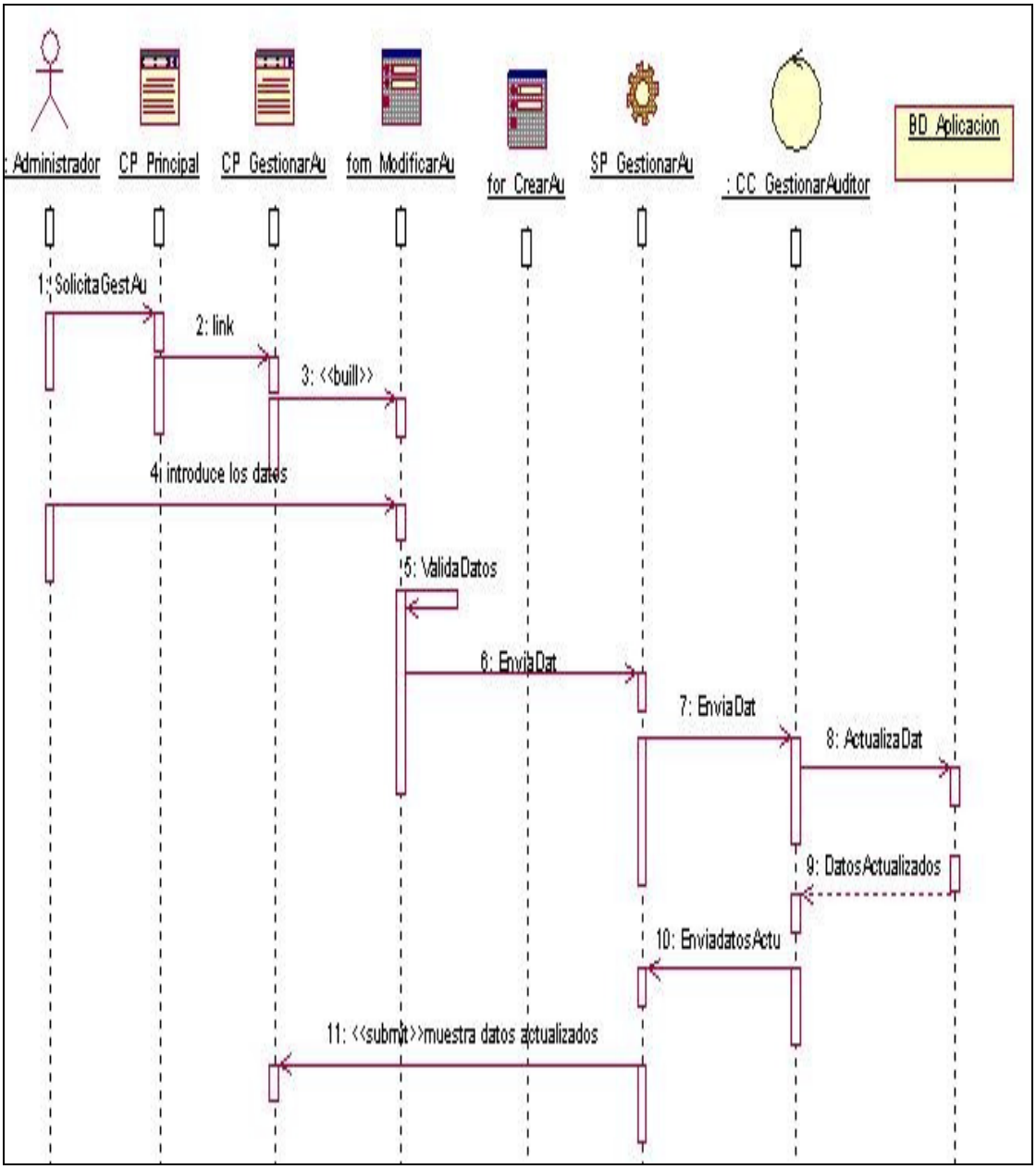


Diagrama de Secuencia Gestionar Violación (Escenario Adicionar Violación).

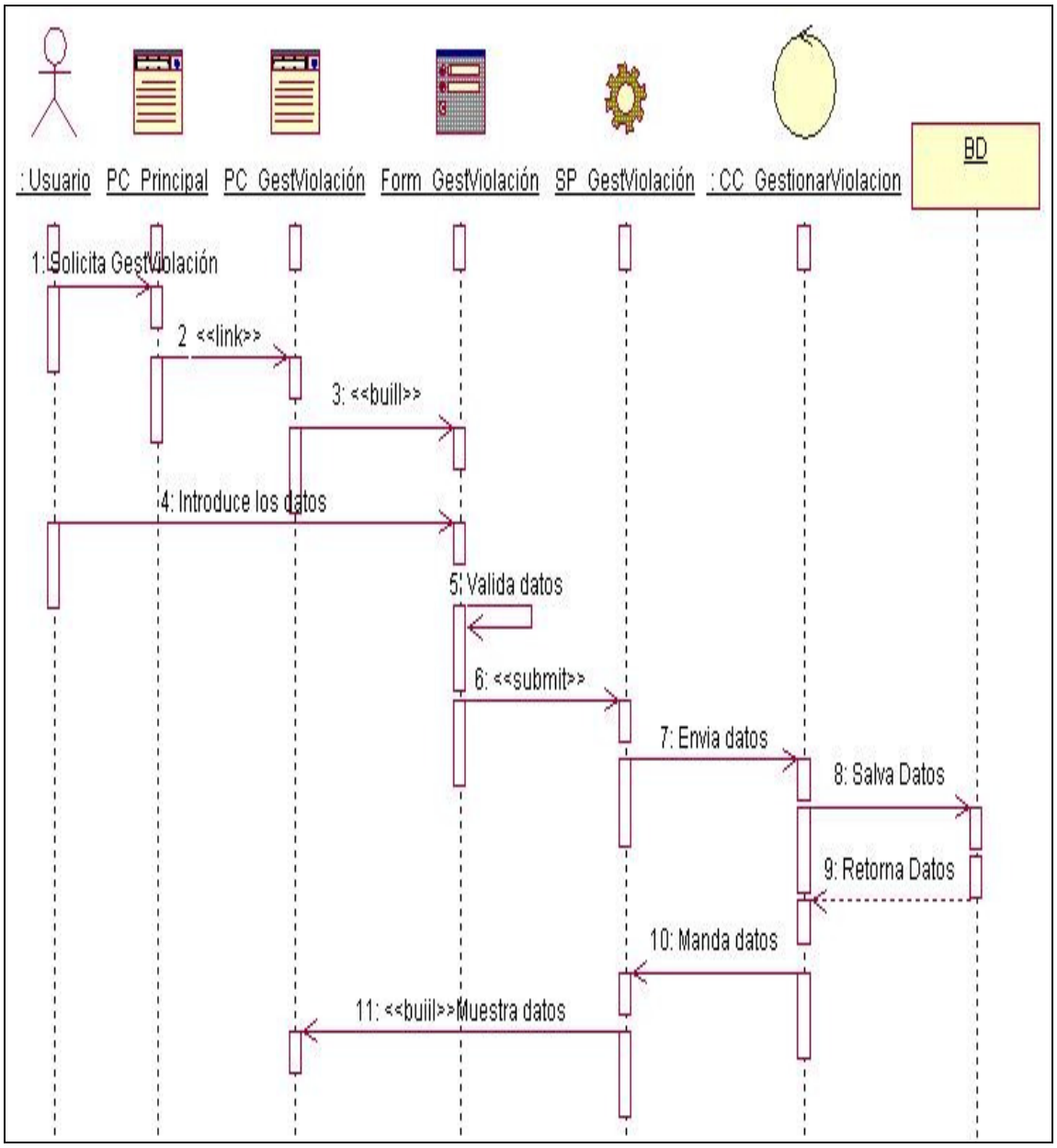


Diagrama de Secuencia Visualizar Datos.