

UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 3



TÍTULO: Diseño de Subsistema de Control de Auditoría para Aplicaciones de Software Empresariales.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

AUTORA: Yunielsy Castillo González

TUTOR: Ing. Informático Yuniel Eliades Proenza Arias

CO-TUTOR: Ing. Informático Oiner Gómez Baryolo

Ciudad de la Habana, julio, 2008

Año 50 de la Revolución

“Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte. ”

Ernesto “Che” Guevara De La Serna.



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 ____ días del mes de _____ del año _____.

Yunielsy Castillo González

Firma del Autor

Yuniel E. Proenza Arias

Firma del Tutor.

DATOS DE CONTACTO

Datos del Tutor:

Ing. Informático Yuniel Eliades Proenza Arias.

Graduado en el 2006 de la carrera de de Ingeniería Informática en la Universidad de Holguín. Profesor de la Universidad de las Ciencias Informáticas. Opta por la Categoría Docente de Instructor. Se ha desempeñado como profesor en las disciplinas de Ingeniería y Gestión de Software y Técnicas de Programación. Actualmente cursa la maestría de Gestión de Proyectos Informáticos.

Datos de Co-Tutor.

Ing. en Ciencias Informáticas Oiner Gómez Baryolo.

Graduado en el 2007 de la carrera Ingeniería en Ciencias Informática en la Universidad de las Ciencias Informáticas (UCI). Actualmente labora en el Centro UCID (integración de las FAR y la UCI) como Especialista General.

A mis padres por depositar en mí plena confianza y apoyarme en estos años de grandes esfuerzos. Sepan que formaron parte de la inspiración y el deseo de hacer realidad este trabajo y obtener un título.

A Mirian Martínez y Carlo Edghill quienes considero mis segundos padres, sin la gran solidaridad que hay en sus corazones y todo el apoyo que me han ofrecido este trabajo no hubiese tenido un principio y menos este fin.

A mi hermana, abuelos, tíos (as) y primos que siempre estuvieron presentes.

A mi tío José Angel por ser padre rector en mi familia, por su fuerza y tesón para que tengamos una vida sin necesidades.

A mi hermanita de universidad Yalena, gracias amiga por estar siempre presente, por apoyarme en todas mis decisiones, por darme sabios consejos, por cuidarme cuando más lo necesité.

A mis amistades que ocupan un lugar en mi corazoncito: Yalena, Yinett, Yanet, Yisel de la Peña, Yisel Sariof, Yisel Guerra, Nemury, Dina, Misleydis, Aray, Omar; me satisface mucho haberlos conocido y haber compartido con ustedes estos años de universidad.

A mi mol, la persona más especial que he conocido. Gracias a ti mi mol por ser el haz de luz, el hilo conductor en mi camino. Gracias por enseñarme que la fortuna de la vida está más allá de lo que se logra porque siempre se puede lograr más. Sin tu apoyo, ayuda, comprensión y amor no hubiese podido lograr con éxito este trabajo.

A mis compañeros de proyecto: Borbón, Omar, Luis Alberto.

A Rolando por su gran ayuda en estos 7 meses.

A mi Co-Tutor por ser 50 % comprensible con mis ideas y por ayudarme a mejorarlas.

A mi tutor por toda la ayuda que me prestó durante la elaboración de este trabajo, por todo el esfuerzo que aportó cuando lo necesité.

A mis vecinos por mostrar interés y preocupación mientras estaba en la universidad, así como acompañar a mi familia en los momentos difíciles.

A los que en algún momento preguntaron ¿Y la tesis que...?

A mis compañeras de apto 110103, por acogerme sin ni siquiera pertenecer a su facultad, gracias por los buenos momentos.

A la Revolución Cubana y en especial al Comandante Fidel Castro, gran inspirador de los estudiantes de la Universidad de las Ciencias Informáticas.

Muchas gracias!!!

Dedico este trabajo con todo el amor del mundo:

A mis padres que tanto han soñado con verlo hecho realidad.

A mis abuelos por darme su cariño, ayuda y aportar en mi educación.

A mi mol. Tú también eres parte de él y siempre estarás en mi corazón.

A mi Virgen de la Caridad del Cobre por siempre protegerme e iluminar el camino hacia el triunfo final.

A mí por el gran esfuerzo que he realizado para llegar hasta el fin...

Con el actual avance de las nuevas tecnologías de la informática y el surgimiento de una nueva era de automatización, cambia por completo el enfoque de las empresas, donde el flujo de información cada vez mayor, rompe con los esquemas de procesamiento y almacenamiento actuales y aumenta en gran medida los requerimientos de clientes cada vez más insatisfechos.

A la vez que se necesita tener mayor alcance para soportar flujos gigantescos de datos en un área creciente las empresas se ven en la obligación de expandir su dominio y surgen empresas distribuidas en extensos territorios y el flujo de información crece más.

El aumento en los volúmenes de información, sus características e importancia para clientes y empresa en general, crea la necesidad de utilizar herramientas que soporten el procesamiento y almacenamiento de manera segura, así como posibilidades de proteger su integridad, prevenir y enfrentar posibles ataques.

En el centro donde se desarrolla el presente trabajo se cuenta con un sistema de seguridad para las aplicaciones existentes; pero no se tiene en cuenta el control de las acciones que realizan los usuarios sobre las aplicaciones, denominadas eventos. El proceso de almacenar información de tales acciones y mostrar reportes se puede denominar control de auditoría.

La pregunta es la siguiente: ¿Cómo lograr mantener la integridad de la información y enfrentar posibles ataques a pesar de implantar mecanismos de seguridad? Con necesidad de dar respuesta a esta pregunta se conforma este trabajo.

Palabras claves: Información, Acción, Evento, Aplicación, Reporte, Auditoría.

INTRODUCCIÓN	1
CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA	4
1.1 INTRODUCCIÓN	4
1.2 OBJETO DE ESTUDIO.....	4
1.2.1 <i>Objetivo estratégico de la organización</i>	4
1.2.2 <i>Flujo actual de los procesos</i>	4
1.3 PROCESOS OBJETOS DE AUTOMATIZACIÓN	5
1.4 SISTEMAS AUTOMATIZADOS EXISTENTES VINCULADOS AL CAMPO DE ACCIÓN	5
1.5 LA AUDITORÍA EN APLICACIONES DE SOFTWARE.....	5
1.6 SISTEMAS UTILIZADOS EN LA AUDITORÍA DE APLICACIONES DE SOFTWARE.....	6
1.7 CONCLUSIONES	9
CAPÍTULO 2 TENDENCIAS Y TECNOLOGÍAS ACTUALES.....	10
2.1 INTRODUCCIÓN	10
2.2 APLICACIONES WEB.....	10
2.2.1 <i>Servicios Web</i>	11
2.3 LENGUAJES DE PROGRAMACIÓN WEB.....	14
2.4 ARQUITECTURA DE SOFTWARE.....	15
2.4.1 <i>Arquitectura cliente/servidor</i>	16
2.4.2 <i>Estilos Arquitectónicos</i>	17
2.5 GESTORES DE BASES DE DATOS.....	21
2.6 METODOLOGÍAS DE DESARROLLO DE SOFTWARE.....	22
2.6.1 <i>RUP</i>	23
2.6.2 <i>XP</i>	24
2.6.3 <i>Metodología utilizada</i>	24
2.7 LENGUAJE DE MODELADO	25
2.8 HERRAMIENTA CASE VISUAL PARADIGM PARA UML.....	25
2.9 CONCLUSIONES	26
CAPÍTULO 3 PRESENTACIÓN DE LA SOLUCIÓN PROPUESTA.....	27
3.1 INTRODUCCIÓN	27
3.2 DEFINICIÓN DE LAS ENTIDADES Y LOS CONCEPTOS PRINCIPALES	28
3.3 REPRESENTACIÓN DEL MODELO DEL DOMINIO.....	29
3.4 DESCRIPCIÓN DE LAS RELACIONES ENTRE LAS ENTIDADES DEL DOMINIO	29

3.5 REQUERIMIENTOS.....	30
3.5.1 Tipos de Requerimientos.....	31
3.6 REQUERIMIENTOS FUNCIONALES	32
3.7 REQUISITOS NO FUNCIONALES	33
3.8 ACTORES DEL SISTEMA A AUTOMATIZAR	36
3.9 DIAGRAMA DE PAQUETES.....	36
3.10 DIAGRAMA DE CASOS DE USO DEL SISTEMA A AUTOMATIZAR.....	37
3.11 DESCRIPCIÓN DE LOS CASOS DE USO	38
3.12 DESCRIPCIÓN DETALLADA DE LOS CASOS DE USOS	41
3.13 CONCLUSIONES	58
CAPÍTULO 4 DISEÑO DE LA SOLUCIÓN PROPUESTA	59
4.1 INTRODUCCIÓN	59
4.2 MODELO DEL ANÁLISIS.....	59
4.2.1 Diagramas de Colaboración de clases del análisis.....	60
4.3 MODELO DE DISEÑO.....	65
4.3.1 Arquitectura del Sistema.....	65
4.3.2 Presentación general	65
4.3.3 Mecanismos de Diseño	66
4.3.4 Diagrama Genérico.....	70
4.4 DIAGRAMAS DE CLASES DEL DISEÑO	74
4.4.1 Diagramas de Secuencias clases del diseño.....	78
4.5 DISEÑO DE LA BASE DE DATOS	79
4.5.1 Modelo lógico de datos.....	79
4.5.2 Modelo físico de datos.....	79
4.6 DIAGRAMA DE DESPLIEGUE.....	80
4.7 MODELO DE IMPLEMENTACIÓN	81
4.7.1 Composición de los paquetes	81
4.7.2 Diagrama Genérico para los casos de uso.....	82
4.7 CONCLUSIONES	84
CONCLUSIONES	85
RECOMENDACIONES.....	86
GLOSARIO DE TÉRMINOS	87

REFERENCIAS BIBLIOGRÁFICAS	90
BIBLIOGRAFÍA.....	92
ANEXO 1 DIAGRAMAS DE CLASES DEL ANÁLISIS.....	93
ANEXO 2 DIAGRAMAS DE SECUENCIA DE LAS CLASES DEL DISEÑO	96

TABLA 2.1 EJEMPLOS DE FRAMEWORK QUE UTILIZAN MVC	18
TABLA 3.1 CONCEPTOS DE AUDITORÍA	28
TABLA 3.2 ACTORES DE SISTEMA	36
TABLA 3.3 BREVE DESCRIPCIÓN DEL CU GESTIONAR EVENTOS A AUDITAR.....	38
TABLA 3.4 BREVE DESCRIPCIÓN DEL CU EXPORTAR DATOS.....	38
TABLA 3.5 BREVE DESCRIPCIÓN DEL CU GENERAR REPORTE	39
TABLA 3.6 BREVE DESCRIPCIÓN DEL CU REGISTRAR EVENTO	39
TABLA 3.7 BREVE DESCRIPCIÓN DEL CU GESTIONAR SERVICIO DE REPORTE.....	39
TABLA 3.8 BREVE DESCRIPCIÓN DEL CU VALIDAR USUARIO.....	40
TABLA 3.9 DESCRIPCIÓN DETALLADA DEL CU GESTIONAR EVENTOS A AUDITAR.....	48
TABLA 3.10 DESCRIPCIÓN DETALLADA DEL CU EXPORTAR DATOS.....	51
TABLA 3.11 DESCRIPCIÓN DETALLADA DEL CU GENERAR REPORTE	55
TABLA 3.12 DESCRIPCIÓN DETALLADA DEL CU REGISTRAR EVENTO	55
TABLA 3.13 DESCRIPCIÓN DETALLADA DEL CU GESTIONAR SERVICIO DE REPORTE.....	56
TABLA 3.14 DESCRIPCIÓN DETALLADA DEL CU VALIDAR USUARIO.....	57
TABLA 4.1 DESCRIPCIÓN DE LAS CLASES UTILIZADAS EN LA VISTA.	67
TABLA 4.2 DESCRIPCIÓN DE LAS CLASES UTILIZADAS EN EL CONTROLADOR.....	68
TABLA 4.3 DESCRIPCIÓN DE LAS CLASES UTILIZADAS EN EL MODELO.	69

FIGURA. 1.1 CONCEPTOS PRESENTES EN LOS LOGS DE AUDITORÍA	6
FIGURA 2.1 FUNCIONAMIENTO MVC.....	19
FIGURA 3.1 MODELO DE DOMINO	29
FIGURA 3.2 DIAGRAMA DE PAQUETES	36
FIGURA 3.3 DIAGRAMA DE CASOS DE USO DEL SISTEMA: PAQUETE <i>ADMINISTRACIÓN DEL SISTEMAS</i>	37
FIGURA 3.4. DIAGRAMA DE CASOS DE USO DEL SISTEMA: PAQUETE <i>SERVICIOS</i>	37
FIGURA 4.1 DIAGRAMA DE COLABORACIÓN DEL CASO DE USO: GESTIONAR EVENTOS A AUDITAR – ESCENARIO REGISTRAR EVENTOS A AUDITAR.....	60
FIGURA 4.2 DIAGRAMA DE COLABORACIÓN DEL CASO DE USO: GESTIONAR EVENTOS A AUDITAR – ESCENARIO MODIFICAR EVENTOS A AUDITAR.....	60
FIGURA 4.3 DIAGRAMA DE COLABORACIÓN DEL CASO DE USO: GESTIONAR EVENTOS A AUDITAR – ESCENARIO ELIMINAR EVENTOS A AUDITAR.....	61
FIGURA 4.4 DIAGRAMA DE COLABORACIÓN DEL CASO DE USO: EXPORTAR DATOS.....	61
FIGURA 4.5 DIAGRAMA DE COLABORACIÓN DEL CASO DE USO: GESTIONAR SERVICIO DE REPORTE.....	62
FIGURA 4.6 DIAGRAMA DE COLABORACIÓN DEL CASO DE USO: REGISTRAR EVENTOS.	62
FIGURA 4.7 DIAGRAMA DE COLABORACIÓN DEL CASO DE USO: VALIDAR USUARIO.....	62
FIGURA 4.8 DIAGRAMA DE COLABORACIÓN DEL CASO DE USO: GENERAR REPORTE – ESCENARIO REPORTE POR SISTEMA.	63
FIGURA 4.9 DIAGRAMA DE COLABORACIÓN DEL CASO DE USO: GENERAR REPORTE – ESCENARIO REPORTE POR USUARIO.	64
FIGURA 4.10 DIAGRAMA DE COLABORACIÓN DEL CASO DE USO: GENERAR REPORTE – ESCENARIO REPORTE POR SISTEMA -USUARIO.....	64
FIGURA 4.11 DIAGRAMA DE COLABORACIÓN DEL CASO DE USO: GENERAR REPORTE – ESCENARIO REPORTE POR SISTEMA - FUNCIONALIDADES - USUARIO.....	65
FIGURA 4.12 VISTA DEL ERP Y ALGUNOS DE LOS MÓDULOS QUE LO COMPONENTE	66
FIGURA 4.13 DIAGRAMA DE CLASES GENÉRICO PARA EXTJS (MECANISMO DE DISEÑO)	67
FIGURA 4.14 DIAGRAMA DE CLASES GENÉRICO PARA ZEND FRAMEWORK (CONTROLADORES).....	68

FIGURA 4.15 DIAGRAMA DE CLASES GENÉRICO PARA DOCTRINE PHP.....	69
FIGURA 4.16 VISTA DE GESTIÓN DE MODELO GENÉRICO PARA MVC	70
FIGURA 4.17 DCD PARA CASO DE USO GENÉRICO (MECANISMO DE DISEÑO)	71
FIGURA 4.18 PROPUESTA FINAL DE DCD PARA CASO DE USO GENÉRICO	73
FIGURA 4.19 DIAGRAMA DE CLASES DEL DISEÑO CASO DE USO GESTIONAR EVENTOS A AUDITAR.....	74
FIGURA 4.20 DIAGRAMA DE CLASES DEL DISEÑO CASO DE USO GENERAR REPORTE	75
FIGURA 4.21 DIAGRAMA DE CLASES DEL DISEÑO CASO DE USO EXPORTAR DATOS	76
FIGURA 4.22 DIAGRAMA DE CLASES DEL DISEÑO CASO DE USO GESTIONAR SERVICIO DE REPORTE.	77
FIGURA 4.23 DIAGRAMA DE CLASES DEL DISEÑO CASO DE USO REGISTRAR EVENTO	78
FIGURA 4.24 DIAGRAMA DE CLASES PERSISTENTES.....	79
FIGURA 4.25 REPRESENTACIÓN DE LA BASE DE DATOS.....	79
FIGURA 4.26 REPRESENTACIÓN DE LA DISTRIBUCIÓN FÍSICA DE LA APLICACIÓN.....	80
FIGURA 4.27 COMPOSICIÓN DE LOS PAQUETES.....	81
FIGURA 4.28 DIAGRAMA GENÉRICO PARA CADA CU.....	82
FIGURA 4.29 DIAGRAMA GENÉRICO PARA CADA CU DE SERVICIO.....	83

Introducción

El Ministerio de las Fuerzas Armadas Revolucionarias cuenta con un conjunto de sistemas para el procesamiento de la información, implementados en diferentes plataformas, cada uno concebido de manera independiente. Surge entonces la necesidad de relacionar todos los sistemas y a la vez controlar el acceso de los usuarios a estos de una manera centralizada así como la auditoría de las principales acciones realizadas. La manera en que esta implementada la seguridad y auditoría actualmente implica numerosas dificultades.

En la entidad ¹se utilizan y se desarrollan numerosas aplicaciones de gestión y de procesos de control que por lo general no tienen implementado un módulo que se encargue de la observación de las operaciones y acciones realizadas por los usuarios una vez que acceden a las mismas. Debido a regulaciones y normas establecidas dentro de la institución, la observación antes mencionada es valiosa para algunas áreas que se encargan de preservar la seguridad e integridad de la información que se maneja en el MINFAR. De manera general algunas de las mayores insatisfacciones se relacionan a continuación:

- Los usuarios realizan diferentes operaciones en las aplicaciones que no son registradas ni controladas.
- Si existe algún tipo de alteración en los datos ya sea con buenas o malas intenciones, resulta imposible verificar la fuente del daño.
- Es posible verificar en algunas aplicaciones las operaciones realizadas a nivel de aplicación pero no las que se realizan a nivel de BD.
- No es posible elaborar reportes de acciones realizadas por los usuarios en diferentes períodos de tiempo, dato importante que utilizan algunas áreas de la institución.

Todo lo anterior está provocado, básicamente, por **la no existencia de un sistema informático que permita controlar los eventos y acciones de los usuarios en las diferentes aplicaciones utilizadas en la entidad, además de gestionar la información obtenida del proceso de auditoría**, esto se puede definir como la causa que provoca la situación problemática y por ende **el problema a resolver**.

¹ Cuando se hace referencia a la palabra "entidad", se está hablando de la centro donde se desarrolla el presente trabajo: UCID (Unidad de Compatibilización, Integración y Desarrollo de Productos para la Defensa)

Objeto de estudio

- Los procesos de auditoría funcional y de control en las aplicaciones dentro del entorno empresarial así como los principales parámetros a medir para su gestión.

Campo de acción

Se deben tener en cuenta los siguientes aspectos:

- Los procesos del control de:
 - ✓ Operaciones sobre la aplicación (Traceo de Web accedidas, tiempo, frecuencia, procedencia).
 - ✓ Operaciones sobre datos (Consultas realizadas a la Base de Datos y ficheros actualizados).

Para dar solución a la situación anteriormente descrita se trazan como **objetivos**:

General

Elaborar el Diseño como base para la implementación del Subsistema de Auditoría que permite controlar los eventos y acciones de los usuarios en las diferentes aplicaciones realizadas en la entidad, así como procesar la información obtenida del proceso de auditoría.

Específicos

- Identificar los principales procesos para la Auditoría de aplicaciones en el entorno empresarial.
- Caracterizar los procesos de control de eventos de los usuarios en las aplicaciones informáticas utilizadas o desarrolladas en el MINFAR.
- Analizar las principales metodologías de desarrollo de software y herramientas asociadas a aplicaciones Web, así como los lenguajes de programación y software de soporte.
- Aplicar la metodología de desarrollo de software y las herramientas seleccionadas como apoyo a la confección de la propuesta de diseño.

- Realizar el modelo de diseño de clases, modelo de implementación y del modelo físico de datos del subsistema.

Como guía de la investigación se propone la siguiente idea a defender:

Si se desarrolla una propuesta de diseño de una aplicación informática que gestione el almacenamiento de la información personalizada de las acciones que se realizan sobre las aplicaciones desarrolladas en la entidad se garantiza entonces la base necesaria y el punto de partida para su implementación.

Capítulo 1 Fundamentación Teórica

1.1 Introducción

En este capítulo se analizan los aspectos fundamentales para la realización del producto propuesto, teniendo en cuenta los elementos del objeto de estudio y el campo de acción. Todo parte con un breve análisis de la auditoría en aplicaciones, detallando conceptos importantes relacionados con el tema, seguido se abunda sobre la situación actual del control de la auditoría en la entidad cliente y las herramientas existentes para dicho control de eventos en el mundo. Se hace un análisis de las metodologías de desarrollo de software, herramientas y lenguajes de modelación para seleccionar los que serán utilizados como base del diseño.

1.2 Objeto de estudio

1.2.1 Objetivo estratégico de la organización

El objetivo fundamental y estratégico de la organización es la preparación del país para la defensa y la lucha armada ante una posible agresión interna o extranjera. El objetivo general de este trabajo se concentra en el aumento del control, mediante la automatización, de todos los recursos materiales, financieros y humanos de la entidad.

1.2.2 Flujo actual de los procesos

Proceso de auditoría

La mayoría de las aplicaciones existentes en la entidad no presentan gestión de auditorías, las que lo tienen implementado lo hacen de manera independiente utilizando diferentes criterios y formas generalmente estos criterios son los que trae por defecto el servidor Web (ficheros logs). También las realizan a nivel de gestor usando triggers, controlando acciones o eventos que ocurren sobre la Base de Datos, acciones sobre ficheros, páginas visitadas, etc.

De manera general el proceso actual se limita a realizar la auditoría a nivel de servidor Web, analizando los ficheros de logs y a nivel de gestor de base de datos mediante aplicaciones desktop y configuración de procedimientos en las bases de datos encargados de almacenar información de la persistencia de datos.

1.3 Procesos Objetos de Automatización

La automatización deseada abarcará los procesos de recepción de la información de la auditoría realizada en todas las aplicaciones, su almacenamiento y posterior uso en la elaboración de reportes detallados. Para esto incluirá los procesos de registro de posibles eventos y la configuración de su auditoría.

1.4 Sistemas automatizados existentes vinculados al campo de acción

La importancia que se le concede a este trabajo es precisamente que es un comienzo para el estudio del tema de auditoría en el centro UCID. Actualmente en ningún módulo del centro existe una aplicación que realice auditoría.

1.5 La auditoría en aplicaciones de software

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

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

En estos ficheros se almacena:

- *Páginas visitadas (Qué)*: Páginas de la aplicación que son visitadas por un usuario determinado o acción realizada.
- *IP del visitante (Desde)*: Dirección de donde proviene la conexión, o sea, dirección de la PC donde está ubicado el usuario que accede a la aplicación.
- *Fecha y hora de la conexión (Cuándo)*: Datos del momento de la conexión a la aplicación (fecha y hora, etc.).
- *Sistemas (Dónde)*: Aplicación en la cual realizó acciones el usuario.

- *Datos del usuario (Quién):* Esto solo sería en el caso de que el usuario este previamente registrado en la aplicación, se registraría el nombre y otros datos

La siguiente figura ilustra lo anteriormente expresado:



Figura. 1.1 Conceptos presentes en los Logs de Auditoría

Actualmente la auditoría en aplicaciones se realiza almacenando conceptos como los mencionados anteriormente y muchos más datos que no son tan usados. Existen diferentes tipos de logs como por ejemplo logs de error, logs de análisis y logs de datos, cada uno con su propia utilidad.

En la actualidad la propia preocupación por el control de estos elementos ha hecho que surjan aplicaciones capaces de procesar toda esa información y mostrar resúmenes estadísticos relevantes así como otra información importante de apoyo al control sobre las aplicaciones Web.

1.6 Sistemas utilizados en la Auditoría de Aplicaciones de software

A nivel mundial existen varios sistemas que realizan el control y auditorías en aplicaciones Web, no todos utilizan las mismas políticas y estrategias pero a manera general han conformado un conjunto de herramientas muy útiles e interesantes. A continuación se presentan algunos de estos sistemas:

HttpBee: Puede crear múltiples subprocesos e integrar un motor de procesamiento de lenguajes script además de una línea de comandos, aparte de esto también puede correr como servicio por lo tanto

puede ser parte de un framework distribuido. Realmente de esta herramienta no hay mucha documentación ni guías por lo que los usuarios deben de tener un alto grado de conocimientos acerca de la materia. (DragonJAR, 2006)

Inguma: Realiza pruebas de penetración, está escrita enteramente en Python. El framework incluye los módulos para descubrir los hosts, incluyendo información sobre ellos, sacar nombres de usuario y las contraseñas por fuerza bruta. Fue orientado inicialmente a atacar los sistemas Oracle pero puede ser utilizado para cualquier otro tipo. Registra los módulos que están siendo utilizados para detectar redes. Es necesario señalar que esta herramienta no trabaja en todos los Sistemas Operativos Windows. (DragonJAR, 2006)

Otras herramientas se encargan de la auditoría Web pero basada generalmente en la creación de estadísticas gráficas avanzadas. Algunas de ellas son:

AWStat (*Advanced Web Statistics*), trabaja desde líneas de comandos, y muestra en pocas páginas, toda la información que se encuentran en registros de una PC (registros de acceso del servidor web). Puede analizar archivos log desde Internet Information Server (formato de log W3C), archivos log de Apache (formato log NCSA combinado/XLF/ELF o formato de log común/CLF).

Algunas de las informaciones que muestra son: (Miguel, 2006)

- Número de visitas y número de visitantes únicos.
- Duración de las visitas y últimas visitas.
- Usuarios autenticados.
- Días de la semana y horas de mayor tráfico (páginas, cantidad de visitas, KB por cada hora y día de la semana).
- Dominios/países de visitantes.
- Páginas más vistas, páginas de entrada y salida.
- Tipos de archivo.
- Navegadores utilizados (páginas, cantidad de visitas, KB por cada usuario, versión, etc.).
- Sistemas Operativos usados.

Webtacorae: Al igual que AWStat genera estadísticas personales de las aplicaciones Web, mostrando así el flujo de visitas a la aplicación y cuál es la de preferencia de los visitantes.

Entre las acciones que realiza se encuentran: (ABCdatos, 1999)

- Recopila la información que pueda interesar sobre visitantes en formato HTML.

- Permite vigilar las entradas de una IP determinada.
- Permite excluir de las estadísticas una IP determinada.
- Permite recibir mensualmente las estadísticas en la dirección de correo que se le indique.
- Permite proteger las estadísticas usando el sistema del servidor Apache: ficheros de seguridad htaccess y htpasswd.
- Permite cambiar la contraseña del sistema de protección de las estadísticas.
- Incorpora un contador global de visitas.
- Incorpora un contador de usuarios online.

Como complemento de esta herramienta esta el GestBitacora que permitirá hacer un mejor manejo de Webtacora.

Partiendo de la existencia a nivel global de varias aplicaciones para controlar los eventos y operaciones que se realizan en la entidad además de la importancia que tienen las mismas para la seguridad de la información que se maneja, y, por supuesto teniendo en cuenta que ninguna de las aplicaciones anteriormente descritas se adapta como elemento solucionador de la situación problemática actual, se toma la decisión de diseñar un sistema para la gestión de la información de los eventos o sucesos en las aplicaciones desarrolladas en la entidad de manera centralizada para el ERP².

² Son sistemas de información gerenciales que integran y manejan muchos de los negocios asociados con las operaciones de producción y de los aspectos de distribución de una compañía comprometida en la producción de bienes o servicios.

1.7 Conclusiones

A lo largo del capítulo se analizaron los elementos que marcan el inicio de la investigación comenzando por tener en cuenta los conceptos que están presentes o asociados de alguna forma con el dominio del problema y así se detallaron otras soluciones existentes. Se está en condiciones ahora de seleccionar otros elementos sobre los cuales se sustentará el diseño.

Capítulo 2 Tendencias y tecnologías actuales

2.1 Introducción

Haber seleccionado la metodología de software es solo el inicio del proceso, aun no se ha definido el ¿qué? ni el ¿con qué? Precisamente estas dos interrogantes responden al tipo de software a desarrollar (convencidos de que es un software lo necesario) y a las herramientas, lenguaje de programación y tecnologías de soporte en las cuales se basará el diseño de la aplicación.

En este capítulo se realiza un estudio detallado de las principales tendencias y tecnologías utilizadas en el proceso de desarrollo de software actualmente con el objetivo de identificar las que pueden influir de una mejor manera en la calidad del software y en la satisfacción de las necesidades del cliente con su puesta en práctica.

2.2 Aplicaciones Web

Se denomina aplicación Web a una aplicación cliente – servidor que usa un navegador Web como programa cliente, y realiza un servicio interactivo para conectar, a través de Internet., servidores a lo largo y ancho del mundo. Estas presentan contenido de forma dinámica basados en los parámetros de las peticiones realizadas por parte del usuario así como tomando en cuenta las limitantes de seguridad. (Shklar, y otros, 2003).

Actualmente se ha incrementado el uso de las aplicaciones distribuidas, y con ello el uso de la Internet por las grandes ventajas que esta ofrece. El futuro de las grandes empresas es integrarse entre ellas y los caminos a la integración se abren por el uso de aplicaciones Web.

Existen numerosas aplicaciones de Software Libre para la implementación de aplicaciones en plataforma Web, generando esto una gran ventaja para el desarrollo de dichas aplicaciones y siendo de gran utilidad para los que desean incursionar en este ventajoso mundo.

Con los elementos actualmente relacionados y teniendo en cuenta las necesidades reales del cliente así como la topografía y composición de la entidad, que se encuentra distribuida por todo el territorio nacional se concluye que el diseño de una aplicación Web es el camino correcto.

2.2.1 Servicios Web

Un Servicio Web no es más que un servicio que se brinda a través de una aplicación software al cual se accede mediante Internet. Al igual que una página está definida por una URL (Uniform Resource Locator o Localizador Uniforme de Recursos), los Servicios Web están definidos por un URI (Uniform Resource Identification o Identificador Uniforme de Recursos). Los "Servicios Web" también están definidos como un sistema de software diseñado para permitir intercambio de información y funcionalidades máquina a máquina en una red.

En general, los Servicios Web son sólo APIs Web (Application Program Interface) que pueden ser accedidas en una red, como internet, y ejecutadas en un sistema remoto. Permiten que diferentes aplicaciones, realizadas con diferentes tecnologías y ejecutándose en toda una variedad de entornos puedan comunicarse e integrarse contribuyendo a sistemas más flexibles y estables, la confidencialidad es uno de sus atributos donde cada paquete de información viaja seguro a través de la red ya que se puede utilizar mecanismos de seguridad como por ejemplo la encriptación de los mensajes.

Generalmente los Servicios Web se basan en una arquitectura de software SOA (Service Oriented Architecture) traduciéndolo a español sería Arquitectura Orientada a Servicios. SOA es una tendencia creciente que intenta reconciliar la visión técnica y de negocios, basándose en estándares abiertos y promoviendo la interoperabilidad entre diversas organizaciones y plataformas de manera eficiente y flexible a los cambios. Actualmente todos los proveedores de tecnología están abocados a soportar este tipo de arquitecturas tanto en empresas pequeñas en crecimiento como en grandes corporaciones.

Es importante mencionar estándares y utilidades que juegan un papel muy importante en la creación de los Servicios Web como por ejemplo el SOAP, XML, SWDL, UDDI, HTML; aclarando que no es de carácter obligatorio hacer uso de estos estándares, solo que es bastante recomendable.

SOAP

Acrónimo de Simple Object Access Protocol, es decir protocolo simple de acceso a objetos. SOAP es un protocolo ligero de mensajes XML que se usa para codificar la información de los mensajes de petición y respuesta de los Servicios Web que se envían a través de una red. Los mensajes SOAP son independientes de los sistemas operativos y de los protocolos, pueden ser transportados usando una variedad de protocolos Internet, incluyendo SMTP y HTTP. Para poner un sencillo ejemplo si se

pretende el uso de Servicios Web a la manera de la programación orientada a objetos, usar un Servicios Web sería como usar una clase, accediendo a sus métodos y atributos. Para usar una función específica se debe instanciar, es decir, llamarla pasando por referencia valores o parámetros, luego se espera un resultado, si es que existe; en este caso ese es el rol que juega SOAP, el de describir esta operación, llamar a la función, pasar los parámetros requeridos y describir cómo será la respuesta y cómo se recibirá.

Algunas de las Ventajas de SOAP son:

- No está asociado con ningún lenguaje: los desarrolladores involucrados en nuevos proyectos pueden elegir desarrollar con el último y mejor lenguaje de programación que exista. SOAP no especifica una API, por lo que su implementación se deja al lenguaje de programación, como en Java, o a la plataforma como Microsoft .Net.
- No se encuentra fuertemente asociado a ningún protocolo de transporte: La especificación de SOAP no describe como se deberían asociar los mensajes de SOAP con HTTP. Un mensaje de SOAP no es más que un documento XML, por lo que puede transportarse utilizando cualquier protocolo capaz de transmitir texto.
- SOAP aprovecha XML para la codificación de los mensajes, en lugar de utilizar su propio sistema de tipo que ya están definidas en la especificación esquema de XML.
- Permite la interoperabilidad entre múltiples entornos: SOAP se desarrolló sobre los estándares existentes de la industria, por lo que las aplicaciones que se ejecuten en plataformas con dichos estándares pueden comunicarse mediante mensaje SOAP con aplicaciones que se ejecuten en otras plataformas.

WSDL

Cubre las funcionalidades de descripción necesarias para las tecnologías de Servicios Web. Como evolución de los esquemas XML, actúa como mecanismo formal para la definición de los servicios, sus interfaces, puntos de acceso, etc. Uno de los aspectos fundamentales es facilitar la generación de código a partir de descripciones WSDL además de ser el secreto por el cual cualquier aplicación puede acceder a un servicio disponible: solo deberá leer un documento XML.

¿Qué es XML?

Extensible Markup Language (Lenguaje Extensible de Marcas). No es más que un conjunto de reglas para definir etiquetas semánticas que nos organizan un documento en diferentes partes. XML es un metalenguaje ³que define la sintaxis utilizada para definir otros lenguajes de etiquetas estructurados.

A pesar de su sencillez aparente, XML está transformando completamente la creación y el uso de software. El Web revolucionó la comunicación entre usuarios y aplicaciones, XML la comunicación entre aplicaciones o, de forma más general, la comunicación entre equipos, pues ofrece un formato de datos universal que permite adaptar o transformar fácilmente la información.

¿Por qué XML es utilizado en los Servicios Web?

- Es un estándar abierto, es decir, que es reconocido mundialmente ya que muchas compañías tecnológicas integran en su software compatibilidad con dicho lenguaje.
- Simplicidad de sintaxis: esto quiere decir que es muy fácil de escribir código en XML y la representación de los datos es casi entendible por cualquier ser humano. Esto lo hace muy flexible a la hora de querer representar datos de cualquier especie, bastará con contar con cualquier editor de texto y aprender unas cuantas intrusiones básicas y ya se está en condiciones de escribir código XML. El hecho de que XML sea tan fácil de codificar y de entender lo hace el lenguaje ideal para utilizarlo en los Servicios Web.
- Independencia del protocolo de Transporte, el hecho de que XML es un lenguaje de Marcado de Texto, no necesita de ningún protocolo de transporte especial, solo necesita de un protocolo que pueda transferir texto o documentos simples.

Para la utilización de servicios Web no se hace obligatorio usar una Arquitectura Orientada a Servicios, estos pueden estar disponibles y funcionales en la red y accesibles entre aplicaciones aun sin utilizar todos los elementos de SOA. Teniendo en cuenta lo anteriormente descrito y la gran cantidad de ventajas que ofrecen, se decide incluir en los principios de diseño la utilización de Servicios Web.

³ Lenguaje natural o formal que se usa para explicar o hablar del lenguaje mismo o de una lengua: las gramáticas formales son metalenguajes.

2.3 Lenguajes de Programación Web

A la par del desarrollo de las aplicaciones Web también está el avance y surgimiento de grandes lenguajes de programación. La misma exigencia de aplicaciones más rápidas, eficientes y seguras ha hecho que se necesiten lenguajes que permitan obtener esto y mayor interoperabilidad. Existen muchos como por ejemplo PHP, ASP, ASP.NET, ColdFusion, Perl, Java Script y si se mencionan muchos más tomaría tiempo. Se hace necesario entonces seleccionar el lenguaje de programación que se utilizará. Para el diseño de la aplicación deseada se escogió PHP. A continuación se presentan algunas características y el por qué de su selección.

PHP

Es un lenguaje de programación gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. Su creador Rasmus Lerdorf ha recibido muchas contribuciones de otros desarrolladores debido a su política de código abierto. Actualmente PHP se encuentra en su versión estable PHP 5.2.3.

PHP puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, incluyendo Linux, muchas variantes Unix (incluido HP-UX, Solaris y OpenBSD), Microsoft Windows y Mac OS X. PHP es soportado por la mayoría de servidores Web de hoy en día, incluyendo Apache, Microsoft Internet Information Server, Personal Web Server, Netscape e iPlanet. Tiene módulos disponibles para la mayoría de estos servidores, para aquellos otros que soporten el estándar CGI, PHP puede usarse como procesador CGI.

En el desarrollo de Servicios Web PHP no se ha quedado atrás. Basándose en SOAP, el protocolo más popular para la creación y consumo de Servicios Web, su principal implementación es NuSOAP, una librería de clases con utilidades y muy práctica para los inicios; pero tiene el inconveniente de tener que actualizarse constantemente y no viene incorporada al PHP. Por esto la mejor opción es el uso del PHP 5.2.3 que desde sus inicios incorpora la extensión `php_soap` para el manejo del protocolo SOAP, además de ser orientado a objetos, ganando en eficiencia, estabilidad y facilidades de uso.

¿Por qué utilizar PHP y no otras opciones?

- Viene acompañado por una excelente biblioteca de funciones que permite realizar cualquier labor (acceso a base de datos, encriptación, envío de correo, gestión de un comercio electrónico, XML, creación de PDF).
- Está siendo utilizado con éxito en varios millones de sitios Web.

- Es multiplataforma, funciona en todas las plataformas que soporten apache.
- Es software libre. Se puede obtener en la Web y su código está disponible bajo la licencia GPL. Este es uno de los aspectos fundamentales para la elección de este lenguaje.

Sus cuatro grandes características: **Velocidad, estabilidad, seguridad y simplicidad.**

- **Velocidad:** No solo la velocidad de ejecución, la cual es importante, sino además no crea demoras en la máquina. Por esta razón no debe requerir demasiados recursos de sistema. PHP se integra muy bien junto a otro software, especialmente bajo ambientes Unix cuando se configura como módulo de Apache.
- **Estabilidad:** La velocidad no sirve de mucho si el sistema se cae cada cierta cantidad de ejecuciones. Ninguna aplicación es 100% libre de bugs (errores de código); pero teniendo de respaldo una increíble comunidad de programadores y usuarios es mucho más difícil para lo bugs sobrevivir. PHP utiliza su propio sistema de administración de recursos y dispone de un sofisticado método de manejo de variables, conformando un sistema robusto y estable.
- **Seguridad:** El sistema debe poseer protecciones contra ataques. PHP provee diferentes niveles de seguridad, estos pueden ser configurados desde el archivo php.ini.
- **Simplicidad:** Se les debe permitir a los programadores generar código productivamente en el menor tiempo posible. Usuarios con experiencia en C y C++ podrán utilizar PHP rápidamente.

Una vez seleccionado el lenguaje se está en condiciones de determinar la base arquitectónica del diseño.

2.4 Arquitectura de software

“Una arquitectura es el conjunto de decisiones significativas sobre la organización del sistema de software, la selección de los elementos estructurales y sus interfaces, con los que se compone el sistema, junto con su comportamiento tal como se especifica en las colaboraciones entre esos elementos, la composición de esos elementos estructurales y de comportamiento en subsistemas progresivamente más amplios, y el estilo de arquitectura que guía esta organización -estos elementos y sus interfaces, sus colaboraciones, y su composición-“. (Booch, 1999)

La Arquitectura de software (AS) constituye la piedra angular para el éxito de cualquier sistema de software, es el principal descriptor de la calidad de software ya que muestra atributos como el rendimiento o fiabilidad. Una arquitectura bien seleccionada ha de satisfacer las necesidades de calidad, es pieza clave en el éxito de los proyectos de software, por lo que un mal uso de la misma es una garantía de fracaso.

Según I, Jacobson, G. Bosch y J. Rumbaugh la arquitectura es el “conjunto de decisiones significativas acerca de la organización de un sistema de software, la selección de elementos estructurales a partir de los cuales se compone el sistema, y las interfaces entre ellos, junto con su comportamiento tal y cual se especifica en las colaboraciones entre estos elementos, la composición entre estos elementos estructurales y de comportamiento en subsistemas progresivamente mayores, y el estilo arquitectónico que guía esta organización.

2.4.1 Arquitectura cliente/servidor

Generalmente, cuando se habla de aplicaciones Web es necesario introducir el tema de la Arquitectura cliente/servidor.

Arquitectura cliente/servidor, consiste en varios clientes distribuidos en diferentes nodos, conectados en una red a uno o varios nodos servidores, donde el servidor puede servir a varios clientes a la vez. En los nodos clientes generalmente se pueden encontrar lo referente a la presentación al usuario y en los nodos servidores la lógica del negocio.

La arquitectura cliente/servidor es una forma de dividir y especializar programas y equipos de cómputo de forma que la tarea que cada uno de ellos realiza se efectúa con la mayor eficiencia posible y permita simplificar las actualizaciones y mantenimiento del sistema. (RÍOS GIL, 2005).

Las características más importantes que se distinguen en el modelo cliente/servidor son: (DUQUE MÉNDEZ, 2006)

- Orientado a servicios. El servidor los ofrece y el cliente los consume.
- Compartición de recursos. Servicios ofrecidos a muchos clientes. Un servidor puede atender muchos clientes que solicitan esos servicios.
- Transparencia de ubicación. El servidor es un proceso que puede residir en el mismo ordenador que el cliente o en uno distinto a lo largo de una red. Un programa puede ser un servidor en un momento y convertirse en un cliente posteriormente.
- Mezcla e igualdad. Esta es una de las más importantes ventajas de este paradigma. Una aplicación cliente/servidor, idealmente es independiente del hardware y de sistemas operativos; mezclando e igualando estas plataformas.
- Interacción a través de mensajes, para envío y respuestas de servicios.
- Servicios encapsulados, exponiendo los servicios a través de interfaces, lo que facilita la sustitución de servidores sin afectar los clientes; permitiendo a la vez una fácil escalabilidad.

2.4.2 Estilos Arquitectónicos

Uno de los aspectos fundamentales dentro de la Arquitectura de Software son los estilos arquitectónicos. Un estilo es un concepto descriptivo que define una forma de articulación u organización arquitectónica. El conjunto de los estilos cataloga las formas básicas posibles de estructuras de software, mientras que las formas complejas se articulan mediante composición de los estilos fundamentales.

Para el diseño de la aplicación es necesario utilizar un estilo arquitectónico, por lo que se realiza a continuación un estudio detallado de los más utilizados en el desarrollo de aplicaciones Web.

Arquitectura en Capas

El patrón Capas se relaciona con la arquitectura lógica; es decir, describe la organización conceptual de los elementos del diseño en grupos, independiente de su empaquetamiento o despliegue físico. (Larman)

Es uno de los patrones más generalizados y utilizados en el desarrollo de aplicaciones web a nivel global y sencillo de implementar. Se resume en lograr:

- Organizar la estructura lógica de gran escala de un sistema en capas separadas de responsabilidades distintas y relacionadas, con una separación clara y cohesiva de intereses como que las capas "más bajas" son servicios generales de bajo nivel, y las capas más altas son más específicas de la aplicación.
- La colaboración y el acoplamiento es desde las capas más altas hacia las más bajas; se evita el acoplamiento de las capas más bajas a las más altas.

Muchos de los frameworks⁴ para el desarrollo de aplicaciones Web que existen hoy en día están implementados bajo otros estilos arquitectónicos y no en 3 capas, que no puede ser violado. ¿Será entonces una buena practica desechar la reutilización de código que plantean los frameworks? No es una buena idea, se propone entonces uno de esos estilos.

⁴ Framework: Es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un *framework* puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Modelo – Vista- Controlador

Es un patrón de diseño que plantea la separación de diferentes clases en dependencia de la función que realizan de modo tal que sea posible manejar dinámicamente la forma en que se procesan solicitudes y se gestiona la manera en que se muestran resultados al usuario final. En otras palabras separa la presentación del dominio de la aplicación.

Es un principio que utilizan muchos frameworks para basar su funcionamiento, la idea de “Don’t call us, we’ll call you” (No nos llame, nosotros lo llamaremos a usted). Esa idea ha hecho que los frameworks que implementan MVC se puedan usar sencillamente implementando interfaces o extendiendo de una clase abstracta que brinda el framework. Muchos de los más usados en PHP son:

Framework	Estilo
Prado	MVC
<i>CakePHP</i>	MVC
<i>Simfony</i>	MVC
Kumbia	MVC
CodeIgniter	MVC
<i>Zend Framework</i>	MVC

Tabla 2.1 Ejemplos de framework que utilizan MVC

A continuación se representa el funcionamiento de dicho estilo arquitectónico:

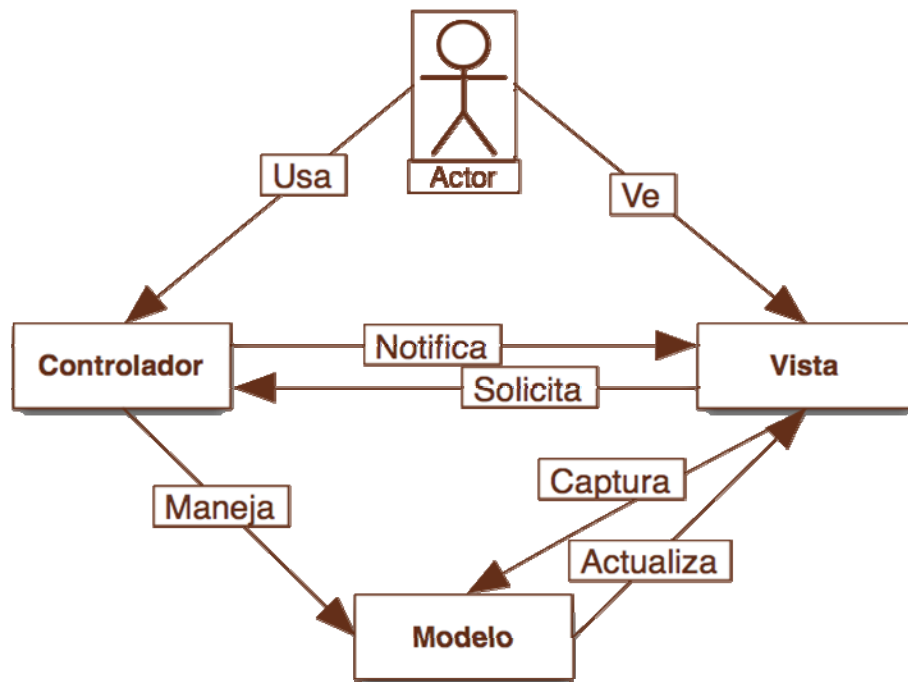


Figura 2.1 Funcionamiento MVC

De la representación anterior se pueden identificar las tres partes componentes:

El Modelo: es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.

La Vista: es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa con el Modelo a través de una referencia al propio Modelo.

El Controlador: es el objeto que proporciona significado a las órdenes del usuario actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo.

Debido a la gran utilidad que tiene y a la importancia que tiene la reutilización de código que brindan

los frameworks se propone utilizar el estilo arquitectónico Modelo – Vista – Controlador. Ahora ¿Qué framework de los mostrados y de otros usados podemos utilizar?.

EXT

Es un framework para JavaScript utilizado en el desarrollo de aplicaciones Web con AJAX. Tiene una librería inmensa que permite configurar las interfaces Web de manera semejante a aplicaciones desktop.

Tiene incluidos la mayoría de los controles de los formularios Web incluyendo Grids para mostrar datos y elementos semejantes a la programación desktop como los formularios, paneles, barras de herramienta, menus y muchos otros. Dentro de su librería de componentes incluye componentes para el manejo de datos, lectura de XML, lectura de datos JSON e implementaciones basadas en AJAX.

Zend Framework

Los frameworks por lo general presentan una estructura organizada que obliga a los programadores a seguir estándares y a trabajar de manera organizada. El uso de estas aplicaciones ha demostrado que la organización de la programación influye notablemente en la calidad de las aplicaciones.

Zend Framework es uno de los más utilizados para PHP y utiliza el estilo MVC como base de su funcionamiento. Es fácilmente integrable a las aplicaciones debido a su composición ya que contiene diferentes clases de gran utilidad como por ejemplo en la búsqueda dinámica de ficheros a incluir.

Cuenta con un importante mecanismo de manejo de controladores y vistas por lo que se propone tenerlo en cuenta para el diseño de estos dos componentes de la arquitectura.

Doctrine PHP

Doctrine es un potente y completo sistema ORM (Object Relational Mapper) para PHP 5.2+ con un DBAL (Data Base Abstraction Layer) incorporado.

Entre muchas otras cosas tienes la posibilidad de exportar una base de datos existente a sus clases correspondientes y también a la inversa, es decir convertir clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos. Por otro lado, como la librería es bastante grande, ésta tiene un método para ser ‘compilada’ al pasar a producción.

Su principal ventaja radica en poder acceder a la base de datos utilizando la programación orientada a objetos (POO) debido a que doctrine utiliza el patrón Active Record para manejar la base de datos, tiene su propio lenguaje de consultas y trabaja de manera rápida y eficiente. Es fácilmente integrado a

los principales frameworks de desarrollo utilizados actualmente, por lo que se propone su uso en el modelo.

2.5 Gestores de Bases de Datos

Una aplicación de software no está compuesta solamente por el componente de la lógica, ni la manera que se muestra un resultado procesado al usuario. Las aplicaciones manejan información importante, la cual es almacenada y procesada. Precisamente de éstas operaciones, entre otras fundamentales, se encargan los Sistemas Gestores de Bases de Datos. Para el diseño de la aplicación si bien no es relevante el SGBD a utilizar si es importante al menos seleccionarlo para tener criterios específicos a la hora de representar mediante modelos la información. A continuación se describen las características fundamentales de algunos de los más utilizados, del gestor seleccionado y los criterios seguidos para su selección.

PostgreSQL

Este gestor de bases de datos posee una estabilidad y confiabilidad legendaria, nunca ha presentado caídas en varios años de operación de alta actividad. Está disponible para 34 plataformas Unix en la última versión estable, existe una versión para Windows usando la plataforma Cygwin. Fue diseñado para ambientes de alto volumen. Escala muy bien al aumentar el número de CPUs y la cantidad de memoria. Soporta transacciones y desde la versión 7.0, claves ajenas con comprobaciones de integridad referencial. Tiene mejor soporte para vistas y procedimientos almacenados en el servidor, además tiene ciertas características orientadas a objetos.

MySQL

Según definición de sus autores: es un servidor de base de datos muy rápido, robusto, multitarea y multiusuario. Tiene enfoque relacional, soporta clientes en C, C++, Eiffel, Java, Perl, PHP, Python y Tcl. Trabaja en diferentes plataformas además de soportar múltiples idiomas. Completo y optimizado uso del SQL.

En cuanto a seguridad confía en la propia del sistema a efectos de robo de las bases de datos, caída del sistema. Usa Listas de Control de Acceso para todas las conexiones, consultas y otras operaciones.

Actualmente este gestor se declara como software libre pero en enero del 2008 fue comprado por la Sun Microsystems caracterizada por comercializar software propietario, lo cual es una razón para no utilizarlo.

Razones para usar Postgres y no MySQL:

- PostgreSQL es una base de datos diseñada para ser de tipo empresarial.
- Algunas características de PostgreSQL aún no están disponibles o estables en MySQL:
 - ✓ Triggers (5.1 rudimentarios).
 - ✓ Vistas (5.0).
 - ✓ Secuencias.
 - ✓ Herencia.
 - ✓ Cursores (5.0).
 - ✓ Procedimientos almacenados (5.0 lenguaje único).

SQL Server

Microsoft SQL Server 7.0 constituye un lanzamiento determinante para los productos de bases de datos de Microsoft. SQL Server es el RDBMS de elección para una amplia gama de clientes corporativos y Proveedores Independientes de Software que construyen aplicaciones de negocios. Las necesidades y requerimientos de los clientes han llevado a la creación de innovaciones de producto significativas para facilitar la utilización, escalabilidad, confiabilidad y almacenamiento de datos. Es software propietario.

Razones para usar Postgres y no SQL-Server:

- Estabilidad, Confiabilidad, SQL Server corre solo en MS-Windows.
- Adherencia a Estándares.

Otras ventajas de PostgreSQL con respecto a SQL Server

- Drivers ODBC.
- Herramientas de migración e integración.

2.6 Metodologías de desarrollo de software.

Partiendo de la necesidad de diseñar una aplicación informática, se hace necesario entonces seleccionar una metodología de desarrollo de software para organizar y guiar todo el proceso de desarrollo hasta la etapa de análisis y diseño.

Actualmente existen ciertas tendencias fundamentadas en la idea de construir sistemas más grandes y complejos. Se quiere de un software que esté mejor adaptado a las necesidades actuales, lo que a su vez hace que el mismo sea más complejo, pero no sólo eso, sino que además se quiere que sea lo más rápido posible.

Existen hoy a nivel mundial muchas metodologías para el desarrollo de software y cada una de ellas tiene su propia forma de realizar este proceso de manera que se obtenga un producto con calidad y con un costo mínimo. Dentro de las metodologías más utilizadas e importantes en el proceso de desarrollo de software a nivel internacional se encuentran: Rational Unified Process (RUP) y Extreme Programming (XP). A continuación se detallan ambas metodologías para una mejor comprensión.

2.6.1 RUP

Rational Unified Process (RUP) es una propuesta de proceso para el desarrollo de software basada en la orientación a objetos, el desarrollo iterativo y la modelación visual usando UML para describir un sistema, lo cual permite incorporar al proceso de desarrollo de software un mejor control de los requerimientos y cambios. Posibilita la distribución del trabajo en diversos frentes de forma simultánea. A pesar de ser una metodología desarrollada directamente para el trabajo con clases y objetos brinda amplias posibilidades con el manejo eficiente del tiempo de diseño e implementación de aplicaciones Web.

Características principales de RUP

- Guiado por Casos de Uso.

Un sistema software ve la luz para dar servicio a sus usuarios. Por tanto, para construir un sistema con éxito se debe conocer lo que sus futuros usuarios necesitan y desean.

Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Representan requerimientos funcionales. Todos los casos de uso juntos constituyen el modelo de casos de uso el cual describe la funcionalidad total del sistema. Los casos de uso guían la arquitectura del sistema y la arquitectura del sistema influye en la selección de los casos de uso. Por tanto, esta característica es de suma importancia, mantiene organizado y bien definido el trabajo y cada uno de los procesos que se realizan en la entidad y que en algún momento serán ejecutados por el sistema dando la posibilidad de darles atención por separado y teniendo en cuenta prioridades.

- Centrado en la arquitectura.

La arquitectura en un sistema software se describe mediante diferentes vistas del sistema en construcción. Este concepto incluye los aspectos estáticos y dinámicos más significativos del sistema. Esta se refleja en los casos de uso pues cada producto tiene tanto una función como una forma, ninguna es suficiente por si sola. Definiendo una buena arquitectura en el sistema, se evitaría cualquier tipo de mal entendido entre desarrolladores y no desarrolladores, mantener una buena estructura de la lógica arquitectónica de un sistema orienta y guía al personal involucrado en el mismo. Esta característica de RUP permite respetar en todo momento y en toda etapa de desarrollo la

línea base de la arquitectura y por tanto limita implicaciones de tiempo en correcciones arquitectónicas y eleva la calidad del software resultante.

- Iterativo e incremental.

Resulta práctico dividir el trabajo en partes más pequeñas o mini proyectos los que no son más que iteraciones que resultan en un incremento o posibles versiones que serán mejoradas a medida que pasa el tiempo.

Una iteración es una secuencia de actividades con un plan establecido y criterios de evaluación, cuyo resultado es una versión del software. Las iteraciones tienen muchos beneficios importantes, reducir el coste del riesgo al coste de un solo incremento, las necesidades del usuario y correspondientes requisitos no deben definirse completamente al principio y permite planificar puntos de control de proyecto e iteraciones por etapas.

2.6.2 XP

La Programación Extrema pertenece a las metodologías ágiles que intervienen en el proceso de desarrollo de software que se basa en la retroalimentación constante entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. La metodología XP surgió como respuesta y posible solución a los problemas derivados del cambio en los requerimientos, en la mayoría de los casos se plantea como una metodología a emplear en los proyectos con riesgos de requisitos muy cambiantes. La generación de documentos o la documentación de cada paso que se realiza en el desarrollo del producto es uno de los elementos más importantes para tener un control o dominio acerca de la marcha del proyecto.

Es una de las metodologías de desarrollo de software más exitosas en la actualidad utilizada para proyectos de corto plazo y un equipo pequeño. 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.

2.6.3 Metodología utilizada

Después de un breve análisis de las metodologías más usadas en el proceso de desarrollo de software se decidió guiar el diseño del sistema utilizando RUP, porque esta metodología, como anteriormente se mencionó, permite un proceso organizado, la mayor cantidad de riesgos se eliminan en cada iteración que se realice y que ésta a su vez puede ser planificada como parte del tiempo total, aunque es importante señalar también que XP tiene una buena característica y es que es una metodología donde el cliente es parte del equipo de trabajo ya que debe estar presente para cualquier duda de los

desarrolladores. En el caso de RUP también existe un vínculo estrecho entre desarrolladores y clientes.

2.7 Lenguaje de modelado

Cuando se trata de desarrollar software y de hacerlo bien siempre se tiene en cuenta la representación mediante esquemas, figuras o modelos de las partes integrantes del mismo. De manera general la historia ha quedado como lección que la utilización de modelos representativos para esbozar las diferentes fases del desarrollo así como los elementos que componen el software ha marcado un punto determinante en la rapidez de elaboración y en la calidad de las aplicaciones. Para esto se utilizan los lenguajes de modelado y en este caso se hace necesario utilizarlos por lo que continuación se analiza a uno de los más consumidos.

RUP utiliza el Lenguaje Unificado de Modelado (Unified Modeling Language, UML) para acomodar todos los esquemas de un sistema de software. Se puede decir que de hecho UML es una parte de RUP, sus desarrollos fueron paralelos.

El UML se define como un lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de software. Es un sistema notacional destinado a los que utilizan conceptos orientados a objetos. UML estandariza los artefactos y la notación, pero no define un proceso oficial de desarrollo. Básicamente UML permite a los desarrolladores visualizar los resultados de su trabajo en esquemas o diagramas estandarizados.

Tiene muchas utilidades, por lo que se toma como determinación utilizar UML como lenguaje de modelado. Se necesita ahora una herramienta que utilice UML como lenguaje y que a la vez permita realizar los modelos necesarios. Para esto tienen mucha utilidad las herramientas CASE (Computer Aided Software Engineering) o Ingeniería de Software Asistida por Computadora.

2.8 Herramienta CASE Visual Paradigm para UML

Visual Paradigm para UML (VP-UML) es una poderosa herramienta CASE visual; la misma no sólo se utiliza para el modelado de software, sino, que además proporciona generación del código e ingeniería inversa. El Generador Instantáneo puede transformar los detalles de un Diagrama de Clases del Diseño en código para los lenguajes de programación más utilizados hoy en día, con un esfuerzo mínimo, dentro de estos lenguajes se pueden mencionar a PHP 5.0, Java, C #, Visual Basic.NET, Action Script 3.0, Python, Ruby y más.

Además de estas breves características que antes se exponen se utiliza Visual Paradigm para modelar el sistema porque es requerimiento específico del cliente.

2.9 Conclusiones

En este capítulo se ha realizado un análisis descriptivo de las tecnologías y tendencias actuales en las cuales se basó el trabajo, exponiendo sobre todo las ventajas que llevan a elegir estas herramientas. Brevemente se expusieron algunas razones por las cuales se utilizaron tanto PostgreSQL como gestor de base de datos y PHP como lenguaje de programación. Se describió cada uno de los elementos integrantes de la futura aplicación centrados en las ventajas de la reutilización de código y rapidez en el desarrollo que brindan los frameworks. Además se hizo un estudio de las metodologías de desarrollo de software y se seleccionó la específica a utilizar por las ventajas que ofrece. Se escogió el lenguaje de modelado para simplificar la representación del sistema y la herramienta a utilizar para realizarla.

Capítulo 3 Presentación de la solución propuesta

3.1 Introducción

El proceso de control de la auditoría tiene suma importancia para supervisar la integridad de la información existente en las aplicaciones que posee la entidad. Dicho proceso actualmente no se tiene en cuenta en los sistemas ubicados en el UCID, por lo que este trabajo está enmarcado en el diseño de una aplicación que realice el control de los eventos de los usuarios sobre las aplicaciones. Una vez conociendo el estado de la entidad y analizando los procesos implicados en la automatización que se quiere se hace una captura de requisitos, sabiendo que, para un sistema constituye uno de los procesos más importantes en la etapa de desarrollo de software pues es donde se definen las funcionalidades del sistema en desarrollo y se analizan detalladamente los requerimientos de software, hardware, etc., necesarios para lograr el correcto funcionamiento del sistema y una gran satisfacción por parte del cliente.

En este capítulo se hace referencia a los principales elementos del proceso de auditoría, sus definiciones y la relación entre ellos, además quedarán definidos y descritos los procesos reales de automatización para el control de auditoría de acuerdo con la petición del cliente final.

3.2 Definición de las entidades y los conceptos principales

Concepto	Descripción
Sistema	Aplicación de software en funcionamiento, accesible al personal de la entidad.
Usuario	Representa a personas o sistemas de la entidad que interactúan con la aplicación.
Acción	Acciones que realizan los usuarios sobre los sistemas, ya sean de acceso al sistema y sus módulos, de acceso a datos, consultas, etc.
Evento	Se registra cada vez que un usuario realiza una acción sobre el sistema, es la manera de ver las acciones de la parte del sistema.
Información	Un conjunto organizado de datos, que puede ser que se encuentre en la aplicación o que se genere después del suceso de un evento.
Reporte	Conjunto de información que se utiliza para controlar el trabajo que se realiza sobre las aplicaciones.

Tabla 3.1 Conceptos de auditoría

3.3 Representación del modelo del dominio

Teniendo en cuenta que en la entidad actualmente no se definen procesos claros para el control de auditoría para aplicaciones Web se decide modelar los conceptos iniciales.

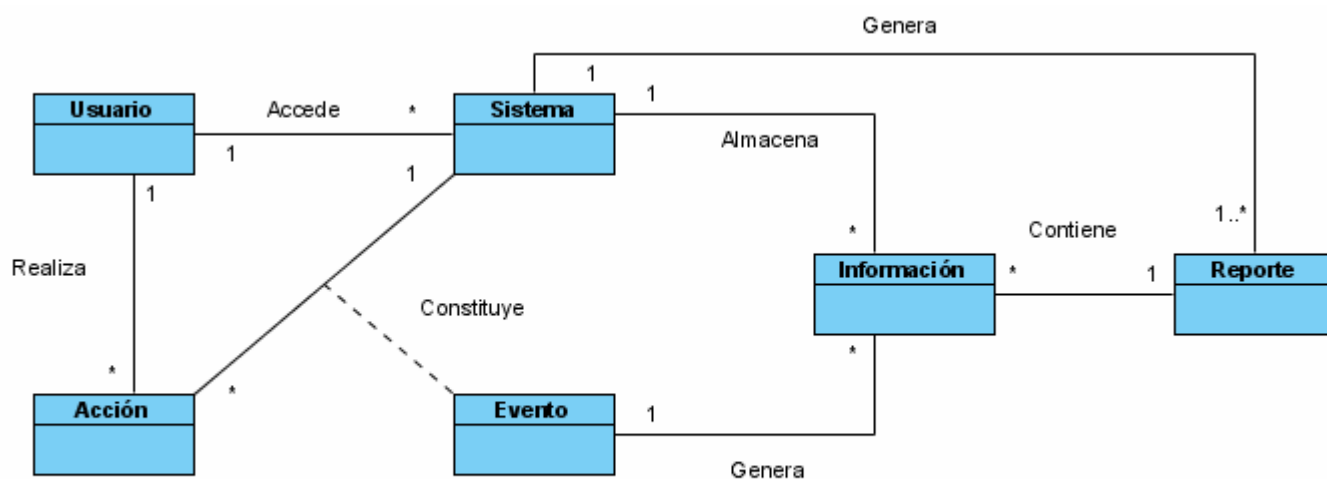


Figura 3.1 Modelo de Dominio

3.4 Descripción de las relaciones entre las entidades del dominio

Una vez representados los conceptos que participaran en el proceso de auditoría: Usuario, Sistema, Información, Acción, Eventos, se describen la relaciones entre ellos.

Los usuarios que propiamente dicho se refiere a las personas que en algún momento accederán a un sistema y que tal sistema almacenará información, una vez que el usuario acceda al sistema realiza uno que otra acción sobre el sistema, donde esta acción constituye un evento y tal evento genera información referente a el trabajo que se hizo sobre la información que contiene el sistema. Hasta aquí queda expuesto el proceso de auditoría en aplicaciones de software, luego, un paso importante dentro del proceso de auditoría es el control de la misma. ¿Cómo se puede controlar la auditoría? La generación de reportes constituye una vía para el control de la auditoría, es decir, un usuario que solicita cerciorarse del trabajo de sus subordinados en el sistema, obtendrá la descripción de las operaciones realizada por cada uno de ellos o por uno en específico, esta descripción es información que fue generada y almacenada en el proceso de auditoría.

3.5 Requerimientos

Uno de los párrafos más citados en la bibliografía de la Ingeniería del Software, dice "La parte más difícil de construir un sistema es precisamente saber qué construir. Ninguna otra parte del trabajo conceptual es tan difícil como establecer los requerimientos técnicos detallados, incluyendo todas las interfaces con personas, máquinas y otros sistemas. Ninguna otra parte del trabajo afecta tanto el sistema si no se realiza correctamente. Ninguna es tan difícil de corregir más adelante. Entonces, la tarea más importante que el Ingeniero de Software hace para el cliente es la extracción iterativa y el refinamiento de los requerimientos del producto" (Bullet 1987).

Existen varios tipos de requerimientos y varias clasificaciones relativas a ellos. Los requerimientos son:

- Las sentencias de necesidades de un usuario que lanzan el desarrollo de un programa o sistema (Flappo and Botta 1995).
- Un estatuto de un servicio o restricción de un sistema (Lavariega and Sommerville Agosto 2005).
- Una necesidad de un usuario o una facilidad necesaria, función o atributo de un sistema que puede ser censado desde una posición externa al sistema (Flappo and Botta 1995).
- Una especificación de lo que debería ser implementado. Son una descripción de cómo el sistema debería comportarse o de una propiedad o atributo del sistema. Puede ser una restricción sobre el proceso de desarrollo del sistema (Lavariega and Sommerville Agosto 2005).

Entre las definiciones que ofrece la IEEE acerca de los requerimientos se pueden destacar las siguientes (Herrera):

- Una condición o capacidad necesaria para un usuario para resolver un problema o alcanzar un objetivo.
- Una condición o capacidad que debe ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, especificación u otro documento formalmente impuesto.
- Una representación documentada de una condición o capacidad dada en los puntos 1 o 2.

Los requerimientos son la base fundamental de cualquier software que se desee desarrollar ya que determinan las capacidades y cualidades que debe cumplir el software para garantizar una buena calidad. Luego de haber analizado las definiciones dichas por varios autores para la investigación que se desarrolla en ese trabajo, se asumirá como definición de requerimientos la planteada por la IEEE.

Una vez entendido qué es un requerimiento es necesario conocer los tipos de requerimientos que existen.

3.5.1 Tipos de Requerimientos

Los requerimientos se pueden dividir en dos grandes grupos como son los Requerimientos Funcionales y los Requerimientos No Funcionales.

Requerimientos Funcionales

- **Requerimientos Funcionales:** Son aquellos que describen las funciones del sistema, todas las actividades o servicios que debe realizar el software y que puede ser comprobada.

Requerimientos No Funcionales

- **Requerimientos no Funcionales:** Son los que limitan al hardware o software bajo el cual el sistema debe operar.(Jacobson, Booch et al. 2004)

Dentro de los **Requerimientos no Funcionales** existen varias categorías como por ejemplo:

- **Requerimientos Físicos:** Son los requerimientos del comportamiento del sistema divididos en requerimientos de: comportamiento, fiabilidad, seguridad, hardware e interfaces de comunicación (López Diciembre 1999).
- **Requerimiento de comportamiento:** Son los que describen todos los aspectos de interfaces entre el software y su medio ambiente (hardware, otro software y humanos) (López Diciembre 1999).
- **Requerimiento de no comportamiento:** Son los atributos de calidad de desarrollo. Incluyen cualquier restricción sobre atributos de construcción estática del sistema, dentro de las cuales están las propiedades de: examinabilidad, mantenibilidad y reusabilidad (López Diciembre 1999).
- **Requerimientos de implantación:** Indican como deberá ser implantado el sistema (Lavariega and Sommerville Agosto 2005).
- **Requerimientos de Rendimiento:** Especifican cuál es el rendimiento mínimo aceptable para el sistema (Lavariega and Sommerville Agosto 2005).
- **Requerimientos de Uso:** Especifican el máximo tiempo aceptable para demostrar el uso del sistema (Lavariega and Sommerville Agosto 2005).
- **Requerimientos de dominio:** Se derivan del dominio del sistema más que de las necesidades específicas de los usuarios. Estándares de organización en el aspecto de los Interfaces de Usuario, con la base de datos (Márquez).

3.6 Requerimientos Funcionales

Como anteriormente se mencionó, los requerimientos funcionales, son capacidades o condiciones que el sistema debe cumplir, por lo que para el desarrollo del sistema que interesa a la entidad se hace un análisis de estas capacidades o condiciones que el mismo debe tener si a continuación se representan.

El sistema debe permitir:

RF1. Configurar Auditoría

- Permitir registrar los eventos auditables de una determinada aplicación.
- Permitir eliminar los eventos auditables de una determinada aplicación.
- Permitir Modificar los eventos auditables de una determinada aplicación.

RF2. Generar reportes

- Permitir generar reporte por un usuario determinado.
- Permitir generar reporte por una fecha determinada.
- Permitir generar reporte por una aplicación determinada.
- Permitir generar reporte por un tipo de evento en específico.

RF3. Exportar datos de forma configurable

- Permitir exportar datos por usuario determinado.
- Permitir exportar datos por fecha determinada.
- Permitir exportar datos por un tipo de evento en específico.
- Permitir exportar datos por una de aplicación determinada.

RF4. Registrar eventos

- ✓ Permitir registrar la auditoría realizada por las aplicaciones de la entidad a través de Servicios Web.

RF5. Gestionar Servicios de Reporte

- Permitir generar reporte por un usuario determinado a través de Servicios Web.
- Permitir generar reporte por una fecha determinada a través de Servicios Web.
- Permitir generar reporte por una aplicación determinada a través de Servicios Web.

- Permitir generar reporte por un tipo de evento en específico a través de Servicios Web.

RF6. Validar Usuario

3.7 Requisitos 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 o confiable.

Apariencia o interfaz externa:

- Diseño sencillo, con pocas entradas, permitiendo que no sea necesario mucho entrenamiento para utilizar el sistema.
- Empleo de los colores: verde, gris, blanco y azul principalmente, que son los definidos en los estándares del proyecto.

Usabilidad:

- El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora.
- El software tendrá siempre la posibilidad de ayuda disponible para cualquier tipo de usuario, lo que le permitirá un avance considerable en la explotación de la aplicación en todas sus funcionalidades.
- Se emplearán barras de progreso para indicar el estado de los procesos que por su complejidad requieran de un tiempo de procesamiento apreciable por los usuarios.

Rendimiento:

- Tiempos de respuestas rápidos al igual que la velocidad de procesamiento de la información, no mayor a los 5 segundos en las actualizaciones y no mayor de 10 para las recuperaciones.

Soporte:

Se requiere un servidor de bases de datos con las siguientes características:

- Soporte para grandes volúmenes de datos y velocidad de procesamiento.
- Tiempo de respuesta rápido en accesos concurrentes.

Portabilidad:

- Necesidad de que el sistema sea multiplataforma.

Seguridad:

- Garantizar que las funcionalidades del sistema se muestren de acuerdo al nivel de usuario que este activo.
- Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.
- Verificación sobre acciones irreversibles (eliminaciones).

Confiabilidad:

- La herramienta de implementación a utilizar tiene soporte para recuperación ante fallos y errores.

Funcionalidad:

- Mínima cantidad de páginas para ejecutar todas las funciones posibles (preferentemente que estén relacionadas).

Implantación

- Entregar toda la documentación asociada al proyecto.
- Organizar el adiestramiento de los usuarios.

Software:

- Apache 2.0 o superior como servidor Web, con módulo PHP 5 disponible y debe estar configurado con la extensión pgsqI incluida.
- PostgreSQL como Sistema Gestor de Base de Datos.
- Y la máquina cliente del usuario debe tener como requerimiento mínimo:
 - ✓ El navegador Mozilla FireFox.
 - ✓ Sistema operativo Windows Advanced Server 2000 o superior; o Linux NOVA.

Hardware

- 128 Mb. de RAM o superior.
- 40 Gb. de disco duro o superior.

- Una computadora que sirva de cliente:
- Pentium a 200 MHz. de velocidad de procesamiento o superior.
- Tarjeta de red.

Políticos culturales

- El sistema solo podrá ser usado en territorio cubano y por las entidades del MINFAR.
- Sólo debe contener palabras en idioma español.
- Debe respetar los términos usados en la especialidad.

Legales

El sistema debe ajustarse y regirse por la ley, decretos leyes, decretos, resoluciones y manuales (órdenes) establecidos, que norman los procesos que serán automatizados:

- Ley No. 75/94 de la Defensa Nacional.
- Decreto-Ley 224 del Servicio Militar, amparado en el capítulo 7 de la ley 75.
- Resolución 46/97 del ministro de las FAR sobre la organización del registro militar de los trabajadores y estudiantes.
- Resolución 47/97 del ministro de las FAR sobre la organización, preparación y aseguramiento de las formaciones especiales.
- Resolución conjunta 1/97 del ministro de trabajo y de finanzas y precio sobre el aseguramiento salarial y financiero al personal movilizado.
- Orden 336 del viceministro Jefe EMG (Estado Mayor General) Manual para el trabajo de los responsables de áreas de atención.
- Orden 337, Manual de personal de las unidades y entidades de las FAR.
- Orden 338 Manual para el trabajo en los Comités Militares.

Aplicación de estándares

- Se utilizarán los estándares de codificación, estándares de diseño para la base de datos y mecanismos de diseño definidos por la entidad (UCID).

3.8 Actores del sistema a automatizar

Nombre del actor	Descripción
Administrador	Es el administrador del sistema, que maneja toda la información de configuración de los sistemas y usuarios.
Sistema Externo	Representa los sistemas externos que interactúan con el sistema de auditoría.

Tabla 3.2 Actores de sistema

3.9 Diagrama de paquetes



Figura 3.2 Diagrama de paquetes

3.10 Diagrama de Casos de uso del sistema a automatizar

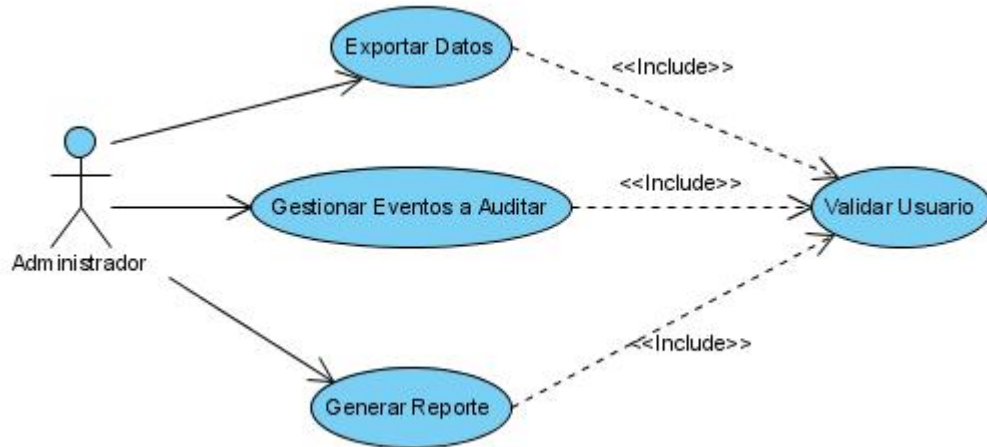


Figura 3.3 Diagrama de casos de uso del sistema: Paquete *Administración del Sistemas*

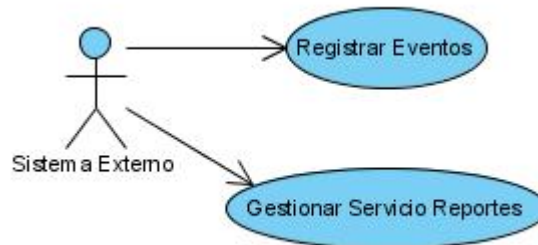


Figura 1.4. Diagrama de casos de uso del sistema: Paquete *Servicios*

3.11 Descripción de los casos de uso

Nombre del caso de uso	Gestionar Eventos a Auditar
Actores	Administrador
Resumen	El caso de uso inicia cuando se desea insertar, modificar o eliminar un evento auditable, el administrador procesa toda la información acerca del evento auditable. El caso de uso finaliza cuando se ha almacenado toda la información acerca del evento.
Precondiciones	El Administrador debe estar autenticado, caso de uso: Autenticar.
Poscondiciones	Se actualiza toda la información referente a los eventos auditables en la BD.

Tabla 3.3 Breve descripción del CU Gestionar Eventos a Auditar

Nombre del caso de uso	Exportar Datos
Actores	Administrador
Resumen	El caso de uso inicia cuando se desea guardar en otro lugar por alguna razón información referente a la auditoría realizada en diferentes aplicaciones de la entidad, el administrador selecciona la información a exportar además seleccionar el destino donde se almacenará dicha información. El caso de uso finaliza cuando la información es almacenada en otro fichero.
Precondiciones	El Administrador debe estar autenticado, caso de uso: Autenticar.
Poscondiciones	Se actualiza la BD con la información restante después del proceso Exportar Datos y se crean nuevos ficheros de Datos.

Tabla 3.4 Breve descripción del CU Exportar Datos

Nombre del caso de uso	Generar Reporte
Actores	Administrador
Resumen	El caso de uso inicia cuando se hace la solicitud de un reporte de auditoría de forma general para una o varias aplicaciones y según un

	<p>criterio en específico, de acuerdo con la petición del administrador el sistema hace una consulta en la BD y devuelve el reporte solicitado. Termina el caso de uso con el envío del documento.</p>
Precondiciones	El Administrador debe estar autenticado, caso de uso: Autenticar.
Poscondiciones	Se muestra el reporte de auditoría solicitado.

Tabla 3.5 Breve descripción del CU Generar Reporte

Nombre del caso de uso	Registrar Evento
Actores	Sistema externo
Resumen	El caso de uso inicia cuando el generador de sucesos de un sistema externo solicita verificar si un evento esta configurado para ser auditado, en caso de que encuentre respuesta positiva envía el evento con los datos del usuario, fecha y hora en que ocurrió, el sistema guarda estos datos. El caso de uso termina cuando se actualiza la BD con el nuevo evento.
Precondiciones	-
Poscondiciones	Se actualiza la BD con el nuevo evento.

Tabla 3.6 Breve descripción del CU Registrar Evento

Nombre del caso de uso	Gestionar Servicio de Reporte
Actores	Sistema externo
Resumen	El caso de uso inicia cuando de un sistema externo se hace la solicitud de un reporte de auditoría, según la petición del actor el sistema hace una consulta en la BD y devuelve el reporte solicitado. Termina el caso de uso al enviarse el reporte por servicio.
Precondiciones	-
Poscondiciones	Se envía el reporte de auditoría solicitado.

Tabla 3.7 Breve descripción del CU Gestionar Servicio de Reporte

Nombre del caso de uso	Validar Usuario
Actores	Administrador
Resumen	El caso de uso inicia cuando el administrador desea realizar alguna operación en el sistema, internamente se verifica que este usuario tiene los privilegios para realizar tal operación. Termina el caso de uso cuando es autorizada o cancelada la operación.
Precondiciones	
Poscondiciones	Se autoriza o cancel la operación.

Tabla 3.8 Breve descripción del CU Validar Usuario

3.12 Descripción detallada de los Casos de Usos

Caso de uso:	Gestionar Eventos a Auditar
Actores:	Administrador
Propósito:	Almacenar información acerca de los eventos que pueden ocurrir en una aplicación.
Resumen:	El caso de uso inicia cuando se desea insertar, modificar o eliminar un evento auditable, el administrador procesa toda la información acerca del evento auditable. El caso de uso finaliza cuando se ha almacenado toda la información acerca del evento.
Precondiciones:	El Administrador debe estar validado, caso de uso: Validar Usuario.
Referencias:	R1
Poscondiciones:	Se actualiza toda la información referente a los eventos auditables en la BD.
Requerimientos Especiales:	-
Interfaces:	

Pantalla Principal: SistCA

Documento sin título - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://10.12.168.222:5800/auditoria%201/plantillas/

Hotmail gratuito Personalizar vínculos Windows Media Windows Ext 2.0 Samples SOAP (Simple Object ...)

SISCA

Sistema de Control de Auditoría

Gestionar Eventos Exportar Datos Generar Reportes

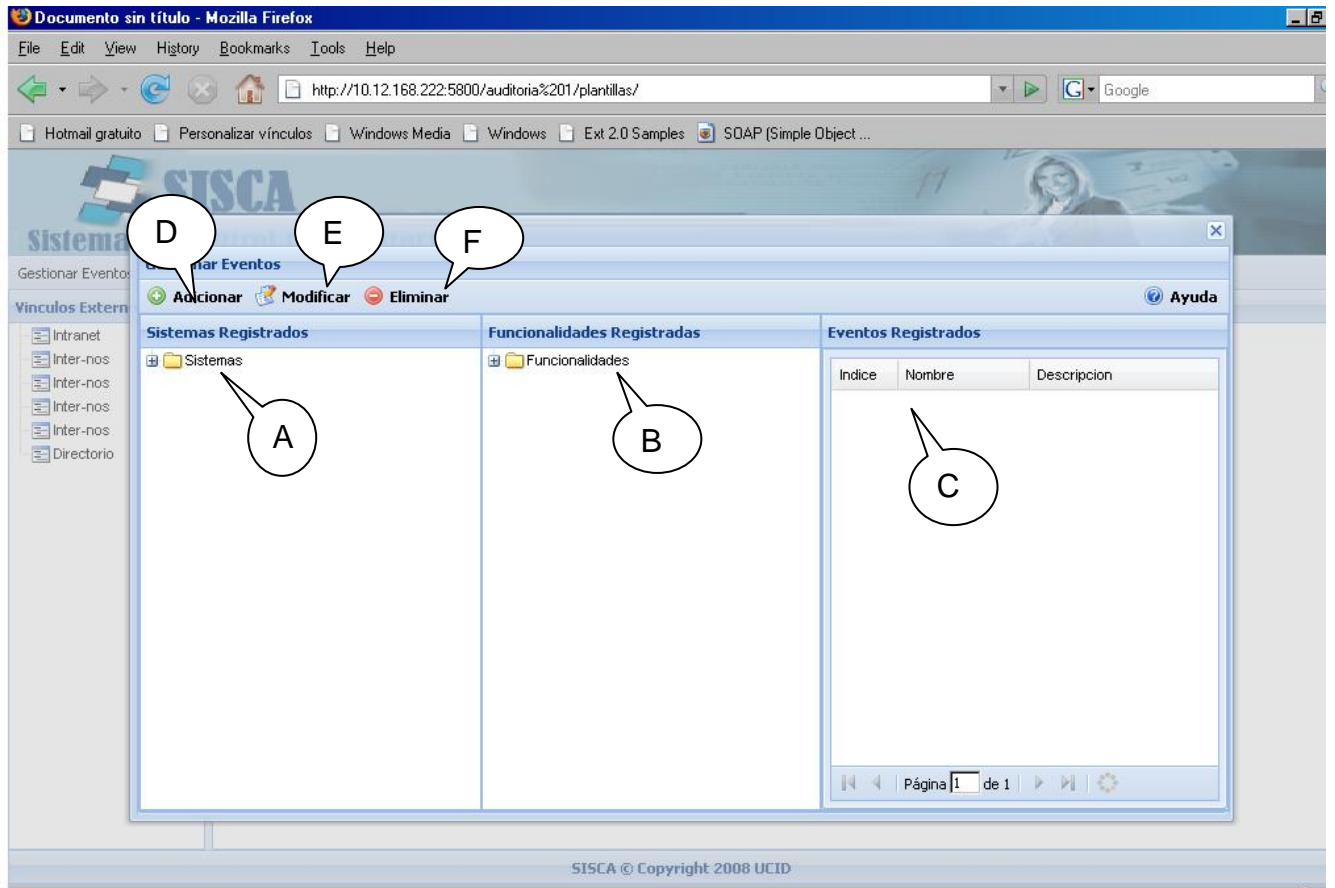
..Bienvenidos al Sistema de Control de Auditoría...

Vinculos Externos

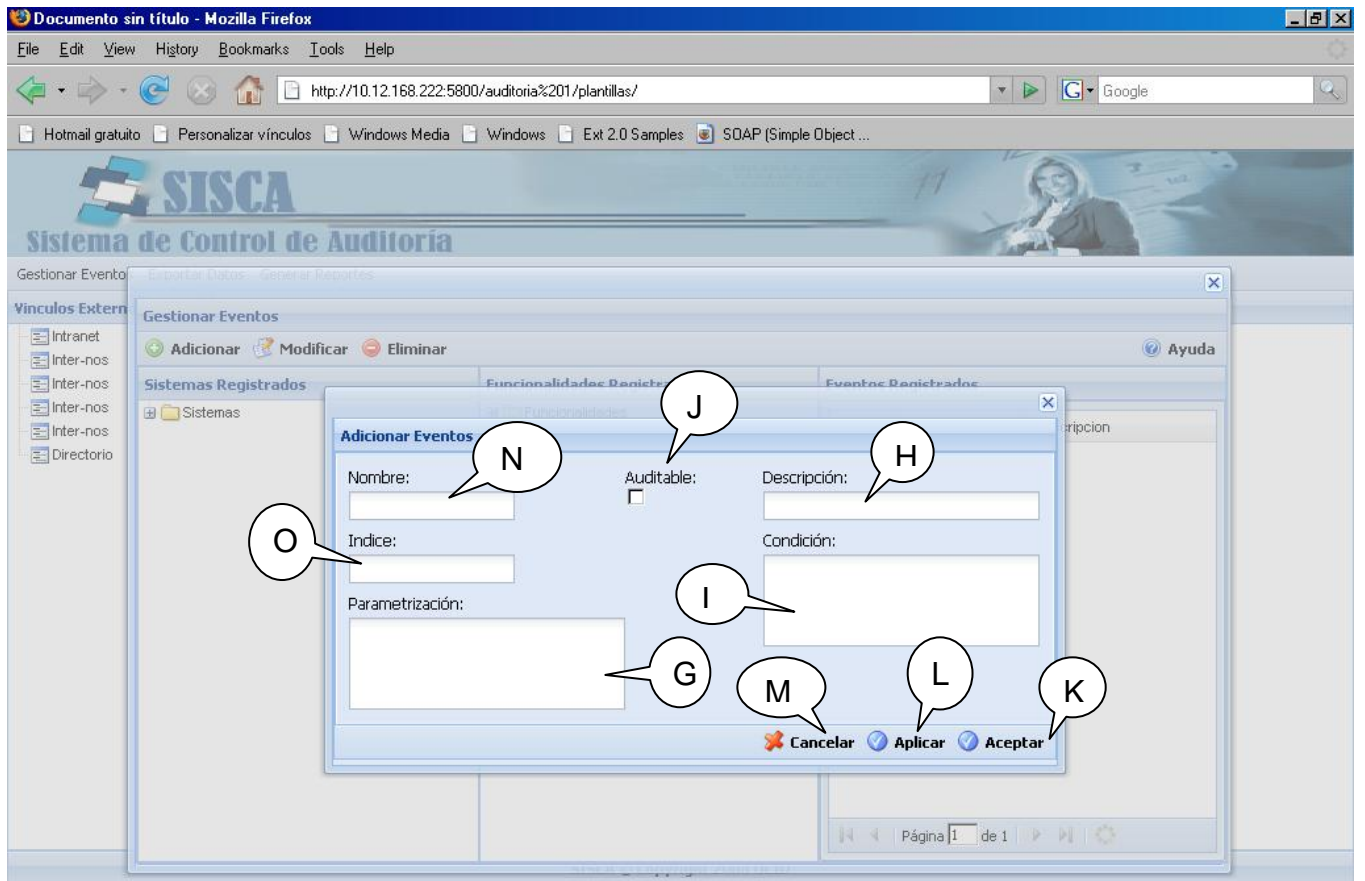
- yanet
- er-nos
- Inter-nos
- Inter-nos
- Inter-nos
- Directorio

SISCA © Copyright 2008 UCID

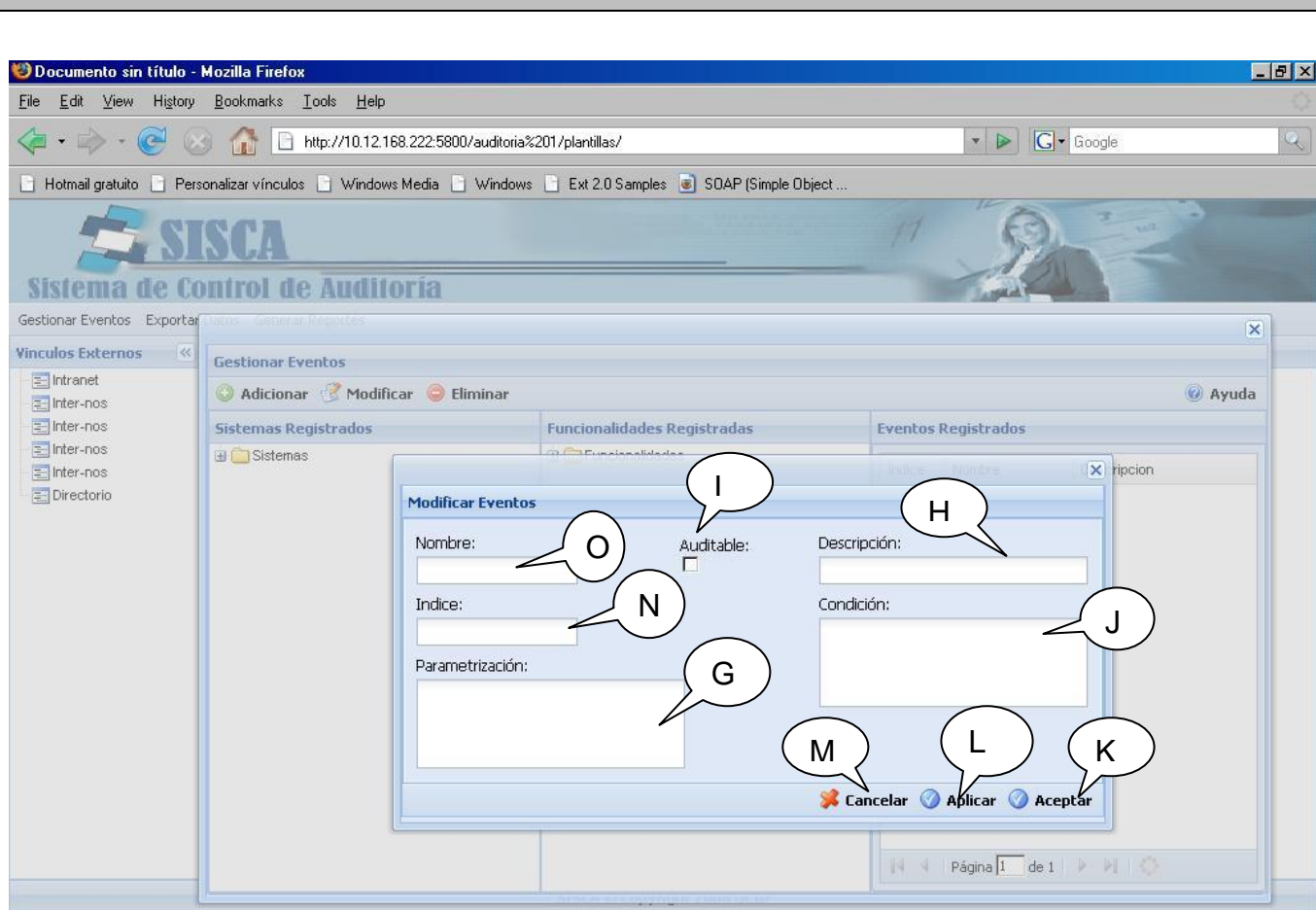
Pantalla Gestionar Eventos



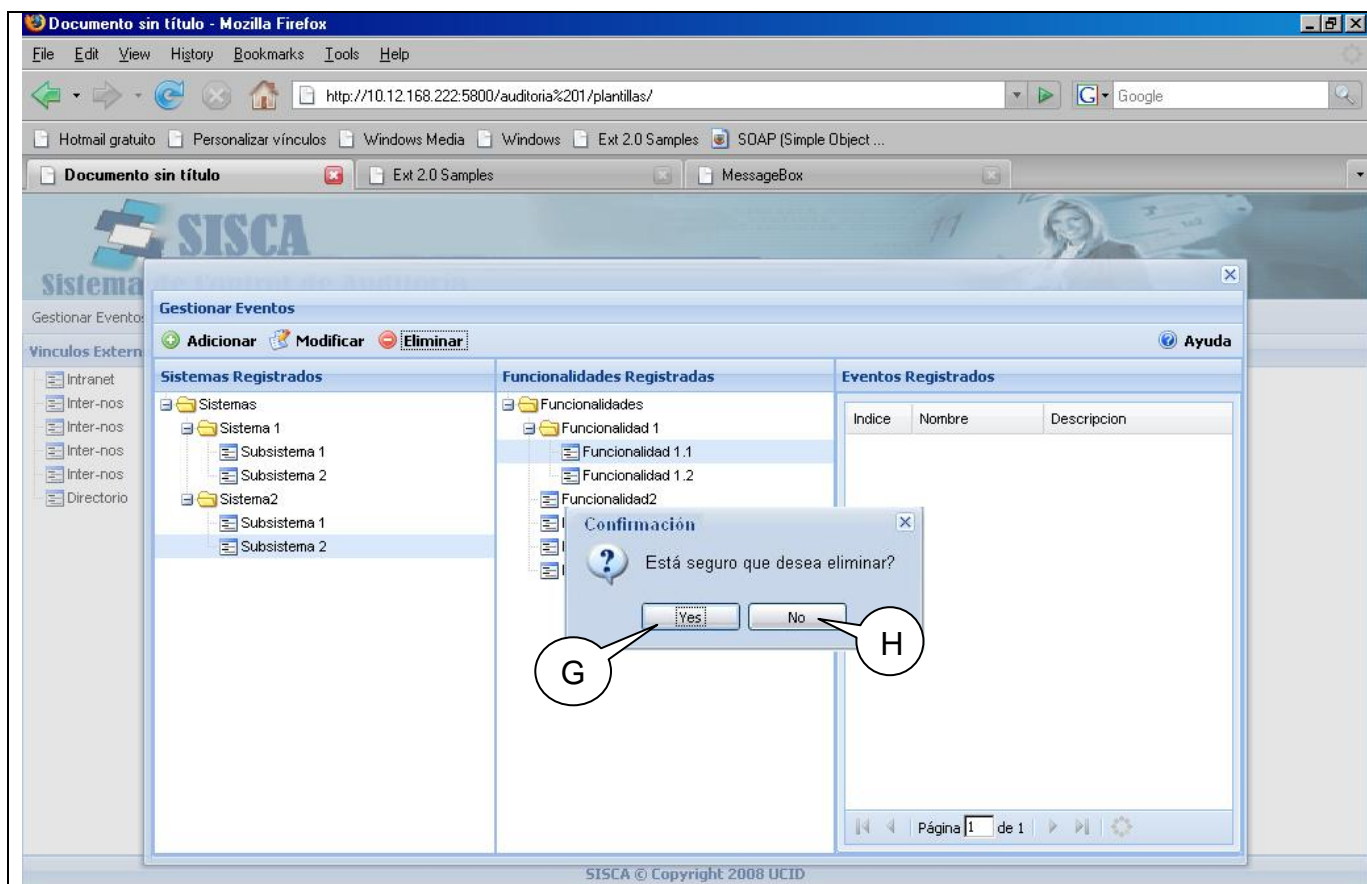
Pantalla Adicionar Eventos



Pantalla Modificar Eventos



Pantalla Eliminar Evento



Curso normal de eventos para el caso de uso:

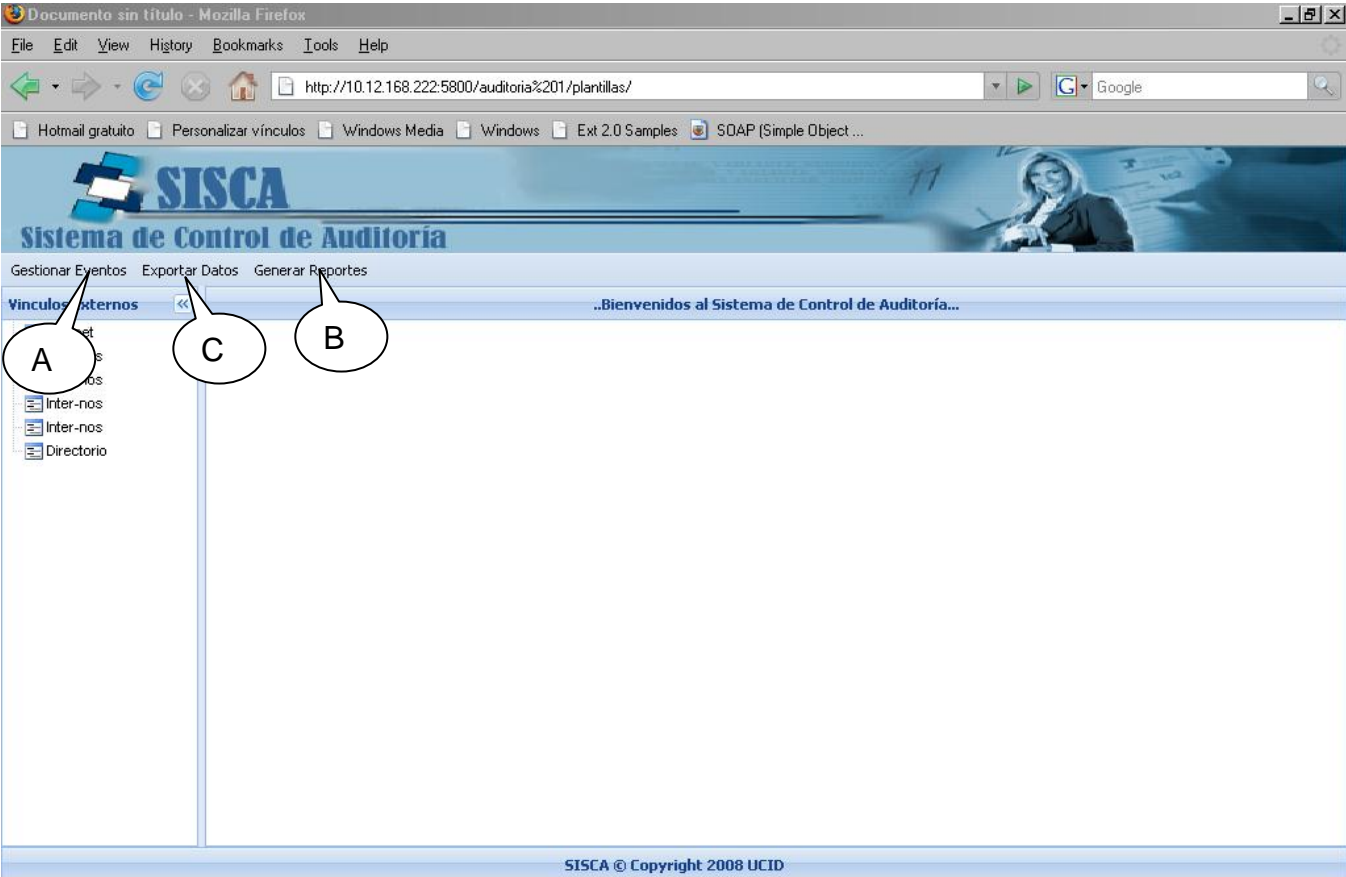
Acción del actor	Respuesta del sistema
<p>1- El actor desea gestionar un Evento.</p> <p>3- Selecciona la opción Gestionar Eventos (A).</p>	<p>2- Muestra la pantalla Principal SISCA.</p> <p>4- Muestra la pantalla Gestionar Eventos.</p> <p>Si el usuario selecciona:</p> <ul style="list-style-type: none"> • Adicionar Evento, ver sección “Adicionar Evento”. • Modificar Evento, ver sección “Modificar Evento”. • Eliminar Evento ver sección “Eliminar Evento”.
Sección: Adicionar Evento	
Acción de Actor	Respuesta del Sistema
<p>1- En la pantalla Gestionar Eventos selecciona</p>	<p>2- Muestra los eventos (C) en caso que existan, los</p>

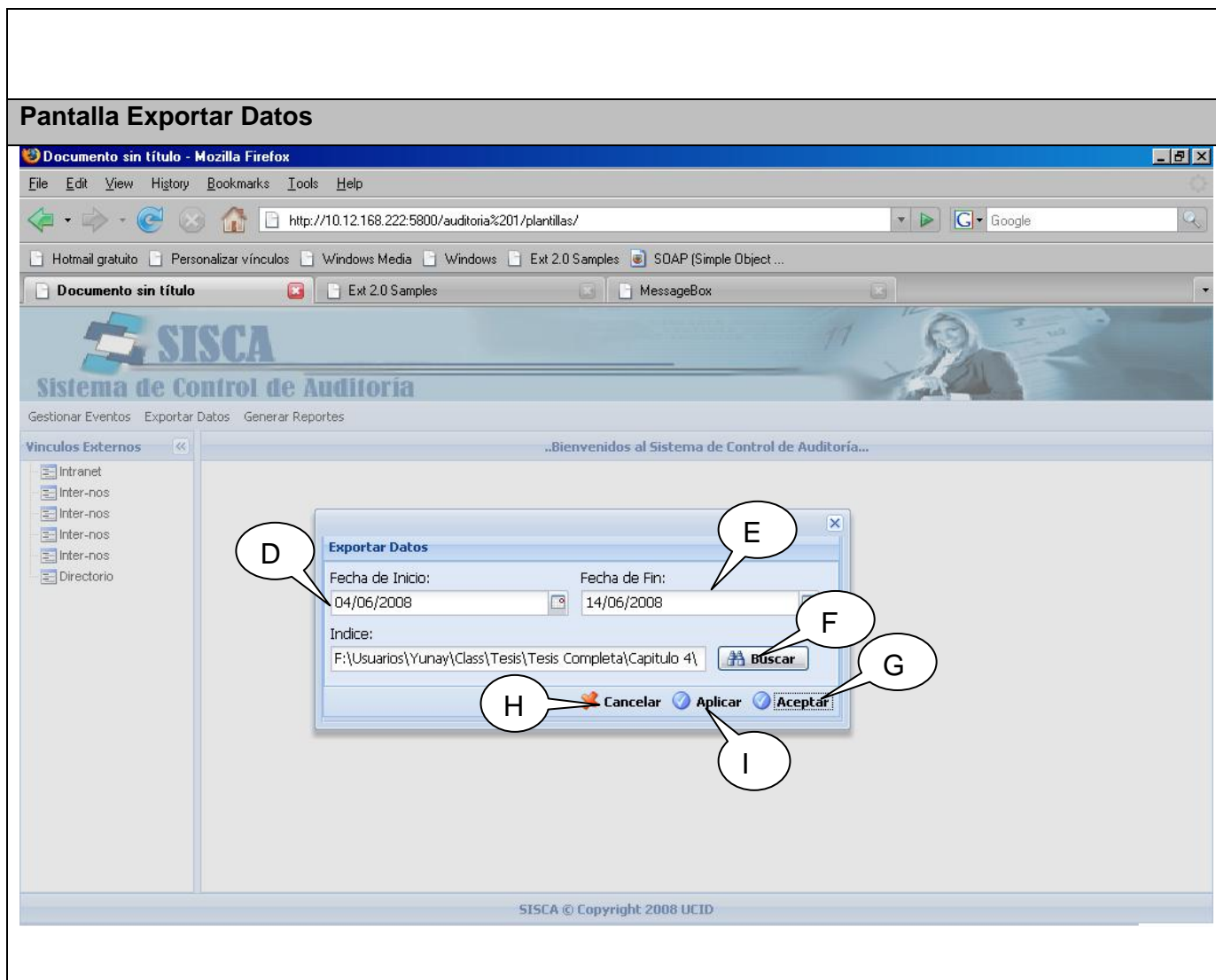
<p>el sistema y funcionalidad (A, B) al que pertenecerá el nuevo evento que desea registrar.</p> <p>3- Verifica que el evento que quiere insertar no está registrado, luego presiona la pestaña Adicionar (D).</p> <p>5- Introduce los datos correspondientes a un evento: Nombre, Índice, Parametrización, Descripción, Condiciones (N, O, G, H, I) y especifica si es auditable o no (J). Luego presiona el botón Aceptar (K).</p>	<p>que pertenecen al sistema y funcionalidad previamente seleccionados.</p> <p>4- Muestra la pantalla Adicionar Evento.</p> <p>6- Procesa los datos que introdujo el actor.</p> <p>7- Muestra el nuevo evento en el grid (C) en la pantalla Gestionar Eventos.</p>
Cursos alternos de la sección:	
<p>5- Si el actor desea registrar mas de un evento puede presionar el botón aplicar, donde el sistema mantiene la misma interfaz Registrar usuario con los campos listos para introducir nuevos datos para un nuevo evento que pertenecerá a el sistema y funcionalidad previamente seleccionada.</p> <p>8- Si Ocurre un Error en los procesos de inserción de eventos el sistema muestra un mensaje de error, y brinda la posibilidad de corregir datos.</p>	
Sección: Modificar Evento	
Acción de Actor	Respuesta del Sistema
<p>1- En la pantalla Gestionar Eventos selecciona el sistema y funcionalidad (A, B) a los cuales pertenece el evento que desea modificar.</p> <p>3- Selecciona el evento que desea modificar. Luego presiona la pestaña Modificar Evento (E).</p> <p>5- Elimina la información actual que desea modificar ya sea el Nombre, Id Evento, Parametrización, Descripción, Condiciones (N, O, G, H, I) y especifica si es auditable o no (J) y luego introduce los nuevos datos según la necesidad. Presiona el botón Aceptar (K).</p>	<p>2- Muestra los eventos (C) que pertenecen al sistema y funcionalidad previamente seleccionados (A, B).</p> <p>4- Muestra la pantalla Modificar Evento con la información actual de los datos.</p> <p>6- Procesa los datos que introdujo el actor.</p> <p>7- Muestra el evento (C) con las nuevas características en la pantalla Gestionar Eventos a Auditar.</p>
Cursos alternos de la sección:	
<p>8- Si Ocurre un Error en los procesos de modificación de eventos el sistema muestra un mensaje de error, y brinda la posibilidad de corregir datos.</p>	

Sección: Eliminar Evento	
Acción de Actor	Respuesta del Sistema
<p>1- En la pantalla Gestionar Eventos selecciona el sistema y funcionalidad (A, B) a los cuales pertenece el evento que desea eliminar.</p> <p>3- Selecciona el evento que desea eliminar. Luego presiona la pestaña Eliminar Evento (F).</p> <p>5- Presiona el botón Si (G).</p>	<p>2- Muestra los eventos (C) que pertenecen al sistema y funcionalidad previamente seleccionados (A, B).</p> <p>4- Muestra un mensaje preguntando al actor si esta seguro que desea eliminar dicho evento (E).</p> <p>6- Realiza el proceso de eliminación del evento seleccionado. Actualiza la lista de los eventos (C) correspondientes a la funcionalidad seleccionada.</p>
Cursos alternos de la sección:	
<p>5- Si el usuario presiona el botón Cancelar (H), se cancela el proceso de eliminación del evento.</p> <p>7- Si Ocurre un Error en los procesos de eliminación de eventos el sistema muestra un mensaje de error, y brinda la posibilidad de corregir datos.</p>	

Tabla 3.9 Descripción detallada del CU Gestionar Eventos a Auditar

Caso de uso:	Exportar Datos
Actores:	Administrador
Propósito: Almacenar en otro fichero información que lleva un período de tiempo en la BD.	
Resumen: El caso de uso inicia cuando se desea guardar en otro lugar por alguna razón información referente a la auditoría realizada en diferentes aplicaciones de la entidad, el administrador selecciona la información a exportar además seleccionar el destino donde se almacenará dicha información. El caso de uso finaliza cuando la información es almacenada en otro fichero.	
Precondiciones:	El Administrador debe estar validado, caso de uso: Validar Usuario.
Referencias:	R3
Poscondiciones:	Se actualiza la BD con la información restante después del proceso Exportar Datos y se crean nuevos ficheros de Datos.

Requerimientos Especiales:	-
Interfaces:	
Pantalla Principal: SistCA	
	



Curso normal de eventos para el caso de uso:

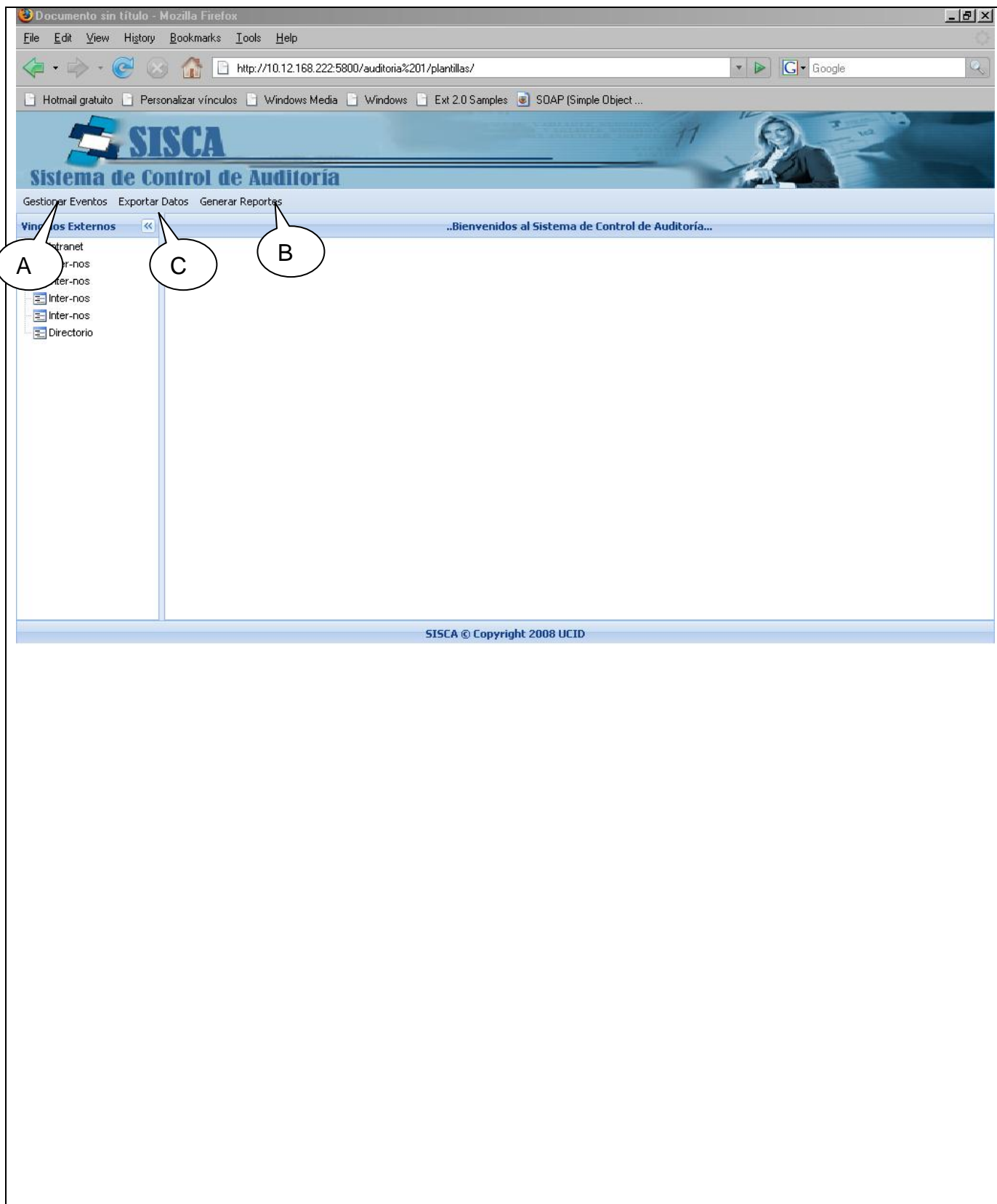
Acción del actor	Respuesta del sistema
<p>1- El actor desea exportar datos.</p> <p>3-Selecciona la opción Exportar Datos (C).</p> <p>5- Selecciona la fecha de inicio y fecha fin (D, E) en que fueron registrados dichos datos.</p> <p>6- Selecciona el índice o destino (F) donde se almacenaran los datos. Presiona el botón Aceptar (G).</p>	<p>2- Muestra la pantalla Principal SISCA.</p> <p>4- Muestra la pantalla Exportar Datos.</p> <p>7- Transfiere los datos seleccionados al destino escogido almacenándolos en un fichero XML.</p>

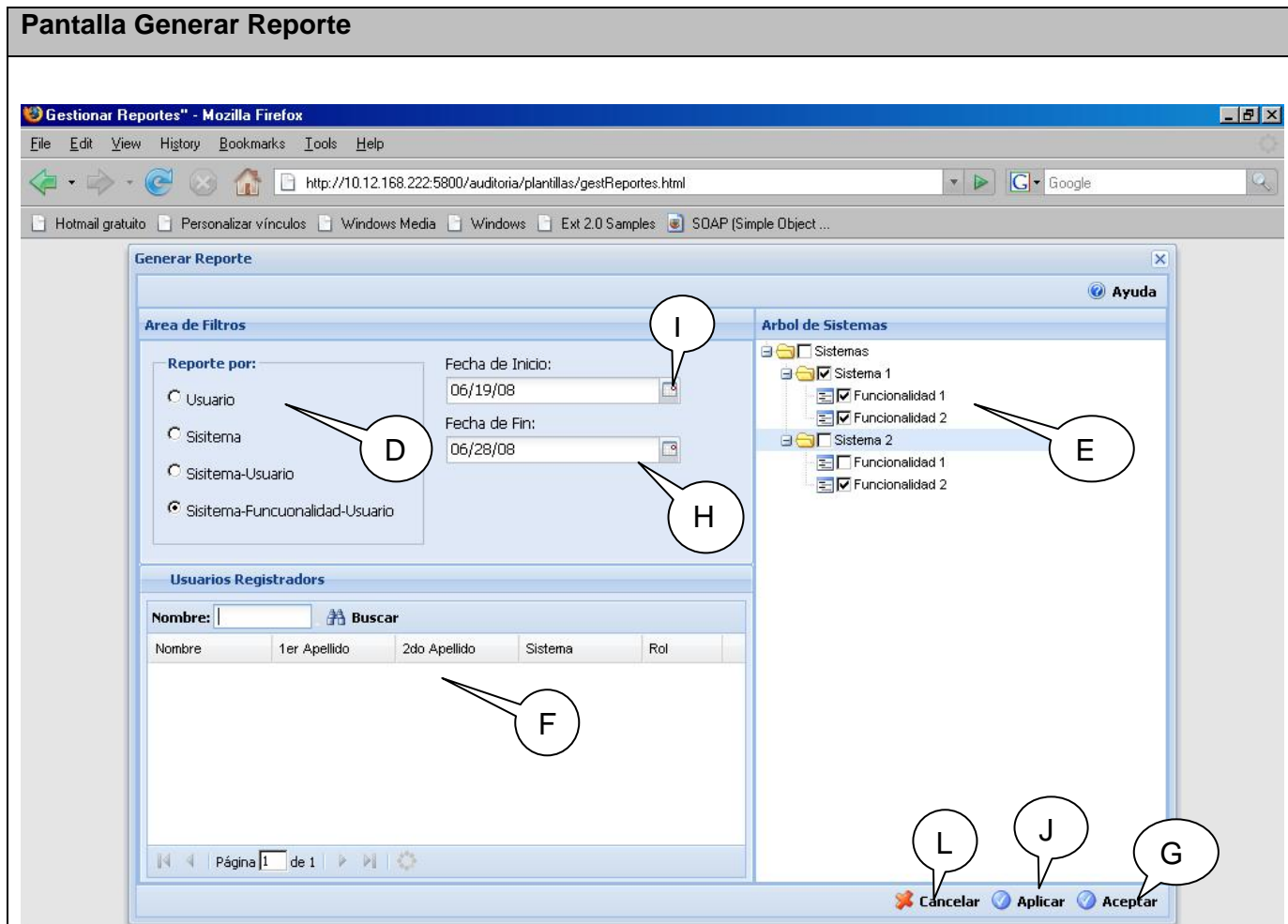
Cursos alternos:

5- Si Ocurre un Error en las fechas seleccionadas el sistema muestra mensaje de error, y brinda la posibilidad de corregir datos.

Tabla 3.10 Descripción detallada del CU Exportar Datos

Caso de uso:	Generar Reporte
Actores:	Administrador
Propósito: Brindar información en forma de reporte acerca de las acciones que realiza uno o varios usuarios sobre una o varias aplicaciones.	
Resumen: El caso de uso inicia cuando se hace la solicitud de un reporte de auditoría de forma general para una o varias aplicaciones y según un criterio en específico, de acuerdo con la petición del administrador el sistema hace una consulta en la BD y devuelve el reporte solicitado. El caso de uso termina cuando se muestra el documento de reporte.	
Precondiciones:	El Administrador debe estar validado, caso de uso: Validar Usuario.
Referencias:	R2
Poscondiciones:	El usuario obtiene un log de auditoría con todos los parámetros que solicitó al inicio del caso de uso.
Requerimientos Especiales:	-
Interfaces:	
Pantalla Principal: SISCA	





Curso normal de eventos para el caso de uso:	
Acción del actor	Respuesta del sistema
<p>1- Desea obtener un Reporte</p> <p>3- Selecciona la opción Generar Reporte (C).</p> <p>5- Selecciona que combinación de reporte desea.</p>	<p>2- Muestra la pantalla principal SISCA.</p> <p>4- Muestra la pantalla Generar Reporte.</p> <p>6- Si el actor selecciona:</p> <ul style="list-style-type: none"> • “Reporte por Usuario” ver sección Reporte por Usuario. • “Reporte por Sistema” ver sección Reporte por Sistema. • “Reporte por Sistema - Usuario” ver sección Reporte por Sistema - Usuario. • “Reporte por Sistema - Funcionalidad - Usuario” ver sección Reporte por Sistema - Funcionalidad

	- Usuario.
Sección: Reporte por Usuario	
<p>7- Solicita un reporte por usuario (D).</p> <p>9- Selecciona en el campo Usuario (F) los usuarios de los cuales le interesa el reporte y presiona el botón Aceptar (G).</p>	<p>8- Activa el campo con todos los usuarios existentes en la entidad (F).</p> <p>10- El sistema realiza las consultas correspondientes a la selección y elabora un documento con el reporte solicitado.</p>
Cursos alternos de la Sección Reporte por Usuario	
<p>9 - El actor puede seleccionar, si lo desea, una fecha de inicio (I) y una fecha fin (H) donde ocurrieron los eventos.</p>	
Sección: Reporte por Sistema.	
<p>7- Solicita un reporte por sistema (D).</p> <p>9- Selecciona en el campo Sistema - Funcionalidades (E) los sistemas de los cuales le interesa el reporte y presiona el botón Aceptar (G).</p>	<p>8- Activa el campo con todos los sistemas existentes en la entidad (E).</p> <p>10- El sistema realiza las consultas correspondientes a la selección y elabora un documento con el reporte solicitado.</p>
Cursos alternos de la Sección Reporte por Sistema	
<p>9 - El actor puede seleccionar, si lo desea, una fecha de inicio (I) y una fecha fin (H) donde ocurrieron los eventos</p>	
Sección: Reporte por Sistema - Usuario.	
<p>7- Solicita un reporte por sistema - usuario (D).</p> <p>9- Selecciona en el campo Sistema - Funcionalidades (E) el o los sistemas de los cuales le interesa el reporte.</p> <p>11- Selecciona en el campo Usuarios (F) los usuarios de los cuales le interesa el reporte y presiona el botón Aceptar (G).</p>	<p>8- Activa el campo Sistema - Funcionalidades (E) con todos los sistemas existentes en la entidad.</p> <p>10- Dinámicamente muestra en el campo Usuarios (F) con los usuarios que tienen privilegios de acceso al o los sistemas seleccionados en el campo Sistema - Funcionalidades (E).</p> <p>12- El sistema realiza las consultas correspondientes a la selección y elabora un documento con el reporte solicitado.</p>
Cursos alternos de la Sección Reporte por Sistema - Usuario	
<p>9 - El actor puede seleccionar, si lo desea, una fecha de inicio (I) y una fecha fin (H) donde ocurrieron los eventos</p>	

Sección: Reporte por Sistema - Funcionalidad - Usuario.	
<p>7- Solicita un reporte por sistema - funcionalidad - usuario (D).</p> <p>9- Despliega en el campo Sistema - Funcionalidades (E) el o los sistema de los cuales le interesa el reporte y selecciona la o las funcionalidades que están registradas dentro de los sistemas antes escogidos.</p> <p>11- Selecciona los usuarios de los cuales le interesa el reporte (F) y presiona el botón Aceptar (G).</p>	<p>8- Activa el campo Sistema - Funcionalidades (E) con todos los sistemas existentes en la entidad.</p> <p>10- Dinámicamente muestra en el campo Usuarios (F) con los usuarios que tienen privilegios de acceso al sistema y funcionalidades seleccionadas en el campo Sistema – Funcionalidades. (E).</p> <p>12- El sistema realiza las consultas correspondientes a la selección y elabora un documento con el reporte solicitado.</p>
Cursos alternos de la Sección Reporte por Sistema - Funcionalidad - Usuario.	
<p>9 - El actor puede seleccionar, si lo desea, una fecha de inicio (I) y una fecha fin (H) donde ocurrieron los eventos</p>	

Tabla 3.11 Descripción detallada del CU Generar Reporte

Caso de uso:	Registrar Evento
Actores:	Sistema externo
Propósito: Centralizar la información de los eventos ocurridos en las aplicaciones de la entidad	
Resumen: El caso de uso inicia cuando el generador de sucesos de un sistema externo solicita verificar si un evento esta configurado para ser auditado, en caso de que encuentre respuesta positiva envía el evento con los datos del usuario, fecha y hora en que ocurrió, el sistema guarda estos datos. El caso de uso termina cuando se actualiza la BD con el nuevo evento.	
Precondiciones:	-
Referencias:	R4
Poscondiciones:	Se actualiza la BD con el nuevo evento.
Curso normal de eventos para el caso de uso:	
Acción del actor	Respuesta del sistema
<p>1- Envía los eventos registrados en su sistema de auditoría.</p>	<p>2- Actualiza la BD con los nuevos eventos registrados.</p> <p>3- Envía un mensaje confirmando que la acción tuvo éxito.</p>

Tabla 3.12 Descripción detallada del CU Registrar Evento

Caso de uso:	Gestionar Servicio de Reporte
Actores:	Sistema externo
Propósito:	Devolver respuesta a la petición de un usuario que desea visualizar las tareas que ha realizado durante un tiempo determinado en una aplicación determinada.
Resumen:	El caso de uso inicia cuando de un sistema externo se hace la solicitud de un reporte de auditoría, según la petición del actor el sistema hace una consulta en la BD y devuelve el reporte solicitado. Termina el caso de uso al enviarse el reporte por servicio.
Precondiciones:	-
Referencias:	R5
Poscondiciones:	Se envía el reporte de auditoría solicitado.
Curso normal de eventos para el caso de uso:	
Acción del actor	Respuesta del sistema
1- Solicita ver un reporte de lo que ha realizado en una fecha y aplicación determinada.	2- Hace una búsqueda según la petición del usuario. 3- Elabora el reporte y por servicio web es enviado al usuario final.

Tabla 3.13 Descripción detallada del CU Gestionar Servicio de Reporte

Caso de uso:	Validar Usuario
Actores:	Administrador
Propósito: Verificar que el usuario tiene privilegios suficientes para operar en el sistema.	
Resumen: El caso de uso inicia cuando el administrador desea realizar alguna operación en el sistema, internamente se verifica que este usuario tiene los privilegios para realizar tal operación. Termina el caso de uso cuando es autorizada o cancelada la operación.	
Precondiciones:	-
Referencias:	R6
Poscondiciones:	Se autoriza o cancel la operación.
Curso normal de eventos para el caso de uso:	
Acción del actor	Respuesta del sistema
1- El actor desea realizar una operación en el sistema.	2- Verifica mediante un servicio web que el usuario tiene los privilegios para trabajar en con dicha funcionalidad del sistema. 3- Autoriza al usuario a realizar la operación que desea.
Cursos alternos:	
3- Es denegada la operación que desea realizar el usuario.	

Tabla 3.14 Descripción detallada del CU Validar Usuario

3.13 Conclusiones

En el capítulo quedan señaladas las características que garantizan un buen despliegue, una buena seguridad y una excelente aceptación del sistema por el cliente final. También se reflejan las funcionalidades que debe realizar el sistema además de una detallada descripción de la interacción entre el usuario final y la aplicación. En presencia de estos elementos se continuará en el próximo capítulo con la propuesta de de solución al problema que se plantea, es decir, el análisis y diseño de cada funcionalidad que se representan en este capítulo.

Capítulo 4 Diseño de la solución propuesta

4.1 Introducción

En este capítulo se realizará una descripción del diseño que se propone para el problema existente. Se abundará acerca de la arquitectura sobre la cual se construirá el sistema. Luego, continúa con los diagramas de las clases de diseño, la confección del modelo de datos, el modelo de componentes y el diagrama de despliegue.

4.2 Modelo del análisis

En la construcción del modelo de análisis se tienen que identificar las clases que describen la realización de los casos de uso, los atributos y las relaciones entre ellas. Con esta información se construye el Diagrama de clases del análisis, que por lo general se descompone para agrupar las clases en paquetes. Esta descomposición tiene impacto por lo general en el diseño e implementación de la solución. Para ver dichos diagramas remítase al **Anexo 1** Diagramas de clases del análisis.

Teniendo en las manos los diagramas de clases de análisis, se continúa con otra fase del flujo de trabajo, y se trata de la realización de los casos de uso, esto no es más que la descripción de como se lleva a cabo y se ejecuta un CU en termino de clases de análisis y sus objetos.

Si bien los diagramas de clases representan las relaciones de manera estática, existen otros diagramas que representan las relaciones dinámicas entre los objetos (instancias, mensajes, llamadas a métodos) y son los llamados diagramas de interacción donde RUP propone que para cada caso de uso se construya un diagrama de clases que muestre las clases del análisis participantes y uno de los dos tipos de diagramas de interacción (Colaboración o Secuencia).

4.2.1 Diagramas de Colaboración de clases del análisis

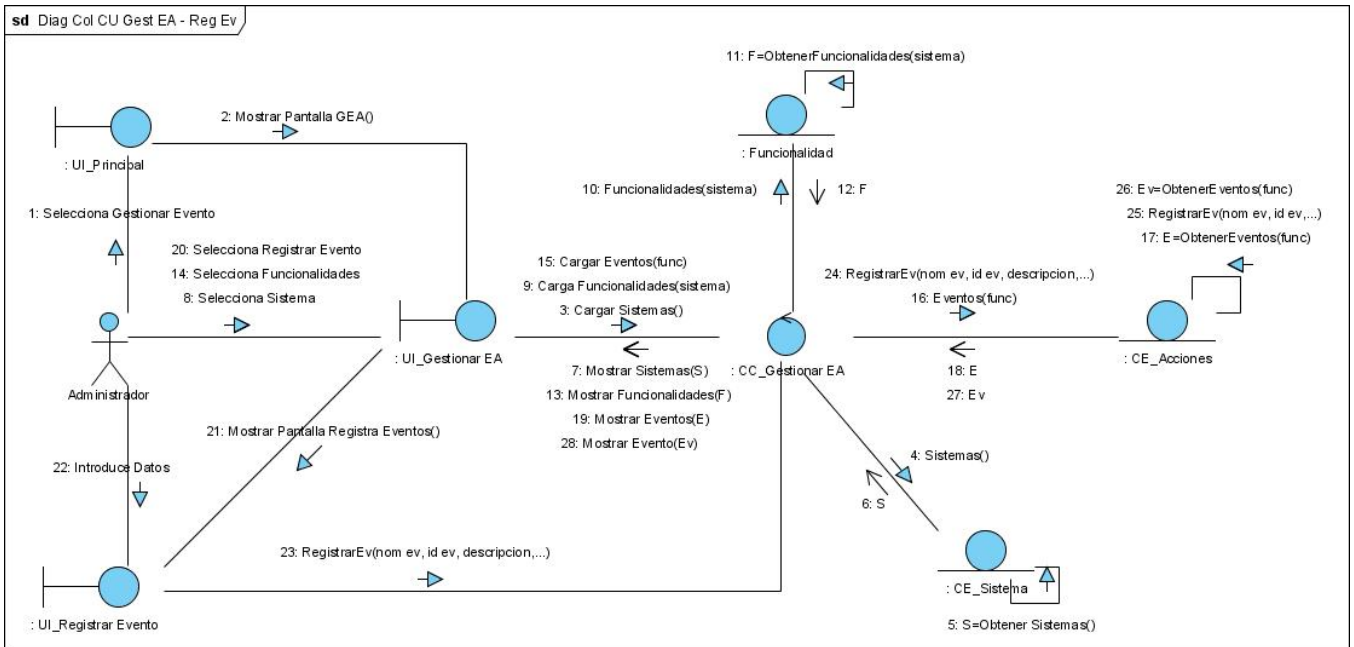


Figura 4.1 Diagrama de colaboración del caso de uso: Gestionar Eventos a Auditar – Escenario Registrar Eventos a Auditar.

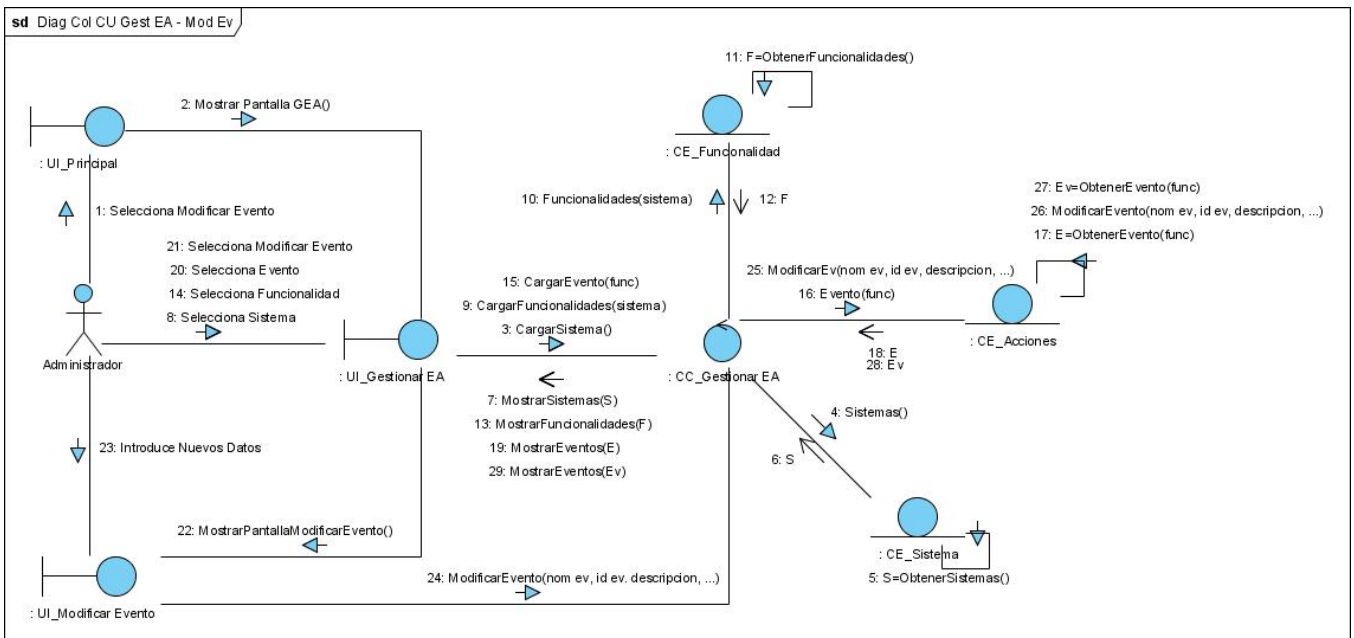


Figura 4.2 Diagrama de colaboración del caso de uso: Gestionar Eventos a Auditar – Escenario Modificar Eventos a Auditar.

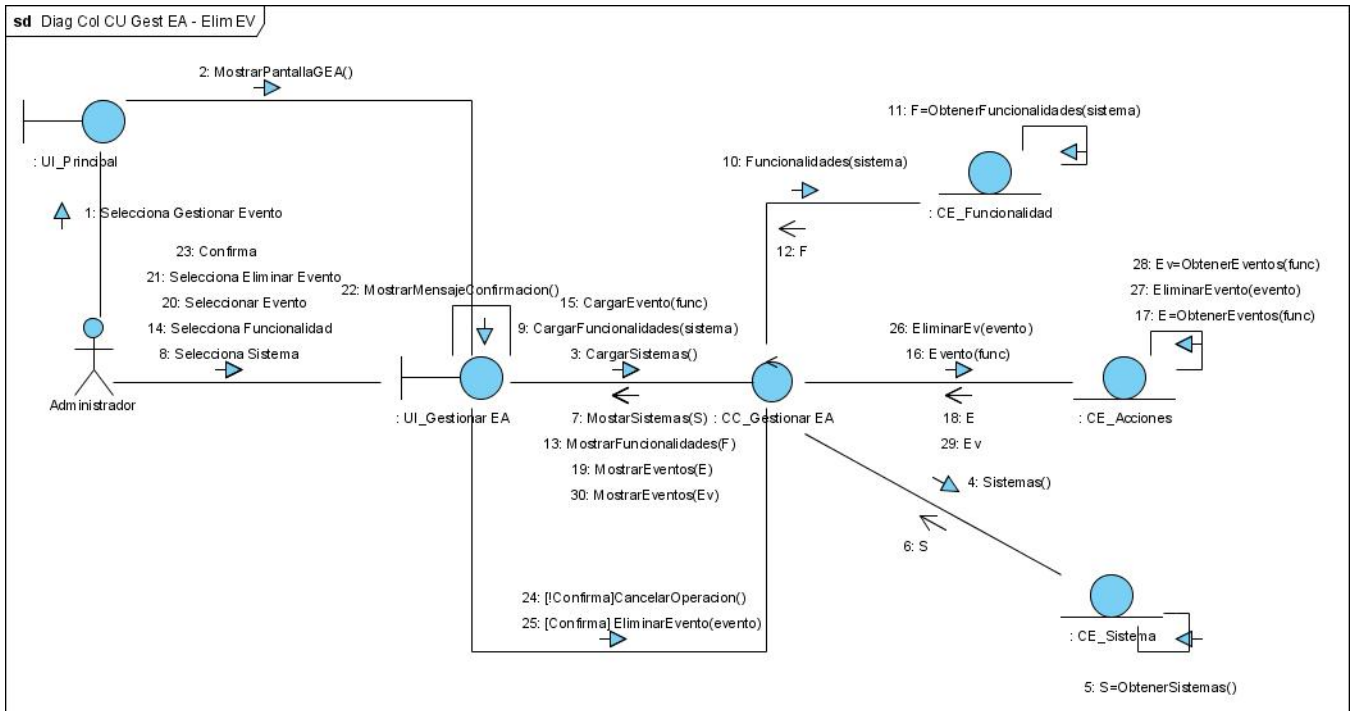


Figura 4.3 Diagrama de colaboración del caso de uso: Gestionar Eventos a Auditar – Escenario Eliminar Eventos a Auditar.

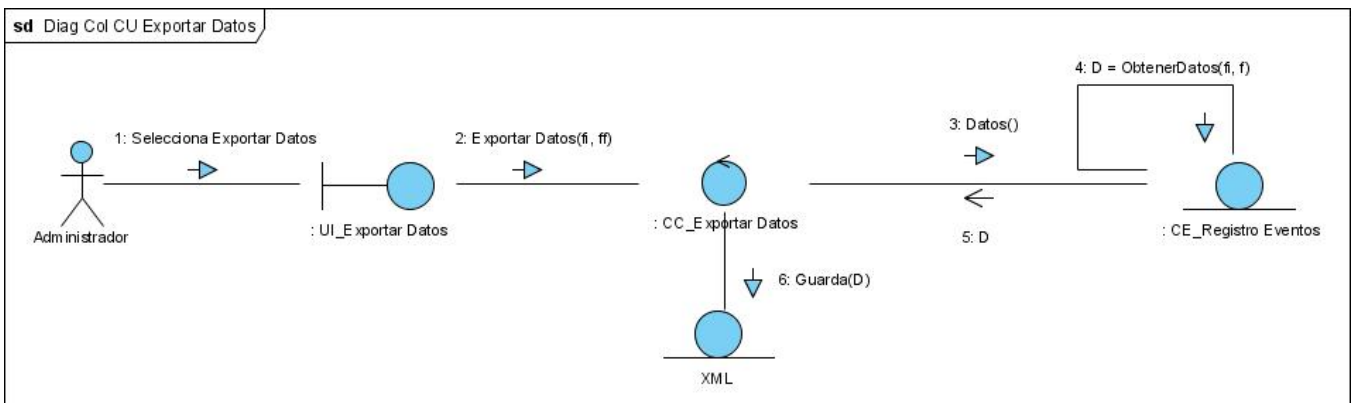


Figura 4.4 Diagrama de colaboración del caso de uso: Exportar Datos.

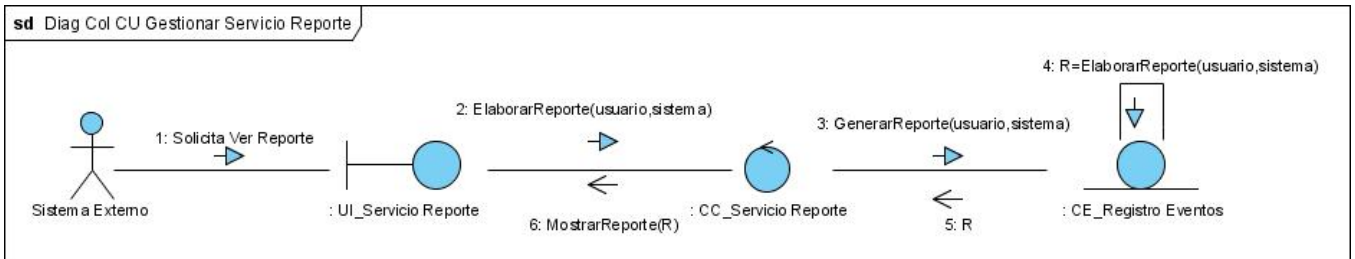


Figura 4.5 Diagrama de colaboración del caso de uso: Gestionar Servicio de Reporte.

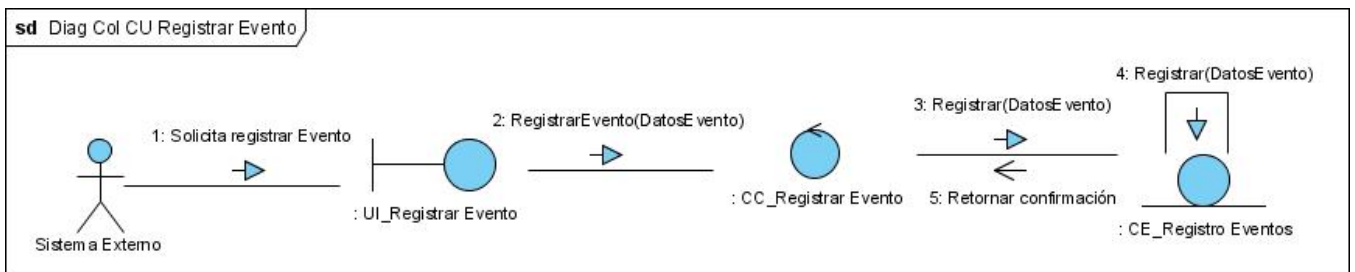


Figura 4.6 Diagrama de colaboración del caso de uso: Registrar Eventos.

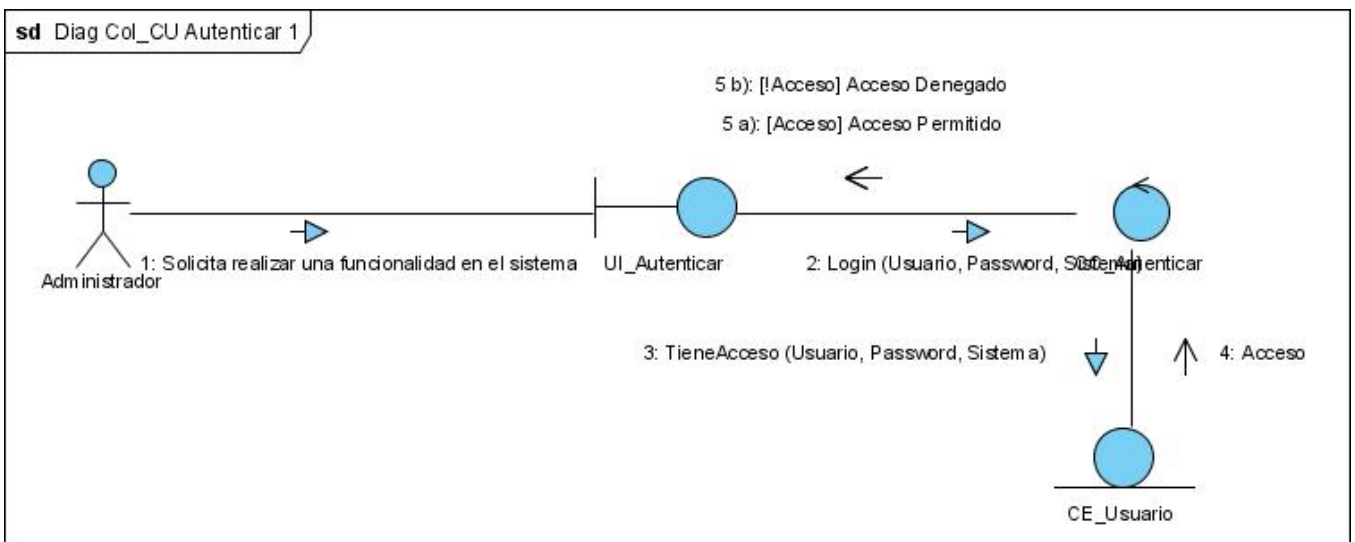


Figura 4.7 Diagrama de colaboración del caso de uso: Validar Usuario.

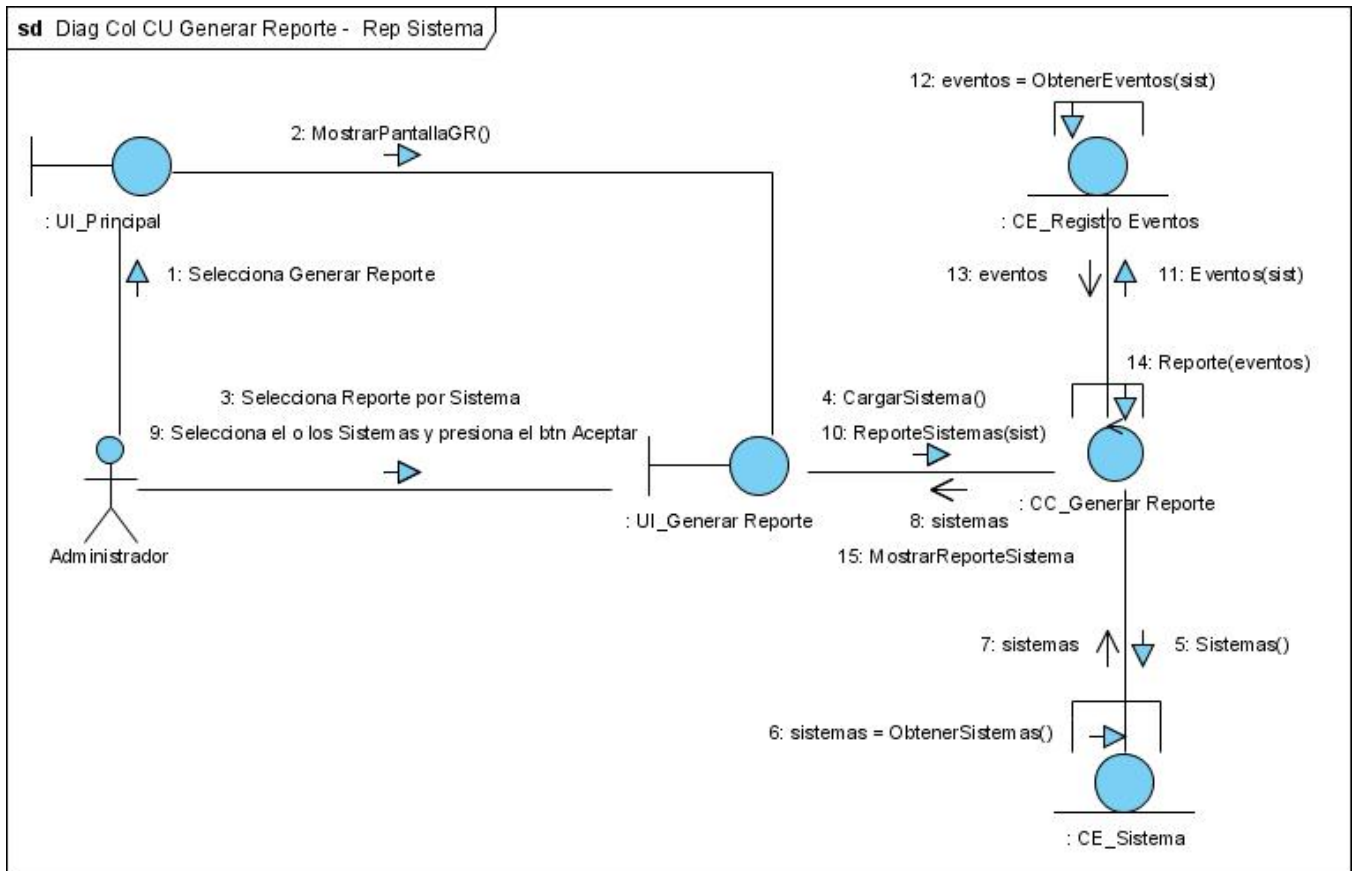


Figura 4.8 Diagrama de colaboración del caso de uso: Generar Reportes – Escenario Reporte por Sistema.

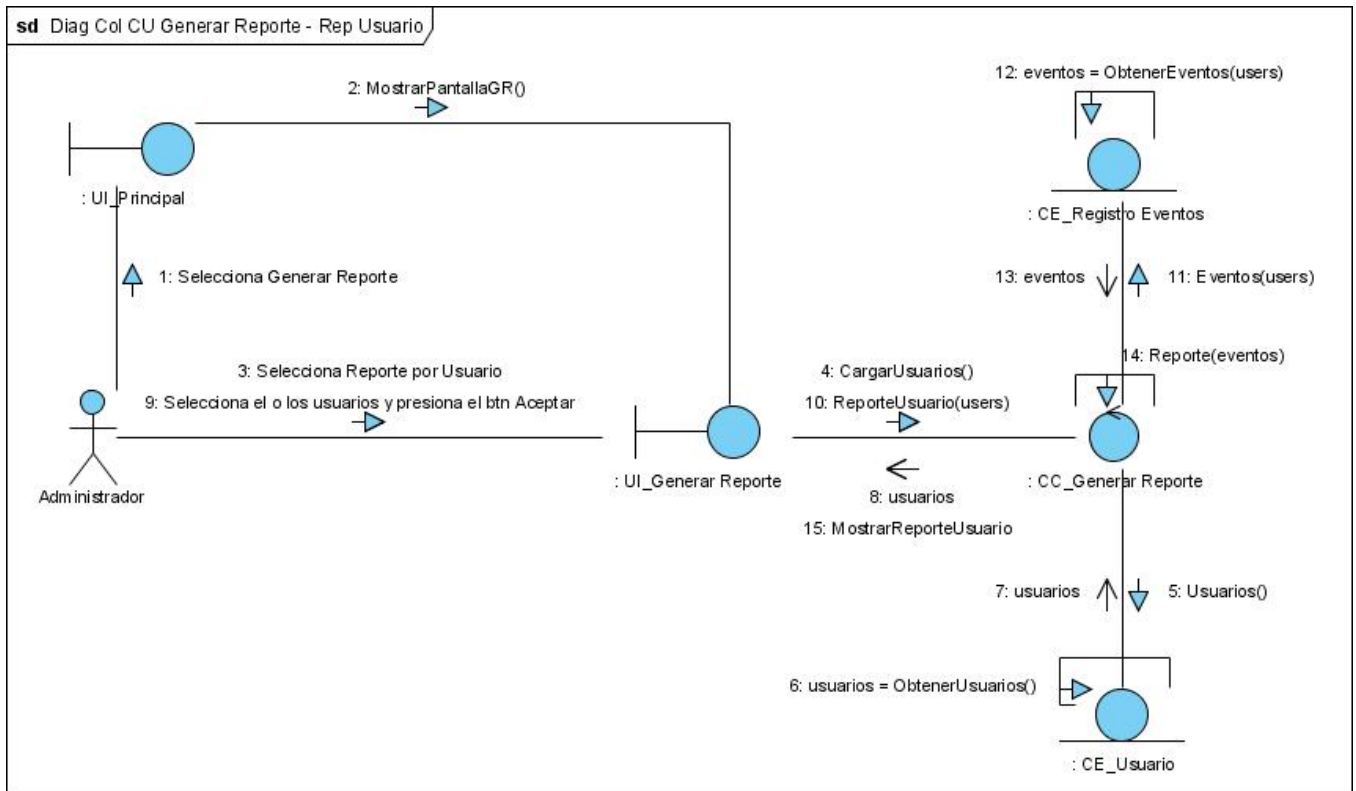


Figura 4.9 Diagrama de colaboración del caso de uso: Generar Reportes – Escenario Reporte por Usuario.

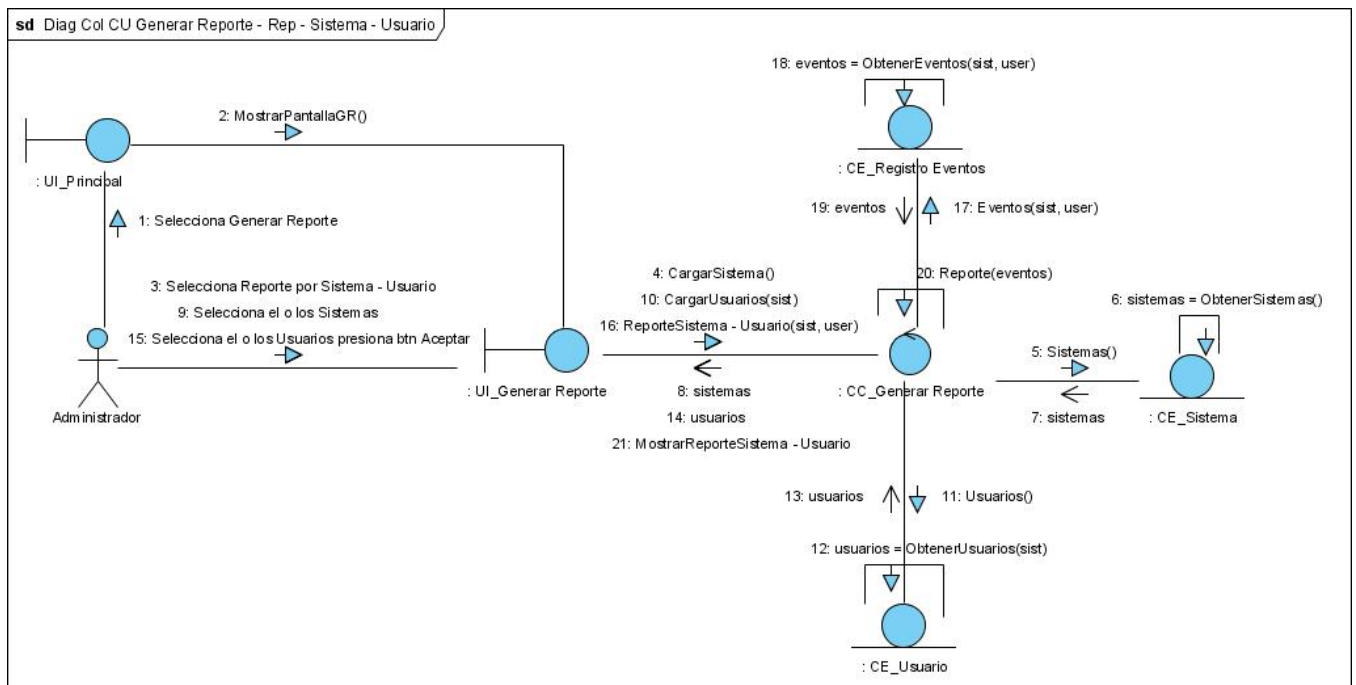


Figura 4.10 Diagrama de colaboración del caso de uso: Generar Reportes – Escenario Reporte por Sistema -Usuario.

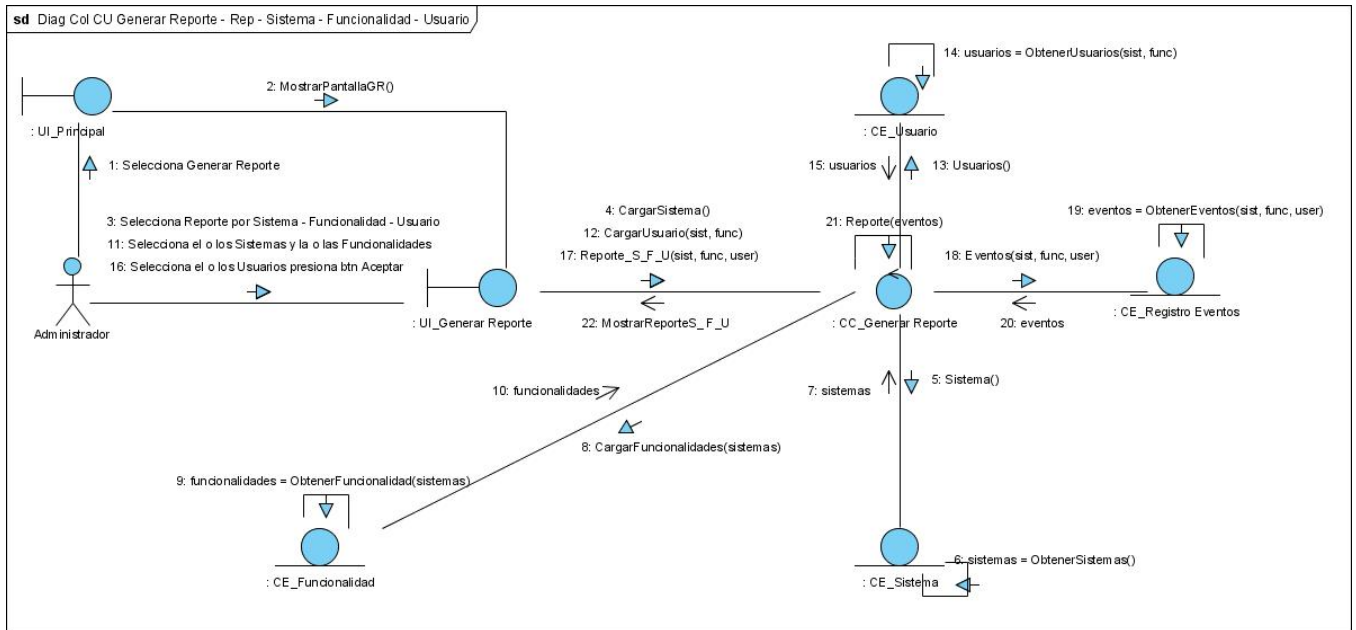


Figura 4.11 Diagrama de colaboración del caso de uso: Generar Reportes – Escenario Reporte por Sistema - Funcionalidades - Usuario.

4.3 Modelo de diseño

4.3.1 Arquitectura del Sistema

Uno de los elementos bases del proceso de desarrollo de software es diseñar la Arquitectura de Software. Esencialmente sobre ella se sustentan todos los mecanismos de diseño y representaciones de la estructura general de la aplicación a desarrollar. De la cohesión, utilidad y flexibilidad de los componentes de la arquitectura dependerán la calidad final y la utilidad del software. La correcta definición del estilo arquitectónico a utilizar, patrones y mecanismos de diseño es la raíz de lo anteriormente descrito.

Tal y como se detalla en el epígrafe 0 del Capítulo 2 el estilo arquitectónico a utilizar es Modelo – Vista – Controlador, incluyendo un conjunto de frameworks que facilitarán el trabajo durante el desarrollo. A continuación de muestran estas características aplicadas al diseño del sistema.

4.3.2 Presentación general

La aplicación diseñada es un pequeño módulo del Subsistema para el Control de la Seguridad y Auditoría en las aplicaciones de la entidad. Para comenzar el diseño este es un elemento muy importante a tener presente, debido a que deberá brindar interoperabilidad a la aplicación. En la figura siguiente se representa la vista del ERP y algunos de los módulos que lo componen.

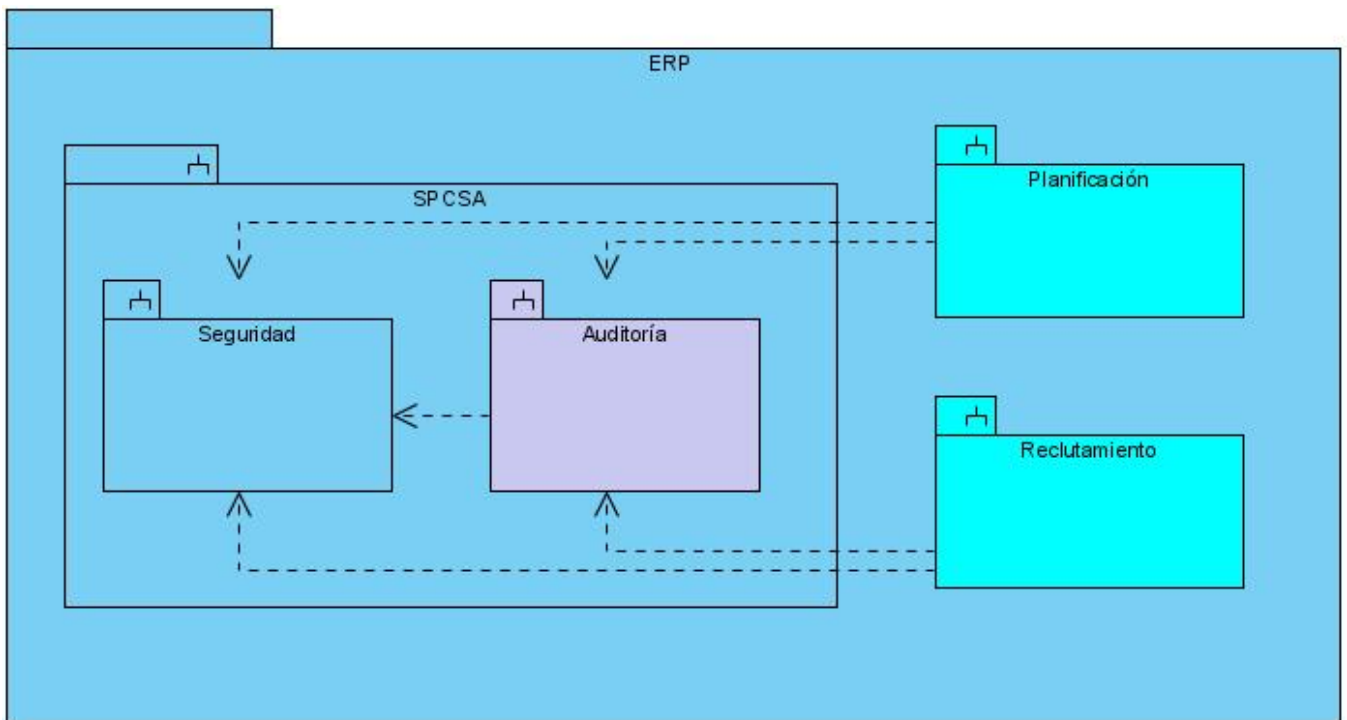


Figura 4.12 Vista del ERP y algunos de los módulos que lo componen

4.3.3 Mecanismos de Diseño

Es muy importante prestar atención al esfuerzo invertido en la modelación, tratando de hacerla lo mas eficientemente posible, y de manera que el resultado obtenido (los modelos), sean comprensibles y permitan una comunicación efectiva.

Un elemento que puede ser favorable en este sentido son los Mecanismos de Diseño, artefacto propuesto y descrito en [RUP_Mec, 2003]⁵. Los mismos reportan beneficios para al menos 3 propósitos:

1. Mantener la homogeneidad en el diseño.
2. Reutilizar soluciones anteriormente probadas.
3. Reutilizar documentación.

Por lo anteriormente descrito se utilizan en la propuesta a continuación.

⁵ [RUP,2003] *Artifact Design Mechanism. Rational Unified Process, v 2003.06.00*. Documentación del Proceso Unificado, version 2003.06.00.

Vista

La Vista no es más que la presentación del sitio, o sea, la parte exterior del mismo, en la práctica se recomiendan que sean ficheros HTML, al que se les puede incrustar, en caso de que sea necesario, código en el lenguaje del servidor. Como se especifica en el epígrafe 2.4.2 en el Capítulo 2 en el diseño de la vista se utiliza EXT. (Ver Epígrafe 0 Cap. 2)

Uso de EXTJS

<u>Clase</u>	<u>Descripción</u>
ext-base	Contiene lo necesario para el manejo del DOM, Ajax, animaciones, etc.
ext-all	Se encarga de la creación de los componentes que presentaran en la vista.
vista	Esta clase es la que mostrará al usuario los componentes que fueron construidos a partir de la petición que realiza la clase js_vista a la clase ext-all.
js_vista	Es la clase que se encarga de personalizar, en función de las necesidades del CUS, el comportamiento de los componentes de Ext.

Tabla 4.1 Descripción de las clases utilizadas en la vista.

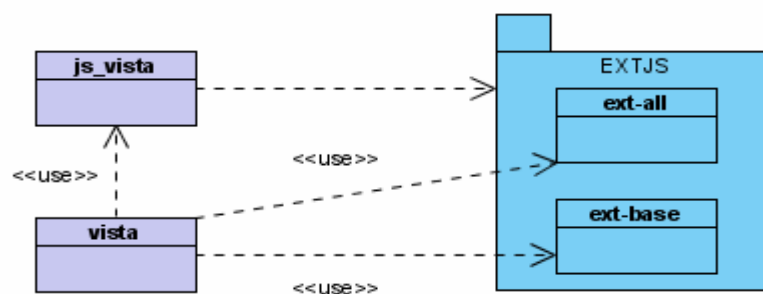


Figura 4.13 Diagrama de Clases Genérico para EXTJS (Mecanismo de Diseño)

Controlador

El Controlador es el coordinador de los recursos de software (Vistas y Modelos) para responder a cada una de las peticiones HTTP realizadas a la aplicación. Son las clases que gestionan el manejo de la lógica del negocio (Dominio). Por lo general incluyen las restricciones y validaciones fundamentales determinadas por las reglas. Para un caso de uso puede representarse una clase controladora o más. En este caso se usa del Zend Framework por las grandes utilidades que brinda. (Ver Epígrafe 0 Cap. 2)

Uso de Zend Framework

<u>Clase</u>	<u>Descripción</u>
Zend_Controller_Action	Zend_Controller_Action es el componente base de la acción del controlador. Cada controlador es una sola clase que hereda de esta clase y podría contener uno o más métodos.
vistaController	Controlador donde se describe la lógica de negocio del caso de uso.
modelController	Esta clase se especializa en el proceso de abstracción de datos al nivel de ORM que brinda Doctrine PHP, este posee la(s) consulta(s) a la BD.

Tabla 4.2 Descripción de las clases utilizadas en el controlador.

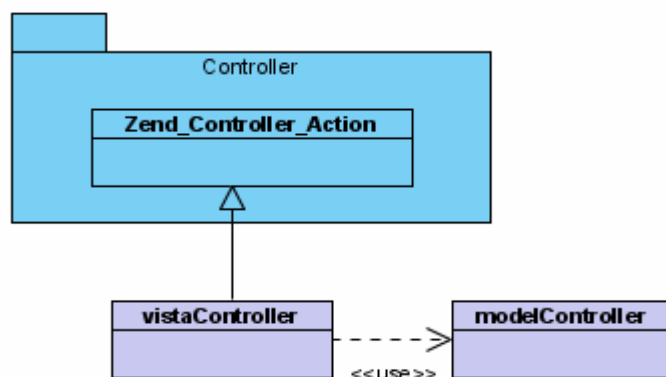


Figura 4.14 Diagrama de Clases Genérico para Zend Framework (controladores)

Modelo

El Modelo representa el proceso de abstracción para acceder al medio de persistencia de datos, en este se programan las clases que interactúan con la base de datos. En este caso se utiliza en el diseño Doctrine.

Uso de Doctrine PHP

<u>Clase</u>	<u>Descripción</u>
Doctrine_Record	Es la clase que, en función del mapeo de la base de datos, genera la interfaz entre las consultas en DQL (Doctrine Query Language) y el SQL nativo de PDO.
Clase_bd	Esta clase posee una instancia de la conexión de la base de datos y en esta se deben escribir las consultas tanto en DQL como aplicando el patrón de registro activo que el mismo marco de trabajo proporciona.

Tabla 4.3 Descripción de las clases utilizadas en el modelo.

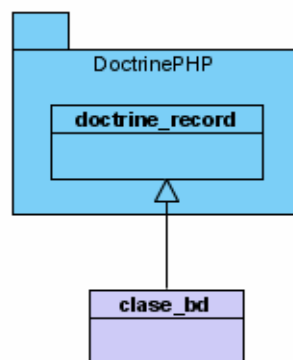


Figura 4.15 Diagrama de Clases Genérico para Doctrine PHP

Modelo – Vista - Controlador

Luego de analizar los elementos principales de cada una de las partes componentes del estilo MVC utilizando características de los frameworks a utilizar se está en condiciones entonces de presentar la propuesta general.

A continuación se propone la vista de Gestión de Modelo de la Arquitectura Base para MVC. Se representan los paquetes fundamentales y el conjunto de clases utilizadas directamente de entre el resto de las contenidas.

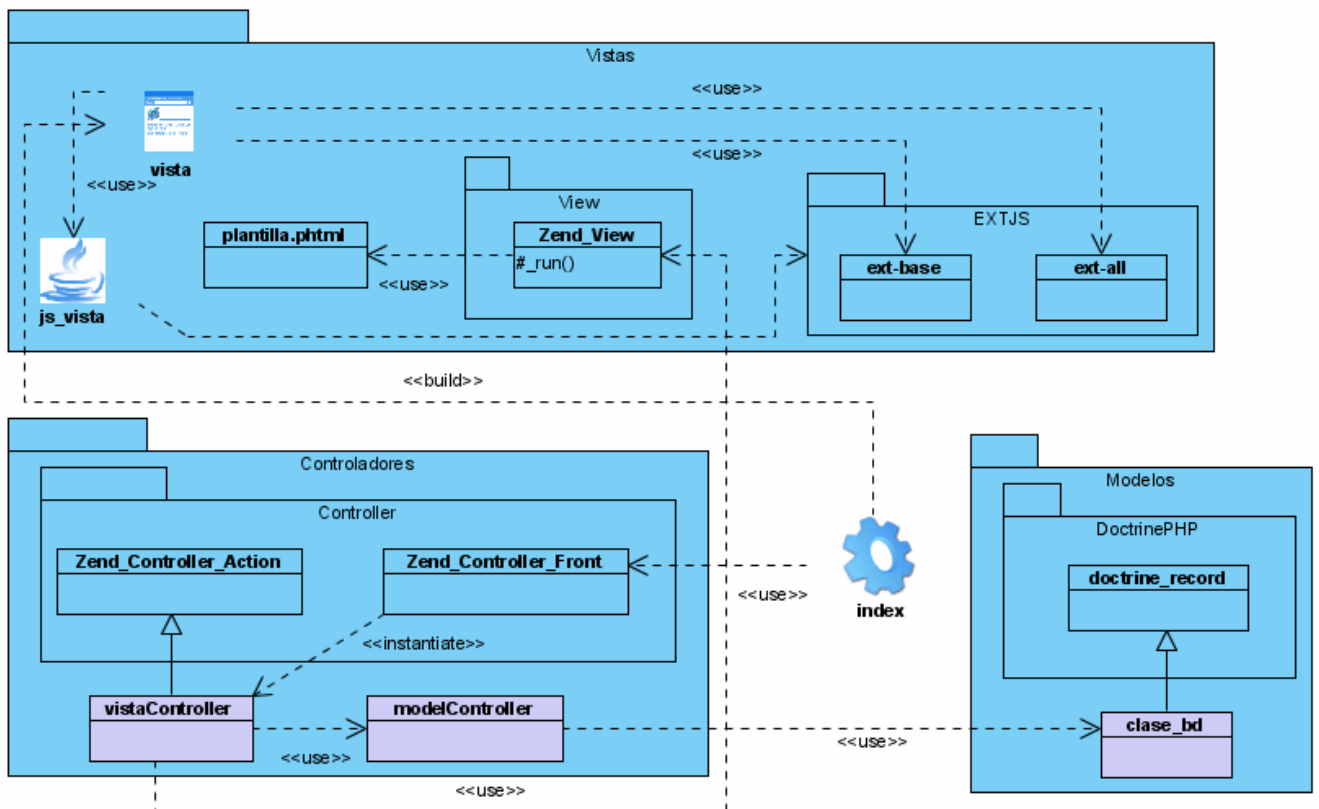


Figura 4.16 Vista de Gestión de Modelo Genérico para MVC

Los paquetes View y Controller son los paquetes incluidos por defecto en la carpeta library del Zend Framework.

4.3.4 Diagrama Genérico

Una vez definida la base de la arquitectura se presenta a continuación la propuesta para los Diagramas de Clases del Diseño para los Casos de Uso del Sistema de manera genérica. En el diseño se propone no volver a especificar patrones de diseño puesto que la reutilización de EXT y de Doctrine incorporan algunos de los más utilizados para la presentación y el manejo de datos.

Se hace uso de los artefactos (clases, paquetes, relaciones, subsistemas) como medio para simplificar diagramas que puedan ser complejos y lograr un mejor entendimiento.

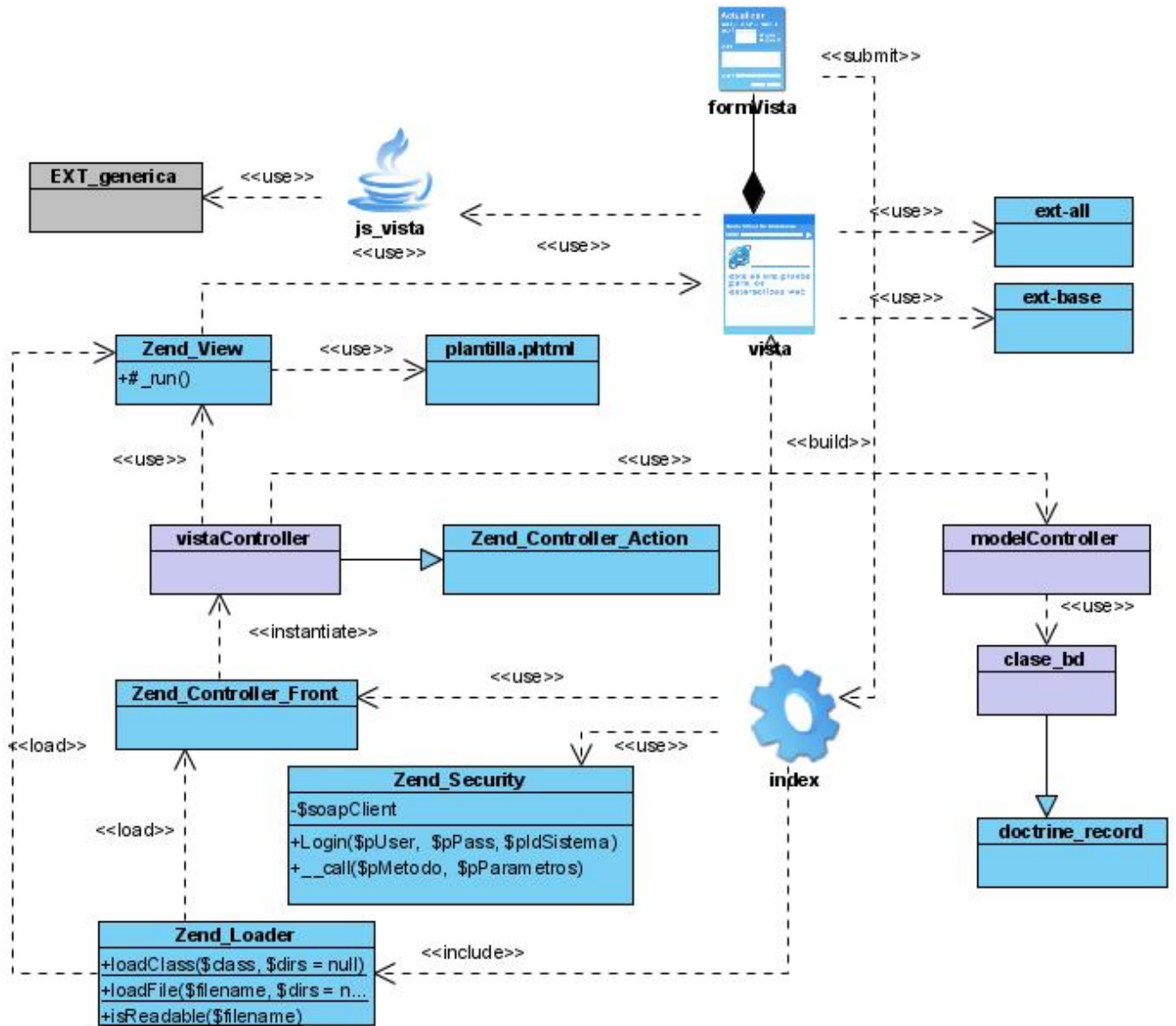


Figura 4.17 DCD para Caso de Uso Genérico (Mecanismo de Diseño)

<u>Clase</u>	<u>Descripción</u>
Zend_Controller_Front	Zend_Controller_Front procesa todas las peticiones recibidas por el servidor y es en última instancia responsable de delegar peticiones a (Zend_Controller_Action). ⁶
Zend_View	Permitirá separar el código que muestra la página del código dentro de las funciones de action además de cargar la vista correspondiente.
Zend_Loader	Tiene las funciones estáticas requeridas que nos permitirán cargar cualquier clase del Zend Framework o de la aplicación.
Zend_Security	Es la interfaz entre el Subsistema de Control de Auditoría y el Sistema Integral de Gestión de Seguridad.

Tabla 4.4 Descripción clases DCD genérico

Analizando entonces la real composición de los paquetes que se reutilizan (ver Figuras 4.13, 4.14, 4.15) y para lograr una simplificación para casos de uso que puedan ser más complejos se presenta como versión definitiva:

⁶ Ver Tabla 4.2 Descripción de las clases utilizadas en el controlador.

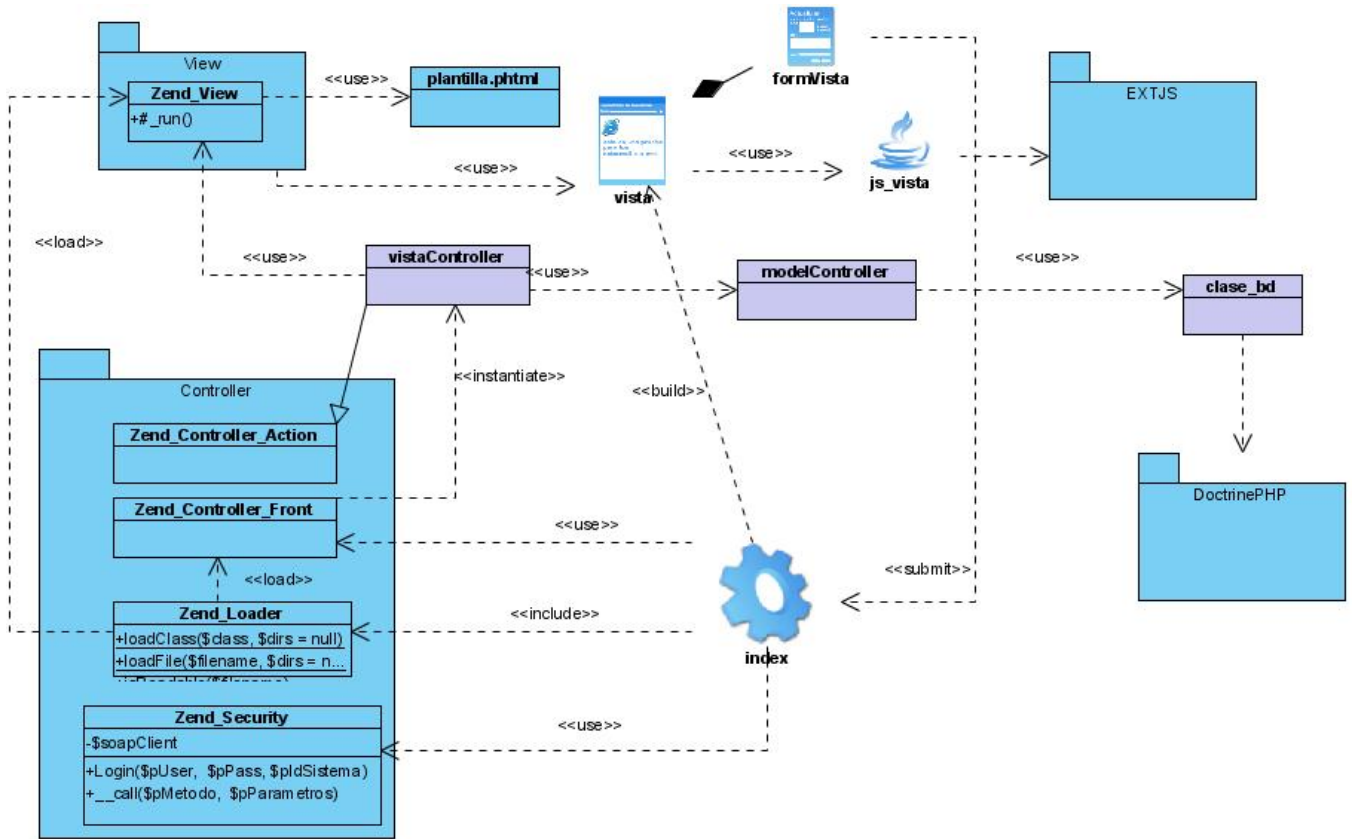


Figura 4.18 Propuesta final de DCD para Caso de Uso Genérico

4.4 Diagramas de clases del diseño

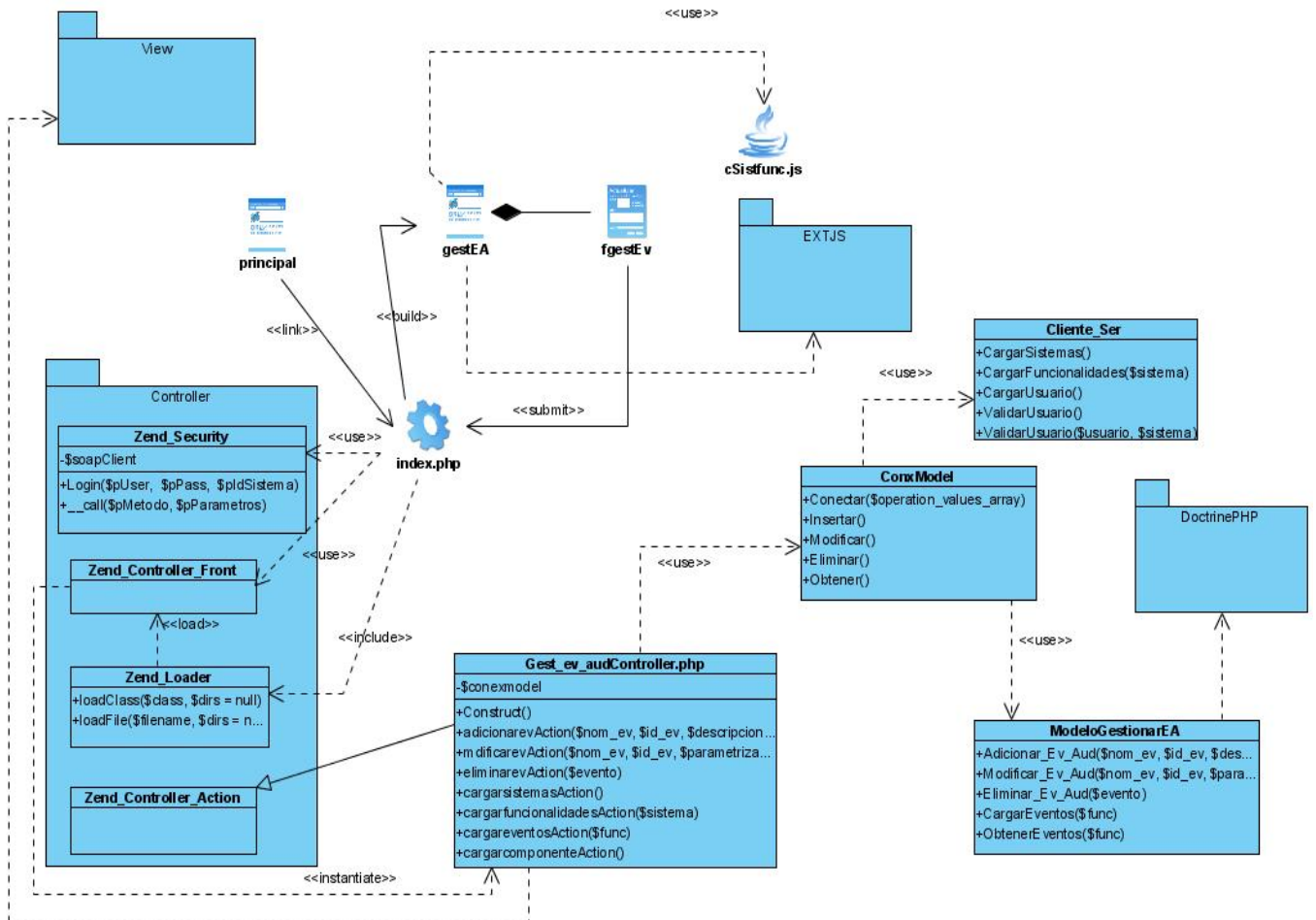


Figura 4.19 Diagrama de Clases del diseño Caso de uso Gestionar Eventos a Auditar

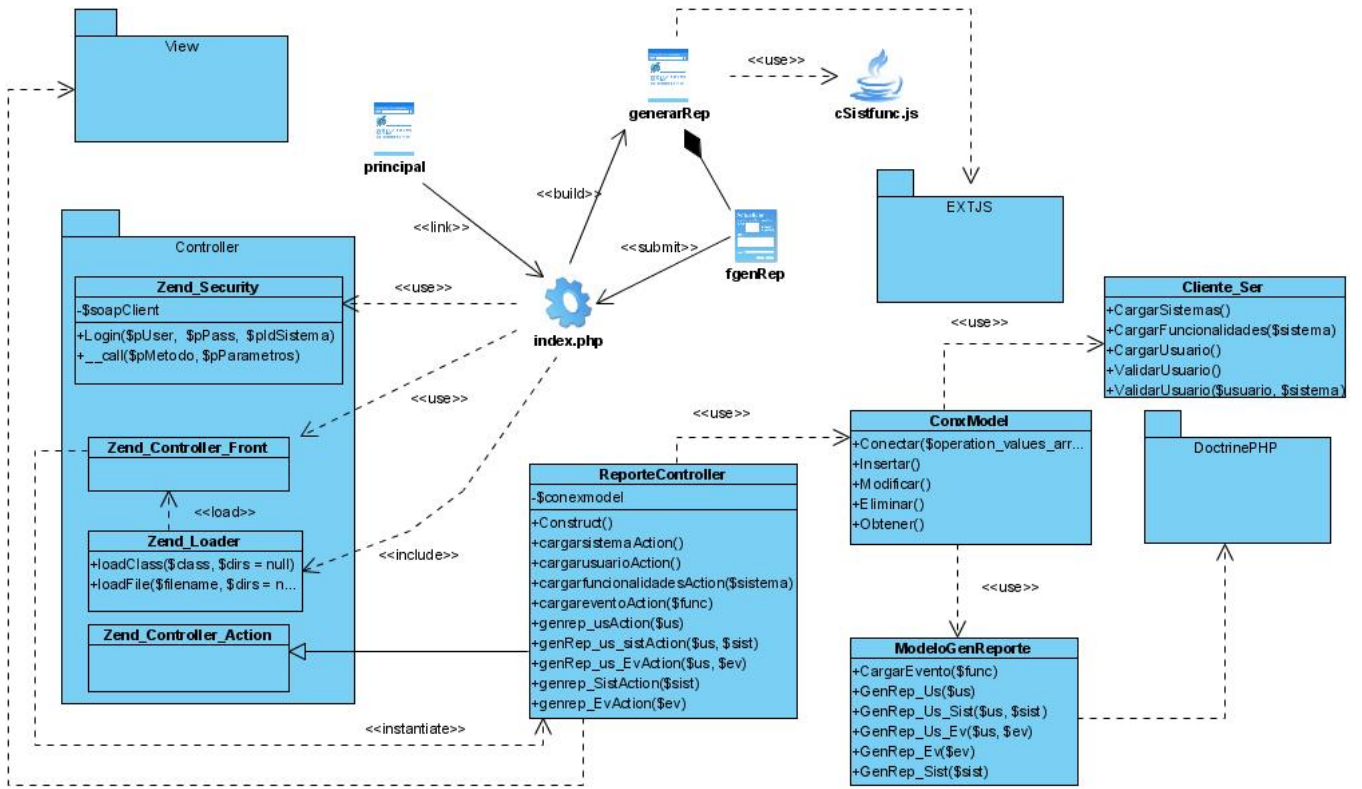


Figura 4.20 Diagrama de Clases del diseño Caso de uso Generar Reporte

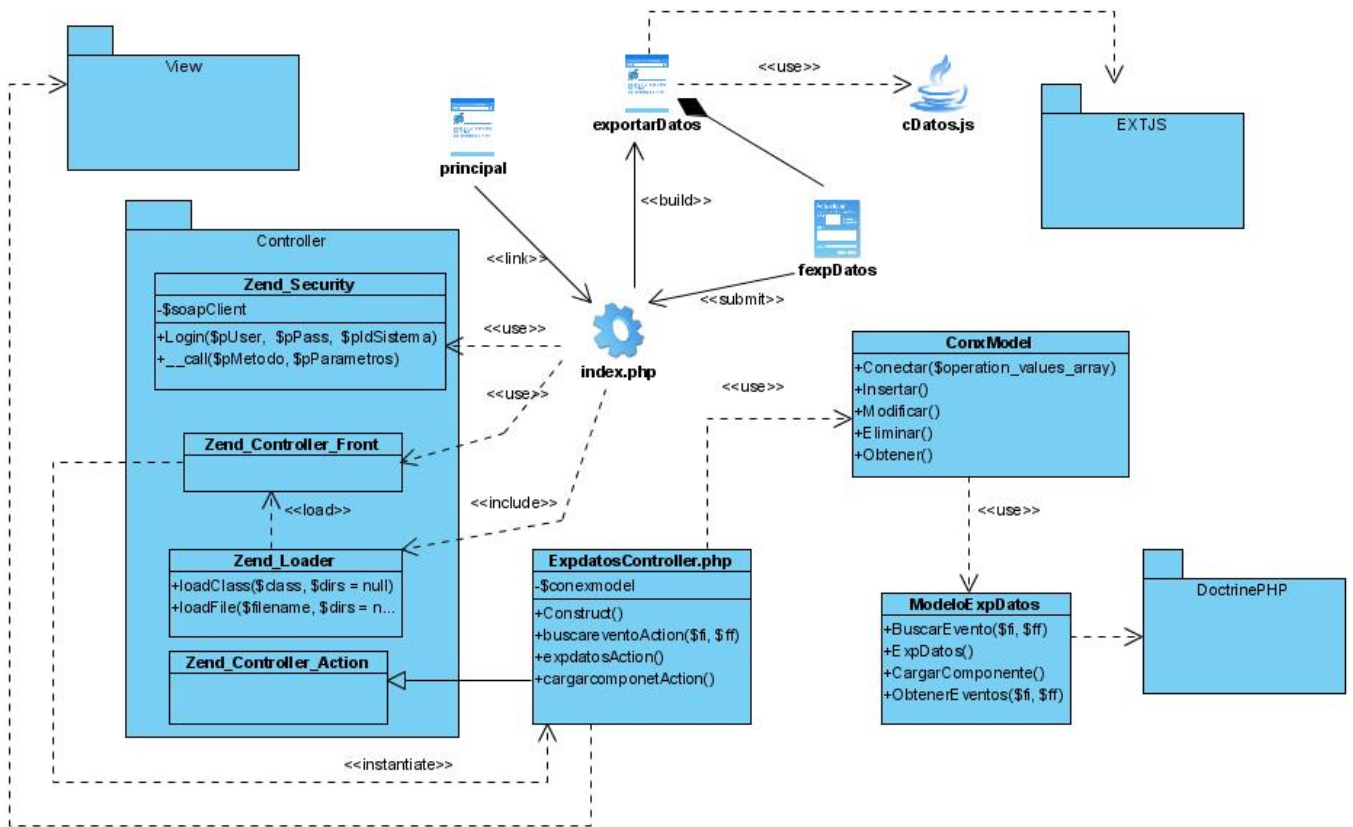


Figura 4.21 Diagrama de Clases del diseño Caso de uso Exportar Datos

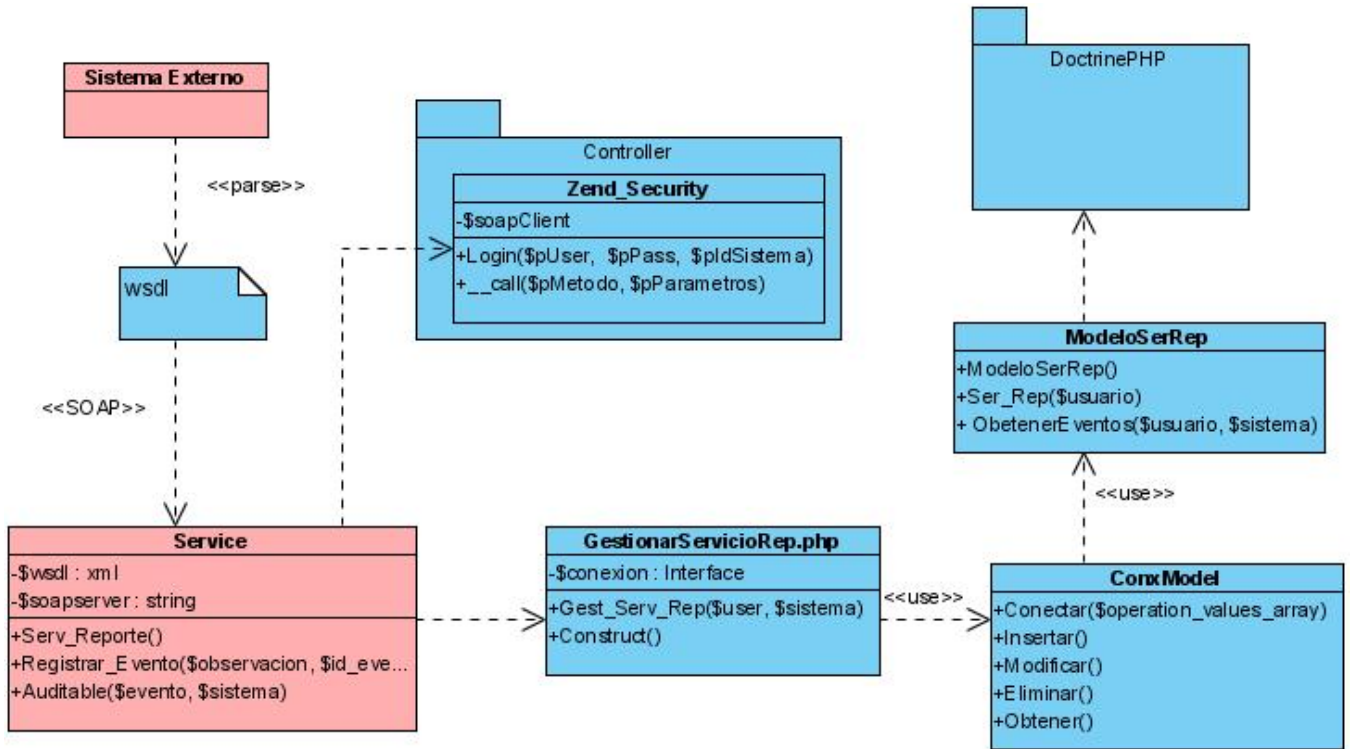


Figura 4.22 Diagrama de Clases del diseño Caso de uso Gestionar Servicio de Reporte.

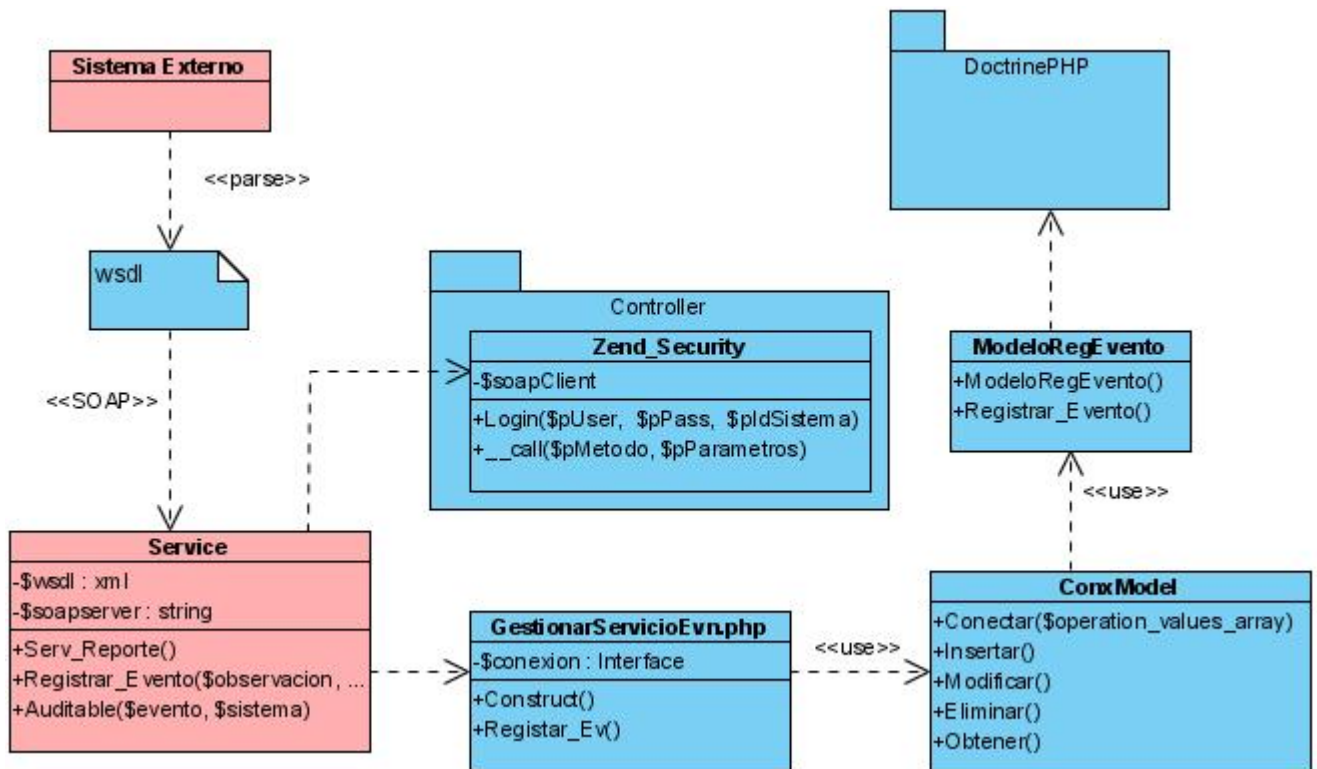


Figura 4.23 Diagrama de Clases del diseño Caso de uso Registrar Evento

4.4.1 Diagramas de Secuencias clases del diseño

Al igual que en el análisis, en el diseño existen los diagramas de interacción (colaboración y secuencia). En este trabajo se decidió realizar los diagramas de secuencias para las clases de diseño, ya que, es más descriptivo en cuanto a la secuencia de eventos que suceden entre objetos de y clases del caso de uso hasta el propio actor. Para ver dichos diagramas remítase al **Anexo 2** Diagramas de Secuencia de las Clases del Diseño.

4.5 Diseño de la Base de Datos

4.5.1 Modelo lógico de datos

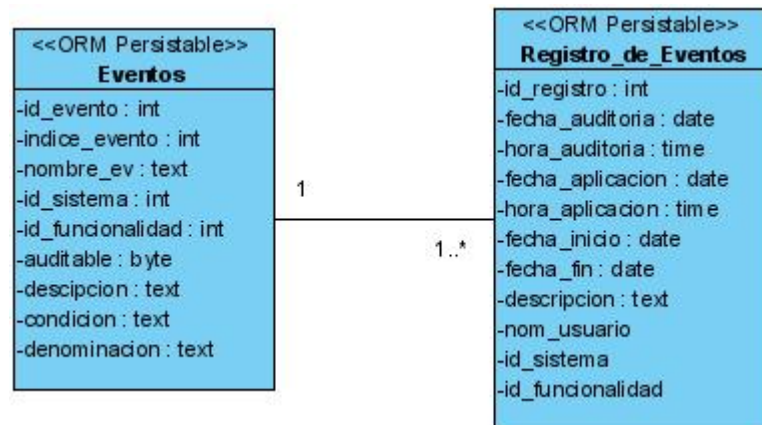


Figura 4.24 Diagrama de clases persistentes.

4.5.2 Modelo físico de datos

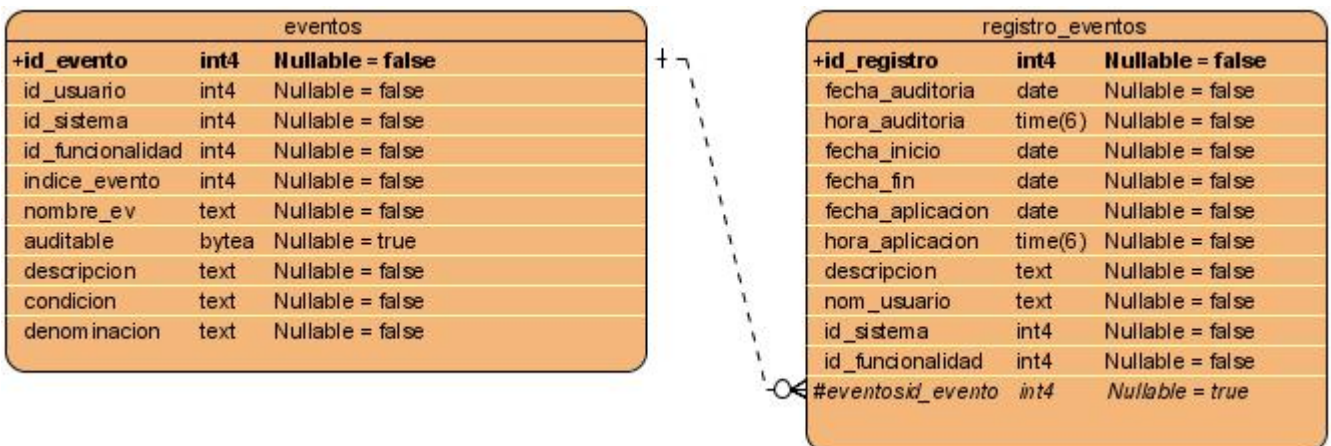


Figura 4.25 Representación de la Base de Datos.

4.6 Diagrama de Despliegue

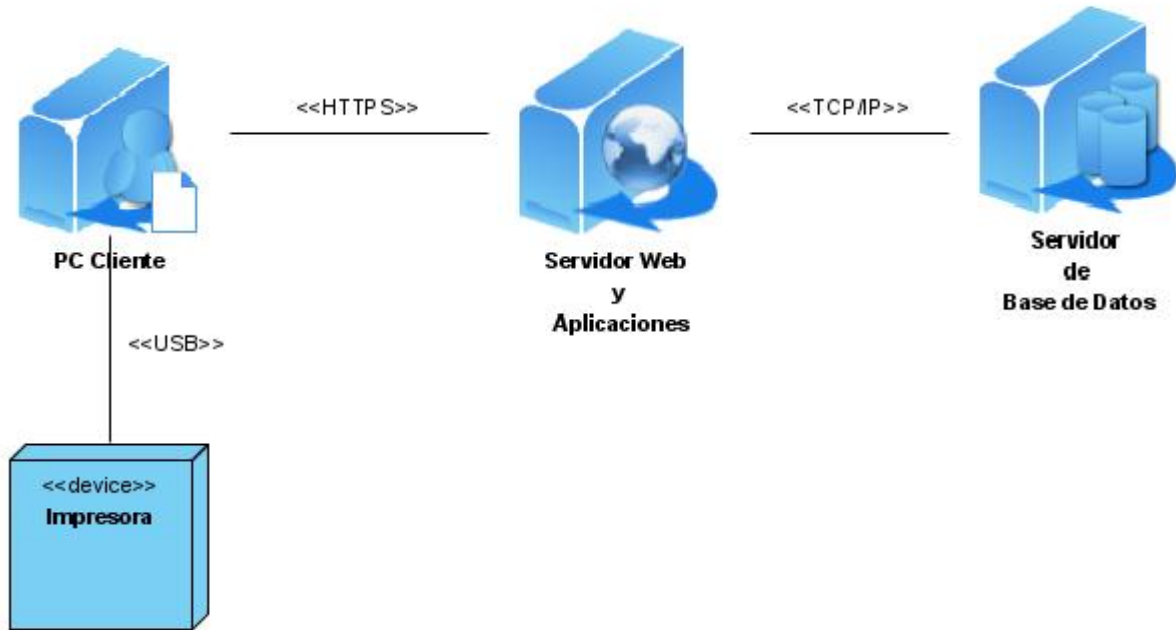


Figura 4.26 Representación de la distribución física de la aplicación.

4.7 Modelo de Implementación

4.7.1 Composición de los paquetes

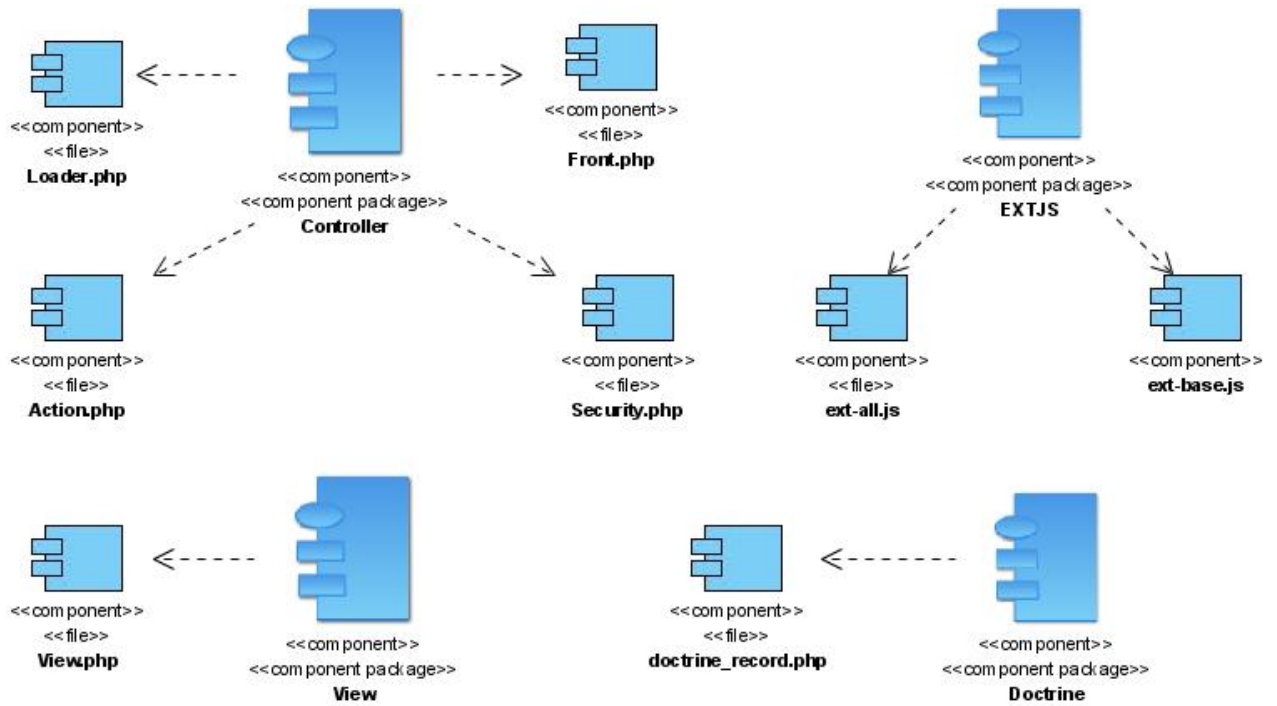


Figura 4.27 Composición de los paquetes.

4.7.2 Diagrama Genérico para los casos de uso

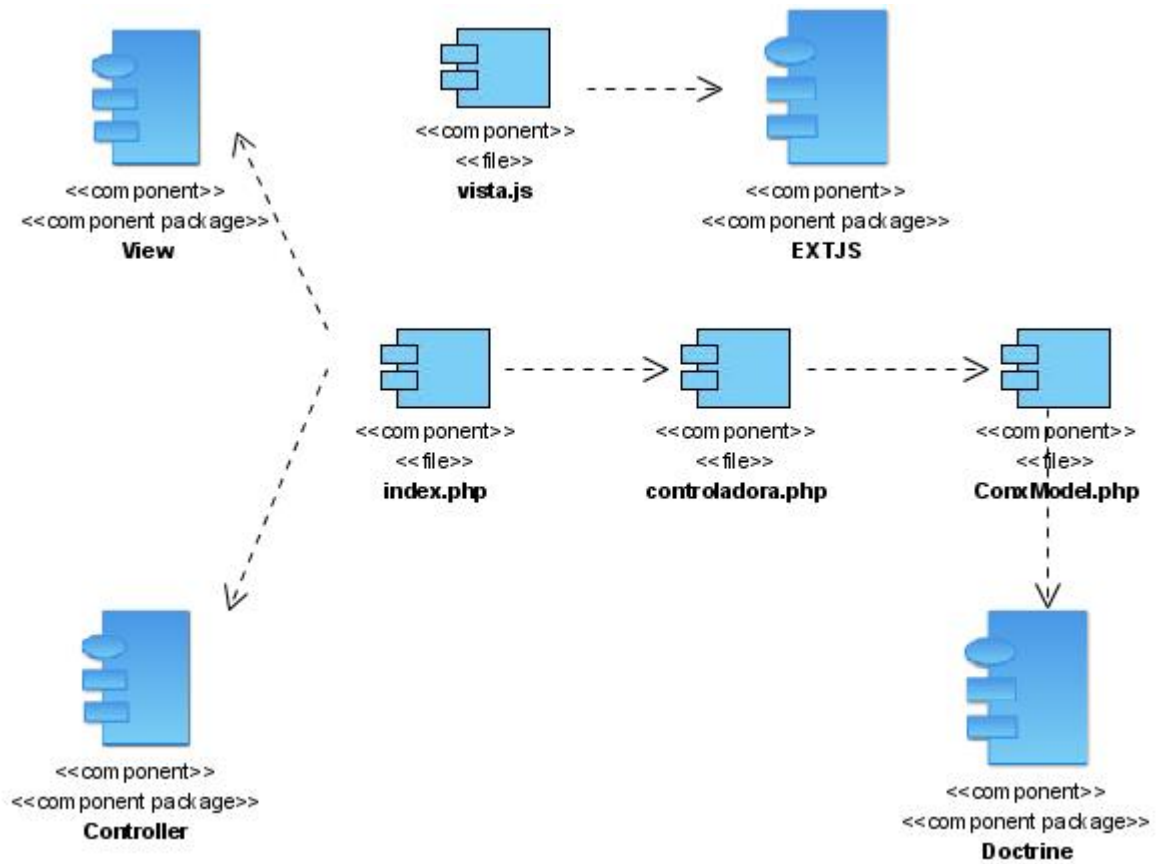


Figura 4.28 Diagrama Genérico para cada CU.

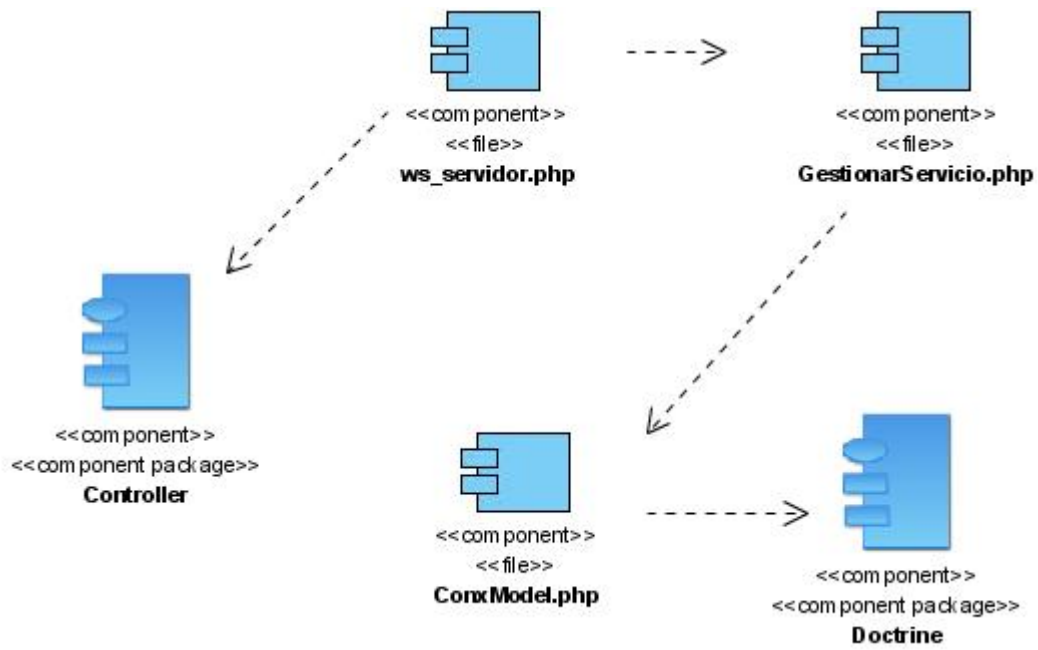


Figura 4.29 Diagrama Genérico para cada CU de Servicio.

4.7 Conclusiones

En este capítulo se ha realizado una descripción de la solución propuesta a través de los distintos artefactos que dispone RUP para la construcción de una aplicación de este tipo, analizando los diagramas de clases, diseño de la base de datos, los principios de diseño propuestos y el modelo de despliegue. Con todo lo antes mencionado se da fin al proceso de análisis para luego pasar a la fase de elaboración del software.

Conclusiones

A partir del estudio de los procesos de auditoría y de su análisis detallado se obtiene el conocimiento necesario para representar, utilizando una metodología de desarrollo, seleccionada luego del análisis de las existentes, los procesos anteriormente mencionados teniendo en cuenta los requerimientos del cliente y realizar el diseño del Subsistema para el Control de Auditoría en las aplicaciones de la entidad.

Con éste se logran los artefactos necesarios para pasar al proceso de implementación del software. Se establece la base de la arquitectura así como los principales mecanismos de diseño, artefactos y modelos que integran la documentación a utilizar por los desarrolladores en la codificación.

Se entrega el modelo de implementación que representa los elementos físicos que integran el sistema, los diferentes ficheros de datos, ejecutables, librerías, módulos y subsistemas de implementación descritos de manera detallada a manera de diagramas.

Se presenta el modelo de datos, elemento esencial para la creación de la base de datos que soportará la información necesaria que manejará el sistema así como la representación de la distribución física de la aplicación utilizando el diagrama de despliegue.

Recomendaciones

Como se ha observado los objetivos trazados para este trabajo han sido logrados, sin embargo, por ser la primera vez que se estudian estos procesos, que trae consigo falta de experiencia en este tema, se recomienda:

- Continuar el desarrollo de la investigación iniciada con este trabajo, sobre los procesos de auditoría con el objetivo de perfeccionar y aumentar las posibles funcionalidades de la aplicación web futura.
- Profundizar más en el tema de los reportes, ya que, es la parte esencial del control de la auditoría.
- Desarrollar la propuesta diseñada y ponerla a prueba en todas las aplicaciones de la entidad así como en las del ministerio.

Glosario de términos

1. **AJAX:** Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente, es decir, en el navegador de los usuarios y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad en la aplicación.
2. **Apache:** El servidor HTTP Apache es un software (libre) servidor HTTP de código abierto para plataformas Unix , Windows, Macintosh y otras, que implementan el protocolo HTTP.
3. **APIs Web:** (Application Programming Interface) son un conjunto de especificaciones para comunicarse con una aplicación, normalmente para obtener información y utilizarla en otros servicios.
4. **Artefactos:** Una parte de la información que es producida, modificada, o usada por un proceso, define un área de responsabilidad, y está sujeta al control de versión. Un artefacto puede ser un modelo, un elemento del modelo, o un documento. Un documento puede adjuntar otros documentos.
5. **C++:** Lenguaje de programación que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos, es una extensión del lenguaje de programación C.
6. **CGI:** Common Gateway Interface, una definición estándar para un interfaz entre un servidor web y un programa externo que permite hacer peticiones de servicio a los programas externos.
7. **CPUs:** Central Processing Unit, (Unidad Central de Procesamiento), es el componente en una computadora digital que interpreta las instrucciones y procesa los datos contenidos en los programas de computadora.
8. **Cygwin:** Es una colección de herramientas para Windows. Proporciona una interfaz de aplicación similar a UNIX. Las herramientas te permiten exportar un número significativo de programas de UNIX a Windows, sin realizar significantes cambios en el código.
9. **Drivers ODBC:** Es un controlador que guiará paso a paso a través de la conexión una llamada a una función de entrada, en la que, dependiendo de los parámetros especificados se realizan unas u otras funciones.
10. **ERP:** Son sistemas de información gerenciales que integran y manejan muchos de los negocios asociados con las operaciones de producción y de los aspectos de distribución de una compañía comprometida en la producción de bienes o servicios.

11. **GPL:** GNU General Public License (La Licencia Pública General de GNU), está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.
12. **HTML:** HyperText Markup Language (Lenguaje de Marcado de Hipertexto), predominante para la construcción de páginas web.
13. **HTTP:** Protocolo de Transferencia de Hipertexto, es el protocolo de transmisión estándar usado en la World Wide Web (www).
14. **IEEE:** Institute of Electrical and Electronics Engineers, Asociación técnico-profesional mundial dedicada a la estandarización. Es la mayor asociación internacional sin fines de lucro formada por profesionales de las nuevas tecnologías, como ingenieros eléctricos, ingenieros en electrónica, científicos de la computación e ingenieros en telecomunicación.
15. **iPlanet:** iPlanet Application Server, proporciona un soporte robusto para los estándares J2EE, además de capacidades claves adicionales a través de servicios como el balance de carga, el control de fallos, y una alta disponibilidad aplicada a estos mismo estándares J2EE.
16. **KB:** Un kilobyte, es una unidad de almacenamiento de información cuyo símbolo es el kB.
17. **Linux:** Es un sistema operativo tipo Unix que se distribuye bajo la Licencia Pública General de GNU (GNU GPL), es decir que es software libre.
18. **Log W3C:** Brinda información necesaria para hacer un seguimiento exhaustivo del tráfico en un determinado sitio web, facilitando la realización de los informes apropiados con los valores oportunos.
19. **Mac OS:** Macintosh Operating System (Sistema Operativo de Macintosh), el primer sistema operativo con una interfaz gráfica de usuario en tener éxito.
20. **Microsoft Internet Information Server:** Servicios de Internet Información Server (IIS) simplifica la publicación de información en Internet o en la Intranet. IIS incluye una amplia gama de funciones administrativas para controlar sitios Web y el servidor Web.
21. **Microsoft Windows:** Es un sistema operativo desarrollado y comercializado por Microsoft.
22. **Mozilla FireFox:** Es un navegador de Internet, con interfaz gráfica de usuario, desarrollado por la Corporación Mozilla.
23. **Netscape:** Navegador web.
24. **OpenBSD:** es un sistema operativo libre tipo Unix, multiplataforma, con un foco especial en la seguridad y la criptografía.

25. **Oracle:** Relational Data Base Management System (RDBMS) es un sistema de gestión de base de datos relacional Oracle Corporation. Se considera a Oracle como uno de los sistemas de bases de datos más completos.
26. **PDF:** Portable Document Format (Formato de Documento Portátil), es un formato de almacenamiento de documentos, desarrollado por la empresa Adobe Systems.
27. **Personal Web Server:** El Personal Web Server (PWS), es un servidor web personal que permite montar un servidor web una PC. De esta manera, se puede correr aplicaciones web directamente sin necesidad de subir el sitio a un servidor online.
28. **Python:** Lenguaje de programación En la actualidad Python se desarrolla como un proyecto de código abierto.
29. **RDBMS:** Es un Sistema Administrador de Bases de Datos Relacionales.
30. **SMTP:** Simple Mail Transfer Protocol (Protocolo Simple de Transferencia de Correo). Protocolo de red basado en texto utilizado para el intercambio de mensajes de correo electrónico entre computadoras o distintos dispositivos (teléfonos móviles, etc.). Está definido como un estándar oficial de Internet.
31. **Software Libre:** (Free Software) es la denominación del software que brinda libertad a los usuarios sobre su producto adquirido y por tanto, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente.
32. **Software:** Es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación.
33. **Solaris:** Es un sistema operativo desarrollado por Sun Microsystems. Es un sistema certificado como una versión de UNIX y puede considerarse uno de los sistemas operativos más avanzados.
34. **Triggers:** Es un evento que se ejecuta cuando se cumple una condición establecida al realizar una operación de inserción (INSERT), actualización (UPDATE) o borrado (DELETE).
35. **Unix:** Es un sistema operativo portable, multitarea y multiusuario.
36. **URI:** Identificador de Recursos Uniforme, una cadena de caracteres compacta para identificar un recurso físico o abstracto. Los URIs que se usan en world-wide web se refieren normalmente como URLs.
37. **URL:** Uniform Resource Locator (Localizador Uniforme de Recursos), es un texto corto que identifica cualquier recurso (servicio, página, documento, dirección de correo, enciclopedia, etc.) accesible dentro de la red.

Referencias Bibliográficas

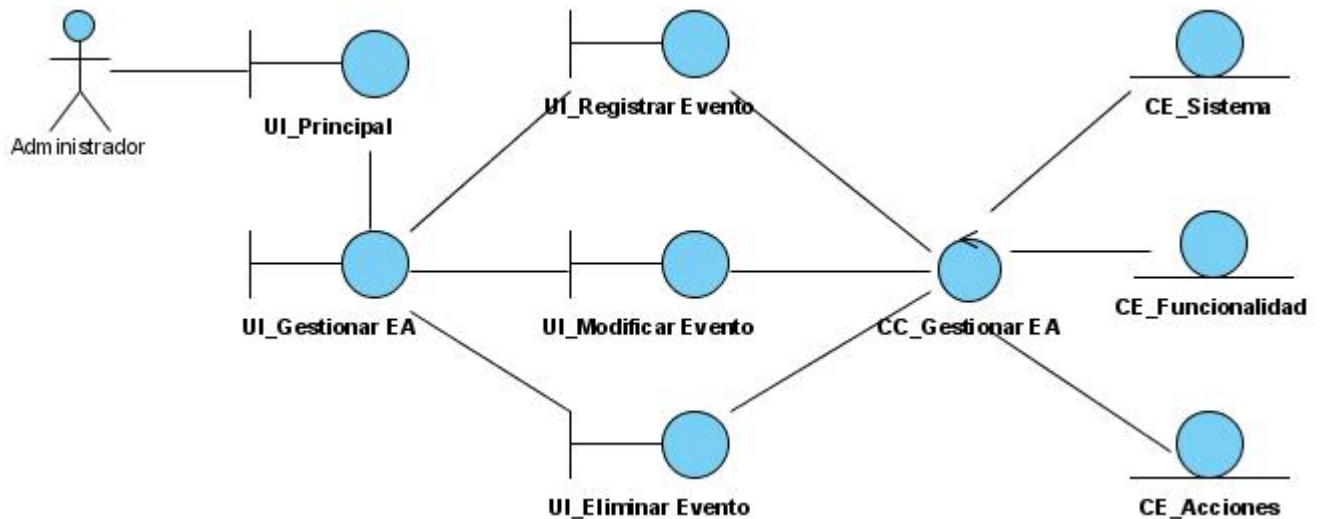
1. **DragonJAR, La Comunidad. 2006.** La Comunidad DragonJAR;. *La Comunidad DragonJAR*; [Online] 2006. [Cited: noviembre 25, 2007.] <http://www.dragonjar.us/httpbee-la-navaja-suiza-de-las-aplicaciones-web.xhtml>.
2. **ABCdatos. 1999.** ABCdatos, Programas y tutoriales que hablan tu idioma. *ABCdatos, Programas y tutoriales que hablan tu idioma.* [Online] 1999. [Cited: diciembre 4, 2007.] <http://www.abcdatos.com/webmasters/programa/z2492.html>.
3. **Booch, G., Rumbaugh, J, and Jacobson, I. 1999.** The Unified Modeling Language User Guide. 1999.
4. **Bullet, S. (1987).** "Essence and Accident in Software Engineering."
5. **DragonJAR, La Comunidad. 2006.** La Comunidad DragonJAR. *La Comunidad DragonJAR.* [Online] 2006. [Cited: noviembre 25, 2007.] <http://www.dragonjar.us/iguna-herramienta-gratuita-para-realizar-de-pruebas-de-penetracion.xhtml>.
6. **Duque Méndez, Néstor D. 2006.** Conceptos de Arquitectura Cliente/Servidor. *Conceptos de Arquitectura Cliente/Servidor.* [En línea] 27 de noviembre de 2006. [Citado el: 10 de mayo de 2008.] <http://www.virtual.unal.edu.co>.
7. **Flappo, P. G. and M. Botta. (1995).** "Planificación de Ingeniería de Requerimientos. Universidad Tecnológica Nacional Facultad Regional Santa Fe. ."
8. **Herrera, L. J.** "Ingeniería De Requerimientos, Ingeniería De Software." From <http://www.monografias.com/trabajos6/resof/resof.shtml>.
9. **Jacobson, I., G. Booch, et al. (2004).** El Proceso Unificado de Desarrollo de Software, Volumen I, Editorial Feliz Varela.
10. **Larman, Craig.** *UML y Patrones. 2da Edición. Cap. 30 Epig. 2. Patrón de Arquitectura: Capas.*
11. **Lavariega, J. C. and G. K. e. I. Sommerville (Agosto 2005)** "Requirements Engineering: Processes and Techniques." Volume, DOI:
12. **López, M. M. d. I. A. S. (Diciembre 1999).** "Análisis de Requerimientos de Software."
13. **Márquez, J. M.** "Introducción a la Ingeniería de Requisitos. Métodos y técnicas."

14. **Miguel Angel. 2006.** Manuales de Hosting LMI. *Manuales de Hosting LMI*. [Online] septiembre 2, 2006. [Cited: noviembre 30, 2007.] <http://manuales.hostinglmi.es/article.php?id=018>.
15. **Ríos Gil, J. J. 2005.** Departamento de Ingeniería Telemática Escuela Politécnica Superior Universidad Calos III de Madrid. *Departamento de Ingeniería Telemática Escuela Politécnica Superior Universidad Calos III de Madrid*. [En línea] 13 de octubre de 2005. [Citado el: 10 de mayo de 2008.] www.it.uc3m.es/docencia/si/material/1_cli-ser_mcfp.pdf. Trabajos citados
16. **Shklar, Leon and Rosen, Richard. 2003.** *Web Application Architecture, Principles, protocols and practices*. s.l. : Editorial Offices, 2003. ISBN 0-471-48656-6, [English].

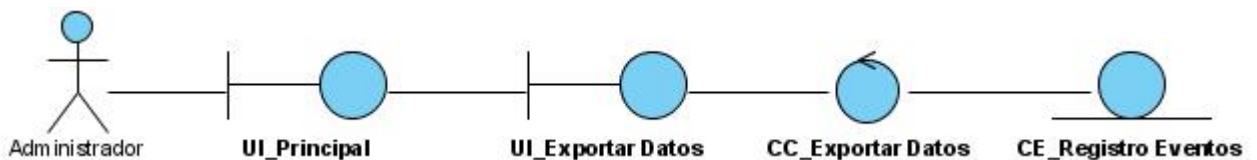
Bibliografía

1. **Booch, G., Rumbaugh, J, and Jacobson, I., 1999.** *The Unified Modeling Language User Guide*. Reading, MA.: Addison-Wesley.
2. **Hernández León, Rolando Alfredo and Coello González, Sayda. 2002.** *EL PARADIGMA CUANTITATIVO DE LA INVESTIGACIÓN CIENTÍFICA*. Ciudad de la Habana : Editorial Universitaria, 2002. ISBN: 959-16-0343-6.
3. **Manual Online de Doctrine PHP** (Idioma Inglés) en:
<http://www.phpdoctrine.org/documentation/manual> Visitado: abril 2008.
4. **Instituto Seguridad Internet. 2001.** Instituto Seguridad Internet. *Instituto Seguridad Internet*. [En línea] 2001. [Citado el: 27 de noviembre de 2007.]
<http://www.instisec.com/publico/auditoria.asp?id=2>.

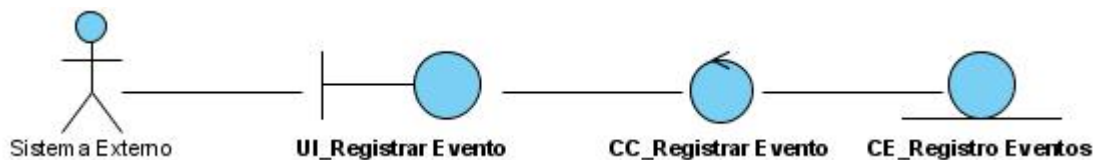
Anexo 1 Diagramas de clases del análisis



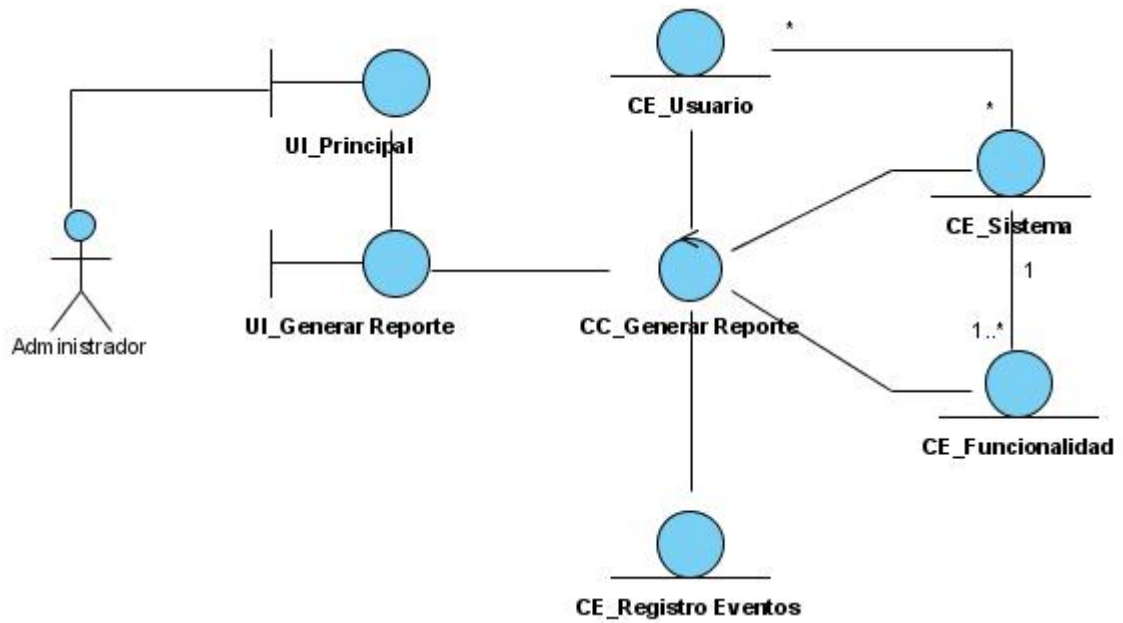
Anexo 1.1 Diagrama de clases del análisis del CUS Gestionar Eventos a Auditar



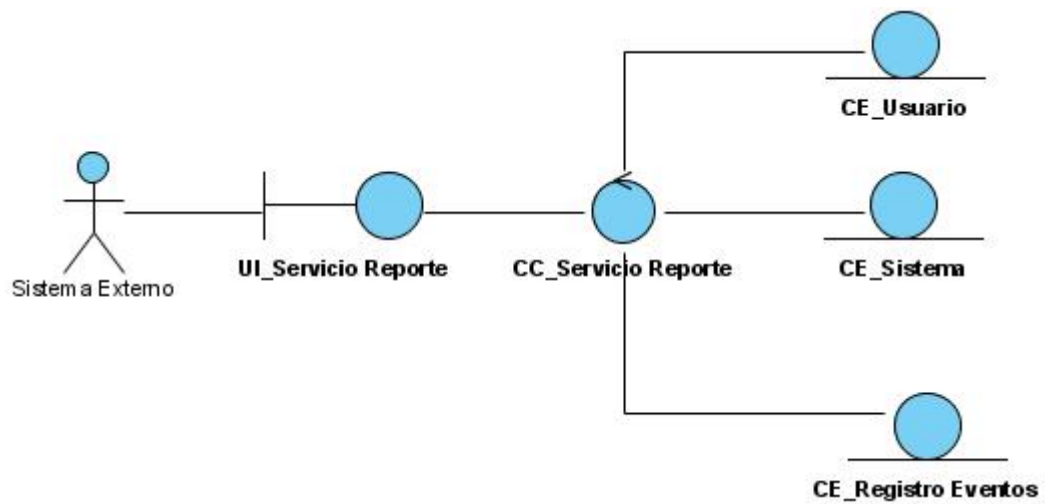
Anexo 1.2 Diagrama de clases del análisis del CUS Exportar Datos



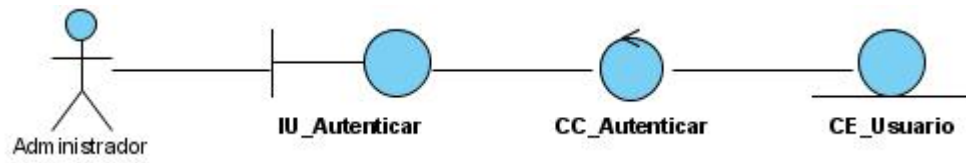
Anexo 1.3 Diagrama de clases del análisis del CUS Registrar Eventos



Anexo 1.4 Diagrama de clases del análisis del CUS Generar Reporte

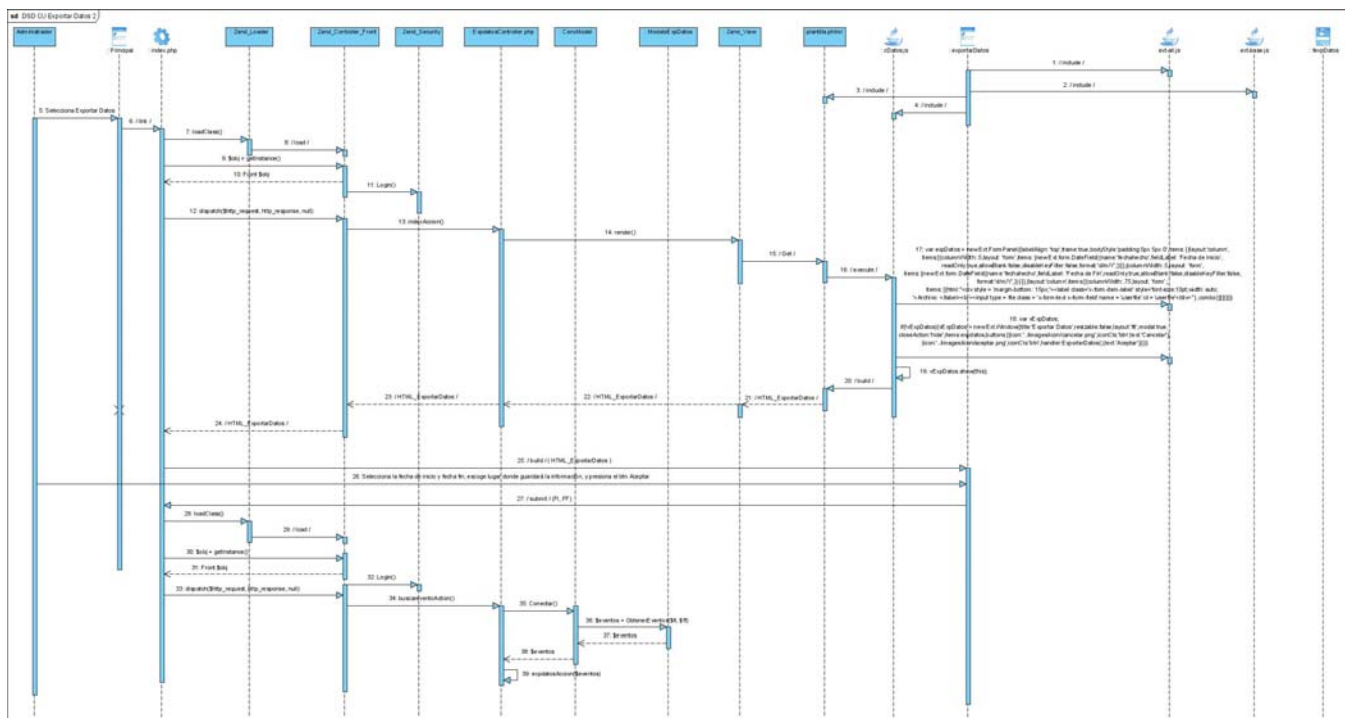


Anexo 1.5 Diagrama de clases del análisis del CUS Gestionar Servicio de Reporte

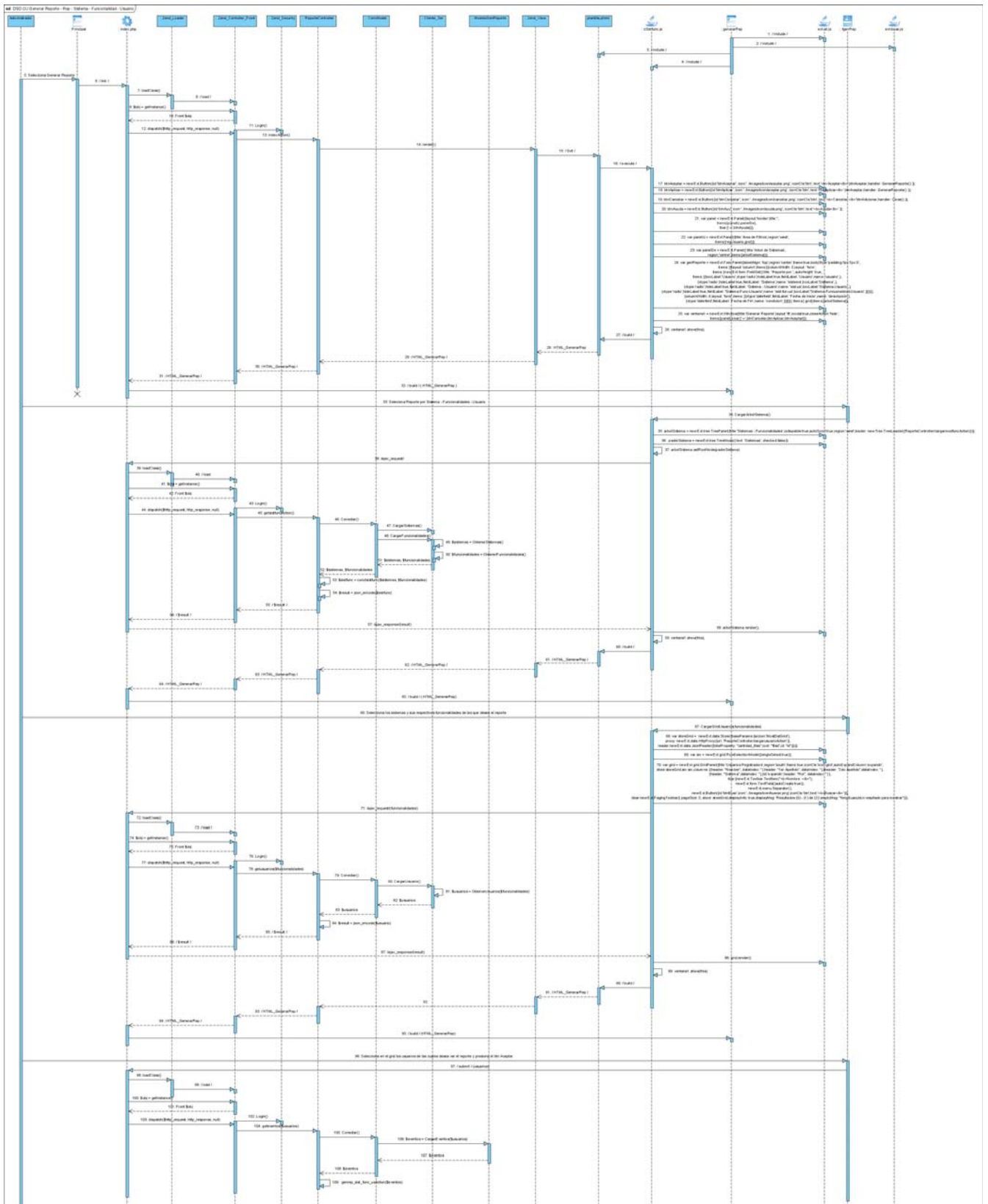


Anexo 1.6 Diagrama de clases del análisis del CUS Validar Usuario

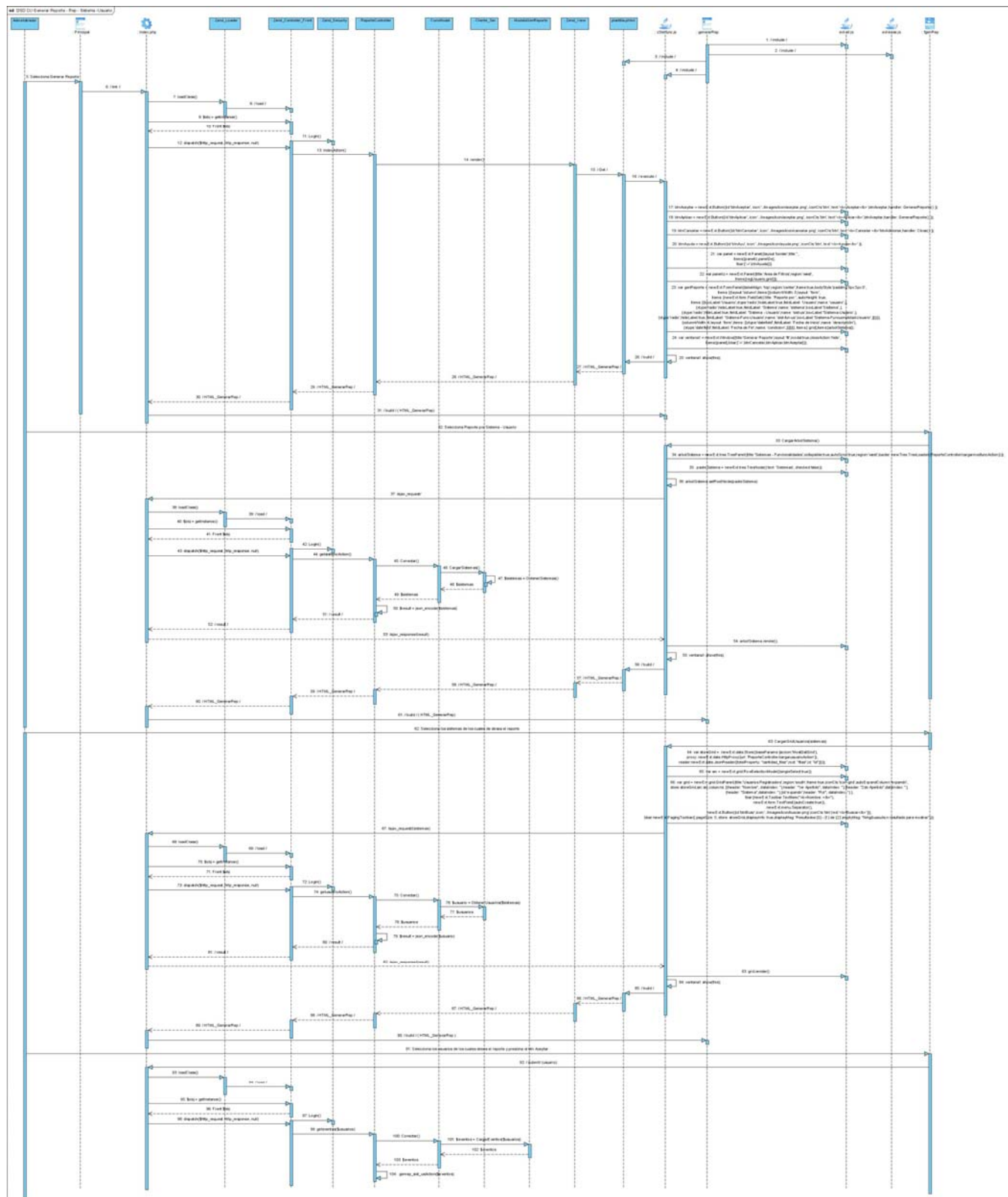
Anexo 2 Diagramas de Secuencia de las Clases del Diseño



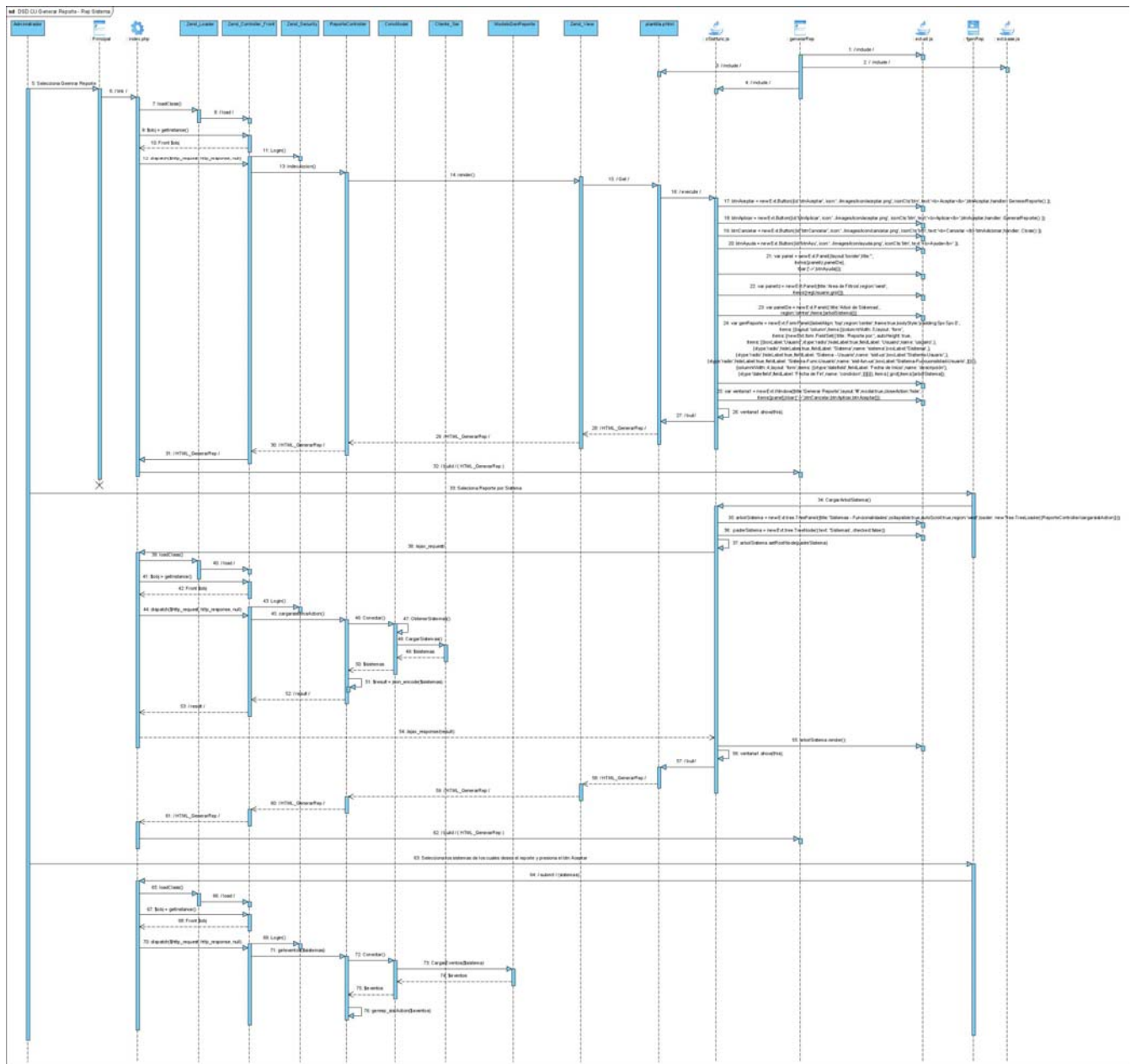
Anexo 2.1 Diagrama SCD del CU Exportar Datos



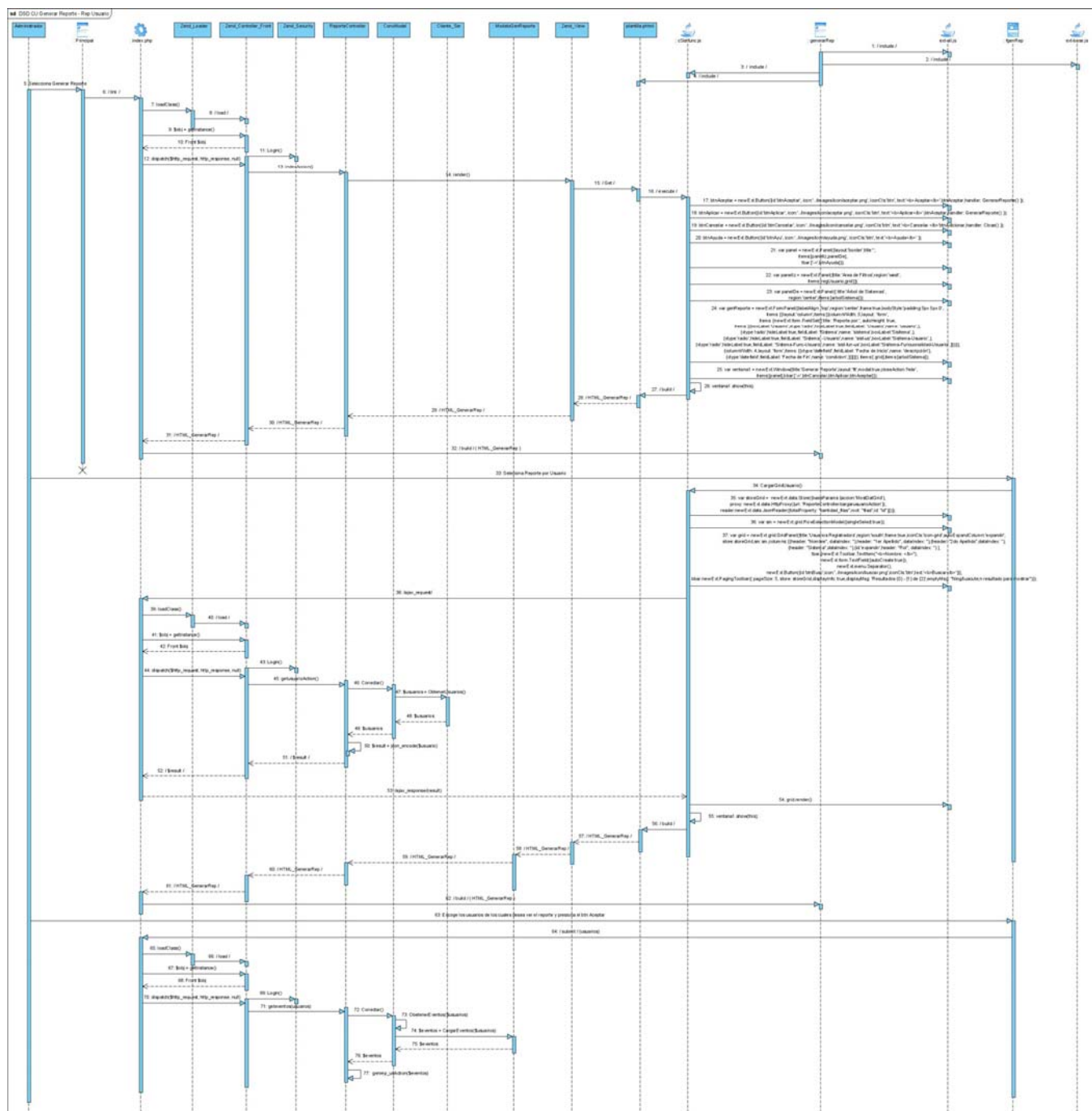
Anexo 2.2 Diagrama SCD del CU Generar Reporte por Sistema – Funcionalidades – Usuario



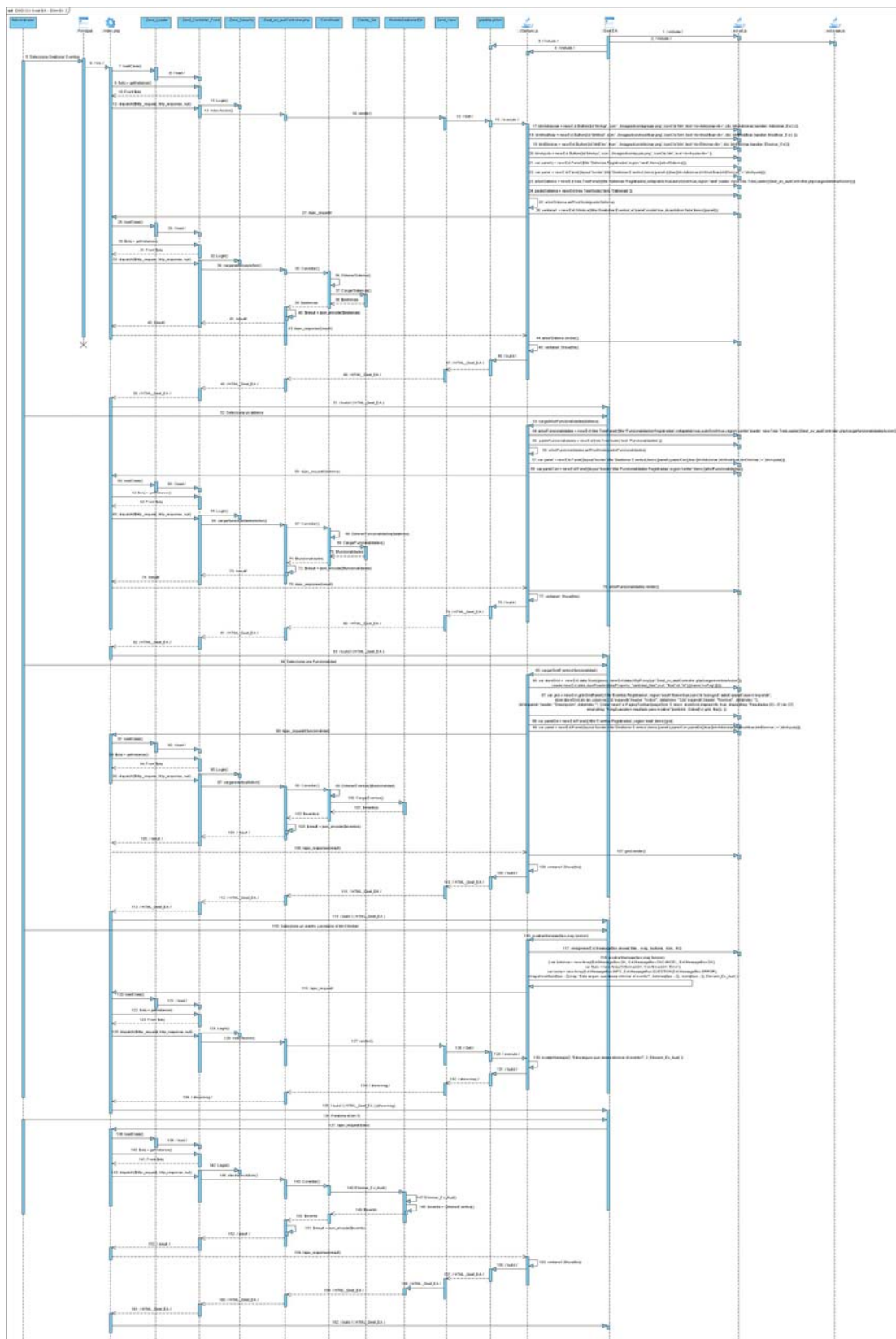
Anexo 2.3 Diagrama SCD del CU Generar Reporte por Sistema – Usuario



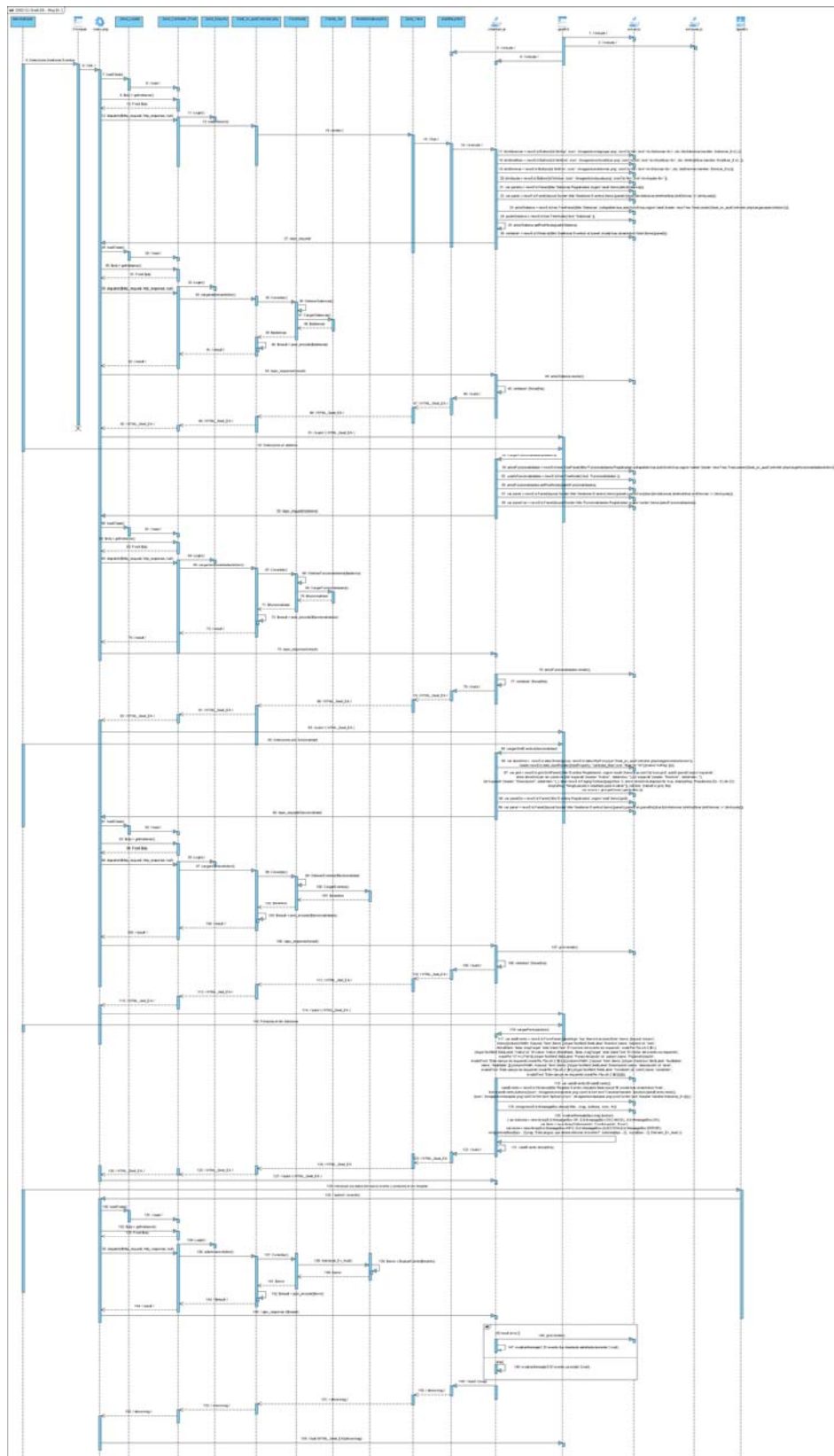
Anexo 2.4 Diagrama SCD del CU Generar Reporte por Sistema



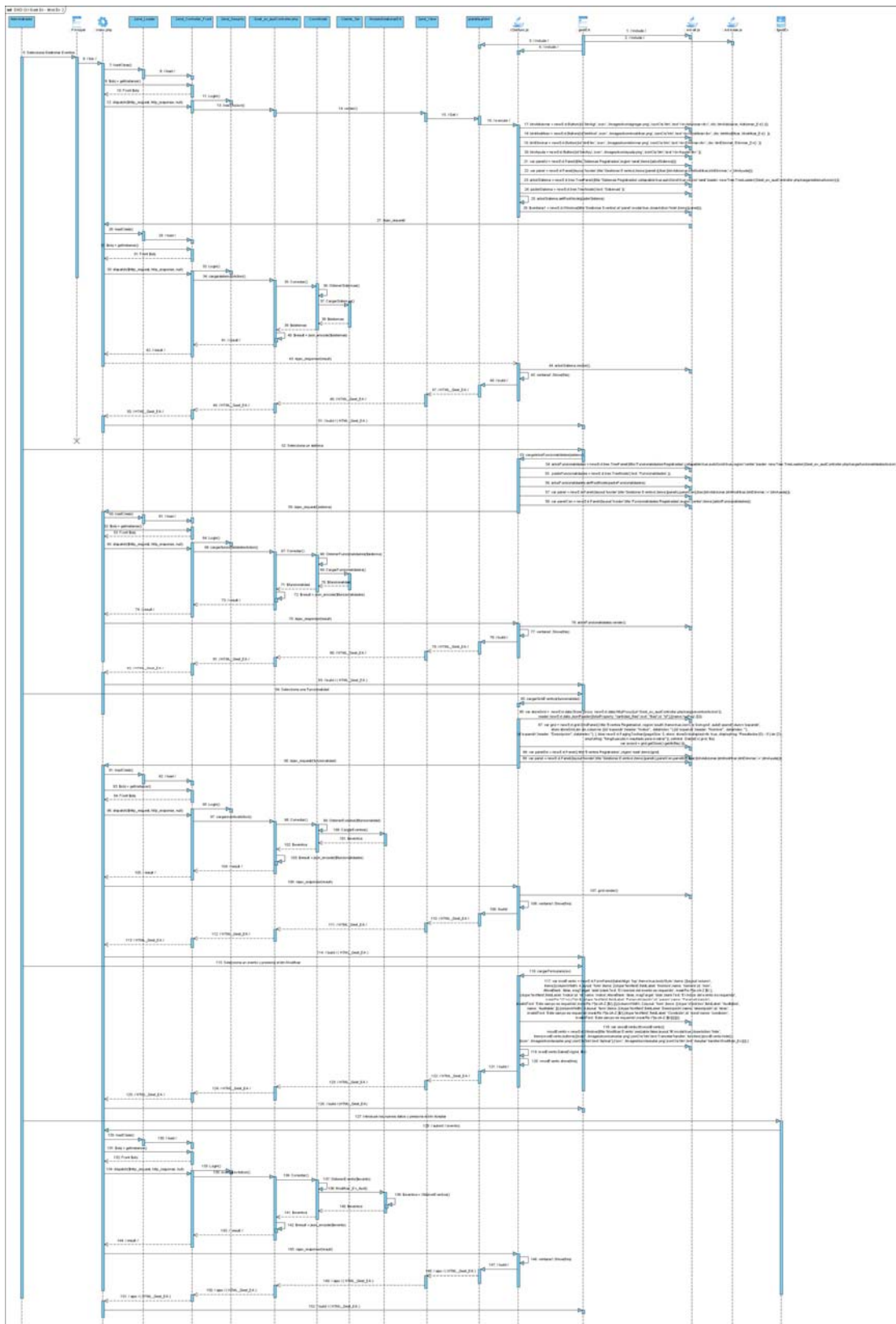
Anexo 2.5 Diagrama SCD del CU Generar Reporte por Usuario



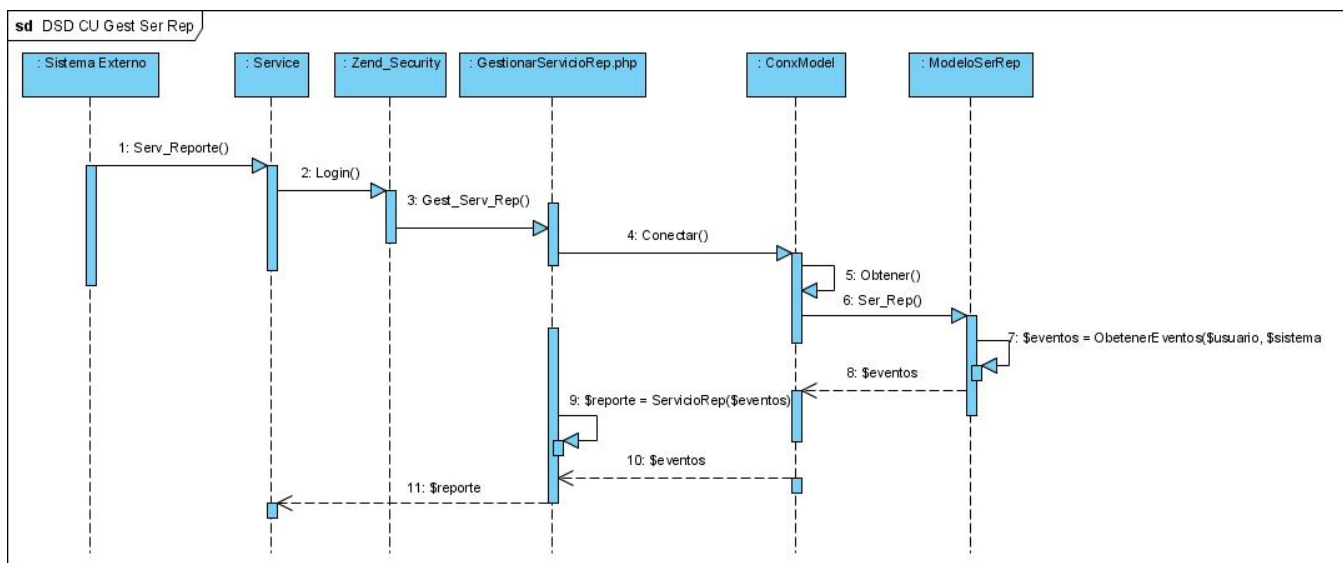
Anexo 2.6 Diagrama SCD del CU Gestionar Eventos a Auditar - Escenario Eliminar Evento.



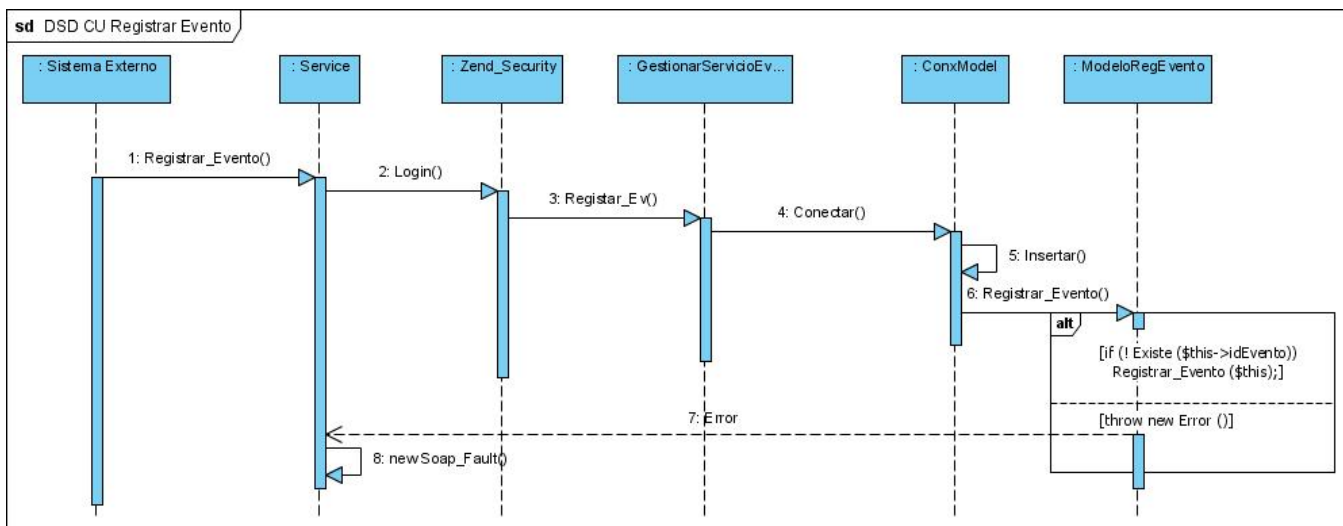
Anexo 2.7 Diagrama SCD del CU Gestionar Eventos a Auditar - Escenario Registrar Evento.



Anexo 2.8 Diagrama SCD del CU Gestionar Eventos a Auditar - Escenario Modificar Evento.



Anexo 2.9 Diagrama SCD del CU Gestionar Servicio de Reporte.



Anexo 2.10 Diagrama SCD del CU Registrar Evento.