Universidad de las Ciencias Informáticas

Facultad 4

Herramienta para la auditoría y control externo de la Plataforma Educativa Zera.

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

Autor:

René Antonio Salomón Díaz

Tutora:

Lic. Teresa González Cruz

Cotutor:

Ing. Daniel Rodríguez Soberats

La Habana 2012

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año_____.

_____

Firma del Autor

René Antonio Salomón Díaz

_____ _____

Firma del Tutor Firma del Co-tutor

Teresa González Cruz Daniel Rodríguez Soberats

Desde que nacemos hay personas que siempre están ahí para apoyar, educar y guiar durante nuestro paso por la vida. Las mismas personas que quieren lo mejor para nosotros, de las cuales aprendemos y escuchamos, las que no quieren que nos equivoquemos y que hagamos las cosas lo mejor que podamos e incluso mucho mejor. Si hay personas a las que hay que agradecer, esas personas son mis padres Rosalía y René que junto con la vida me han dado su corazón, amor y cariño.

A mis otros padre María y Sarbelio que han guiado mi paso por la vida y siempre han confiado en mí.

A mis abuelos Rafaela y Tony por su apoyo y confianza, así como a Aracelis y René, Gloria y Armando que hoy no están conmigo en vida pero siempre lo van a estar en alma.

A mis tíos Miguel y Clarita por sus consejos apoyo y cariño que han sido otros padres para mí y han participado en cada momento de mi formación universitaria.

A mis tíos Hilda y José los cuales han sido maravillosos conmigo, dándome siempre su voto de confianza.

A mis tíos Mandy y Rosandra los que quiero mucho.

A mis hermanos de corazón Laritza, Liudmila y Sarbelio, los cuales siempre han confiado en mí.

A mis primos Jorgito, Marquito, Abrancito y la pequeñita Daniela.

A mis primos Miguelitico, Raudel, Sondra y Whitney.

A la familia Mulén Tito y en especial al abuelito Jonás.

A mis tutores Teresa y Daniel por su apoyo y seguimiento ofrecido, al ser incondicionales.

Al profesor Edgar por su ayuda y tutoría oportuna.

A mis amigos y compañeros del grupo 8106 y en especial a Ruben y Laurien, amigos incondicionales. Y en los nuevos grupos formados a todo lo largo de la carrera amigos como Ernesto y Yuleisis por su apoyo, guía y preocupaciones.

A mis amigos de Alfaomega en especial a Yerandy, Adrian, Miguel y Yoandy.

A todos los profesores que han sido parte de mi formación a todo lo largo de la carrera, mis respeto y admiración a cada uno de ellos.

A la profesora Mayi por su apoyo en cada momento en que lo necesité.

A la Profesora Maritza, siempre humana y atenta de los reclamos pertinentes.

A todos los que han colaborado y confiado en todos mis pasos.

A todas las personas que han apoyado mi formación como hijo, hombre y amigo siempre les profesaré gratitud.

Dedico este resultado con el mayor cariño:

A mis padres que han guiado mi empeño y a todos mis familiares

Resumen

La investigación presentada, responde a la necesidad de elaborar una herramienta informática que le permita a la empresa Albet S.A, realizar la auditoría externa de la Plataforma Educativa Zera.

En el estudio de ingeniería se abordan conceptos como la importancia de la auditoría en el proceso de desarrollo de software, se valoran aspectos teóricos que sustentan el análisis del estado del arte de los principales software de auditoría, los criterios para la selección de las herramientas y tecnologías para aplicar la auditoría informática a la Plataforma Educativa Zera. Estudio de conceptos asociados al negocio, requisitos funcionales y no funcionales, descripción de los casos de uso del sistema, análisis, implementación y las pruebas las cuales resultan la garantía de la terminación del software.

Con el propósito de auditar externamente la Plataforma Educativa Zera mediante la automatización de los procesos se utilizan servicios web, RUP como metodología de desarrollo teniendo en cuenta una modelación del negocio. Se enfatiza en el análisis y diseño del sistema a desarrollar, a través del modelado de los artefactos necesarios que contribuyan a la implementación. Son estimados en ilustraciones diagramas de clases en todo el ciclo de desarrollo de software, estándares y principios valorados para diseñar la interfaz de la aplicación, como artefactos que conforman la propuesta de solución.

ÍNDICE DE CONTENIDO

INTRODUCCIÓN *1*

CAPÍTULO 1 ..... *7*

FUNDAMENTACIÓN TEÓRICA *7*

Introducción *7*

1.1 Surgimiento y Evolución de la Auditoría **7**

1.2 Definición de Auditoría **8**

1.3 Importancia de la Auditoría **10**

1.4 Tipos de Auditoría **11**

1.5 Clases de Auditoría **12**

1.6 Técnicas de auditoría más utilizadas **13**

1.7 Estudios de herramientas similares **15**

1.7.1 ACL 15
1.7.2 IDEA 16
1.7.3 Versat Sarasola 17
1.7.4 Criterio de selección sobre herramientas similares 17

1.8 Tecnologías de acceso a la información **18**

1.8.1 Servicios web 18
1.8.2 WSDL 20
1.8.3 *XML* 20
1.8.4 *SOAP* 20
1.8.5 *Criterio de selección de la tecnología de acceso a la información.* 21

1.9 Metodologías para el desarrollo de software **21**

1.9.1 *OpenUP* 22
1.9.2 *Proceso Unificado de Software* 22
1.9.3 *Extreme Programing* 24
1.9.4 *Selección de la metodología de software* 25

1.10 Herramienta para el modelado **26**

1.10.1 *Lenguaje de Modelado Unificado (UML)* 26
1.10.2 *Día Project* 27
1.10.3 *Umbrello* 27
1.10.4 *Visual Paradigm* 28
1.10.5 *Selección de la herramienta para el modelado* 29

1.11 Framework **29**

1.11.1 *Zend Framework* 29
1.11.2 *Symfony* 30
1.11.3 *Selección del Framework* 31

1.12 Lenguaje de desarrollo 31
1.12.1 Tecnologías del lado del cliente 32
1.12.2 Tecnología del lado del servidor 33
1.12.3 Selección del lenguaje de programación 35

1.13 Sistema Gestor de Base de Datos 35
1.13.1 SQLite 36
1.13.2 MySQL 36
1.13.3 PostgreSQL 38
1.13.4 Selección de la Base de Datos 39

1.14 Servidor Web Apache 39

1.15 IDEs (Entorno de desarrollo integrado) 40
1.15.1 NetBeans 6.9 40
1.15.2 Eclipse 41
1.15.3 Selección del IDE de desarrollo 42

1.16 Conclusiones Parciales 42

CAPÍTULO 2 43

CARACTERÍSTICAS DEL SISTEMA 43

Introducción 43

2.1 Descripción del Sistema Propuesto. 43

2.2 Modelo de Dominio 44

2.3 Requisitos del software 46
2.1.1 *Requisitos Funcionales 46*
2.1.2 Requisitos no funcionales del sistema 49

2.2 Modelo de Casos de Uso del Sistema. 51
2.2.1 Descripción de los actores del sistema. 51

2.2.2 Patrones de Casos de Uso (CU) 52

2.2.3 Diagrama de Casos de Uso del Sistema 53

2.2.4 Descripción de los Casos de Uso del Sistema 53

2.3 Conclusiones Parciales 64

CAPÍTULO 3 65

ANÁLISIS Y DISEÑO DEL SISTEMA 65

Introducción 65

3.1 Modelo de análisis 65
3.1.1 Diagramas de clases 66
3.1.2 Diagramas de clases del análisis 66

3.2 Patrón arquitectónico Modelo – Vista – Controlador en "Symfony" **67**
3.3 Aplicación de los patrones de diseño en Symfony **69**
3.4 Modelo de diseño **69**
3.4.1 Diagrama de clase del diseño. 71
3.5 Diseño de la base de datos **71**
3.6 Descripción de las tablas de la base de datos. **72**
3.7 Conclusiones parciales. **74**
CAPÍTULO 4 ..... ***75***
Implementación y Prueba ***75***
4 Introducción ***75***
4.1 Implementación **75**
4.2 Vista de Despliegue **76**
4.3 Diagrama de Componentes **76**
4.4 Pruebas de Software **77**
4.4.1 Métodos de Prueba 79
4.4.2 Diseño de Casos de Prueba 79
4.5 Resultados Obtenidos **83**
Conclusiones generales ***86***
Recomendaciones **87**
Referencias Bibliográficas ***88***
Bibliografía ***92***
GLOSARIO DE TÉRMINOS ***98***
Anexos ***99***

ÍNDICE DE FIGURAS

Figura 1: Diagrama de la Metodología RUP..... 23

Figura 2: Representación del Modelo de Dominio..... 45

Figura 3: Diagrama de Casos de Uso del Sistema..... 53

Figura 4: DCA_CU Gestionar Usuario..... 66

Figura 5: DCA_ CU Incluir Contrato..... 67

Figura 6: DCA CU Modificar Contrato. 67

Figura 7: DCD_CU Gestionar Usuario..... 71

Figura 8: Diseño de la base de datos. 72

Figura 9: Diagrama de Despliegue..... 76

Figura 10: Diagrama de Componentes. 77

ÍNDICE DE TABLA

Tabla 2: Descripción de los actores del sistema. 52

Tabla 3: Descripción del CU Gestionar Usuario. 58

Tabla 4: Descripción del CU Incluir Contrato. 61

Tabla 5: Descripción del CU Modificar Contrato..... 63

Tabla 6: Descripción de la tabla Usuario..... 73

Tabla 7: Descripción de la tabla tb_agreement..... 73

Tabla 8: Diseño de Caso de Prueba Gestionar Contrato. 83

Tabla 9: Pruebas internas..... 84

Tabla 10: Pruebas cruzadas..... 84

INTRODUCCIÓN

En la actualidad existe una gran competencia en el mercado del software, incidiendo en el constante desarrollo de las “Tecnologías de la Información y las Comunicaciones” (TIC). El uso de las TIC en la enseñanza en diferentes niveles estimula el conocimiento, ofrece la oportunidad de desarrollar capacidades intelectuales y creativas; de este modo la informática contribuye a configurar una visión del mundo actual e influye en el comportamiento. La sociedad que ha irrumpido desde las tres últimas décadas del siglo XX hasta la actualidad, ha acelerado las transformaciones tecnológicas y su aplicación en diferentes esferas de la vida, apreciándose cambios que repercuten en las condiciones materiales e intelectuales. En este acelerado desarrollo hacia la superación, tienen grandes influencias los procesos de auditoría y control informático de los medios y procedimientos que se siguen en una empresa.

La auditoría se torna muy útil en la dinámica de la sociedad contemporánea. El perfeccionamiento de las técnicas, la aplicación de tecnologías y el desarrollo de los medios de información permiten realizar una mejor gestión, administración y toma de decisiones para lograr un adecuado control interno en la institución donde se realiza la auditoría, de modo que implique fiabilidad y seguridad. La auditoría significa control, organización, revisión y enjuiciamiento de la calidad de los sistemas computarizados para lograr eficiencia y eficacia organizativa, evidenciándose en la práctica en la gestión de la información. El control interno de la función informática se ejecuta a partir de herramientas favorables y procedimientos factibles, como por ejemplo de software de auditoría se encuentran ACL e IDEA y los procedimientos como la división de la misma en clases, aspectos argumentados en los epígrafes 1.4 y 1.8. En la auditoría la informática contribuye a la toma de decisiones, pero su aplicación es importante en diversas esferas, especialmente en los mecanismos económicos.

El Decreto Ley No. 159 de la Auditoría de Cuba, define la auditoría como un proceso sistemático, que consiste en obtener y evaluar objetivamente evidencias sobre las afirmaciones relativas a los actos o eventos de carácter económico – administrativo, con el fin de determinar el grado de

correspondencia entre esas afirmaciones y los criterios establecidos, para luego comunicar los resultados a las personas interesadas. (1)

En Cuba, la empresa Albet Ingeniería y Sistemas, S.A, conocida como Albet, S.A. es una de las encargadas de la comercialización de software. Entre los productos que comercializa se encuentran los desarrollados por la "Universidad de Ciencias Informáticas" (UCI). A través de Albet S.A. la empresa mexicana Alfaomega realizó un contrato para elaborar un software educativo, y así surge como proyecto el desarrollo de una Plataforma Educativa llamada Zera.

El objetivo principal del negocio realizado por Albet S.A. y Alfaomega tiene sus fundamentos en la confección de la Plataforma Educativa Zera bajo la integración de "Hiperentornos de Aprendizaje" (HEA) que está conformado por contenidos educativos tales como: Matemática, Física, Química y Biología. Estos están basados en tecnología hipermedia[1] desarrolladas por pedagogos y especialistas del Ministerio de Educación de Cuba (MINED) y reflejadas en una serie llamada Colección Futuro con el propósito de atender las necesidades didácticas de estudiantes de preuniversitario. La confección de la Plataforma Educativa Zera le permitirá a la empresa Alfaomega comercializar la Colección Futuro.

La parte cubana como propietaria de Zera representada por Albet S.A. recibirá un por ciento de ganancia sobre las ventas llamadas regalías[2] por la explotación comercial de los productos mencionados. La comercialización se realizará a través de un contrato entre Alfaomega y un cliente final (Escuela que solicite el uso de la Plataforma Educativa Zera).

La parte mexicana representada por Alfaomega estará obligada a presentar el contrato realizado con terceros donde esté escrito legalmente el pago de una escuela.

Un aspecto primordial dentro de dicha plataforma lo constituye el hecho de garantizar la entrada de dinero al país, de manera que se puedan prevenir fraudes internos. Actualmente no se tiene un control informático sobre los contratos entre Alfaomega y una escuela, debido a esto, el

[1] Resultado de la combinación de hipertexto y multimedia, en la cual se incluye texto, imagen, video, entre otros.
[2] Pago que Albet S.A. recibirá por la explotación comercial de Zera.

personal encargado de recibir las regalías ha detectado varios problemas en cuanto al control del proceso administrativo de Zera, que son puntualizados a continuación:

- No se tiene una auditoría externa que permita revisar la administración de la Plataforma Educativa Zera.
- Carece de reportes sobre las ventas de programa de estudio y las licencias asociadas a este.
- Carece de reportes que muestren las ganancias de la plataforma sobre el total de ventas de los programa de estudio.
- Ausencia de control sobre las ganancias por los diferentes tipos de licencias y por programas de estudios.

Por lo antes mencionado se hace necesario encontrar una solución que permita preservar las finanzas del país, así como llevar el control del proceso económico administrativo de la Plataforma Educativa Zera.

Basado en la problemática planteada se formula el **problema a resolver** en la siguiente interrogante:

¿Cómo informatizar la auditoría y control externo de la Plataforma Educativa Zera?

Definiendo el **objeto de estudio** como**,** “Los procesos de auditorías en sistemas informáticos”. El **campo de acción** de la investigación es “El proceso de auditoría externa de la Plataforma Educativa Zera”.

Siendo el **Objetivo general,** “Desarrollar un software que le permita a Albet S.A. aplicar la auditoría externa a la Plataforma Educativa Zera”. La **Idea a Defender** está sustentada en, “Mediante una aplicación que permita la auditoría externa, se controlaran las finanzas de la Plataforma Educativa Zera”

A partir del objetivo general definido, se derivan los siguientes **objetivos específicos**:

- Realizar un análisis del estado del arte de los sistemas informáticos que realizan auditorías.
- Desarrollar el análisis y diseño del sistema propuesto.
- Implementar el sistema, de acuerdo con la estructura de diseño definida.
- Validar el software desarrollado.

Al finalizar el presente trabajo, se espera como posibles resultados un software que permita auditar externamente la Plataforma Educativa Zera y facilite:

- Obtener una cuantía exacta de las ganancias con el fin de garantizar la economía de la Plataforma Educativa Zera.
- Prevenir fraudes internos.
- Obtener la documentación de todo el proceso de investigación y desarrollo que tribute a posteriores versiones del software desarrollado.

Para dar cumplimiento a los objetivos específicos propuestos, se ejecutan las siguientes **tareas de investigación**:

- Caracterización de las diferentes herramientas de auditorías en el mundo.
- Selección de los elementos teóricos y conceptuales a tener en cuenta en la elaboración de la propuesta.
- Identificación de los requerimientos funcionales y no funcionales para desarrollar el software propuesto.
- Elaboración del modelo de caso de uso del sistema.
- Confección del diagrama de clases de análisis y diseño para cada caso de uso.
- Confección del diagrama de componentes del sistema.
- Implementación de los diagramas de clases del análisis y el diseño.

- Aplicación de las pruebas funcionales al sistema.

Métodos de investigación utilizados:

Los métodos científicos que fueron útiles en el desempeño de las tareas científicas se muestran a continuación:

- El método **Histórico y Lógico.** En la presente investigación se utiliza este método para realizar el estudio del estado del arte, o sea, para investigar acerca de otras aplicaciones o soluciones similares y de los lenguajes y metodologías de desarrollo de software existentes, así como los *frameworks* (marcos de trabajo) y herramientas de desarrollo; describir la metodología, herramientas y lenguajes a utilizar en el análisis, diseño e implementación del software de auditoría propuesto.
- El método **Analítico-Sintético**. Se utilizó para el análisis de la teoría y extracción de los principales conceptos a incluir en el marco teórico, además para el estudio y análisis de la información, lo que permitió obtener los elementos principales relacionados con la auditoría en la Plataforma Educativa ZERA.

El contenido del trabajo presenta la siguiente **Estructura Capitular:**

Capítulo 1: Fundamentación Teórica.

Fundamentación del marco teórico y referencial de la investigación donde se realiza un análisis de la evolución de la auditoría para comprender cómo con el desarrollo tecnológico varía la auditoría informática; se exponen las definiciones de términos como bases teóricas y lógicas importantes con las cuales opera la investigación. Realizándose un estudio de los siguientes aspectos:

1- Principales conceptos y valoraciones de auditoría, así como un análisis del estado del arte.

2- Particularidades de la auditoría informática.

3- Evaluación de las principales herramientas y técnicas para llevar a cabo el desarrollo de una auditoría informática.

Capítulo 2: Características del Sistema.

En su contenido se realiza una descripción de las características del sistema. Se obtiene el modelo de dominio, se realiza levantamiento de requisitos, el diagrama de casos de uso del sistema y como artefacto fundamental las descripciones de los mismos.

Capítulo 3: Análisis y diseño del Sistema.

En este capítulo se realiza el análisis y diseño del sistema, así como los diagramas de clases web, que brindan una visión de cómo debe estructurarse el producto una vez terminada la investigación. Además, se realiza el modelo de datos y la descripción de cada una de sus tablas.

Capítulo 4: Implementación y Pruebas.

Los aspectos del desarrollo son evaluados en este capítulo. Describe cómo está implementado el sistema, definiéndose, los tipos de pruebas y los casos de pruebas que se le realizarán al software.

1

CAPÍTULO 1

FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capítulo se realiza un estudio sobre el estado del arte de la auditoría informática. Se exponen aquellos conceptos que permitan la comprensión de la investigación, realizándose un análisis de los conceptos asociados al dominio del problema, así como la descripción del objeto de estudio como precedente teórico importante. Se desarrollará también el análisis y descripción de las distintas herramientas y tecnologías con el objetivo de seleccionar aquellas que puedan brindar mejores resultados en el desarrollo de la investigación.

1.1 Surgimiento y Evolución de la Auditoría

Para emprender el estudio de la auditoría en la informática es oportuno explicar sus orígenes como comprensión histórica y lógica de su desenvolvimiento. En la investigación realizada se han consultado diversas fuentes de información como libros y materiales digitalizados, como resultado de investigaciones acerca del tema que aportan datos interesantes para el estudio planteado.

El término “auditoría” fue asimilado al idioma castellano luego de su amplia y profunda difusión en el idioma inglés, deriva etimológicamente del latín “audire” que significa oír. Se ha estimado que la auditoría aparece desde los tiempos remotos en que comenzó a desarrollarse el comercio, resultado de la necesidad de registrar y controlar las acciones en este sentido; se atribuye el

término auditoría a la acción de controlar, examinar y verificar si una información u acción es confiable, veraz y oportuna.

En Cuba a partir de 1902 con el predominio del capital norteamericano, surge la necesidad de Auditores y contadores de alta calificación, los cuales provenían del extranjero, ya que la gran mayoría de las grandes empresas organizadas en Cuba eran sucursales de Compañías norteamericanas.

En 1927 aparecieron las primeras firmas privadas de auditores cubanos y surgieron instituciones públicas como tribunal de cuentas, también el Ministerio de Haciendas y el Banco Nacional de Cuba que se dedicaban a realizar auditorías de balances o financieras tanto privadas como públicas. En los últimos años y con la aparición de las Empresas privadas y mixtas se ha ampliado el campo de acción de los auditores. Actualmente existe el Ministerio de Auditoría y Control de la República de Cuba, creado el 25 de Abril del 2001, como un Organismo de la Administración Central del Estado encargado de dirigir, ejecutar y controlar la aplicación de la política del estado y del gobierno en materia de auditoría gubernamental, Fiscalización y Control gubernamental; así como para regular, organizar, dirigir y controlar, metodológicamente, el Sistema Nacional de Auditoría. (2)

1.2 Definición de Auditoría

En la investigación fueron analizadas diversas valoraciones acerca de la auditoría, las cuales declaran perspectivas específicas. Se han tenido en cuenta para la realización del trabajo algunas de las más relevantes debido a su integridad, las mismas se encuentran a continuación.

La IEEE, afirma que la auditoría es “un examen independiente de un producto de software, proceso del software, o sistema de procesos del software para determinar conformidad con especificaciones, estándares, acuerdos contractuales, u otros criterios”. (3)

La auditoría puede definirse como una técnica de control, dirigida a valorar, el control interno y la observancia de las Normas Generales de Contabilidad. Comprende un examen independiente de

los registros de contabilidad y otra evidencia relacionada con una entidad para apoyar la opinión experta imparcial sobre la confiabilidad de los estados financieros. (4)

Otro enfoque declara a la auditoría, como la acción de verificar si la información financiera, operacional y administrativa que se presenta, es confiable, veraz y oportuna. Es revisar que los hechos, fenómenos y operaciones se den en la forma en que fueron planeados, que las políticas y lineamientos establecidos han sido observados y respetados; también que se cumplan con obligaciones fiscales, jurídicas y reglamentarias en general. Es evaluar la forma de cómo se administra y opera teniendo al máximo el aprovechamiento de los recursos. (5)

Acerca del estudio de la auditoría en relación con los procesos informáticos, existen diversos criterios de especialistas, cuyos modos de interpretación no siempre coinciden. Estos contrastes de ideas tienen relación con el desempeño del auditor (por ejemplo en la gestión financiera, el dilema de auditar sistemas informatizados, entre otras acciones).

Como antecedente, la comprensión acerca de la auditoría infiere un análisis de lo general a lo particular en su evolución histórica. Se ha concebido la auditoría como mecanismo importante de examen y control de los estados financieros, en lo cual el auditor expresa su punto de vista.

La asociación americana de auditoría la declara como “proceso sistemático para obtener y evaluar de manera objetiva las evidencias relacionadas con informes sobre actividades económicas y otros acontecimientos” (6)

Según las diversas concepciones hay puntos convergentes respecto a entender la auditoría como examen de los registros contables y la determinación de la situación financiera por el auditor calificado, que informa las evidencias entre los hechos reflejados y los estados de las finanzas con el rigor necesario. Estas tienen un carácter interno y externo.

La auditoría en su devenir histórico ha alcanzado otras dimensiones más allá de los procesos económicos trascendiendo a la actuación social, aplicándose a otras ramas del conocimiento, tales son la medicina, los estudios del medio ambiente, procesos constructivos, la informática, entre otros.

El Comité Internacional de prácticas de auditoría concibió su aplicabilidad mediante el uso de computadoras, para el procesamiento de información financiera. Otras investigaciones asumieron la auditoría de programas para examinar la eficiencia técnica, seguridad y optimización con recursos informáticos.

Para la valoración teórica ha sido útil el estudio de las definiciones anteriores, fuente de información que ofrece argumentos que permiten enunciar el criterio de esta investigación:

La "Auditoría" es la ciencia que se encarga de evaluar, revisar y controlar si los métodos y procedimientos que se siguen en una empresa están correctamente definidos y aplicados, si aseguran el cumplimiento de planes, leyes, políticas y reglamentos de manera que la organización los esté cumpliendo adecuadamente.

1.3 Importancia de la Auditoría.

El desarrollo tecnológico en la modernidad ha condicionado el desempeño de la sociedad en el dominio de diversas esferas de la vida. La informatización invade las comunicaciones, los medios de información, los diseños y controles de industrias, bancos, transporte aéreo, terrestre, hospitales, la enseñanza en todos los niveles; todo ello resultado de su constante perfeccionamiento, lo cual demanda de controles racionales y eficientes, la protección y seguridad de las informaciones.

Debido a la complejidad y variedad que adquiere la informática a través de redes de información, libros, cuentas, expresiones artísticas, en un mundo cada vez más conectado, si bien es resultado del talento e ingenio humano que dinamiza la información, medios de control y organización de diversos procesos; también suceden fraudes, delitos informáticos, virus dañinos que destruyendo lo logrado. (7)

Para contrarrestar el impacto y riesgos en la informática se elaboran métodos, procedimientos que resguardan las redes mediante un sistema de aseguramiento desde la óptica teórica y práctica. La expansión de los medios de informatización condicionó la necesidad de métodos y sistemas de auditoría aplicados a la informática y los sistemas de comunicación.

Las auditorías en los negocios son muy importantes, por cuanto la gerencia sin la práctica de una auditoría no tiene plena seguridad de que los datos económicos registrados realmente son verdaderos y confiables. La auditoría define con bastante razonabilidad la situación real de la empresa, además evalúa el grado de eficiencia y eficacia con que se desarrollan las tareas administrativas y el grado de cumplimiento de los planes y orientaciones de la gerencia. Contribuyendo así a una adecuada toma de decisiones. (8)

1.4 Tipos de Auditoría

Es necesario comprender la importancia de obtener resultados específicos, lograr propósitos o realizar objetivos, cumpliendo con estándares predeterminados de cantidad, calidad, tiempo, costo y de servicio. Los tipos de auditorías enmarcan estos estándares, las cuales se muestran a continuación.

La Norma ISO 19011:2002 establece los siguientes tipos de auditorías:

- Auditorías internas o externas en una organización.
- Auditorías de primera, segunda y tercera parte.
- Auditorías individuales, combinadas o conjuntas.
- Auditorías de riesgos de gestión y de actividades. (9)

Para tener una comprensión de la auditoría externa y los objetivos que en ella se aplican, es necesario conocer el funcionamiento de las auditorías internas.

Auditoría interna: Es aquella cuyo ámbito de su ejercicio se da por auditores que dependen o son empleados de la misma organización en que se practica. El resultado de su trabajo es con propósitos internos o de servicio para la misma organización. Tiene como objetivo apoyar a los miembros de la organización en el desempeño de sus actividades y como beneficios, permite determinar si los sistemas y procedimientos establecidos son efectivos, facilitando continuamente la efectividad de los controles establecidos y haciendo recomendaciones para el mejoramiento de las políticas, procedimientos, sistemas, etcétera.

Auditoría Externa: Realizada por personas ajenas al personal de la empresa. Consiste en el examen de los estados financieros independientes, mediante la aplicación de unos procedimientos sujetos a unas normas generalmente aceptadas. Su objeto es expresar una opinión sobre la razonabilidad con que dichos documentos presentan la situación contable, los resultados de sus operaciones y los cambios en su posición financiera conforme a los principios de contabilidad generalmente aceptados y aplicados con uniformidad. (10)

Las auditorías, ya sean internas o externas están conformadas por clases, las cuales se tratarán en el siguiente epígrafe.

1.5 Clases de Auditoría

Según define la autora Adelkys Rosa Sánchez Gómez en su libro “Auditoría y control interno”, existen diversas clases de auditorías entre las que se encuentran:

- **Financiera:** Consiste en el examen y evaluación de los documentos, operaciones, registros y estados financieros de la entidad, para determinar si éstos reflejan, razonablemente, su situación financiera y los resultados de sus operaciones, así como el cumplimiento de las disposiciones económico – financieras, con el objetivo de mejorar los procedimientos relativos a la gestión y el control interno.
- **Operacional:** Consiste en el examen y evaluación que se realiza a una entidad para establecer el grado de economía, eficiencia y eficacia en la planificación, control y uso de los recursos y comprobar la observancia de las disposiciones pertinentes, con el objetivo de verificar la utilización más racional de los recursos y mejorar las actividades y materias examinadas.
- **De Cumplimiento:** Consiste en verificar el cumplimiento de las disposiciones legales, reglamentarias y normativas aplicables en la ejecución de las actividades desarrolladas por una empresa.

- **De Rendimiento:** Consiste en el examen y evaluación de los documentos, operaciones, registros y estados financieros de la entidad, para determinar los resultados obtenidos en el proceso de auditoría
- **Revisiones especiales:** Según Pablo Emilio Hurtado Flores es una categoría mixta que incluye otro tipo de auditorías que no son las clases definidas, como la auditoría informática, la cual se concibe como el examen que se practica a los recursos computarizados de una empresa, comprendiendo:
- Muestreo de programas para extraer datos para pruebas de auditoría.
- Distribución de los equipos.
- Estructura del departamento de informática y utilización de los mismos.
- Procedimientos analíticos, por ejemplo, identificar inconsistencias o fluctuaciones importantes.
- Pruebas de controles de aplicación, por ejemplo, pruebas del funcionamiento de un control programado.
- Rehacer cálculos o revisar auditorías anteriores realizadas por los sistemas de contabilidad de la entidad. (11)

Enmarcando la investigación en las clases de auditoría, se ha decidido realizar las de cumplimiento y financiera, debido a que se efectuará un control sobre el cumplimiento del contrato pactado entre Alfaomega y Albet S.A. y variables financieras específicas como las licencias y programas de estudios.

1.6 Técnicas de auditoría más utilizadas

Las técnicas de auditoría, son los métodos prácticos de investigación y prueba que el auditor utiliza para lograr la información y comprobación necesaria para poder emitir su opinión profesional. Estas contribuyen a agilizar y dinamizar la ejecución del control como proceso.

Técnicas de la auditoría más utilizadas:

- Comparación: Es observar la similitud o diferencia existente entre dos o más elementos.
- Observación: Es el examen ocular para cerciorarse cómo se ejecutan las operaciones.
- Verificación Oral: Confirmación, verificación y expresión.
- Indagación: Es el acto de obtener información verbal sobre un asunto mediante la búsqueda y consulta de datos.
- Entrevistas: Pueden ser efectuadas al personal de la empresa auditada o personas beneficiarias de los programas o proyectos en intercambio comunicativo.
- Encuestas: Pueden ser útiles para recopilar información de un gran universo de datos o grupos de personas, revelan indicadores cuantitativos y cualitativos.
- Análisis: Proceso lógico del conocimiento que consiste en la separación y evaluación crítica, objetiva y minuciosa de los elementos o partes que conforman una operación, actividad, transacción o proceso, con el fin de establecer su naturaleza, su relación y conformidad con los criterios normativos y técnicos existentes. La síntesis es el proceso lógico que integra el análisis realizado.(12)
- Confirmación: Es la técnica que permite comprobar la autenticidad de los registros y documentos analizados, a través de información directa y por escrito otorgada por funcionarios que participan o realizan las operaciones sujetas a examen. Al confirmar se constata veracidad.
- Conciliación: Implica hacer que concuerden dos conjuntos de datos relacionados, separados e independientes.
- Comprobación: Se aplica en el curso de un examen, con el objeto de verificar la autenticidad y legitimidad de las operaciones efectuadas por una empresa.

- Inspección: Es el examen físico y ocular de activos, obras, documentos y valores, con el objeto de establecer su existencia y autenticidad. (13)

El estudio de las técnicas de auditoría favoreció en la recopilación de información para la toma de decisión de las variables financieras a utilizar.

1.7 Estudios de herramientas similares

En la actualidad existen varios software de auditoría, importantes en los medios o contextos en que se aplican en correspondencia con sus propósitos, de ellos se destacan como líderes en el mercado mundial ACL e IDEA y en el uso nacional Versat Sarasola.

1.7.1 ACL

Este software de auditoría financiera brinda prestaciones de análisis de datos, que permiten que las organizaciones aseguren la precisión, con funciones integradas de creación de informes. Llega prácticamente a los datos de cualquier fuente, en cualquier sistema, mediante una interfaz de usuario consistente.

Con la tecnología de ACL las organizaciones pueden asegurar un eficiente cumplimiento, además, reducir el riesgo y los fraudes, contener los costos, minimizar las fugas de ingresos y aumentar su rentabilidad.

Los profesionales de control y auditoría señalan constantemente a ACL como el software específico para auditorías preferido para la extracción y el análisis de datos, la detección de fraudes y el monitoreo continuo.

Entre algunas de las desventajas que presenta se encuentra que es una aplicación de escritorio, y por eso su acceso se limita al ordenador donde están instaladas, son dependientes del sistema operativo que utilice el ordenador y sus capacidades (video, memoria, entre otros), requieren instalación y actualización personalizada y suelen tener requerimientos especiales de software y librerías. (14)

1.7.2 IDEA

Es una herramienta de PC basada en la Interrogación de Archivos para ser utilizada por auditores, contadores, investigadores y personal de seguridad informática. Analiza los datos de diversas maneras y admite la extracción, el muestreo y la manipulación de datos para identificar errores, problemas, cuestiones específicas y tendencias. Es eficiente en auditorías financieras, y la misma puede controlar cuentas por pagar, cuentas por cobrar, activos fijos, ventas y cobros.

También se emplean para realizar pruebas específicas como fraude, seguridad informática y dirección contable.

Las principales funcionalidades del IDEA son:

1. Importar datos desde un amplio rango de tipos de archivo.
2. Crear diferentes vistas de los datos y reportes.
3. Llevar a cabo análisis específicos de los datos como:
   - 3.1 Cálculo de estadísticas diversas.
   - 3.2 Detección de omisiones.
   - 3.3 Sumarizaciones.
   - 3.4 Anticuación de fechas.
4. Efectuar diversidad de cálculos.
5. Obtener muestras usando diversas técnicas de muestreo.
6. Unir y comparar diferentes archivos de datos.
7. Crear tablas pívot para análisis multidimensional.
8. Generar automáticamente un historial completo documentando los análisis realizados.(15)

IDEA cuenta con funciones para aritmética, textos, fechas, horas, estas funciones le permiten ejecutar operaciones con:

- Fechas.
- Cálculos financieros y estadísticos.
- Búsquedas de textos.(16)

Utilizando IDEA se pueden obtener varias ventajas, algunas de estas pueden ser:

- Incremento del alcance de las investigaciones llevando a cabo pruebas que no pueden ser efectuadas manualmente.
- Incremento de la cobertura (verificación de un gran número de elementos cubriendo potencialmente el 100% de las transacciones de un año o más).
- Mejor información (por ejemplo análisis adicionales o perfil de los datos).
- Permite ahorro significativo de tiempo. (17)

1.7.3 Versat Sarasola

El programa Versat-Sarasola, sistema cubano de contabilidad confiable, permite enviar información eficaz, de forma inmediata, desde lugares apartados, a la vez que ofrece mayor organización, control y disciplina en cada gestión. Fue éste el primer sistema de contabilidad cubano certificado, en cuya evaluación participaron el Ministerio de Finanzas y Precios, consultorías internacionales y el organismo encargado de la seguridad informática. Es un sistema económico integrado. Constituido por 12 módulos: Configuración y seguridad, Contabilidad general y de gastos, Costos y procesos, Análisis económico empresarial, Control de activos fijos, Finanza y caja, Planificación y presupuestos, Control de inventarios, Pago de salario, Paquete de gestión, Contratación y Auditoría. (18) Presenta el inconveniente de que debe ser utilizado en Windows, que es una aplicación de escritorio y utiliza como sistema gestor de base de datos SQL, por lo que no cumple con el principio de independencia tecnológica que promueve el país.

1.7.4 Criterio de selección sobre herramientas similares

Después de realizado el estudio del arte de las herramientas similares, se considera que es necesaria la realización de un software que cumpla con las especificaciones puntuales que se

han desarrollado en el convenio Albet-Alfaomega. Debido a que los software antes mencionados son propietarios y costosos, no cumplen con las proyecciones de la UCI de migrar a software libre, además, no cumplen con determinados requisitos como por ejemplo:

- Se necesita conectar a través de servicios web a la Plataforma Educativa Zera.
- Generar reportes específicos del software. Varias funciones de comparación, tales como comparar las cantidades físicas de los inventarios de un contrato con las cantidades existentes en la base de datos del sistema auditado.
- Hacer búsquedas de alguna información en particular, la cual cumpla ciertos criterios y que se encuentre dentro de la base de datos del software que se audita.

1.8 Tecnologías de acceso a la información

La comunicación y el acceso a la información en un ser humano dependen de sus ventanas de percepción. Las ayudas tecnológicas en su desarrollo, apuntan día a día a problemas específicos; se enmarcan según la necesidad que suplen y el tipo de discapacidad a la cual sirven.

1.8.1 Servicios Web

Un servicio web es una entidad programable que proporciona alguna funcionalidad determinada, y es accesible a cualquier número de sistemas que usen las normas de Internet. Un servicio web puede ser usado internamente por una aplicación o ser publicado hacia Internet; estos servicios permiten la ejecución de sus funcionalidades sin importar la plataforma, sistema operativo, o lenguaje en el cual estén implementados.

Los servicios web se pueden utilizar para intercambiar datos en redes de ordenadores como Internet, la interoperabilidad se consigue mediante la adopción de estándares abiertos. Las organizaciones OASIS y W3C son los comités responsables de la arquitectura y reglamentación de los servicios web; para mejorar la interoperabilidad entre distintas implementaciones de servicios web se ha creado el organismo WS-I encargado de desarrollar diversos perfiles para definir de manera más exhaustiva estos estándares. (19)

En la medida que surgen y perfeccionan las herramientas de desarrollo, los clientes se enfrentan con la difícil tarea de integrar una amplia variedad de aplicaciones distribuidas con diferentes entornos de ejecución y desarrollo. Debido a la importancia que representa en la actualidad este aspecto se han desarrollado varios enfoques y tecnologías para facilitar la integración, gracias a los estándares abiertos y al énfasis en la comunicación y colaboración entre personas y aplicaciones y a sus múltiples usos los Servicios web son muy utilizados. (20)

Los servicios web tienen una interfaz descriptiva en un formato que puede ser procesado por una máquina (específicamente WSDL) y otros sistemas que interactúan con el servicio web empleando mensajes SOAP (por sus siglas de Simple Object Access Protocol), WSDL y SOAP se detallarán en los epígrafes 1.8.2 y 1.8.4.

Los servicios web brindan grandes ventajas dentro de las aplicaciones distribuidas como son:

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Los servicios web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Al apoyarse en HTTP, los servicios web pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.
- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados. (21)

Atendiendo a las ventajas antes mencionadas, que brindan el uso de servicio web, se incluye su utilización en el desarrollo de la solución propuesta.

Los servicios web pueden ser usados a través de dos protocolos fundamentalmente: RPC y SOAP. El protocolo RPC tiene una dependencia de la plataforma y la tecnología que se utiliza para ejecutar los servicios web, en cambio SOAP es un protocolo que permite la independencia de plataforma y tecnología. (22)

1.8.2 WSDL

WSDL es un formato XML (XML se argumenta en el siguiente epígrafe) que describe los servicios de red como un conjunto de puntos finales que procesan mensajes contenedores de información orientada tanto a documentos como a procedimientos. Las operaciones y los mensajes se describen de forma abstracta y después se enlazan a un protocolo de red y a un formato de mensaje concreto para definir un punto final de red. Los puntos finales concretos relacionados se combinan en puntos finales abstractos (servicios). WSDL es extensible, lo que permite la descripción de puntos finales de red y sus mensajes, independientemente de los formatos de los mensajes o protocolos de red utilizados para comunicarse. (23)

1.8.3 XML

El Lenguaje Extensible de Marcas, abreviado XML (eXtensible Markup Language), describe una clase de objetos de datos llamados documentos XML y describe parcialmente el comportamiento de los programas de computadora que los procesan. XML es un "perfil de aplicación" o una forma restringida de SGML, el Lenguaje Estándar Generalizado de Marcación [ISO 8879]. Por construcción los documentos XML, son documentos SGML conformados.

Los documentos XML están compuestos por unidades de almacenamiento llamadas entidades, que contienen tanto datos analizados como no analizados. Los datos analizados están compuestos de caracteres, algunos de los cuales de la forma datos carácter, y otros de la forma etiquetas. Las etiquetas codifican una descripción de la estructura de almacenamiento del documento y su estructura lógica.

1.8.4 SOAP

SOAP (Simple Object Access Protocol): Protocolo Simple de Acceso a Objetos se trata de un protocolo basado en XML, que propicia la interacción entre varios dispositivos y que tiene la capacidad de transmitir información compleja, ofreciendo funciones muy útiles a usuarios del medio web. Los datos pueden ser transmitidos a través de diferentes protocolos como HTTP, SMTP, entre otros. SOAP especifica el formato de los mensajes. (24)

- Algunas de las ventajas de SOAP son: No está asociado con ningún lenguaje: No especifica una API, por lo que la implementación de esta se deja al lenguaje de programación y la plataforma.
- No se encuentra fuertemente asociado a ningún protocolo de transporte: Su especificación no describe cómo se deberían asociar los mensajes de SOAP con HTTP. Dichos mensajes no son más que un documento XML, por lo que puede transportarse utilizando cualquier protocolo capaz de transmitir texto.
- Aprovecha los estándares existentes en la industria: Utiliza los estándares existentes para satisfacer sus necesidades.
- Permite la interoperabilidad entre múltiples entornos: Debido a que aprovecha los estándares existentes en la industria, las aplicaciones que se ejecuten en plataformas con dichos estándares pueden comunicarse mediante mensaje SOAP con aplicaciones que se ejecuten en otras plataformas.

1.8.5 Criterio de selección de la tecnología de acceso a la información.

La especificación de SOAP define el formato de los mensajes para comunicar el software propuesto con la Plataforma Educativa Zera, normalmente dichos mensajes son una forma de comunicación entre uno y otro. Es el protocolo estándar para la comunicación en Internet además de que requiere muy poco soporte en tiempo de ejecución para funcionar adecuadamente.

1.9 Metodologías para el desarrollo de software

Un proceso de software detallado y completo suele denominarse Metodología, define Quién debe hacer, Qué, Cuándo y Cómo debe hacerlo. Las metodologías se desarrollan con el objetivo de dar solución a los problemas existentes en la producción de software, que cada vez son más complejos. Estas engloban procedimientos, técnicas, documentación y herramientas que se utilizan en la creación de un producto de software.

1.9.1 OpenUP

Es una versión ágil, que aplica propuestas iterativas e incrementales dentro del ciclo de vida, tratando de ser manejable en relación con RUP. Plantea que se debe tener un software ya funcional o lo que es lo mismo, un proyecto ejecutable en un período de tiempo corto. Además, debe utilizar solo los procesos que sean necesarios y en este caso se necesita de personas y profesionales que sean capaces de distinguir entre lo necesario y lo que no es necesario para que el proyecto no tenga errores y con eso evitar entregar al usuario final un producto de mala calidad; aunque este tipo de método plantea además que no se deben utilizar demasiados artefactos y sobre todo que el proyecto debe acoplarse a las necesidades del usuario pudiendo ser este modificado, mejorado y extendido. (25)

1.9.2 Proceso Unificado de Software

El Proceso Unificado de Software (RUP) es un marco genérico que puede especializarse para una gran variedad de proyectos de software hacia diferentes áreas de aplicación, diferentes tipos de organización, diferentes niveles de aptitud y diferentes tamaños de proyectos. Esta metodología de desarrollo utiliza el lenguaje de modelado UML para especificar, visualizar, construir y documentar los artefactos que este genera.

RUP divide en 4 fases el desarrollo del software, las cuales terminan con sus hitos correspondientes:

- Inicio: En esta etapa se determina la visión del proyecto.
- Elaboración: El objetivo de esta etapa es determinar la arquitectura óptima para el desarrollo del ciclo de vida del proyecto.
- Construcción: El objetivo es llegar a obtener la capacidad operacional inicial.
- Transición: El objetivo es llegar a obtener la liberación del proyecto.

Esta metodología también cuenta con flujos de trabajo, los cuales se muestran en la figura 1:

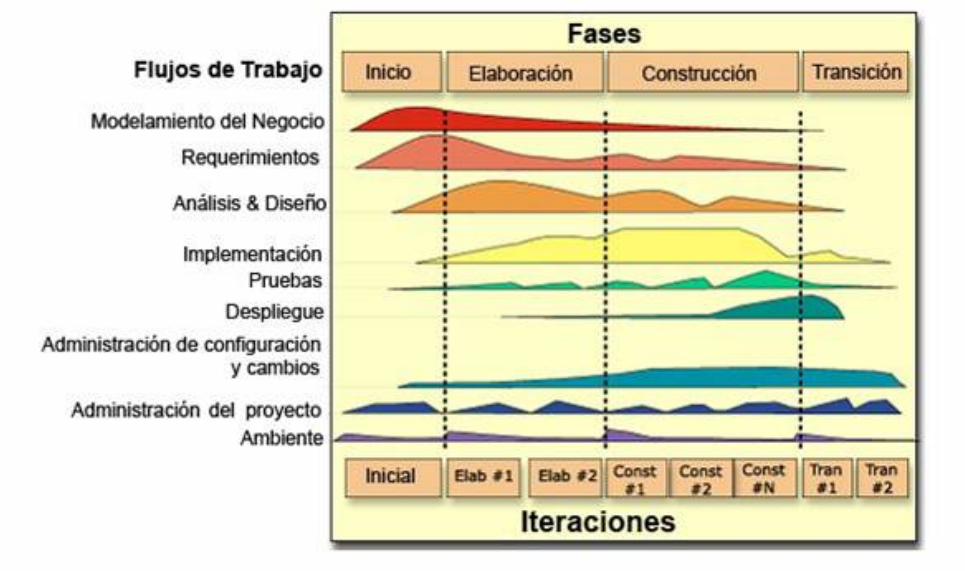

Figura 1: Diagrama de la Metodología RUP.

Los verdaderos aspectos definitorios del Proceso Unificado se resumen en 3 frases claves: Dirigido por Casos de Uso, Centrado en la Arquitectura e Iterativo e Incremental. Esto es lo que hace notable al Proceso Unificado.

Dirigido por Casos de Uso. El proceso de desarrollo sigue una trayectoria que avanza a través de los flujos de trabajo generados por los casos de uso, estos especifican y diseñan el principio de cada iteración y son la fuente mediante la cual los ingenieros de pruebas construyen los casos de prueba. Los casos de uso describen la funcionalidad total del sistema, pensada en términos de la importancia que tiene la misma para el usuario (no solamente la funcionalidad en sí).

Centrado en la Arquitectura. La arquitectura y los casos de uso son procesos que se desarrollan en paralelo, ya que los casos de uso guían la arquitectura del sistema y esta influye en la selección de los casos de uso, pues se desea una arquitectura viable a la hora de implementar los casos de uso. La arquitectura involucra los elementos más significativos del sistema y está influenciada entre otros por las plataformas de software, los sistemas operativos, los sistemas de gestión de base de datos, además de otros como sistemas heredados y requerimientos no funcionales.

Iterativo e Incremental. Se recomienda dividir el proyecto en ciclos o iteraciones a través de cada una de las fases por las que se transita, dentro de las cuales se realizan varias iteraciones en un número variable según el proyecto. La terminación de cada fase ocurre en el hito correspondiente a cada una, donde se evalúa que se hayan cumplido los objetivos de la fase en cuestión. Otros de los beneficios de la iteración:

- Reduce el coste del riesgo al coste de un solo incremento.
- Menos riesgo de no sacar el producto al mercado en fecha.
- Acelera el ritmo de desarrollo.

Las necesidades del usuario y correspondientes requisitos no se definen completamente al principio. Se requieren iteraciones sucesivas. (26)

1.9.3 Extreme Programing

Extreme Programing (XP) es una de las metodologías de desarrollo de software más exitosas en la actualidad empleadas en proyectos de corto plazo y pequeños equipos. 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.

Algunas características que muestra son:

- Desarrollo iterativo e incremental, pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión.
- Programación en parejas. Se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que la mayor calidad del código escrito de esta manera -el código es revisado y discutido mientras se escribe- es más importante que la posible pérdida de productividad inmediata.

- Frecuente integración del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- Corrección de todos los errores antes de añadir nueva funcionalidad.
- Refactorización del código. Reescribir ciertas partes del código para aumentar su legibilidad sin modificar su comportamiento.
- Propiedad del código compartida. Este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto
- Simplicidad en el código. Cuando todo funcione se podrá añadir funcionalidad si es necesario; la simplicidad y la comunicación son extraordinariamente complementarias.

Con más comunicación resulta más fácil identificar qué se debe y qué no se debe hacer. Mientras más simple es el sistema, menos tendrá que comunicar sobre este. (27)

Por ser una metodología utilizada para proyectos de corta duración no se puede utilizar en la realización del trabajo por la gran envergadura que tiene el mismo. El equipo de trabajo del proyecto necesita que los documentos generados sean específicos para la continuidad de la realización del mismo y XP es una metodología ágil que genera menos documentación.

1.9.4 Selección de la metodología de software

Se determinó que XP no se ajusta al proceso en cuestión, a pesar de su utilidad, tiene como principal objetivo producir el software a corto plazo además no genera la documentación suficiente para posteriores versiones del software. Está más orientada a las personas que a las herramientas y plantea que el usuario debe ser un integrante más del equipo de desarrollo.

Se selecciona la metodología RUP debido a que la misma genera abundante documentación, lo cual resulta de vital importancia para el desarrollo del Software Auditoría, la información generada serviría de referencia para las nuevas versiones que se implementen a futuro, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de

sistemas orientados a objetos. Favorece realizar las mejores prácticas en ingeniería de Software y está centrado en la arquitectura y guiado por los casos de uso.

1.10 Herramienta para el modelado

Los lenguajes de modelado constituyen una guía muy importante para la representación de la estructura de los sistemas informáticos. Consiste en un conjunto de vistas, diagramas, símbolos y mecanismos generales o reglas que indican cómo utilizar los elementos.

1.10.1 Lenguaje de Modelado Unificado (UML)

UML (Unified Modeling Language) es un lenguaje de modelado gráfico que propicia organizar, construir y documentar los elementos que forman un sistema de software. Permite modelar artefactos conceptuales como lo son procesos de negocio y funciones de sistema, además de artefactos concretos tales como escribir clases en un lenguaje determinado, esquemas de bases de datos y componentes de software reusables.

UML es un lenguaje para especificar y no un método o un proceso. Aunque es flexible y permite aplicarse en una gran variedad de metodologías de desarrollo, no especifica en sí cual se debe utilizar.

Para modelar, UML utiliza diagramas que se pueden clasificar en dos tipos: diagramas de estructura que comprenden los diagrama de clases, diagrama de componentes, diagrama de objetos, diagrama de despliegue y diagrama de paquetes; también los diagramas de comportamientos entre los que se encuentran diagrama de actividades, diagramas de casos de uso, diagramas de estados, diagrama de secuencia, diagrama de colaboración, entre otros. (29)

Proporciona una organización en los procesos de diseño de tal forma que los analistas, clientes, desarrolladores y otras personas involucradas en el desarrollo del sistema lo comprendan y convengan con él. UML se apoya para su modelado en las herramientas CASE que permiten automatizar el proceso de diseño y desarrollo de software. CASE (Computer Aided Software Engineering) es un conjunto de ayudas para el desarrollo de programas informáticos desde la

planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. (30)

Entre las herramientas de modelado que utilizan UML están:

- ArgoUML.
- Poseidon.
- MagicDraw UML.
- Visual Paradigm.
- Umbrello
- Rational Rose.
- Día Project.

1.10.2 Día Project

Día Project es un software desarrollado por la GNOME Foundation, que se utiliza para el modelado de diagramas y que fue liberado bajo la Licencia Pública General (GPL) del proyecto GNU (acrónimo recursivo que significa GNU No es Unix).

Este software está inspirado en el potente programa privativo del paquete Microsoft Office, denominado Visio. Se puede utilizar para dibujar diferentes tipos de diagramas como por ejemplo: diagramas entidad-relación, diagramas UML, organigramas, diagramas de red, entre otros. Además asiente añadir nuevas formas definiendo simples archivos XML (Lenguaje de Marcas Extensible) en la configuración y utilizando un subconjunto del lenguaje para describir gráficos vectoriales bidimensionales SVG (del inglés Scalable Vector Graphics) en dibujar la nueva forma. (31)

1.10.3 Umbrello

Es una herramienta rápida, ligera y sencilla de usar, no se pone lenta cuando los proyectos son enormes. Requiere de más opciones de generación de documentación de los diseños modelados

dentro de ella. Soporta los diagramas UML estándares. Es una herramienta que tiene buen soporte para la generación de código, desde el mismo programa es posible generar todo o parte y lo hace muy bien. Uno de los defectos encontrados es la falta de libertad en las conexiones ya que estas se colocan automáticamente cuando se mueven los objetos. (32)

Actualmente Umbrello permite instalarse en diferentes plataformas y posee más de 30 idiomas diferentes, gracias a su licencia original GPL. Brinda la posibilidad de:

- Copiar, cortar y pegar los objetos de los diferentes diagramas, puede copiar los objetos como imágenes PNG de forma que pueda insertarlos en cualquier tipo de documento.
- Exportar como imagen un diagrama completo.
- Imprimir diagramas individuales.
- Organizar mejor la maqueta, especialmente en los proyectos grandes.

1.10.4 Visual Paradigm

Es una herramienta multiplataforma de modelado. Soporta el ciclo de vida completo del desarrollo de software, ayuda a una rápida construcción de aplicaciones de calidad y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Ventajas de Visual Paradigm:

- Entorno de creación de diagramas para UML 2.0.
- Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.

- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad en múltiples plataformas.
- Producto de calidad.
- Soporta aplicaciones web.
- Varios idiomas.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones. (33)

1.10.5 Selección de la herramienta para el modelado

Se utiliza Visual Paradigm en su versión 8.0, la cual cumple con el Standard UML 2.0 y 2.1, lo que permite principalmente, en comparación con las versiones anteriores un modelado del proceso de negocio o BPMN (Notación de Modelado del Proceso de Negocio). Posibilita modelar todo el ciclo de desarrollo del software y es multiplataforma. También es de las herramientas estudiadas en las que el equipo de desarrollo presenta más dominio, y es de los productos CASE la que más se ajusta a las herramientas antes mencionadas.

1.11 Framework

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.

1.11.1 Zend Framework

Es uno de los más utilizados en el desarrollo de aplicaciones y servicios web con PHP. Hoy día Zend, constituye la variante open source que brinda la compañía de igual nombre y que se integra fácilmente al IDE Zend Studio, para la construcción de aplicaciones web modernas, robustas y seguras. Entre las principales ventajas y desventajas que muestra son.

Ventajas:

- Permite desarrollar aplicaciones bajo el estilo arquitectónico Modelo Vista Controlador.
- Integra funciones para el manejo de archivos PDF y canales RSS.
- Ofrece soluciones para el acceso eficiente a diferentes gestores de base de datos.
- Contiene librerías de clases, bajo la filosofía de la programación orientada a objetos y soporta PHP5 para autenticación y filtrado de datos.

Desventajas:

- Los formulario generados por Zend Framework son limitados en cuanto a diseño.
- La dependencia del código fuente no permite generar completamente el esquema del modelo de una aplicación. (34)

1.11.2 Symfony

Symfony es un completo framework diseñado para optimizar gracias a sus características el desarrollo de las aplicaciones web. Separa la lógica del negocio, la lógica del servidor y la presentación de la aplicación web, proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja; automatiza las tareas más comunes permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web. Symfony está desarrollado completamente con PHP 5; ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel, es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows.

Symfony se diseñó para que se ajustara a los siguientes requisitos:

- Fácil de instalar y configurar en la mayoría de plataformas y con la garantía de que funciona correctamente en los sistemas Windows y Unix estándares.
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de los casos pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de *"convenir en vez de configurar"*, en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de las "*mejores prácticas"* y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros. (35)

1.11.3 Selección del Framework

Una vez realizado el estudio de algunos de los principales framework, la investigación se ha inclinado por la utilización de Symfony en su versión 2.0.10, ya que éste es independiente de plataforma, es compatible con la mayoría de los gestores de bases de datos y posee una amplia bibliografía y foros en español a los que se les puede hacer referencias en cualquier situación o duda.

1.12 Lenguaje de desarrollo

Para el desarrollo de la presente investigación y debido a que la misma se centrará en desarrollar un software que le permita a Albet S.A auditar externamente la Plataforma Educativa Zera, se propone utilizar en la implementación del mismo la tecnología web. Con la utilización de este tipo

de tecnología se pueden realizar tareas sencillas sin necesidad de descargar ni instalar ningún programa. También se pueden usar desde cualquier sistema operativo porque solo es necesario tener un navegador.

1.12.1 Tecnologías del lado del cliente

Un lenguaje del lado cliente es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio. Pero nuestra página no se verá bien si la computadora cliente no tiene instalados los *plug-in* adecuados. El código, tanto del hipertexto como de los scripts, es accesible a cualquiera y ello puede afectar a la seguridad.

JavaScript: Es un lenguaje de programación utilizado para crear pequeños programas encargados de realizar determinadas acciones dinámicas dentro del ámbito de una página web. Técnicamente es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios; aunque con este nombre, no guarda ninguna relación directa con el lenguaje de programación Java.

Por otra parte pone a disposición del programador todos los elementos que forman parte de la página web, para que el mismo pueda acceder a ellos y modificarlos dinámicamente. Se utilizará este lenguaje para la implementación de algunos aspectos dinámicos como por ejemplo la validación de formularios y la creación de ventanas popup con el propósito de mostrar algún tipo de información. (36)

XHTML (Lenguaje de Marcado de Hipertexto Extensible), es una versión más estricta y limpia que HTML, nace precisamente con el objetivo de remplazar a este último lenguaje ante su limitación de uso, con las, cada vez más abundantes, herramientas basadas en el Lenguaje de Marcado Extensible (XML, por sus siglas en inglés).

XHTML extiende HTML 4.0 combinando la sintaxis de HTML, diseñado para mostrar y describir datos, con la de XML. Su objetivo es avanzar en el proyecto del World Wide Web Consortium,

para lograr una web semántica, donde la información y la forma de presentarla estén claramente separadas.

Las principales ventajas del XHTML sobre el HTML son:

- Se pueden incorporar elementos de distintos espacios de nombres XML (como MathML y Scalable Vector Graphics).
- Un navegador, no necesita implementar heurísticas para detectar qué desea escribir el autor.
- Como es XML se pueden utilizar fácilmente con herramientas creadas para procesamiento de documentos XML genéricos (editores, XSLT, etc.).

CSS (Hojas de Estilo en Cascadas). "Es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos.

CSS se utiliza para dar estilo a documentos HTML y XML, separando el contenido de la presentación. Los estilos definen la forma de mostrar los elementos HTML y XML. CSS. Permite a los desarrolladores web controlar el estilo y el formato de múltiples páginas web al mismo tiempo." (37)

1.12.2 Tecnología del lado del servidor

Un lenguaje del lado del servidor es aquel que se ejecuta en el servidor web, justo antes de que se envíe la página a través de Internet al cliente. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente.

PHP 5.3.3 (Hypertext Preprocessor, procesador de hipertexto por su traducción al español) es un idioma artificial concebido para crear programas de computador, admitiendo incorporar algoritmos

con precisión. Está formado de un conjunto de símbolos y reglas tanto sintácticas como semánticas que especifican su estructura y el significado de sus elementos y expresiones. Cada lenguaje de programación tiene sus propias características, las cuales lo hacen más potente o más débil para determinada finalidad. El estudio de los mismos garantiza contar con los recursos apropiados para codificar la solución a desarrollar.

PHP es un lenguaje interpretado de propósito general ampliamente utilizado, diseñado especialmente para aumentar e incrementar el dinamismo de las páginas web y puede ser incrustado dentro de código HTML (HyperText Markup Language, Lenguaje de Marcado de Hipertexto por su traducción al español). Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida al usuario final, por lo que PHP convierte una página estática en una dinámica. Puede ser desplegado en la mayoría de los servidores web y plataformas sin costo alguno. La nueva versión de PHP posee una serie de nuevas características como son:

- Es un lenguaje multiplataforma.
- Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- El código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- Posee una amplia documentación entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de programación orientada a objetos.
- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución. (38)

Active Server Pages (ASP) es una tecnología del lado servidor de Microsoft para páginas web generadas dinámicamente, que ha sido comercializada como un anexo a Internet Information Server (IIS). ASP tiene muchas ventajas: (39)

- La más relevante es que reemplaza la forma tradicional de intercambiar información entre usuarios.
- Active Server Pages posibilita que el código sea incrustado en un documento HTML y que corra en el servidor.
- ASP es compatible con múltiples plataformas.

Se han expuesto algunas de las ventajas de ASP, pero para utilizar esta tecnología del lado del servidor, se debe contar con servidores de bases de datos propietarios de Microsoft, como el Internet Information Server, por lo que no se puede crear una aplicación libre usando ASP.

1.12.3 Selección del lenguaje de programación

De acuerdo con las características y comparaciones anteriormente descritas PHP en su versión 5.3.3 es mucho más favorecido, debido a su libre uso resulta más provechoso a la hora de realizar una migración de Sistemas Operativos, tiene mayor velocidad de ejecución, es compatible con más de 25 plataformas incluyendo diferentes versiones de Unix y Windows. Se ha convertido en el lenguaje de programación más versátil, por tanto es el adecuado para implementar la propuesta de sistema de este trabajo.

1.13 Sistema Gestor de Base de Datos

Un Sistema Gestor de Base de Datos no es más que un software cuyo objetivo es proporcionar una interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Están compuestos por un lenguaje de definición de datos, uno de manipulación de datos y uno de consulta.

1.13.1 SQLite

SQLite es un sistema completo de bases de datos que soporta múltiples tablas, índices, triggers y vistas. No necesita un proceso separado funcionando como servidor ya que lee y escribe directamente sobre archivos que se encuentran en el disco duro. El formato de la base de datos es multiplataforma e indistintamente se puede utilizar el mismo archivo en sistemas de 32 y 64 bits.

La base de datos se almacena en un único fichero a diferencia de otros DBMS[3] que hacen uso de varios archivos. SQLite emplea registros de tamaño variable de forma tal que se utiliza el espacio en disco que es realmente necesario en cada momento.

El código fuente está pensado para que sea entendido y accesible por programadores promedio. Todas las funciones y estructuras están bien documentadas.

Existe un programa independiente de nombre sqlite que puede ser utilizado para consultar y gestionar los ficheros de base de datos SQLite, también sirve como ejemplo para la escritura de aplicaciones utilizando la biblioteca SQLite.

Uno de los puntos que se señala como una desventaja, es que el modelo tradicional de utilizar un proceso servidor ofrece mayor protección ante aplicaciones que utilizan la base de datos y que pudieran tener fallos de programación; un mal manejo de punteros en un cliente no pueden llegar a corromper la memoria en el servidor, en cambio al ser SQLite parte integral de la aplicación que se utiliza, un problema de programación podría transferir las consecuencias al medio que contiene la información almacenada.

1.13.2 MySQL

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario que implementa diversas Interfaces de Programación de Aplicaciones (APIs) la cual permite que aplicaciones puedan acceder a las bases de datos, independientemente del lenguaje en que

[3] Sistemas de gestión de bases de datos (en inglés *database management system*, abreviado *DBMS*)

estén implementadas. La última versión conocida es la 5.1 bajo licencia GPL y licencia comercial. El gestor incluye soporte para conexiones seguras entre los clientes MySQL y el servidor, utilizando el protocolo Secure Sockets Layer (SSL).

Contiene un interfaz Open Data Base Connectivity (ODBC), llamado MyODBC que permite a cualquier lenguaje de programación que soporte ODBC comunicarse con las bases de datos MySQL. Es muy utilizado en aplicaciones web como MediaWiki o Drupal, en plataformas Linux/Windows-Apache-MySQL-PHP/Perl/Python (WAMP), y por herramientas de seguimiento de errores como Bugzilla.

Entre sus ventajas se incluyen:

- Coste: El coste de MySQL es gratuito para la mayor parte de sus usos y su servicio de asistencia resulta económico.
- Asistencia: MySQL AB ofrece contratos de asistencia a precios razonables y existe una nutrida y activa comunidad MySQL.
- Velocidad: MySQL es mucho más rápido que la mayor parte de sus rivales.
- Funcionalidad: MySQL dispone de muchas de las funciones que exigen sus desarrolladores profesionales, como compatibilidad completa con ACID,
- Compatibilidad para la mayor parte de SQL ANSI, volcados online, duplicación, funciones SSL e integración con la mayor parte de sus entornos de programación. Así mismo, se desarrolla y actualiza de forma mucho más rápida que muchos de sus rivales.
- Portabilidad: MySQL se ejecuta en la inmensa mayoría de sistemas operativos y, la mayor parte de sus casos, sus datos se pueden transferir de un sistema a otro sin dificultad.
- Facilidad de uso: MySQL resulta fácil de utilizar y de administrar. Gran parte de las viejas bases de datos presentan problemas por utilizar sistemas obsoletos, lo que complica innecesariamente las tareas de administración. Las herramientas de MySQL son potentes y flexibles, sin sacrificar su capacidad de uso.”

1.13.3 PostgreSQL

Es el sistema de gestión de base de datos relacional, orientada a objetos de código abierto más avanzado del mundo. PostgreSQL, resulta un potente sistema de base de datos objeto/relacional. Cuenta con más de 15 años de desarrollo activo y una arquitectura probada que se ha ganado una sólida reputación de confiabilidad, integridad de los datos, y corrección, funciona en todos los principales sistemas operativos, incluyendo Linux, UNIX y Windows. Es altamente escalable, tanto en la enorme cantidad de datos que puede administrar, como en el número de usuarios concurrentes que puede soportar.

Posee numerosas ventajas entre ellas se pueden mencionar:

- Instalación ilimitada.
- Mejor soporte que los proveedores comerciales.
- Estabilidad y confiabilidad legendarias.
- Extensible pues el código fuente está disponible para todos sin costo.
- Multiplataforma.
- Diseñado para ambientes de alto volumen.
- Herramientas gráficas de diseño y administración de bases de datos. Existen varias herramientas gráficas de alta calidad para administrar las bases de datos (pgAdmin, pgAccess) y para hacer diseño de bases de datos (Tora, Data Architect).
- Altamente extensible, pues soporta operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario.
- Soporte SQL Comprensivo, soporta la especificación SQL99, e incluye características avanzadas tales como las uniones (joins) SQL92.
- Soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.

- Usa la tecnología PostgreSQL MVCC, o Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control), para evitar bloqueos innecesarios.
- Usa una arquitectura proceso-por-usuario cliente/servidor (Cliente/Servidor).
- Usa Write Ahead Logging (WAL) para incrementar la dependencia de la base de datos al registro de cambios antes de que estos sean escritos en la base de datos, garantizando que existirá un registro de las transacciones a partir del cual se puede restaurar la base de datos". (40)

1.13.4 Selección de la Base de Datos

En este caso se decidió tomar PostGreSQL, ya que se caracteriza por su estabilidad, potencia, robustez, facilidad de administración e implementación de estándares, además de poseer una gran documentación en español en su sitio web oficial y foros de internet. Es el motor de base de datos de código abierto más potente del momento, y en sus últimas versiones iguala o supera otras bases de datos comerciales. Se puede ejecutar en la gran mayoría de los sistemas operativos existentes en la actualidad, entre ellos Windows, Linux y Unix en todas sus variantes (AIX, BSD,HP-UX, SGI IRIZ, Mac OS x, Solaris, Tru64).

1.14 *Servidor Web Apache*

Apache es un servidor web de código abierto. Es el ejemplo de software libre de mayor éxito, por delante incluso del kernel Linux. Desde hace años, más del 60% de los servidores web de Internet emplean Apache (41)

El servidor Apache presenta entre otras características: mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido. Apache es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor web Apache, existen gran cantidad de módulos Apache disponibles para su utilización.

Además comparte con el lenguaje de programación PHP muchas de sus características, como son la gratuidad, su sencillez de manejo y su versatilidad, ya que se pueden instalar sobre Unix o sobre Windows. Aunque existen versiones de Apache para prácticamente todas las plataformas.

Principales características de Apache.

- Fiabilidad: Alrededor del 90% de los servidores con más alta disponibilidad funcionan con Apache.
- Gratuidad: Apache es totalmente gratuito, y se distribuye bajo la licencia Apache Software License, favoreciendo la modificación del código.
- Extensibilidad: Se pueden añadir módulos para ampliar las amplias capacidades de Apache. Existe una amplia variedad de módulos, que proporcionan generar contenido dinámico con PHP, Java, Perl, entre otros, además de monitorizar el rendimiento del servidor. Estos módulos pueden ser creados por cualquier persona con conocimientos de programación. (42)

El desarrollo del sistema que se utilizará el servidor web Apache en su versión 2.2.16 por su gran compatibilidad con el lenguaje de programación PHP. Además, de las múltiples ventajas que este ofrece a la hora de publicar una aplicación web.

1.15 IDEs (Entorno de desarrollo integrado)

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, o sea, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

1.15.1 NetBeans 6.9

Es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans, es un producto

libre y gratuito sin restricciones de uso. Esta plataforma es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones. NetBeans ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Entre las características de la plataforma están:

- Administración de las interfaces de usuario (ej. menús y barras de herramientas).
- Administración de las configuraciones del usuario.
- Administración del almacenamiento (guardando y cargando cualquier tipo de dato).
- Administración de ventanas.
- Framework basado en asistentes. (43)

1.15.2 Eclipse

El principal objetivo de Eclipse es proporcionar mecanismos, reglas que puedan ser seguidas por los fabricantes para integrar de manera transparente sus herramientas. Mediante APIs interfaces, clases y métodos, se exponen estos mecanismos. (44)

Características

- Soportar las herramientas proporcionadas por diferentes fabricantes de software independientes (ISV's). Soportar herramientas que permitan manipular diferentes contenidos (HTML, Java, C, JSP, EJB, XML, etc.)
- Facilitar una integración transparente entre todas las herramientas y tipos de contenidos sin tener en cuenta al proveedor.
- Proporcionar entornos de desarrollo gráfico (GUI) o no gráficos.

- Ejecutarse en una gran variedad de sistemas operativos, incluyendo Windows y Linux. 6. Eclipse emplea módulos (en inglés plug-in) para proporcionar toda su funcionalidad.

1.15.3 Selección del IDE de desarrollo

El IDE seleccionado es NetBeans en su versión 6.9 ya que el mismo es de código abierto y presenta mejores cualidades y funciones para el desarrollo web, es multiplataforma, se complementa perfectamente con el framework a utilizar, consta de una gran comunidad de desarrolladores y ofrece una amplia documentación y recursos de capacitación, así como una variada selección de *plugin*. Además permite la creación de aplicaciones web con la nueva versión de PHP 5.

1.16 Conclusiones Parciales

En relación con la información analizada, se emiten juicios como reflexiones teóricas en las que se sustenta el contenido de la investigación propuesta, entre ellos se encuentra:

- La importancia de la Auditoría en el proceso de desarrollo de software, así como las ventajas que brinda.
- Análisis del estado del arte de los principales software usado por los auditores, y la selección de las diferentes herramientas con el objetivo de aplicar la Auditoría Informática a la Plataforma Educativa Zera.

El estudio de este capítulo fundamenta la selección de las herramientas de trabajo para definir una arquitectura final la cual se muestra a continuación:

Como tecnología de acceso a la información se escogió SOAP, RUP como metodología de desarrollo, Visual Paradigm como herramienta para el modelado, Symfony como *framwork* de desarrollo, como lenguaje de programación JavaScript, HTML, y PHP, PostgreSQL para la base de datos y NetsBean como IDE de desarrollo.

2

CAPÍTULO 2

CARACTERÍSTICAS DEL SISTEMA

Introducción

Este capítulo está dedicado a conocer con una mayor claridad todos los procesos que intervienen en el flujo de trabajo del modelamiento del negocio, así como las percepciones asociadas al mismo. Se describen, todos los conceptos y la manera en que se desarrollan en la actualidad, se exponen los requisitos funcionales del sistema, el diagrama de casos de usos del sistema, la descripción de los casos de usos del sistema y un prototipo de interfaz de usuario por cada caso de uso crítico.

2.1 Descripción del Sistema Propuesto.

Para dar solución al problema planteado, se ha decidido desarrollar una herramienta (constituye un software para auditar la Plataforma Educativa Zera) cuyo objetivo fundamental es permitir a los usuarios (administradores y auditores) monitorizar la administración de Zera.

Las presentaciones que se realicen con este software, tienen por finalidad enfocar la atención de los auditores hacia el proceso de administración que realicen las escuelas con los diferentes programas de estudios. Los mismos se conforman por licencias de tipo profesor, estudiantes, cortesía y cada licencia tendrá un costo, que será controlado y evaluado en cada proceso de

auditoría que se realice. También brindará un ambiente agradable, sencillo y cargado de facilidades entre las que se encuentran:

- Mantener los contratos realizados en una base de datos, lo que permitirá una mayor organización de los mismos.
- Permitirá al auditor cliente realizar las auditorías cuando estime conveniente, esto apoyaría el control del cumplimiento de un contrato.
- Propiciará tener un informe de los resultados de los controles realizados.

2.2 Modelo de Dominio

La disciplina de modelado de negocio es la primera que propone RUP dentro del ciclo de desarrollo de un software, tiene su mayor peso durante la Fase de Inicio debido a que permite conocer los procesos existentes actuales de cualquier entidad o empresa para la que se vaya a desarrollar el sistema. Es en este flujo de trabajo donde se conocen a fondo cómo son iniciadas cada una de las actividades de un sector determinado, y a través del modelado de estos procesos se obtiene una visión más amplia del negocio existente. RUP propone, que para esta disciplina se modele el negocio siempre y cuando los procesos sean fácilmente identificables, en el caso de que no puedan percibirse se realice entonces un modelo de dominio que englobe los principales conceptos que sean encontrados en este modelo, y se proceda al Flujo de Trabajo de Requerimientos.

Debido a que no se han identificado objetivamente los problemas se decide desarrollar el modelo de dominio para capturar los tipos de objetos más importantes que existen o los eventos que suceden en el entorno donde estará el sistema, pretendiendo comprender mejor los conceptos asociados a éste. Se expondrá un vocabulario común para poder entender el contexto en que se encuentra el sistema, a continuación se describen los conceptos identificados.

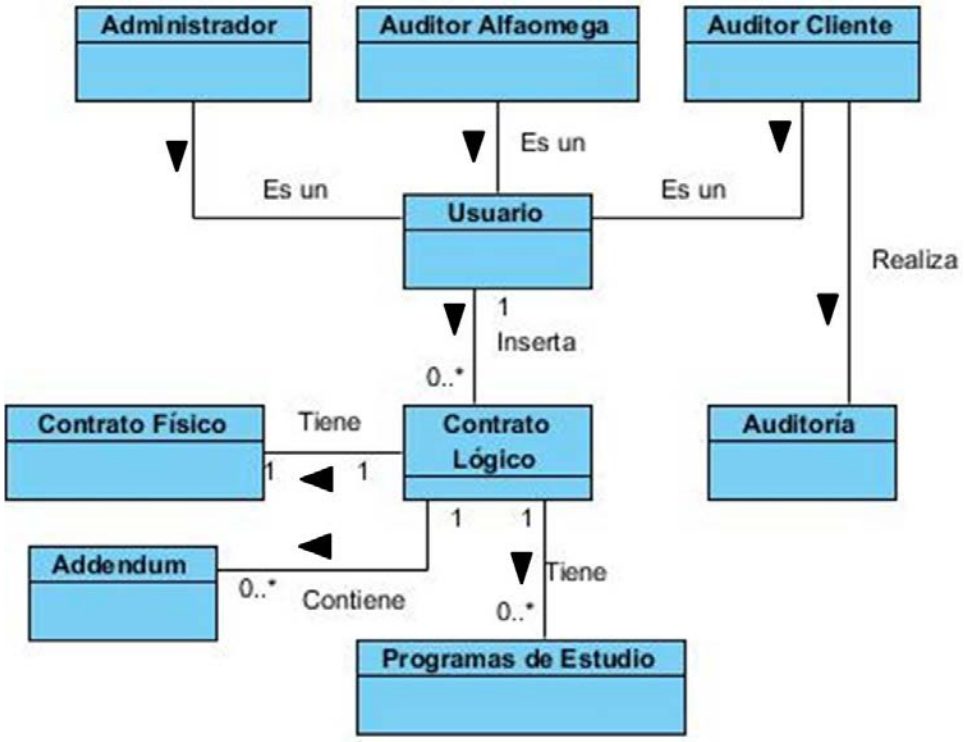

Figura 2: Representación del Modelo de Dominio.

En la figura 2 se representan los principales conceptos que resultan de interés para el problema planteado, a continuación se detallan cada uno de los elementos que lo componen:

- Usuario: Es el destinado a realizar las tareas comunes de los usuarios Auditor Alfaomega, Auditor cliente y el Administrador.
- Auditor Alfaomega: Usuario mexicano destinado a incluir los contratos realizados entre una escuela y la empresa Alfaomega.
- Auditor cliente: Usuario cubano destinado a auditar la Plataforma Educativa Zera.
- Administrador: Usuario destinado a realizar las tareas de administración.
- Auditar: Se denomina auditar, a la forma que tiene el auditor cliente de verificar la veracidad del contrato, y que esté en correspondencia con la base de datos de la Plataforma Educativa Zera.

- Contrato Físico: Es el documento establecido para recoger bajo qué acuerdos una escuela utilizará los programas de estudio.
- Contrato Lógico: Es el contrato realizado por el auditor en el software a desarrollar.
- Addendum: Suplementos de un contrato.
- Programas de Estudio: Contenidos educativos o HEA tales como matemática, física, química y biología.

2.3 Requisitos del software

"Un requisito es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste. En el otro extremo, es una definición detallada y formal de una función del sistema." (45)

2.1.1 *Requisitos Funcionales*

Los requisitos funcionales permiten expresar una especificación más detallada de las responsabilidades del sistema que se propone. Ellos ayudan a determinar de una manera clara, lo que debe hacer el mismo.

RF-1 Autenticar Usuario. El sistema debe propiciar autenticar los usuarios.

RF-2 Gestionar Usuario.

2.1 Incluir Usuario. Para ello se deben introducir mediante un formulario los datos pertenecientes al usuario, mostrados a continuación:

- Usuario: Identificador del usuario.
- Nombre.
- Apellidos.
- Contraseña.
- Correo.
- Teléfono.

- Activo: Muestra si el usuario está activo.
- Rol

2.2 Ver datos de Usuario. Para ver los datos de usuario se accede al mismo.

2.3 Modificar Usuario. Se podrá acceder a través de vínculo editar, pertenecientes al usuario que se desea modificar.

2.4 Eliminar Usuario. Se accede a través de vínculo eliminar perteneciente a dicho usuario.

RF-3 Ver Rol.

RF-4 Modificar Rol

RF-5 Incluir Contrato. Para ello se debe introducir los datos pertenecientes al contrato. Esta opción estará habilitada según los permisos del usuario autenticado. A través de un formulario se introducirán los siguientes datos.

- Número de contrato.
- Descripción.
- Moneda de pago.
- Escuela. Escuela que solicita el contrato.
- Estado. Muestra si el contrato está en negociación o en ejecución.
- Contrato. Documento firmado entre una escuela y Alfaomega, puede ser en formato word o pdf.

RF-6 Ver Contrato. Para ver los datos del contrato se accede al mismo, el sistema visualizará las opciones de modificar y eliminar contrato en dependencia de los permisos del usuario autenticado.

RF-7 Modificar Contrato. Se podrá acceder a través del vínculo editar perteneciente al contrato que se desea modificar.

RF-8 Eliminar Contrato. Se accede a través de vínculo eliminar perteneciente al contrato.

RF-9 Cerrar Contrato. Se podrá acceder a través del vínculo cerrar contrato, perteneciente a dicho contrato.

RF-10 Auditar Contrato. Se podrá acceder a través del vínculo auditar contrato, perteneciente a dicho contrato.

RF-11 Gestionar Programa de estudio. Para ello se deben haber introducido los datos pertenecientes al contrato. Luego se pasará a la gestión del Programa de Estudio.

11.1 Incluir Programa de Estudio. Para ello se deben introducir mediante un formulario los datos pertenecientes al programa de estudio, mostrados a continuación:

- Nombre del programa de estudio.
- Descripción.
- % de ganancia del programa de estudio.
- Cantidad de licencias de estudiantes.
- Costo de licencias de estudiantes.
- Cantidad de licencias de profesores.
- Costo de licencias de profesores.
- Cantidad de licencias de cortesía.
- Costo de licencias de cortesía.

11.2 Ver datos de Programa de Estudio. Para ver los datos del Programa de Estudio se accede al mismo dentro del contrato.

11.3 Modificar Programa de Estudio. Se podrá acceder a través de vínculo “Editar Programas”, perteneciente al programa de estudio que se desea modificar.

11.4 Eliminar Programa de Estudio. Se accede a través de botón eliminar perteneciente a dicho programa de estudio.

2.1.2 Requisitos no funcionales del sistema

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Estos requisitos en la aplicación han sido agrupados por categorías y se muestran a continuación.

Usabilidad

La herramienta propuesta podrá ser usada por cualquier persona autorizada y que posea conocimientos básicos en el manejo de una computadora. Debe ser una herramienta sencilla, manejable y fácil de usar para el usuario.

Fiabilidad

La precisión y exactitud de las salidas del sistema se corresponden con la calidad y exactitud de la información contenida en las base de datos desde donde se extraigan los datos. El sistema no es responsable por la falta de veracidad de dicha información, algunos errores que pueden resultar críticos son:

- Que salgan de funcionamiento las bases de datos desde donde se extraen los datos externos o que no exista conectividad hacia ellas.
- Que falle el servidor donde se despliegue la solución.

Eficiencia

La eficiencia del sistema depende en gran medida de la velocidad de conexión a la base de datos donde se encuentre, así como del volumen de información contenido en las mismas. Sin embargo el sistema debe mantener tiempos de respuestas en un marco razonable, con tal fin se implementará un mecanismo de paginación de los reportes que permitirá que los mismos sean obtenidos de manera progresiva disminuyendo así el tráfico por la red de la información y las consultas que devuelven grandes cantidades de *tuplas*.

Restricciones de diseño e implementación

El sistema deberá ser implementado en el lenguaje de programación PHP versión 5.3 o superior. Como marco de trabajo se utilizará Symfony en su versión 2.0.11, el cual propone una arquitectura modular en tres capas: modelo, vista y controlador. Unas de las bibliotecas fundamentales en el desarrollo de la herramienta es la de EXT JS versión 3.3.0 la cual es una librería en Javascript que logra el diseño de interfaces visuales interactivas usando tecnologías como AJAX y ofrece la posibilidad de crear aplicaciones web con la apariencia de aplicaciones de escritorio. Como IDE de desarrollo se utilizará NetBeans 6.9 y como gestor de base de datos se utilizará PostgreSQL 9.0.3.

Requerimientos de Software

El servidor donde se instalará la aplicación y la base de datos debe cumplir con los siguientes requisitos.

- Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 11.04 o superior.
- Paquetes: apache2, php5, libapache2-modphp5, php5-cli, php5-mysql, php5-pgsql, php5-sqlite3, sqlite3, php5-xsl, php5-gd.
- PostgreSQL versión 9.0.3.
- PGAdminIII o algún administrador para PostgreSQL.
- Usuario con privilegios para instalar la base de datos.

Requerimientos de Hardware

El servidor donde se instalará la aplicación debe cumplir con los siguientes requisitos.

- Procesador Intel Pentium 4 1.7 GHz.
- 1GB de memoria RAM.
- 10 GB de espacio en disco duro.

El servidor donde se instalará la base de datos debe cumplir con los siguientes requisitos

- Procesador Intel Pentium 4 1.7 GHz.
- 1GB de memoria RAM.
- 10 GB de espacio en disco duro.

La máquina utilizada para acceder a la aplicación debe cumplir con los siguientes requisitos

- Procesador 1.7 GHz, o superior.
- 512 MB RAM.
- 5 GB de espacio en disco duro.

2.2 Modelo de Casos de Uso del Sistema.

En esta sección se muestra el modelo de casos de uso del sistema y en él se detallan los diagramas de caso de uso del sistema que representan gráficamente a los procesos y su interacción con los actores.

2.2.1 Descripción de los actores del sistema.

Actor	Descripción
Administrador	Este rol tiene permisos para gestionar usuarios, lo cual permite realizar acciones de inserción, eliminación y modificación de los mismos. Además se encarga de ver los roles y modificar algunos de sus parámetros y puede realizar las funciones de los otros actores.
Auditor Alfaomega	Este Rol tiene como permiso incluir y ver los contratos en el sistema y modificar algunos de sus parámetros.

Auditor Cliente	Usuario máximo responsable de la fidelidad de la auditoría, interviene en la gestión de la exploración de una Auditoría, en la ejecución y terminación de Auditorías y Comprobaciones, como parte del proceso de auditoría analiza gran cantidad de datos, utilizando diferentes técnicas para obtener un informe final que le permita arribar a conclusiones. La función principal es la de auditar la Plataforma Educativa Zera.

Tabla 1: Descripción de los actores del sistema.

2.2.2 Patrones de Casos de Uso (CU)

Son comportamientos que deben existir en el sistema, ayudan a describir qué es lo que el sistema debe hacer, es decir, describen el uso del sistema y cómo este interactúa con los usuarios. Estos patrones son utilizados generalmente como plantillas que describen como deberían ser estructurados y organizados los casos de uso, son patrones que capturan mejores prácticas para modelar casos de uso. Entre algunos de los patrones a utilizar están los siguientes:

Patrón CRUD

El patrón CRUD Completo consiste en un caso de uso para administrar la información (CRUD Información), permitiendo modelar las diferentes operaciones para administrar una entidad de información, tales como crear, leer, cambiar y eliminar o dar de baja. (46)

Múltiples Actores

Roles comunes. Puede suceder que los dos actores jueguen el mismo rol sobre el CU. Este rol es representado por otro actor, heredado por los actores que comparten este rol. Es aplicable

cuando, desde el punto de vista del caso de uso, solo exista una entidad externa interactuando con cada una de las instancias del caso de uso. (47)

2.2.3 Diagrama de Casos de Uso del Sistema

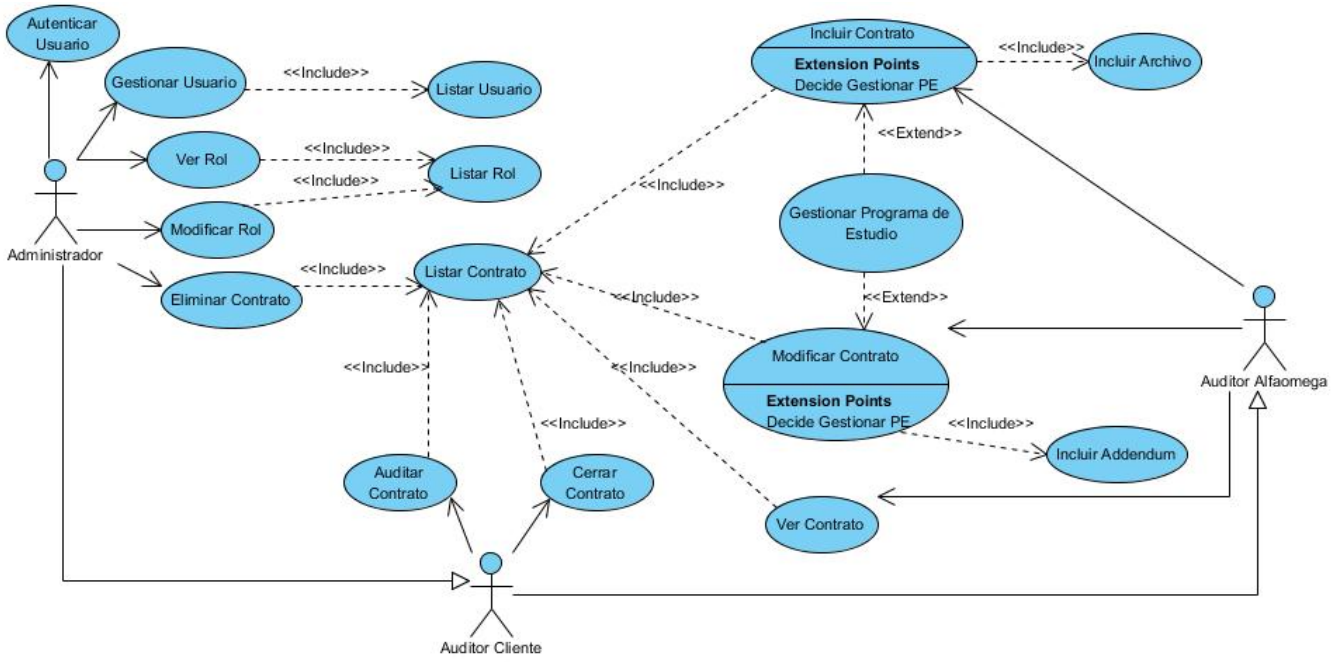

Figura 3: Diagrama de Casos de Uso del Sistema.

2.2.4 Descripción de los Casos de Uso del Sistema

A continuación se muestran las descripciones de los CUS críticos.

Caso de Uso:	Gestionar Usuario
Actores:	Administrador (Inicia)
Resumen:	El caso de uso inicia cuando el administrador solicita insertar, ver, modificar o eliminar un usuario. El sistema brinda dichas posibilidades y finaliza el caso de uso.
Precondiciones:	Usuario autenticado.
Referencias	RF-2

<table>
<tr><td>Prioridad</td><td colspan="2">Crítico</td></tr>
<tr><td colspan="3">Sección “Principal”</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>1. Solicita insertar, ver, modificar o eliminar un usuario.</td><td colspan="2">2. Ejecuta alguna de las siguientes acciones:
a) Si el administrador decide insertar un usuario, ir a la sección "Nuevo Usuario".
b) Si el administrador decide ver un usuario, ir a la sección "Ver Usuario".
c) Si el administrador decide modificar un usuario, ir a la sección "Editar Usuario".
d) Si el administrador decide eliminar un usuario, ir a la sección "Eliminar Usuario”.</td></tr>
<tr><td colspan="3">Sección “Insertar Usuario”</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>1. Selecciona la opción nuevo usuario desde la lista de usuario.</td><td colspan="2">2. Muestra la interfaz usuario con un formulario solicitando los siguientes datos:
• Usuario: Identificador del usuario.
• Nombre.
• Apellidos.
• Contraseña.
• Correo.
• Teléfono.</td></tr>
</table>

	• Activo: Muestra si el usuario está activo. • Rol. Y permite: • Guardar: Permite guardar los datos del nuevo usuario. • Cancelar: Cancela la creación del usuario.
3. Introduce los datos solicitados y selecciona la opción Guardar.	4. Comprueba que los campos contengan información.
	5. Almacena los datos del usuario.
	6. Permite ver los datos introducidos por el administrador.
Flujos Alternos	
Flujo Alterno 3a"Selecciona la opción Guardar"	
Acción del Actor	**Respuesta del Sistema**
	3a. Almacena los datos, muestra los datos del usuario y finaliza el caso de uso.
Flujo Alterno 3b"Selecciona la opción Cancelar"	
Acción del Actor	**Respuesta del Sistema**
	3b. Regresa al paso 1 del Flujo Normal de Eventos de la sección "Principal" y finaliza el caso de uso.
Flujo Alterno 4a"Campos vacíos"	
Acción del Actor	**Respuesta del Sistema**
	4a. Emite un mensaje indicando que se deben llenar los campos obligatorios, regresa al paso 2 del Flujo Normal de Eventos de la sección

	"Insertar Usuario" y finaliza el caso de uso.
Sección "Ver Usuario"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción ver un usuario.	2. Muestra la interfaz usuario con un formulario mostrando los siguientes datos: • Nombre. • Apellidos. • Usuario. • Está activo. • Está logueado. • Última vez que accedió. • Fecha de creado. • Correo. • Teléfono. Y permite: • Cancelar: Cancela la muestra de datos del usuario.
3. Selecciona la opción cancelar.	4. Regresa a la lista de usuario.
Flujos Alternos	
Flujo Alterno 3a"Selecciona la opción Cancelar"	
Acción del Actor	**Respuesta del Sistema**
	3a. Regresa al paso 1 del Flujo Normal de Eventos de la sección "Principal" y finaliza el caso de uso.
Sección "Modificar Usuario"	
Flujo Normal de Eventos	

Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción editar usuario desde la lista de usuario.	2. Muestra la interfaz usuario con un formulario solicitando los siguientes datos: • Usuario: Identificador del usuario. • Nombre. • Apellido. • Contraseña. • Correo. • Teléfono. • Activo: Muestra si el usuario está activo. • Rol. Y permite: • Guardar: Permite guardar los datos del nuevo usuario. • Cancelar: Cancela la creación del usuario.
3. Modifica los datos deseados y selecciona la opción Guardar.	4. Comprueba que los campos contengan información.
	5. Actualiza los datos del usuario.
	6. Muestra los datos del usuario modificado y finaliza el caso de uso.
Flujos Alternos	
Flujo Alterno 3a“Selecciona la opción Guardar”	
Acción del Actor	**Respuesta del Sistema**
	3a.1. Almacena los datos, muestra los datos del usuario y finaliza el caso de uso.
Flujo Alterno 3c“Selecciona la opción Cancelar”	

Acción del Actor	Respuesta del Sistema
	3c.1. Regresa al paso 1 del Flujo Normal de Eventos de la sección "Principal" y finaliza el caso de uso.
Flujo Alterno 4a"Campos vacíos"	
Acción del Actor	**Respuesta del Sistema**
	4a.1. Emite un mensaje indicando que se debe llenar los campos obligatorios, regresa al paso 2 del Flujo Normal de Eventos de la sección "Modificar usuario" y finaliza así el caso de uso.
Sección "Eliminar Usuario"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Elige el usuario que desea eliminar y selecciona la opción "Eliminar".	2. Muestra el siguiente mensaje: "Confirma que desea eliminar".
3. Acepta el mensaje de confirmación.	4. Elimina los datos del usuario y finaliza así el caso de uso.
Flujos Alternos	
Flujo Alterno 3a"Selecciona la opción Cancelar"	
Acción del Actor	**Respuesta del Sistema**
	3a.1. Cancela la eliminación, regresa al curso y finaliza el caso de uso.

Tabla 2: Descripción del CU Gestionar Usuario.

Caso de Uso:	Incluir Contrato
Actores:	Auditor Alfaomega, Auditor Cliente o Administrador (Inician)
Resumen:	El caso de uso inicia cuando el Auditor Alfaomega, Auditor Cliente o Administrador solicitan incluir un contrato. El sistema brinda dicha posibilidad

<table>
<tr><td></td><td colspan="2">y finaliza el caso de uso.</td></tr>
<tr><td>Precondiciones:</td><td colspan="2">Usuario autenticado.</td></tr>
<tr><td>Referencias</td><td colspan="2">RF-5</td></tr>
<tr><td>Prioridad</td><td colspan="2">Crítico</td></tr>
<tr><td colspan="3">Sección “Principal”</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>3. Solicita incluir un contrato.</td><td colspan="2">4. Ejecuta la siguiente acción:
e) Si el Auditor Alfaomega, Auditor Cliente o Administrador decide incluir un contrato, ir a la sección "Nuevo Contrato".</td></tr>
<tr><td colspan="3">Sección “Incluir Contrato”</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>7. Selecciona la opción nuevo contrato desde la lista de contrato.</td><td colspan="2">8. Muestra la interfaz contrato con un formulario mostrando los siguientes datos:
• Número de contrato.
• Descripción.
• Moneda de pago.
• Escuela. Escuela que solicita el contrato.
• Estado. Muestra si el contrato está en negociación o en ejecución.
• Contrato. Documento firmado entre una escuela y Alfaomega, puede ser en formato word o pdf.</td></tr>
</table>

	Y permite: • Guardar: Admite guardar los datos del nuevo contrato. • Cancelar: Cancela la creación del contrato. • Insertar programas de estudio. Si el usuario así lo desea, ir al CU extendido gestionar Programa de Estudio y a la sección de adicionar programa de estudio.
9. Introduce los datos solicitados y selecciona la opción Guardar.	**10.** Comprueba que los campos contengan información.
	11. Almacena los datos del contrato.
	12. Accede a ver los datos introducidos por el usuario.
Flujos Alternos	
Flujo Alterno 3a“Selecciona la opción Guardar”	
Acción del Actor	**Respuesta del Sistema**
	3a. Almacena los datos, muestra los datos del contrato y finaliza el caso de uso.
Flujo Alterno 3a“Selecciona la opción Cancelar”	
Acción del Actor	**Respuesta del Sistema**
	3b. Regresa al paso 1 del Flujo Normal de Eventos de la sección “Principal” y finaliza el caso de uso.
Flujo Alterno 4a“Campos vacíos”	
Acción del Actor	**Respuesta del Sistema**

	4a. Emite un mensaje indicando que se deben llenar los campos obligatorios, regresa al paso 2 del Flujo Normal de Eventos de la sección "Incluir Contrato" y finaliza el caso de uso.

Tabla 3: Descripción del CU Incluir Contrato.

Caso de Uso:	Modificar Contrato
Actores:	Auditor Alfaomega, Auditor Cliente o Administrador (Inician)
Resumen:	El caso de uso inicia cuando el Auditor Alfaomega, Auditor Cliente o Administrador solicitan modificar un contrato. El sistema brinda dicha posibilidad y finaliza el caso de uso.
Precondiciones:	Usuario autenticado.
Referencias	RF-7
Prioridad	Crítico
Sección "Principal"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Solicita modificar un contrato.	2. Ejecuta la siguiente acción: a) Si el Auditor Alfaomega, Auditor Cliente o Administrador decide modificar un contrato, ir a la sección "Editar Contrato".
Sección "Modificar Contrato"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción editar contrato desde	2. Muestra la interfaz contrato con un formulario

la lista de contrato.	mostrando los siguientes datos: • Número de contrato. • Descripción. • Moneda de pago. • Escuela. Escuela que solicita el contrato. • Estado. Muestra si el contrato está en negociación o en ejecución. • Contrato. Documento firmado entre una escuela y Alfaomega, puede ser en formato word o pdf. Y permite: • Modificar en negociación. Si el contrato está en negociación permite modificar todos los datos excepto la escuela que realiza el contrato. • Modificar en ejecución. Si el contrato está en ejecución solo permitirá modificar la descripción. • Guardar: Permite guardar los datos modificados del contrato. • Cancelar: Cancela la modificación del contrato. • Modificar programas de estudio. Si el usuario así lo desea, ir al CU extendido Gestionar Programa de Estudio. • Adicionar *addendum*. Permite al usuario adicionar el documento donde se

	muestren las modificaciones de contrato, puede ser en formato word o pdf.
3. Modifica los datos deseados y selecciona la opción Guardar.	4. Comprueba que los campos contengan información.
	5. Almacena los datos modificados del contrato.
	6. Permite ver los datos modificados por el usuario.
Flujos Alternos	
Flujo Alterno 3a"Selecciona la opción Guardar"	
Acción del Actor	**Respuesta del Sistema**
	3a. Almacena los datos, muestra los datos del contrato y finaliza el caso de uso.
Flujos Alternos	
Flujo Alterno 3b"Selecciona la opción Cancelar"	
Acción del Actor	**Respuesta del Sistema**
	3b. Regresa al paso 1 del Flujo Normal de Eventos de la sección "Principal" y finaliza el caso de uso.
Flujo Alterno 4a"Campos vacíos"	
Acción del Actor	**Respuesta del Sistema**
	4a. Emite un mensaje indicando que se deben llenar los campos obligatorios, regresa al paso 2 del Flujo Normal de Eventos de la sección "Incluir Contrato" y finaliza el caso de uso.

Tabla 4: Descripción del CU Modificar Contrato.

2.3 Conclusiones Parciales

En el presente capítulo partiendo de la propuesta de RUP para el desarrollo de software, se realizó una modelación del sistema, para mejor comprensión de la estructura y dinámica de los procesos que se desarrollan en el entorno hacia el cual se ejecuta la aplicación. Además, se muestra un listado de requisitos funcionales y no funcionales del sistema en general, como resultado de esto se obtuvo el diagrama de casos de uso del sistema donde se definieron los actores y casos de uso así como las relaciones entre los mismos.

3

CAPÍTULO 3

ANÁLISIS Y DISEÑO DEL SISTEMA

Introducción

En el presente capítulo se expone el análisis y diseño del sistema que se va a desarrollar. Se efectúa el modelado de los artefactos necesarios que contribuyan a la implementación del sistema. Estos artefactos son el Modelo de Análisis y el Modelo de Diseño, los cuales se encargan de describir en términos de clases cómo deben funcionar los casos de uso del sistema.

3.1 Modelo de análisis

El modelo de análisis es la primera representación técnica del sistema, por lo que puede considerarse como una primera aproximación al modelo de diseño, y es por tanto una entrada fundamental para las actividades de diseño e implementación subsiguientes. A partir de su realización se refinan y estructuran los requisitos obtenidos con anterioridad proporcionando una visión general del sistema. El modelo de análisis debe lograr tres objetivos primarios:

- Describir lo que requiere el cliente.
- Establecer una base para la creación de un diseño de software.
- Definir un conjunto de requisitos con los que se pueda validar una vez que se construye el software. (48)

3.1.1 Diagramas de clases

Los diagramas de clases son diagramas de estructura estática que muestran las clases del sistema y sus interrelaciones, estos son el pilar básico del modelado con UML. Siendo utilizados tanto para mostrar lo que el sistema puede hacer (análisis), como para mostrar cómo puede ser construido (diseño).

Los diagramas de clases son los más utilizados en el modelado de sistemas orientados a objetos. Un diagrama de clases muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Estos diagramas también son la base para un par de diagramas relacionados: los diagramas de componentes y los de despliegue. Los diagramas de clases son importantes no sólo para visualizar, especificar y documentar modelos estructurales, sino también para construir sistemas ejecutables, aplicando ingeniería directa e inversa. (49)

A continuación se muestran los principales diagramas de clases del análisis para varios casos de uso del sistema.

3.1.2 Diagramas de clases del análisis

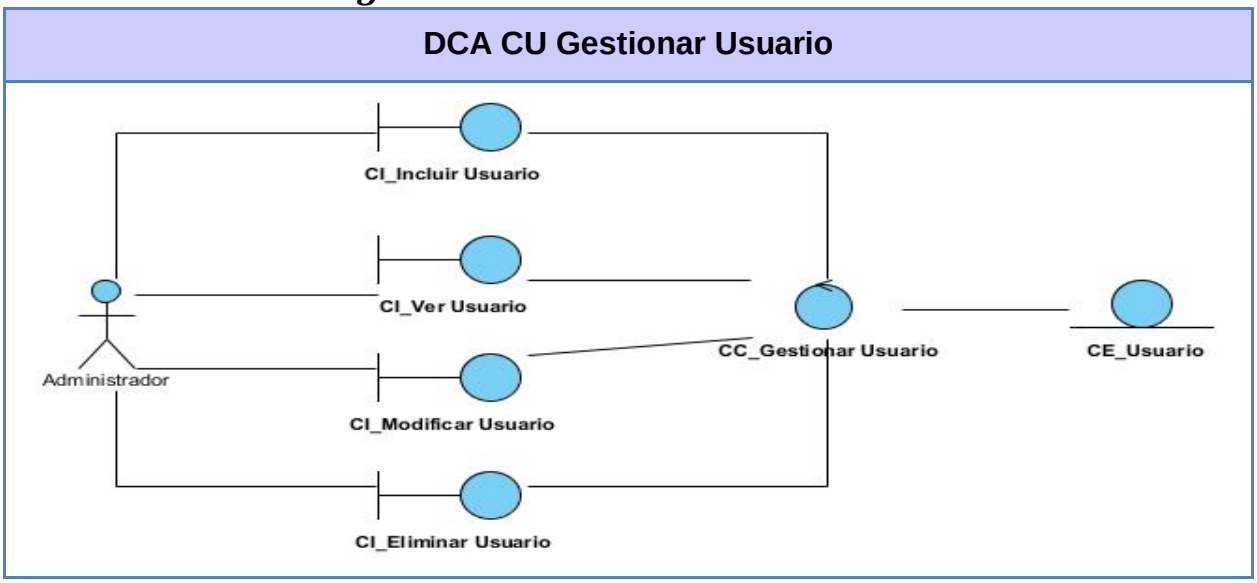

Figura 4: DCA_CU Gestionar Usuario.

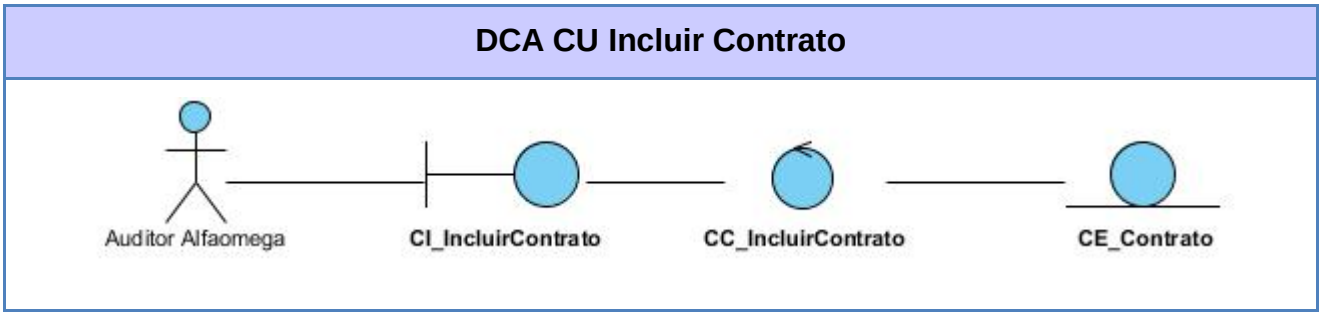

Figura 5: DCA_ CU Incluir Contrato.

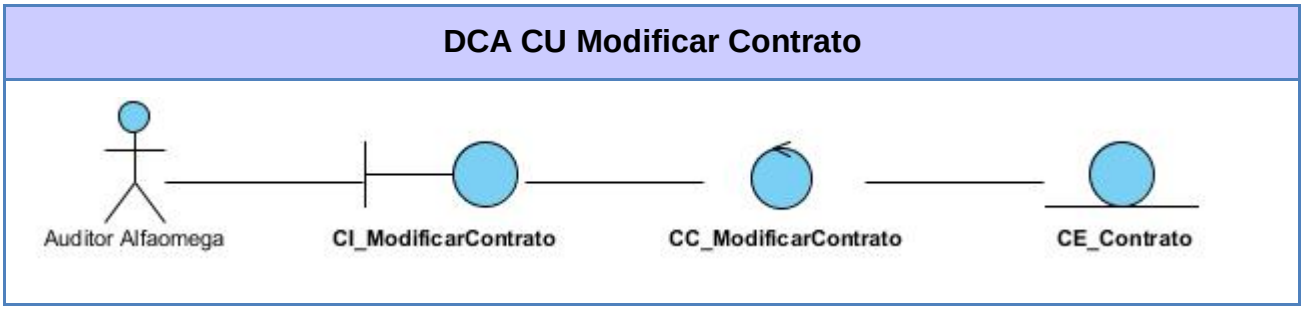

Figura 6: DCA CU Modificar Contrato.

3.2 Patrón arquitectónico Modelo – Vista – Controlador en "Symfony"

Con el MVC se consigue la separación de responsabilidades.

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

El principio más importante de la arquitectura MVC es la separación del código del programa en tres capas, dependiendo de su naturaleza. La lógica relacionada con los datos se incluye en el modelo, el código de la presentación en la vista y la lógica de la aplicación en el controlador.

La capa del modelo se puede dividir en la capa de acceso a los datos y en la capa de abstracción de la base de datos. De esta forma, las funciones que acceden a los datos no utilizan

sentencias ni consultas que dependen de una base de datos, sino que utilizan otras funciones para realizar las consultas. Así, si se cambia de sistema gestor de bases de datos, solamente es necesario actualizar la capa de abstracción de la base de datos.

La capa de la vista se separa en un layout y en una plantilla; pues las páginas web suelen contener elementos que se muestran de forma idéntica a lo largo de toda la aplicación, el layout es global en toda la aplicación o al menos en un grupo de páginas, la plantilla sólo se encarga de visualizar las variables definidas en el controlador, el contenido de la plantilla se integra en el layout, o si se mira desde el otro punto de vista, el layout decora la plantilla. Este comportamiento es una implementación del patrón de diseño “Decorator”.

El controlador suele tener mucho trabajo, Entre las tareas comunes se encuentran el manejo de las peticiones del usuario, el manejo de la seguridad, cargar la configuración de la aplicación, etcétera. Por este motivo, el controlador se divide en un controlador frontal, que es único para cada aplicación, y las acciones, que incluyen el código específico del controlador de cada página.

Active Record : Una de las virtudes del patrón es representar de forma Orientada a Objetos los datos de una Base de Datos Relacional modelo conocido también como ORM o “Object-Relational Mapping”, definiendo interfaces sencillas para acceder y manipular esos datos.

Active Table: El Active Table es un objeto que puede persistir otros objetos en medios no volátiles, como bases de datos. La función de una clase que implementa este patrón es de comunicarse con la base de datos y regresar objetos como los definidos en la aplicación.

Singleton: Su función es almacenar una referencia a todos los objetos que forman el núcleo de Symfony.

Front Controller: Al situarse dentro del patrón MVC, el 'Front Controller' es el componente que recibe los requerimientos, los envía a los elementos encargados de procesar la lógica y luego lo envía nuevamente a la vista (esta vez incluyendo los datos obtenidos). Ofrece un punto de entrada único para toda la aplicación. Así, en caso de que sea necesario impedir el acceso a la aplicación, solamente es necesario editar el script correspondiente al controlador frontal. (50)

3.3 Aplicación de los patrones de diseño en Symfony

El framework Symfony utiliza en su implementación una serie de patrones de diseño, los cuales clasifican y describen formas de solucionar problemas específicos y comunes del diseño orientado a objetos. A continuación se explicarán algunos de los patrones utilizados directamente en la solución.

Observer (Observador): Este patrón permite definir una dependencia entre objetos de forma que cuando un objeto cambia de estado, todos sus objetos dependientes son notificados y actualizados.

Strategy (Estrategia): El objetivo de este patrón es definir un grupo de clases que representan un conjunto de posibles comportamientos, estos pueden ser fácilmente intercambiados en una aplicación, modificando la funcionalidad en cualquier instante. Este patrón se emplea en la gestión los contratos, modificando un comportamiento específico dependiendo del tipo de usuario que haga uso de las funcionalidades incluir, modificar, eliminar y ver contrato.

Decorator (Decorador): El objetivo de este patrón es extender la funcionalidad de un objeto dinámicamente. Se utiliza a la hora de cargar el contenido dinámico de las plantillas del contrato, dentro de otra plantilla más general, la que extiende su funcionamiento a la primera.

Creador: Este patrón se usa para establecer la responsabilidad de quién es el encargado de crear objetos de una clase determinada. Es usado en la mayoría de las acciones de los módulos de symfony, creando objetos entidades, de formularios y de clases de abstracción a la base de datos.

3.4 Modelo de diseño

Los diagramas de clases que se presentan para el diseño del sistema propuesto, han sido elaborados bajo el criterio de distribuirlos por paquetes, resultando evidenciar los diferentes estados y etapas por los que transcurren las acciones de auditorías y control, así como los documentos que se manipulan de dichas acciones de control.

Los diagramas de clases del diseño son una representación estática de cómo va a fluir la interacción entre, peticiones del cliente, o sea, su interacción con la interfaz, que son en este caso las páginas clientes, la entrada de datos a través de los formularios y el acceso a los mismos, donde entran a accionar las páginas servidoras para dar respuesta a dichas peticiones. Se representa en cada uno de los diagramas, una clase interfaz principal que contiene un formulario donde se listan las acciones de control, pues para llegar a ejecutar las operaciones sobre dichas acciones de control se hace necesario el uso de este componente; por lo que se consideró necesario incluir para un mayor entendimiento del diseño web. El diagrama de clases web, fue definido, a partir de los diferentes casos de uso del sistema y empleando las extensiones de UML para web.

3.4.1 Diagrama de clase del diseño.

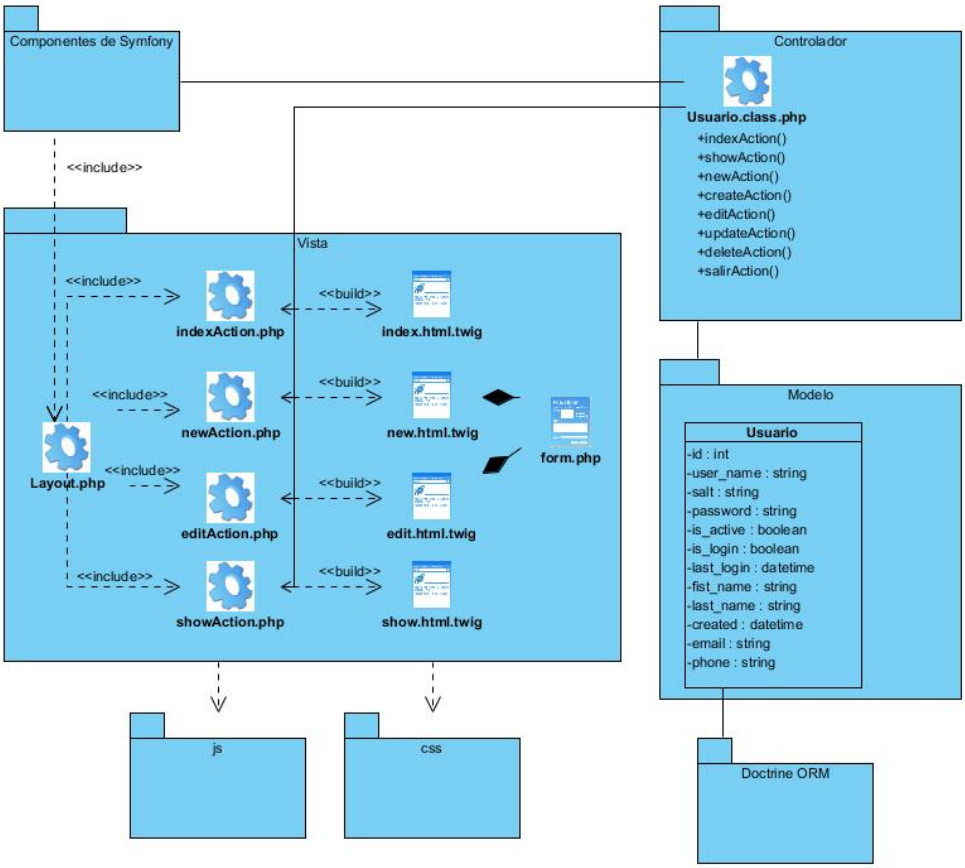

Figura 7: DCD_CU Gestionar Usuario.

3.5 Diseño de la base de datos

En este epígrafe se muestra el diseño de la base de datos del sistema propuesto a través de los esquemas de la base de datos generados a partir de este, con el modelo de datos.

Figura 8: Diseño de la base de datos.

3.6 Descripción de las tablas de la base de datos.

Tb_User		
Descripción: Esta tabla se encarga de guardar los datos del usuario.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador del usuario
username	varchar	Usuario dentro del software
Salt	varchar	Genera un código aleatorio para la contraseña.
password	varchar	Permite dar seguridad al usuario.
first_name	varchar	Primer nombre del usuario dentro del software.
last_name	varchar	Apellidos del usuario dentro del software.
created	date	Muestra la fecha en que es dado de alta el usuario.
description	text	Describe bajo qué condiciones se encuentra el

		usuario en el software.
email	varchar	Correo que utilizará el usuario para la comunicación con el sistema.
phone	varchar	Número telefónico del usuario (celular).
is_active	bool	Muestra si el usuario está activo.
is_login	bool	Muestra si el usuario está dentro del software.
last_login	timestamp	Muestra el tiempo que el usuario lleva dentro del software.

Tabla 5: Descripción de la tabla Usuario.

Tb_Agreement		
Descripción: Esta tabla se encarga de guardar los principales datos del contrato.		
Atributo	**Tipo**	**Descripción**
Id	integer	Identificador del contrato.
nro_agreement	varchar	Número especial que identifica al contrato.
description	text	Describe el contrato.
school_name	varchar	Nombre de la escuela que realiza el contrato.
created_agreement	datatime	Muestra la fecha en que es dado de alta el contrato.
state	Varchar	Estado que tiene el contrato. Muestra si está activo o no.

Tabla 6: Descripción de la tabla tb_agreement.

3.7 Conclusiones parciales

El capítulo abordado expuso los resultados de la etapa de Análisis y Diseño, son presentados los diagramas de clases del análisis y del diseño, además se explican los estándares y principios que fueron tenidos en cuenta a la hora de diseñar la interfaz de la aplicación, se ilustran los diagramas que se desarrollan para apoyar el diseño de la Base de Datos, es decir el Diagrama de Entidad Relación. Todos estos artefactos conforman una propuesta de solución para implementar el software de auditoría de la Plataforma Educativa Zera.

4

CAPÍTULO 4

Implementación y Prueba

4 Introducción

En este capítulo se realiza el diagrama de componentes para estructurar el modelo de implementación y se muestra la ubicación física de los nodos de procesamiento a través del diagrama de despliegue. Se realizan, pruebas de caja negra para comprobar el correcto funcionamiento del software, se verifica que la entrada de datos se acepte de forma adecuada y que se produzca un resultado correcto.

4.1 Implementación

La implementación es cómo desarrollar y organizar los componentes basándose en las especificaciones del diseño. Es la realización de las especificaciones técnicas o algoritmos como un programa, componente software, u otro sistema de cómputo. Dentro de los principales objetivos que se desea alcanzar en una etapa de implementación se encuentran:

- Definir la organización del código, en términos de los subsistemas de implementación organizados en capas.

- Implementar los elementos de diseño en términos de los elementos de implementación (archivos de origen, binarios, programas ejecutables y otros).

- Probar y desarrollar componentes como unidades.(49)

4.2 Vista de Despliegue

La vista de despliegue muestra las relaciones físicas de los distintos nodos que componen un sistema y muestra la forma en que están ubicados los componentes sobre dichos nodos, formando un grafo de nodos unidos por conexiones de comunicación. Un nodo puede contener instancias de componentes, software, objetos y procesos. En general un nodo será una unidad.

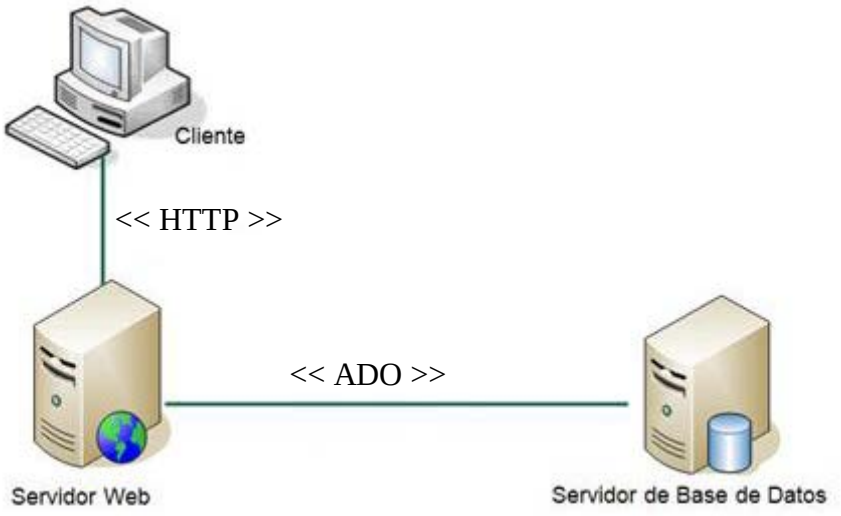

Figura 9: Diagrama de Despliegue.

Cliente: Ordenador cliente que se conecta a través de un navegador web al servidor donde estará instalado el software de auditoría.

Servidor web: Servidor donde estará instalado el software de auditoría.

Servidor de Base de Datos: Servidor donde se encuentra instalado la base de datos a auditar (Plataforma Educativa Zera).

4.3 Diagrama de Componentes

Un diagrama de componentes es un diagrama desarrollado en el lenguaje unificado de modelado (UML). Este diagrama representa como un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes.

Normalmente los diagramas de componentes contienen:

- Componentes: es una parte física de un sistema (módulo, base de datos, programa ejecutable, etc.). Se puede decir que un componente es la materialización de una o más clases, porque una abstracción con atributos y métodos pueden ser implementados en los componentes.

Los componentes se pueden agrupar en paquetes, además puede haber entre ellos relaciones de dependencia como generalización, asociación, agregación, realización.

- Interfaces: es el lazo de unión entre varios componentes.
- Paquetes o subsistemas.

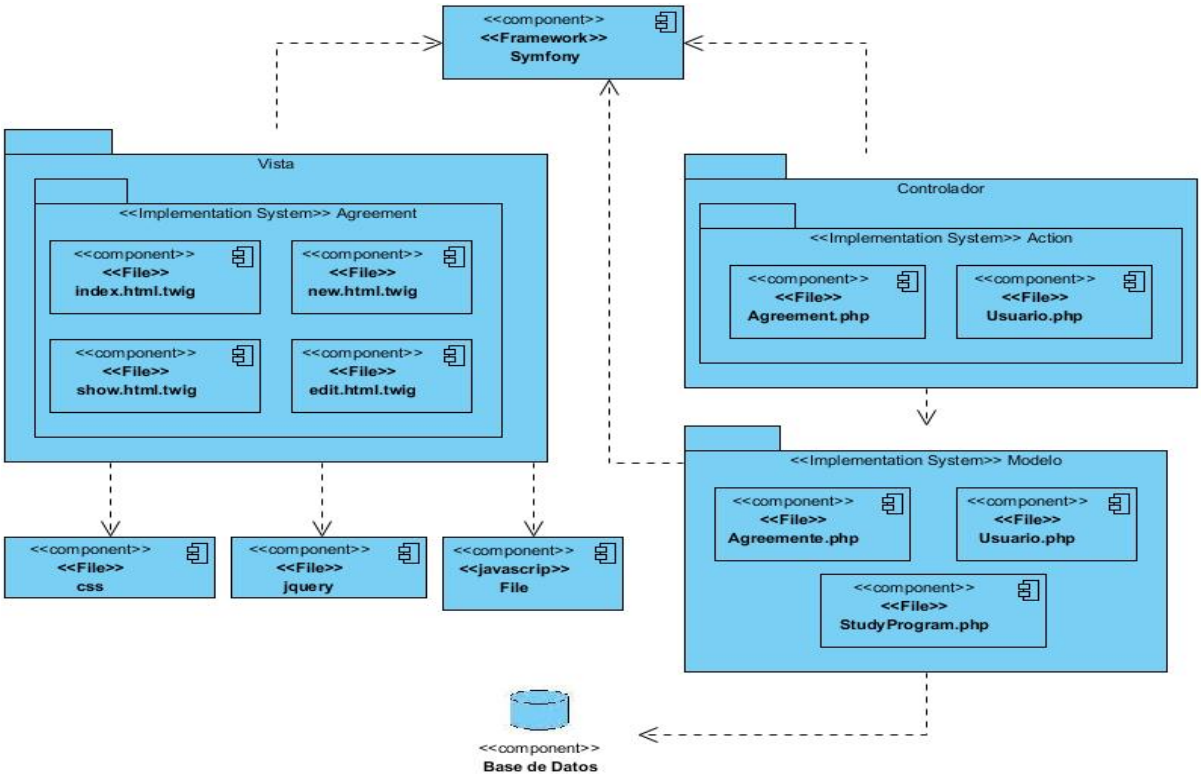

Figura 10: Diagrama de Componentes.

4.4 Pruebas de Software

La prueba de software es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación.

El objetivo de las pruebas al sistema es comprobar la integración del sistema de información globalmente, verificando el funcionamiento correcto de las interfaces entre los distintos subsistemas que lo componen y con el resto de los sistemas de información con los que se comunica. En la realización de estas pruebas es importante comprobar la cobertura de los requisitos, dado que su incumplimiento puede comprometer la aceptación del sistema por el equipo de operación responsable de realizar las pruebas de implantación del sistema, que se llevarán a cabo en el proceso Implantación y Aceptación del Sistema.

Pruebas Unitarias: Comienzan con la prueba de cada módulo. Una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de estos funcione correctamente por separado. El objetivo de las pruebas unitarias es aislar cada parte del programa y mostrar que las partes individuales son correctas. Proporcionan un contrato escrito que el fragmento de código debe satisfacer. Estas pruebas aisladas proporcionan cinco ventajas básicas; fomentan el cambio, simplifican la integración, documentan el código, separan la interfaz del código y hacen que los errores estén más acotados y sean fáciles de localizar. (51)

Pruebas de Integración: A partir del esquema del diseño, los módulos probados se vuelven a probar combinados para probar sus interfaces. Pruebas integrales o pruebas de integración son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las pruebas unitarias. Únicamente se refieren a la prueba o pruebas de todos los elementos unitarios que componen un proceso, hecha en conjunto, de una sola vez. Consiste en realizar pruebas para verificar que un gran conjunto de partes del software funcionan juntos. (52)

Pruebas del Sistema: El software ensamblado totalmente con cualquier componente hardware que requiera, se prueba para comprobar que se cumplen los requisitos funcionales. Cualquier pieza de software completo, desarrollado o adquirido, puede verse como un sistema que debe probarse, ya sea para decidir acerca de su aceptación, para analizar defectos globales o para estudiar aspectos específicos de su comportamiento, tales como seguridad o rendimiento. Este tipo de pruebas estudia el producto completo.

4.4.1 Métodos de Prueba

RUP propone dos métodos fundamentales: caja blanca y caja negra. A continuación se describen ambos métodos, haciéndose mayor énfasis en las pruebas de Caja Negra ya que serán las más utilizadas en la comprobación de la solución.

Pruebas de Caja Negra

Las pruebas de caja negra, también denominadas pruebas de comportamiento, se centran en los requisitos funcionales del software. O sea, permiten al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Estas pruebas no son una alternativa a las técnicas de pruebas de caja blanca, más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores que se llevan a cabo sobre la interfaz del software. El objetivo es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene, saber qué es lo que hace el software pero sin entrar en detalles de código, es decir, que es lo que hace, y no cómo lo hace (no se ve el código). Estas pruebas se centran principalmente en los requisitos funcionales del software y permiten encontrar: (53)

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

4.4.2 Diseño de Casos de Prueba

Diseño de Casos de Prueba: CU Gestionar Contrato
El caso de uso inicia cuando el Auditor solicita insertar, ver, editar o cerrar un contrato. En caso

de que seleccione la opción insertar, el sistema dará la posibilidad de insertar los datos del contrato según los permisos correspondientes al auditor autenticado. Si el actor selecciona la opción ver, el sistema mostrará los datos del contrato correspondiente. Si el actor elige la opción modificar, el sistema mostrará los datos del contrato a modificar según los permisos, y una vez realizados los cambios el sistema actualizará los datos. Si el actor selecciona la opción cerrar, el sistema ocultara los datos del contrato en cuestión, y concluye el CU cuando ha sido realizada la solicitud.

Condiciones de ejecución

El usuario debe haberse autenticado previamente como Auditor.

Debe de existir la escuela a la que se le va a realizar el contrato en la base de dates de Zera.

Para incluir un Contrato, el auditor debe llenar los datos correspondientes de acuerdo a los permisos correspondientes.

Para ver un Contrato este debe estar seleccionado previamente y el actor debe tener el permiso para ver el elemento.

Para modificar un Contrato debe estar seleccionado previamente y el actor debe ser el responsable temporal de la acción.

Para eliminar un Contrato debe estar seleccionado previamente y el actor debe tener el permiso de eliminar el elemento.

Escenario	Descripción	Campos	Respuesta del sistema
EC 1.1 “Selecciona acción”	Selecciona la opción de realizar una acción sobre un usuario.		Brinda la posibilidad de realizar las siguientes acciones: • Incluir nuevo Contrato • Ver los datos del Contrato Ver SC 2: “Ver datos del Contrato • Modificar los datos del Contrato Ver SC 3: “Modificar datos del Contrato

			• Eliminar Contrato Ver SC 4: “Eliminar Contrato
EC 1.2 “Selecciona incluir”	Selecciona la opción de incluir un nuevo Contrato.		Brinda la posibilidad de introducir o seleccionar los datos siguientes: • Nro_Contrato. Identificador de un Contrato • Descripción • Escuela. • Responsable. Persona que inserta el Contrato. Además permite: • Guardar los datos. • Cancelar la operación en cualquier momento.
EC 1.3 “Introduce datos”	Selecciona o introduce los datos para añadir el Contrato. Selecciona la opción de guardar los datos.	V	Valida los datos. Crea un nuevo Contrato. Muestra los datos del Contrato incluido. Ver SC 2: “Ver datos del Contrato” Muestra un mensaje de confirmación.

EC 1.4 “Selecciona cancelar”	Selecciona la opción de Cancelar.		Regresa al listado de Contrato Muestra un mensaje de confirmación.
EC 1.5 “Campos vacíos”	Existen datos incompletos.	I (Vacío) V (Vacío) V I (Vacío)	Muestra un mensaje de información. Muestra un indicador sobre los campos vacíos. Regresa al EC 1.3
EC 1.6 “Campos incorrectos”	Existen datos incorrectos.	I **(Vacío)** *V*	Muestra un mensaje de información. Muestra un indicador sobre

			cada uno de los campos incorrectos. Regresa al EC 1.3
		V	
		I	
		(Vacío)	
EC 1.7 "Ya existe"	El Contrato ya existe.	V	Muestra un mensaje de información. Muestra un indicador sobre el campo. Regresa al EC 1.3

Tabla 7: Diseño de Caso de Prueba Gestionar Contrato.

4.5 Resultados Obtenidos

Durante el desarrollo del software se realizaron pruebas unitarias para ir comprobando el funcionamiento. Estas no se planificaron ni se registraron sus resultados, fueron haciéndose a medida que la solución se desarrollaba. Para evaluar la solución se realizaron varias iteraciones donde se probó el software íntegramente. Prestando gran atención a las pruebas de integración con vista a probar funcionalidades que relacionan al software con la Plataforma Educativa Zera. Se especifican las principales pruebas realizadas a la aplicación haciendo uso del método de caja negra, estas pertenecen al nivel de prueba de sistema y se detallan a continuación.

Pruebas Internas: Son realizadas por el equipo de calidad interna del proyecto (en este caso las analistas) con el fin de entregar un producto con la menor cantidad de errores posibles. Se centraron en el cumplimiento de las funcionalidades descritas, en el listado de requerimientos y de casos de uso del sistema.

Módulo	Caso de Uso	No Conformidades			
		Alta	Media	Baja	Total
Auditoría	6	1	3	11	15

Tabla 8: Pruebas internas.

Pruebas Cruzadas: Fueron realizadas al sistema por el equipo de desarrollo del proyecto (analistas de diferentes módulos), con el fin de encontrar la mayor cantidad de errores posibles en término de validaciones, formato de los campos, etc.

Módulo	Caso de Uso	No Conformidades			
		Alta	Media	Baja	Total
Auditoría	6	1	3	11	15

Tabla 9: Pruebas cruzadas.

Conclusiones Parciales

En este capítulo se realizó el modelo de implementación con el propósito de mostrar los componentes del sistema y sus relaciones a través del diagrama de componentes. Mediante el diagrama de despliegue se mostró la configuración de los nodos de procesamiento en tiempo de ejecución y los vínculos de comunicación entre ellos. Además, se realizaron los casos de prueba para validar que los requisitos fueron implementados correctamente y se describen los casos de prueba que verifican el correcto funcionamiento del sistema a través de las pruebas de caja negra.

Conclusiones generales

Después de haber realizado el presente trabajo, se logró dar respuesta a los objetivos específicos que fueron definidos para desarrollar el sistema propuesto, cumpliendo así con las distintas etapas que se definieron para su realización, lográndose el objetivo de elaborar un software que le permita a Albet S.A. aplicar la auditoría externa a la Plataforma Educativa Zera.

Se realizó un estudio sobre los conceptos asociados a la investigación, entre los que se encuentran auditoría, sistemas de gestión de auditorías, servicios web, entre otros. Mediante la metodología utilizada se logró organizar y dirigir el desarrollo del sistema así como se escogió la arquitectura para la realización del software.

En la disciplina Modelamiento del Negocio se obtuvo el modelo del dominio, se definieron los requisitos funcionales y los no funcionales, el modelo de caso de uso del sistema, así como las descripciones de sus casos de uso.

En el análisis y diseño se obtuvieron los diagramas de clases del análisis así como los del diseño, se realizaron los diagramas de clases del diseño de los casos de usos más significativos para la arquitectura.

En la disciplina Implementación y Prueba se obtuvo el Diagrama de Despliegue del sistema, al igual que los diagramas de componentes de la lógica del negocio y presentación.

Se realizaron los casos de prueba a los casos de usos más significativos obteniendo resultados satisfactorios, cumpliendo así con las exigencias del sistema. Se logró desde el software auditar la Plataforma Educativa Zera mostrando un informe con los datos existentes en cada contrato, también se mostró el por ciento de ganancias por cada programa de estudio, y las ganancias por cada licencia de tipo profesor, estudiante y cortesía.

Mediante la aplicación propuesta en correspondencia con las pruebas de rigor permitirá la auditoría externa y se controlaran las finanzas de la Plataforma Educativa Zera.

Recomendaciones

Resulta oportuno valorar el resultado presentado, debido a la pertinencia del tema, se recomienda:

- Considerar el resultado de la aplicación propuesta para implementar futuras mejoras en la labor de la Plataforma Educativa Zera.
- Continuar la investigación del tema para su profundización en estudios de postgrado, y posibilidad de diversificar la aplicación.
- Transmitir experiencias a otros proyectos que utilicen metodología, herramientas y tecnologías similares.
- Considerar el contenido de la Auditoría Informática y sus aplicaciones para contribuir a enriquecer la formación profesional de la carrera de Ingeniería Informática en Pregrado mediante un Curso Optativo, y en la superación de postgrado para el ejercicio de la profesión.

Referencias Bibliográficas

1. **Decreto Ley 159 De la Auditoría.** *Principales Disposiciones Legales y Regulaciones sobre la Auditoría.* Bibliografía. La Habana : s.n., 2001.

2. **Gómez, Adelkys Rosa Sánchez.** GestioPolis. *GestioPolis.* [En línea] Octubre de 2005. [Citado el: 17 de Noviembre de 2011.] http://www.gestiopolis.com/canales5/fin/defigaud.htm. CC BY-NC-SA 3.0.

3**. Reviews, IEEE Standard for Software.** IEEE. *IEEE.* [En línea] Octubre de 2006. [Citado el: 25 de Noviembre de 2011.] http://www.virusprot.com/glosarioc.html.

4**. Calidad, Empresa de Software para el Control de la Auditoría.** Empresa de Software para el Control de la Calidad. *Wilsoft.* [En línea] Mayo de 2011. [Citado el: 21 de Diciembre de 2011.] http://www.wilsoft-la.com/QAction.htm.

5. **http://www.ecured.cu.** [En línea] [Citado el: 3 de Marzo de 2012.] http://www.ecured.cu/index.php/Información_(auditoría)

6. **Encinosa, Lázaro J. Blanco.** *Auditoría y Sistemas Informáticos.* La Habana : Félix Varela., 2008.

7. **Borghi, Alicia.** Coyuntura Economica. *Coyuntura Economica.* [En línea] 7 de Mayo de 2010. [Citado el: 16 de Noviembre de 2011.] http://coyunturaeconomica.com/empresas/resumen-de-la-auditoría.

8. **Hurtado, Pablo Emilio Flores.** Curso elemental de auditoría. Universidad Internacional de La Rioja. *Curso elemental de auditoría. Universidad Internacional de La Rioja.* [En línea] 29 de Noviembre de 2005. [Citado el: 28 de Enero de 2012.] http://www.mailxmail.com/curso-elemental-auditoría/importancia-auditoría.

9. *Norma Internacional ISO 19011*. ISO 19011:2002, Bibliografia. Suiza : s.n., 2002.

10. Ibídem

11. **Hurtado, Pablo Emilio Flores.** Curso elemental de auditoría. Universidad Internacional de La Rioja. *Curso elemental de auditoría. Universidad Internacional de La Rioja.* [En línea] 29 de Noviembre de 2005. [Citado el: 28 de Enero de 2012.] http://www.mailxmail.com/curso-elemental-auditoría/importancia-auditoría.

12. **Fernando Lopez, Pedro Castaño.** Buenas Tareas. *Buenas Tareas.* [En línea] 15 de Noviembre de 2009. [Citado el: 27 de Enero de 2012.] http://www.buenastareas.com/ensayos/ Auditoría-Informatia/1288930.html.

13. **Moro., Ismaila López Sotolongo. Jorge León.** *SIGAC. Subsistema de Auditoría Gubernamental. Análisis y Diseño.* La Habana : s.n., 2008.

14. **Fernando Lopez, Pedro Castaño.** Buenas Tareas. *Buenas Tareas.* [En línea] 15 de Noviembre de 2009. [Citado el: 27 de Enero de 2012.] http://www.buenastareas.com/ensayos/ Auditoría-Informatia/1288930.html.

15. Ibídem

16. Eniac. *Software para Auditoría Interna y Auditoría de Sistemas.* [En línea] [Citado el: 15 de Enero de 2012.] http://www.eniac.com/productos/acl.htm.

17. **Santos, Humberto Suárez y Alás, Pedro Manuel Verdecia.** *Arquitectura de Software General de Auditoría.* La Habana : s.n., 2008.

18. **del Toro Ríos, José Carlos y González Brito, Henry Raúl.** *Documento Visión proyecto ERP Cuba v2.0.* La Habana: s.n., 2008.

19. **Ortiz, MSc. Kadir Hector.** *Plataforma para el control del uso de software educativos.* Cienfuegos : s.n., 2010.

20. **Benedí, Jennifer Pérez.** *Web Services. Departamento de Sistemas Informáticos y Computación. 18,* Valencia, España : s.n, 2008, Vol. XXVI.

21. **http://www.ecured.cu.** [En línea] [Citado el: 3 de Marzo de 2012.] http://www.ecured.cu/index.php/ Servicios_Web

22. **http://www.w3.org/** [En línea] [Citado el: 21 de Febrero de 2012.] http://www.ecured.cu/ TR/wsdl20-primer.

23. **Encinosa, Lázaro J. Blanco.** *Auditoría y Sistemas Informáticos.* La Habana : Félix Varela., 2008.

24. **C., Benjamín González.** Desarrollo web. [En línea] [Citado el: 2 de Febrero de 2012.] http://www.desarrolloweb.com/articulos/1557.php.

25. Todotecnologia.com. *METODOLOGIA OPEN UP.* [En línea] Noviembre de 2009. [Citado el: 3 de Abril de 2012.] http://todotecnology.blogspot.com/search/label/open UP.

26. Object Managment Group. *Object Managment Group.* [En línea] 2 de Octubre de 2007. [Citado el: 2 de Febrero de 2012.] http://www.uml.org/.

27. **Escribano, Gerardo Fernández.** [En línea] 09 de Diciembre de 2002. [Citado el: 6 de Diciembre de 2011.] http://www.dsi.uclm.es/asignaturas/42551/trabajosAnteriores/Presentacion-XP.pdf

28. **Larman, Craig.** Visual Paradigm for UML. *Visual Paradigm for UML.* [En línea] Febrero de 2007. [Citado el: 12 de Enero de 2012.] http://www.visual-paradigm.com/product/vpuml.

29. Object Managment Group. *Object Managment Group.* [En línea] 2 de Octubre de 2007. [Citado el: 2 de Febrero de 2012.] http://www.uml.org/.

30. **Giraldo, Luis y Zapata, Yuliana.** *Herramientas de desarrollo de Ingeniería de SW en linux.* 24 : Septiembre, 2005.

31. **Tugores, Miquel Àngel Herrera.** Modelando con UML en Linux. *Modelando con UML en Linux.* [En línea] 20 de Enero de 2005. [Citado el: 11 de Enero de 2012.] http://www.wikilearning.com/articulo/modelando_con_uml_en_linux-introduccion/330-1.

32. **Giraldo, Luis y Zapata, Yuliana.** *Herramientas de desarrollo de Ingeniería de SW en linux.* 24 : Septiembre, 2005.

33. **http://www.ecured.cu.** [En línea] [Citado el: 3 de Marzo de 2012.] http://www.ecured.cu/index.php/Visual_Paradigm

34. **Lago, Ramiro.** Introduccion a JSF. *Proactiva Calidad.* [En línea] Mayo de 2007. [Citado el: 15 de Enero de 2012.] http://www.proactiva-calidad.com/java/jsf/introduccion.html.

35. **Potencier, Fabien.** Symfony2. [En línea] 23 de Agosto de 2011. [Citado el: 6 de Febrero de 2012.] http://symfony.com/.

36. **Santos, Humberto Suárez y Alás, Pedro Manuel Verdecia.** *Arquitectura de Software General de Auditoría.* La Habana : s.n., 2008.

37. **Pérez, J. E.** Introducción al CSS. *Introducción al CSS.* [En línea] 2008. [Citado el: 16 de Enero de 2012.] http://librosweb.es.

38. **Hernández, Yasmany Hernández y Infante, Jenny Frometa.** *Arquitectura de Software para el Sistema de Gestión de Reportes Dinámicos.* Ciudad Habana : s.n., 2009. s.n.

39. **ULTRASIST.** ¿Porqué utilizamos ASP? [En línea] 2012. [Citado el: 23 de Enero de 2012.] http://www.ultrasist.com.mx/tecnologias/asp.htm.

40. **Ayuda.com, Manuales de.** Manuales de Ayuda. [En línea] 23 de Julio de 2011. [Citado el: 8 de Febrero de 2012.] http://www.manualesdeayuda.com/manuales/bases-de-atos/postgresql/caracteristicas-de-postgresql-01844.html.

41. **ÁLVAREZ, S. and HERNANDEZ, A.** "Metodología para el desarrollo de aplicaciones con tecnología Orientada a Objetos utilizando notación UML.". [En línea] [Citado el: 10 de Febrero de 2012.] http://www.colombiaaprende.edu.co/html/mediateca/1607/articles-75593_archivo.pdf.

42. **http://httpd.apache.org/.** [En línea] [Citado el: 15 de Marzo de 2012.] http://httpd.apache.org/.

43. **http://bitsbeta.com.** [En línea] 5 de 12 de 2011. [Citado el: 10 de 1 de 2012.] http://bitsbeta.com/netbeans-ide-grafica-programacion.

44. **Garrido, Jesús Manuel Montero.** *Plataforma Eclipse.* 2004.

45. **http://www.ecured.cu.** [En línea] [Citado el: 3 de Marzo de 2012.] http://www.ecured.cu/index.php/Requisitos_de_Software.

46. Ibídem

47. Ibídem

48. Ibídem

49. **Vilas, Ana Fernandez.** Grupo de Redes e Ingeniería del Software. GRIS. [En línea] 5 de Octubre de 2008. [Citado el: 23 de Febrero de 2012.]

50. **García, Liudmila de la Caridad Labrada y González, Dairon Javier Soler.** *Módulo Auditorías y Comprobaciones Especiales. Análisis y Diseño.* La Habana : s.n., 2008.

51. **Olivera, lilian de.** Mail.com. *Control de calidad en la empresa.* [En línea] 16 de Mayo de 2008. [Citado el: 21 de Mayo de 2012.] http://www.mailxmail.com/curso-control-calidad-empresa/fase-implementacion.

52. **James Rumbaugh, Ivar Jacobson, Grady Booch.** *El lenguaje Unificado de Modelado. Manual de Referencia.* s.l. : Addison Wesley.

53. Ibídem

Bibliografía

Gómez, Adelkys Rosa Sánchez. GestioPolis. *GestioPolis.* [En línea] Octubre de 2005. [Citado el: 17 de Noviembre de 2011.] http://www.gestiopolis.com/canales5/fin/defigaud.htm. CC BY-NC-SA 3.0.

Decreto Ley 159 De la Auditoría. *Principales Disposiciones Legales y Regulaciones sobre la Auditoría.* Bibliografía. La Habana: s.n., 2001.

Borghi, Alicia. Coyuntura Economica. *Coyuntura Economica.* [En línea] 7 de Mayo de 2010. [Citado el: 16 de Noviembre de 2011.] http://coyunturaeconomica.com/empresas/resumen-de-la-auditoría.

Norma Internacional ISO 19011. ISO 19011:2002, Bibliografia. Suiza : s.n., 2002.

Encinosa, Lázaro J. Blanco. *Auditoría y Sistemas Informáticos.* La Habana: Félix Varela., 2008.

Pablo Ortiz Sanchidrián, Lourdes Riestra Alba. [En línea] 25 de Junio de 2009. [Citado el: 21 de Diciembre de 2011.] http://oa.upm.es/1387/1/PFC_PABLO_ORTIZ_LOURDES_RIESTRA.pdf.

Tugores, Miquel Àngel Herrera. Modelando con UML en Linux. *Modelando con UML en Linux.* [En línea] 20 de Enero de 2005. [Citado el: 11 de Enero de 2012.] http://www.wikilearning.com/articulo/modelando_con_uml_en_linux-introduccion/330-1.

Escribano, Gerardo Fernández. [En línea] 09 de Diciembre de 2002. [Citado el: 6 de Diciembre de 2011.] http://www.dsi.uclm.es/asignaturas/42551/trabajosAnteriores/Presentacion-XP.pdf.

Lago, Ramiro. Proactiva Calidad. *Proactiva Calidad.* [En línea] Mayo de 2007. [Citado el: 15 de Enero de 2012.] http://www.proactiva-calidad.com/java/jsf/introduccion.html.

Reviews, IEEE Standard for Software. *IEEE.* [En línea] Octubre de 2006. [Citado el: 25 de Noviembre de 2011.] http://www.virusprot.com/glosarioc.html.

Calidad, Empresa de Software para el Control de la Auditoría. Empresa de Software para el Control de la Calidad. *Wilsoft.* [En línea] Mayo de 2011. [Citado el: 21 de Diciembre de 2011.] http://www.wilsoft-la.com/QAction.htm.

Morejon, carlos. Auditoría de Sistemas. *Auditoría de Sistemas.* [En línea] 3 de Junio de 2011. [Citado el: 23 de Enero de 2012.] http://carlos-auditoría.blogspot.com/2011/06/tecnicas-de-auditoría-asistida-por.html.

Fernando Lopez, Pedro Castaño. Buenas Tareas. *Buenas Tareas.* [En línea] 15 de Noviembre de 2009. [Citado el: 27 de Enero de 2012.] http://www.buenastareas.com/ensayos/ Auditoría-Informatia/1288930.html.

Administración de Empresas. *Administración de Empresas.* [En línea] 2009. [Citado el: 27 de Enero de 2012.]

Franklin, Enrique B. *Auditoría Administrativa.* México: 1era. Edición., 2000. McGraw Hil.

Ivar Jacobson, Rumbaugh James and Buch,Grady Buch. *El proceso Unificado del desarrollo del software.* s.l. : PEARSON, 2000. 12/212.

Object Managment Group. *Object Managment Group.* [En línea] 2 de Octubre de 2007. [Citado el: 2 de Febrero de 2012.] http://www.uml.org/.

Larman, Craig. Visual Paradigm for UML. *Visual Paradigm for UML.* [En línea] Febrero de 2007. [Citado el: 12 de Enero de 2012.] http://www.visual-paradigm.com/product/vpuml.

Web Services. *Departamento de Sistemas Informáticos y Computación,*. **Benedí, Jennifer Pérez.** 18, Valencia, España : s.n, 2008, Vol. XXVI.

Giraldo, Luis y Zapata, Yuliana. HERRAMIENTAS DE DESARROLLO DE INGENIERIA DE SW PARA LINUX. *HERRAMIENTAS DE DESARROLLO DE INGENIERIA DE SW PARA LINUX.* [En línea] 24 de Septiembre de 2005. [Citado el: 20 de Enero de 2012.] http://hugolopez.phi.com.co/docs/download/file=Giraldo-Zapata-Herramientas de ISW.pdf,_id=17.

Tugores, Herrera M.Á. Modelando con UML en Linux - Umbrello - Wikilearning,". *Modelando con UML en Linux - Umbrello - Wikilearning,"*. [En línea] 2005.

Pérez, J. E. Introducción al CSS. *Introducción al CSS.* [En línea] 2008. [Citado el: 16 de Enero de 2012.] http://librosweb.es.

Hernández, Yasmany Hernández y Infante, Jenny Frometa. *Arquitectura de Software para el Sistema de Gestión de Reportes Dinámicos.* Ciudad Habana : s.n., 2009. s.n.

Pimentel, Annia Rivero y Hernández, Antonio Dominguez. *Sistema Informático para la Gestión de Auditoría y Control (SIGAC). Módulo de Planificación.* La Habana : s.n., 2009.

Hurtado, Pablo. [En línea] Mayo de 2009. [Citado el: 28 de Enero de 2012.]

Decreto Ley 159 De la Auditoría. *Principales Disposiciones Legales y Regulaciones sobre la Auditoría.* Bibliografía. La Habana : s.n., 2001.

Gómez, Adelkys Rosa Sánchez. GestioPolis. *GestioPolis.* [En línea] Octubre de 2005. [Citado el: 17 de Noviembre de 2011.] http://www.gestiopolis.com/canales5/fin/defigaud.htm. CC BY-NC-SA 3.0.

Reviews, IEEE Standard for Software. IEEE. *IEEE.* [En línea] Octubre de 2006. [Citado el: 25 de Noviembre de 2011.] http://www.virusprot.com/glosarioc.html.

Calidad, Empresa de Software para el Control de la Auditoría. Empresa de Software para el Control de la Calidad. *Wilsoft.* [En línea] Mayo de 2011. [Citado el: 21 de Diciembre de 2011.] http://www.wilsoft-la.com/QAction.htm.

http://www.ecured.cu. [En línea] [Citado el: 3 de Marzo de 2012.] http://www.ecured.cu/index.php/Información_(auditoría)

Encinosa, Lázaro J. Blanco. *Auditoría y Sistemas Informáticos.* La Habana : Félix Varela., 2008.

Borghi, Alicia. Coyuntura Economica. *Coyuntura Economica.* [En línea] 7 de Mayo de 2010. [Citado el: 16 de Noviembre de 2011.] http://coyunturaeconomica.com/empresas/resumen-de-la-auditoría.

Hurtado, Pablo Emilio Flores. Curso elemental de auditoría. Universidad Internacional de La Rioja. *Curso elemental de auditoría. Universidad Internacional de La Rioja.* [En línea] 29 de Noviembre de 2005. [Citado el: 28 de Enero de 2012.] http://www.mailxmail.com/curso-elemental-auditoría/importancia-auditoría.

Norma Internacional ISO 19011. ISO 19011:2002, Bibliografia. Suiza : s.n., 2002.

Hurtado, Pablo Emilio Flores. Curso elemental de auditoría. Universidad Internacional de La Rioja. *Curso elemental de auditoría. Universidad Internacional de La Rioja.* [En línea] 29 de Noviembre de 2005. [Citado el: 28 de Enero de 2012.] http://www.mailxmail.com/curso-elemental-auditoría/importancia-auditoría.

Fernando Lopez, Pedro Castaño. Buenas Tareas. *Buenas Tareas.* [En línea] 15 de Noviembre de 2009. [Citado el: 27 de Enero de 2012.] http://www.buenastareas.com/ensayos/ Auditoría-Informatia/1288930.html.

Moro., Ismaila López Sotolongo. Jorge León. *SIGAC. Subsistema de Auditoría Gubernamental. Análisis y Diseño.* La Habana : s.n., 2008.

14. **Fernando Lopez, Pedro Castaño.** Buenas Tareas. *Buenas Tareas.* [En línea] 15 de Noviembre de 2009. [Citado el: 27 de Enero de 2012.] http://www.buenastareas.com/ensayos/ Auditoría-Informatia/1288930.html.

Eniac. *Software para Auditoría Interna y Auditoría de Sistemas.* [En línea] [Citado el: 15 de Enero de 2012.] http://www.eniac.com/productos/acl.htm.

Santos, Humberto Suárez y Alás, Pedro Manuel Verdecia. *Arquitectura de Software General de Auditoría.* La Habana : s.n., 2008.

Ortiz, MSc. Kadir Hector. *Plataforma para el control del uso de software educativos.* Cienfuegos : s.n., 2010.

Benedí, Jennifer Pérez. *Web Services. Departamento de Sistemas Informáticos y Computación. 18,* Valencia, España : s.n, 2008, Vol. XXVI.

http://www.w3.org/ [En línea] [Citado el: 21 de Febrero de 2012.] http://www.ecured.cu/ TR/wsdl20-primer.

Encinosa, Lázaro J. Blanco. *Auditoría y Sistemas Informáticos.* La Habana : Félix Varela., 2008.

C., Benjamín González. Desarrollo web. [En línea] [Citado el: 2 de Febrero de 2012.] http://www.desarrolloweb.com/articulos/1557.php.

Todotecnologia.com. *METODOLOGIA OPEN UP.* [En línea] Noviembre de 2009. [Citado el: 3 de Abril de 2012.] http://todotecnology.blogspot.com/search/label/open UP.

Object Managment Group. *Object Managment Group.* [En línea] 2 de Octubre de 2007. [Citado el: 2 de Febrero de 2012.] http://www.uml.org/.

Escribano, Gerardo Fernández. [En línea] 09 de Diciembre de 2002. [Citado el: 6 de Diciembre de 2011.] http://www.dsi.uclm.es/asignaturas/42551/trabajosAnteriores/Presentacion-XP.pdf

Larman, Craig. Visual Paradigm for UML. *Visual Paradigm for UML.* [En línea] Febrero de 2007. [Citado el: 12 de Enero de 2012.] http://www.visual-paradigm.com/product/vpuml.

Object Managment Group. *Object Managment Group.* [En línea] 2 de Octubre de 2007. [Citado el: 2 de Febrero de 2012.] http://www.uml.org/.

Giraldo, Luis y Zapata, Yuliana. *Herramientas de desarrollo de Ingeniería de SW en linux.* 24 : Septiembre, 2005.

Tugores, Miquel Àngel Herrera. Modelando con UML en Linux. *Modelando con UML en Linux.* [En línea] 20 de Enero de 2005. [Citado el: 11 de Enero de 2012.] http://www.wikilearning.com/articulo/modelando_con_uml_en_linux-introduccion/330-1.

Giraldo, Luis y Zapata, Yuliana. *Herramientas de desarrollo de Ingeniería de SW en linux.* 24 : Septiembre, 2005.

http://www.ecured.cu. [En línea] [Citado el: 3 de Marzo de 2012.] http://www.ecured.cu/index.php/Visual_Paradigm

Lago, Ramiro. Introduccion a JSF. *Proactiva Calidad.* [En línea] Mayo de 2007. [Citado el: 15 de Enero de 2012.] http://www.proactiva-calidad.com/java/jsf/introduccion.html.

Potencier, Fabien. Symfony2. [En línea] 23 de Agosto de 2011. [Citado el: 6 de Febrero de 2012.] http://symfony.com/.

Santos, Humberto Suárez y Alás, Pedro Manuel Verdecia. *Arquitectura de Software General de Auditoría.* La Habana : s.n., 2008.

Pérez, J. E. Introducción al CSS. *Introducción al CSS.* [En línea] 2008. [Citado el: 16 de Enero de 2012.] http://librosweb.es.

Hernández, Yasmany Hernández y Infante, Jenny Frometa. *Arquitectura de Software para el Sistema de Gestión de Reportes Dinámicos.* Ciudad Habana : s.n., 2009. s.n.

ULTRASIST. ¿Porqué utilizamos ASP? [En línea] 2012. [Citado el: 23 de Enero de 2012.] http://www.ultrasist.com.mx/tecnologias/asp.htm.

Ayuda.com, Manuales de. Manuales de Ayuda. [En línea] 23 de Julio de 2011. [Citado el: 8 de Febrero de 2012.] http://www.manualesdeayuda.com/manuales/bases-de-atos/postgresql/caracteristicas-de-postgresql-01844.html.

ÁLVAREZ, S. and HERNANDEZ, A. "Metodología para el desarrollo de aplicaciones con tecnología Orientada a Objetos utilizando notación UML.". [En línea] [Citado el: 10 de Febrero de 2012.] http://www.colombiaaprende.edu.co/html/mediateca/1607/articles-75593_archivo.pdf.

http://httpd.apache.org/. [En línea] [Citado el: 15 de Marzo de 2012.] http://httpd.apache.org/.

Vilas, Ana Fernandez. Grupo de Redes e Ingeniería del Software. GRIS. [En línea] 5 de Octubre de 2008. [Citado el: 23 de Febrero de 2012.]

García, Liudmila de la Caridad Labrada y González, Dairon Javier Soler. *Módulo Auditorías y Comprobaciones Especiales. Análisis y Diseño.* La Habana : s.n., 2008.

Olivera, lilian de. Mail.com. *Control de calidad en la empresa.* [En línea] 16 de Mayo de 2008. [Citado el: 21 de Mayo de 2012.] http://www.mailxmail.com/curso-control-calidad-empresa/fase-implementacion.

James Rumbaugh, Ivar Jacobson, Grady Booch. *El lenguaje Unificado de Modelado. Manual de Referencia.* s.l. : Addison Wesley.

GLOSARIO DE TÉRMINOS

AI: Auditoría informática.

Cliente: Son las personas para las cuales se elabora un producto determinado.

Calidad: Calidad es el grado en el que un conjunto de características inherentes cumple con los requisitos que se plantean en cualquier esfera.

Herramienta: Software que se utiliza para automatizar las actividades definidas en el proceso.

Proyecto: Es un esfuerzo temporal emprendido para crear un producto o servicio único.

Proceso: Es un conjunto de actividades o eventos que se realizan o suceden con un determinado fin; proceso de desarrollo de software (PDS): Es la definición del conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto para transformar los requisitos de usuario en un producto.

Requisitos del software: Son las funciones, servicios y restricciones operativas del sistema.

Sistema: Colección de unidades conectadas que se organiza para lograr un propósito. El sistema es el “modelo completo”.

Software: Se refiere a los programas y datos almacenados en un ordenador.

Tecnología: Conjunto de teorías y de técnicas que permiten el aprovechamiento práctico del conocimiento científico.

Usuario: Persona que utiliza o trabaja con algún objeto o que es destinataria de algún servicio público o privado, empresarial o profesional.

API: Interfaz de programación de aplicaciones, conjunto de funciones y procedimientos (o métodos, si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Anexos

Anexo1. Descripciones textuales de casos de uso

Caso de Uso:	Ver Contrato
Actores:	Auditor Alfaomega, Auditor Cliente o Administrador (Inician)
Resumen:	El caso de uso inicia cuando el Auditor Alfaomega, Auditor Cliente o Administrador solicitan ver un contrato. El sistema brinda dicha posibilidad y finaliza el caso de uso.
Precondiciones:	Usuario autenticado.
Referencias	RF-6
Prioridad	Crítico

Sección "Principal"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción ver un contrato.	2. Ejecuta la siguiente acción: a) Si el Auditor Alfaomega, Auditor Cliente o Administrador decide ver un contrato, ir a la sección "Ver Contrato".
Sección "Ver Contrato"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción ver contrato desde la lista de contrato.	2. Muestra la interfaz contrato con un formulario mostrando los siguientes datos: • Número de contrato. • Descripción. • Moneda de pago.

	• Escuela. Escuela que solicita el contrato. • Estado. Muestra si el contrato esta en negociación o en ejecución. • Contrato. Documento firmado entre una escuela y Alfaomega, puede ser en formato word o pdf. Y permite: • Cancelar: Cancela la muestra de datos del contrato
3. Selecciona la opción cancelar.	4. Regresa a la lista de contrato.
Flujos Alternos	
Flujo Alterno 3a"Selecciona la opción Cancelar"	
Acción del Actor	**Respuesta del Sistema**
	3a. Regresa al paso 1 del Flujo Normal de Eventos de la sección "Principal" y finaliza el caso de uso.

Tabla 10: Ver Contrato

Caso de Uso:	Eliminar Contrato
Actores:	Administrador (Inicia)
Resumen:	El caso de uso inicia cuando el Administrador solicita eliminar un contrato. El sistema brinda dicha posibilidad y finaliza el caso de uso.
Precondiciones:	Usuario autenticado.
Referencias	RF-8
Prioridad	Crítico
Sección "Principal"	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Solicita eliminar un contrato.	2. Ejecuta la siguiente acción: f) Si el Administrador decide Eliminar un contrato, ir a la sección "Eliminar Contrato".
Sección "Eliminar Contrato"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
5. Elige el contrato que desea eliminar y selecciona la opción "Eliminar".	6. Muestra el siguiente mensaje: "¿Estás seguro que desea Eliminar el contrato?".
7. Acepta el mensaje de confirmación.	8. Elimina los datos del contrato y finaliza así el caso de uso.
Flujos Alternos	
Flujo Alterno 3a"Selecciona la opción Cancelar"	
Acción del Actor	**Respuesta del Sistema**
	3a.1. Cancela la eliminación, regresa al curso y finaliza el caso de uso.

Tabla 11: Eliminar Contrato

Caso de Uso:	Auditar Contrato
Actores:	Auditor Cliente o Administrador (Inician)
Resumen:	El caso de uso inicia cuando el Auditor Cliente o Administrador solicitan auditar un contrato. El sistema brinda dicha posibilidad y finaliza el caso de uso.
Precondiciones:	Usuario autenticado.
Referencias	RF-10

Prioridad	Crítico
Sección “Principal”	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Solicita auditar un contrato.	2. Ejecuta la siguiente acción: g) Si el Auditor Cliente o Administrador deciden auditar un contrato, ir a la sección " Auditar Contrato".
Sección “Auditar Contrato”	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Elige el contrato que desea auditar y selecciona la opción “Auditar”.	2. Muestra una interfaz con la comparación de los programas de estudios correspondientes al contrato y a los existentes en Zera con los siguientes datos: • Nombre del programa de estudio. • Descripción. • % de ganancia del programa de studio. • Cantidad de licencias de estudiantes. • Costo de licencias de estudiantes. • Cantidad de licencias de profesores. • Costo de licencias de profesores. • Cantidad de licencias de cortesía. • Costo de licencias de cortesía. • Ganancias del programa de studio.
	3. Muestra los datos de los programas de

	estudios que no coincidan de color rojo y finaliza así el caso de uso.

Tabla 12: Auditar Contrato

Caso de Uso:	Gestionar Programa de Estudio
Actores:	Auditor Alfaomega, Auditor Cliente o Administrador (Inician)
Resumen:	El caso de uso inicia cuando el Auditor Alfaomega, Auditor Cliente o Administrador solicitan insertar, ver, modificar o eliminar un programa de estudio. El sistema brinda dichas posibilidades y finaliza el caso de uso.
Precondiciones:	Usuario autenticado. Debe haberse ejecutado con anterioridad la sección “Incluir Contrato” o la sección “Modificar Contrato” del CU Incluir Contrato o Modificar Contrato.
Referencias	RF-11
Prioridad	Crítico
Sección “Principal”	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
5. Solicita insertar, ver, modificar o eliminar un programa de estudio.	6. Ejecuta alguna de las siguientes acciones: h) Si el usuario autenticado decide insertar un programa de estudio, ir a la sección "Adicionando Programas". i) Si el usuario autenticado decide ver un programa de estudio, ir a la sección "Editar Programas". j) Si el usuario autenticado decide modificar un programa de estudio, ir a la sección "Editar Programas".

	k) Si el usuario autenticado decide eliminar un programa de estudio, ir a la sección "Editar Programas" opción eliminar programa de estudio.
Sección "Insertar Programa de Estudio"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
13. Selecciona la opción adicionar programa de estudio desde la sección "Adicionando Programas".	14. Muestra la interfaz programas de estudio con los formularios de todos los programas existente en Zera solicitando de cada uno los siguientes datos: • Nombre del programa de estudio. • Descripción. • % de ganancia del programa de estudio. • Cantidad de licencias de estudiantes. • Costo de licencias de estudiantes. • Cantidad de licencias de profesores. • Costo de licencias de profesores. • Cantidad de licencias de cortesía. • Costo de licencias de cortesía. Y permite: • Guardar: Permite guardar los datos del nuevo programa de estudio. • Cancelar: Cancela la creación del programa de estudio.
15. Introduce los datos solicitados y selecciona la opción Guardar.	16. Comprueba que los campos contengan información.

	17. Almacena los datos del programa de estudio.
	18. Permite ver los datos introducidos por el usuario autenticado.
Flujos Alternos	
Flujo Alterno 3a"Selecciona la opción Guardar"	
Acción del Actor	**Respuesta del Sistema**
	3a. Almacena los datos, muestra los datos del programa de estudio y finaliza el caso de uso.
Flujo Alterno 3b"Selecciona la opción Cancelar"	
Acción del Actor	**Respuesta del Sistema**
	3b. Regresa al paso 1 del Flujo Normal de Eventos de la sección "Principal" y finaliza el caso de uso.
Flujo Alterno 4a"Campos vacíos"	
Acción del Actor	**Respuesta del Sistema**
	4a. Emite un mensaje indicando que se deben llenar los campos obligatorios, regresa al paso 2 del Flujo Normal de Eventos de la sección "Insertar Programa de Estudio" y finaliza el caso de uso.
Sección "Ver Programa de Estudio"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
5. Selecciona la opción ver editar programas.	6. Muestra la interfaz programas de estudio con los formularios de todos los programas existente en el contrato mostrando de cada

	uno los siguientes datos: • Nombre del programa de estudio. • Descripción. • % de ganancia del programa de estudio. • Cantidad de licencias de estudiantes. • Costo de licencias de estudiantes. • Cantidad de licencias de profesores. • Costo de licencias de profesores. • Cantidad de licencias de cortesía. • Costo de licencias de cortesía. Y permite: • Eliminar. Elimina el programa de estudio que el usuario autenticado desee. • Cancelar: Cancela la muestra de datos del programa de estudio.
7. Selecciona la opción cancelar.	8. Regresa a la lista de contrato.
Flujos Alternos	
Flujo Alterno 3a"Selecciona la opción Cancelar"	
Acción del Actor	**Respuesta del Sistema**
	3a. Regresa al paso 1 del Flujo Normal de Eventos de la sección "Principal" y finaliza el caso de uso.
Sección "Modificar Programa de Estudio"	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
7. Selecciona la opción editar programas.	9. Muestra la interfaz programas de estudio con los formularios de todos los programas

	existente en el contrato mostrando de cada uno los siguientes datos: • Nombre del programa de estudio. • Descripción. • % de ganancia del programa de estudio. • Cantidad de licencias de estudiantes. • Costo de licencias de estudiantes. • Cantidad de licencias de profesores. • Costo de licencias de profesores. • Cantidad de licencias de cortesía. • Costo de licencias de cortesía. Y permite: • Guardar: Permite guardar los datos del programa de estudio modificado. • Cancelar: Cancela la modificación del programa de estudio.
8. Modifica los datos deseados y selecciona la opción Guardar.	9. Comprueba que los campos contengan información.
	10. Actualiza los datos del programa de estudio y finaliza el caso de uso.
Flujos Alternos	
Flujo Alterno 3a“Selecciona la opción Guardar”	
Acción del Actor	**Respuesta del Sistema**
	3a.1. Almacena los datos, muestra los datos del programa de estudio y finaliza el caso de uso.
Flujo Alterno 3c“Selecciona la opción Cancelar”	
Acción del Actor	**Respuesta del Sistema**

	3c.1. Regresa al paso 1 del Flujo Normal de Eventos de la sección “Principal” y finaliza el caso de uso.
Flujo Alterno 4a“Campos vacíos”	
Acción del Actor	**Respuesta del Sistema**
	4a.1. Emite un mensaje indicando que se debe llenar los campos obligatorios, regresa al paso 2 del Flujo Normal de Eventos de la sección “Editar Programas” y finaliza así el caso de uso.
Sección “Eliminar Programa de Estudio”	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
9. Elige el programa de estudio que desea eliminar y selecciona la opción “Eliminar”.	10. Muestra el siguiente mensaje: “Confirma que desea eliminar”.
11. Acepta el mensaje de confirmación.	12. Elimina los datos del programa de estudio y finaliza así el caso de uso.
Flujos Alternos	
Flujo Alterno 3a“Selecciona la opción Cancelar”	
Acción del Actor	**Respuesta del Sistema**
	3a.1. Cancela la eliminación, regresa al curso y finaliza el caso de uso.

Tabla 13: Gestionar Programa de Estudio

Caso de Uso:	Ver Rol
Actores:	Administrador (Inicia)
Resumen:	El caso de uso inicia cuando el administrador solicita ver un rol. El sistema brinda dicha posibilidad y finaliza el caso de uso.

<table>
<tr><td>Precondiciones:</td><td colspan="2">Usuario autenticado.</td></tr>
<tr><td>Referencias</td><td colspan="2">RF-3</td></tr>
<tr><td>Prioridad</td><td colspan="2">Secundario</td></tr>
<tr><td colspan="3">Sección “Principal”</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>7. Solicita ver un rol.</td><td colspan="2">8. Ejecuta la siguiente acción:
l) Si el administrador decide ver un rol, ir a la sección "Ver Rol".</td></tr>
<tr><td colspan="3">Sección “Ver Rol”</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>19. Selecciona la opción ver rol desde la lista de roles.</td><td colspan="2">20. Muestra la interfaz con un formulario mostrando los siguientes datos:
• Rol.
• Descripción.
Y permite:
• Cancelar: Cancela la muestra del rol.</td></tr>
<tr><td>21. Selecciona la opción cancelar.</td><td colspan="2">22. Regresa a la lista de roles.</td></tr>
<tr><td colspan="3">Flujos Alternos</td></tr>
<tr><td colspan="3">Flujo Alterno 3a“Selecciona la opción Cancelar”</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td></td><td colspan="2">3a. Regresa al paso 1 del Flujo Normal de Eventos de la sección “Principal” y finaliza el caso de uso.</td></tr>
</table>

Tabla 14: Ver Rol

<table>
<tr><td>Caso de Uso:</td><td colspan="2">Modificar Rol</td></tr>
<tr><td>Actores:</td><td colspan="2">Administrador (Inicia)</td></tr>
<tr><td>Resumen:</td><td colspan="2">El caso de uso inicia cuando el administrador solicita modificar un rol. El sistema brinda dichas posibilidades y finaliza el caso de uso.</td></tr>
<tr><td>Precondiciones:</td><td colspan="2">Usuario autenticado.</td></tr>
<tr><td>Referencias</td><td colspan="2">RF-4</td></tr>
<tr><td>Prioridad</td><td colspan="2">Secundario</td></tr>
<tr><td colspan="3">Sección “Principal”</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>1. Solicita modificar un rol.</td><td colspan="2">2. Ejecuta la siguiente acción:
m) Si el administrador decide modificar un rol, ir a la sección "Editar Rol".</td></tr>
<tr><td colspan="3">Sección “Modificar Rol”</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>1. Selecciona la opción editar rol desde la lista de roles.</td><td colspan="2">2. Muestra la interfaz con un formulario mostrando los siguientes datos (Solo es editable la descripción):
• Rol.
• Descripción.
Y permite:
• Cancelar: Cancela la muestra del rol</td></tr>
<tr><td>3. Modifica los datos deseados y selecciona la opción Actualizar.</td><td colspan="2">4. Actualiza los datos del rol.</td></tr>
<tr><td></td><td colspan="2">5. Muestra los datos del rol modificado y</td></tr>
</table>

	finaliza el caso de uso.
Flujos Alternos	
Flujo Alterno 3a“Selecciona la opción Cancelar”	
Acción del Actor	**Respuesta del Sistema**
	3a. Regresa al paso 1 del Flujo Normal de Eventos de la sección “Principal” y finaliza el caso de uso.

Tabla 15: Modificar Rol

Anexo 2. Diagramas de clases del análisis

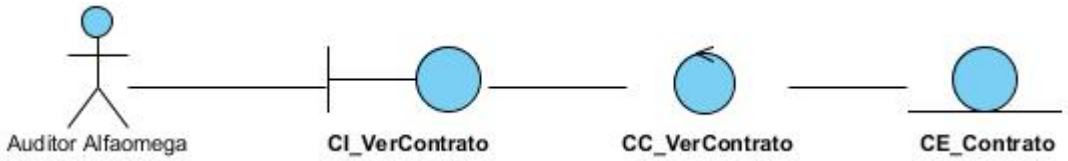

Figura 11: Ver Contrato

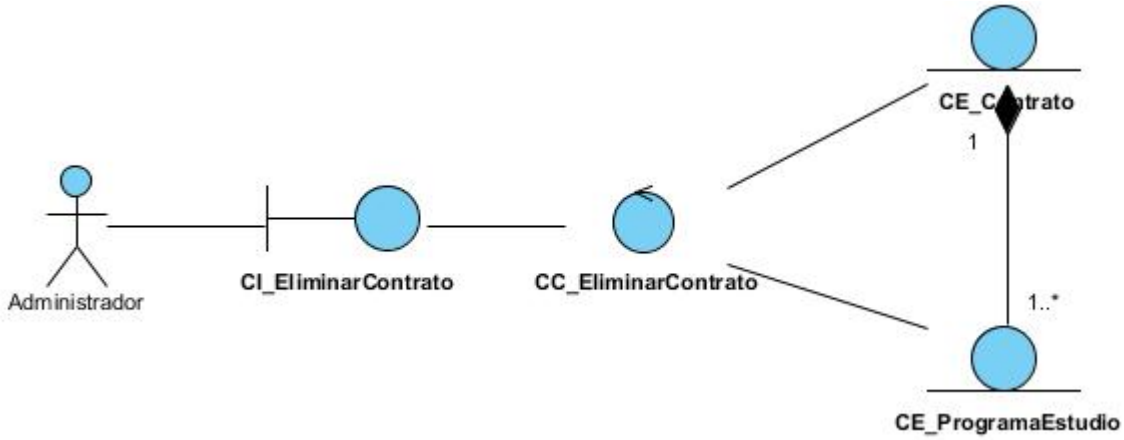

Figura 12: Eliminar Contrato

Figura 13: Cerrar Contrato

Figura 14: Auditar Contrato

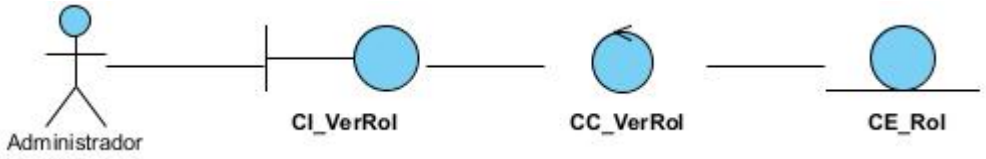

Figura 15: Ver Rol

Figura 16: Modificar Rol

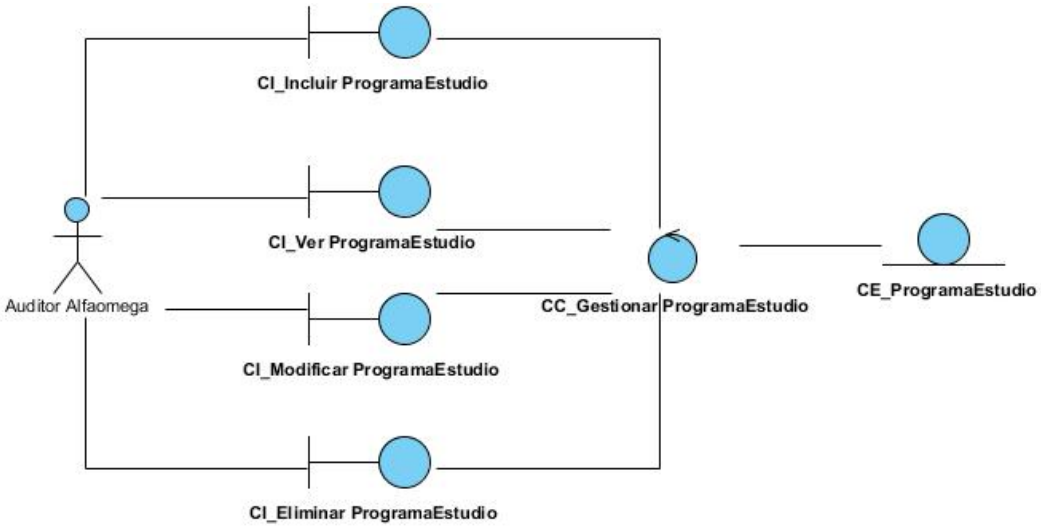

Figura 17: Gestionar Programa de Estudio

Anexo 3. Diagramas de clases del diseño

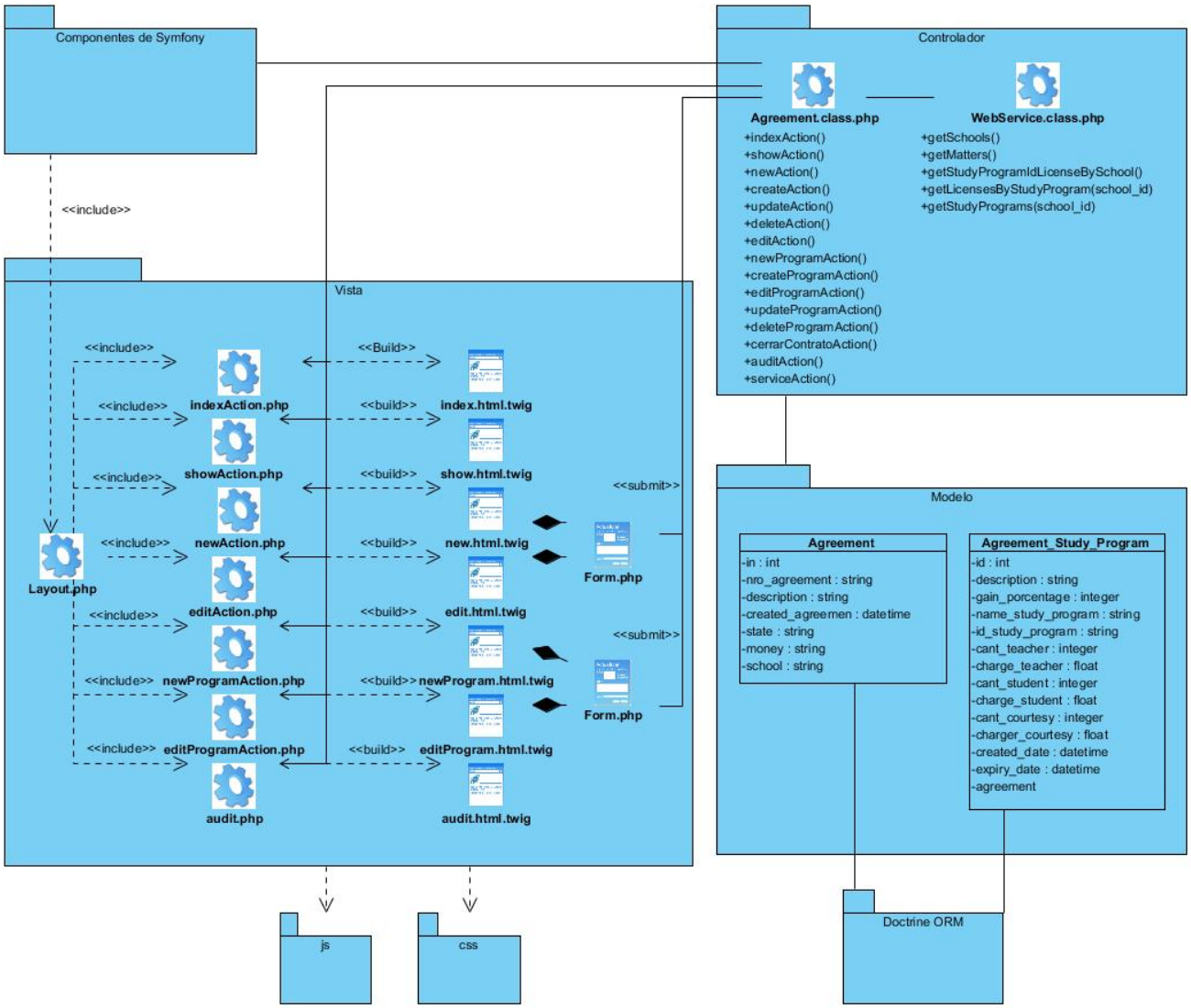

Figura 18: Gestionar Contrato y Programa de Estudio

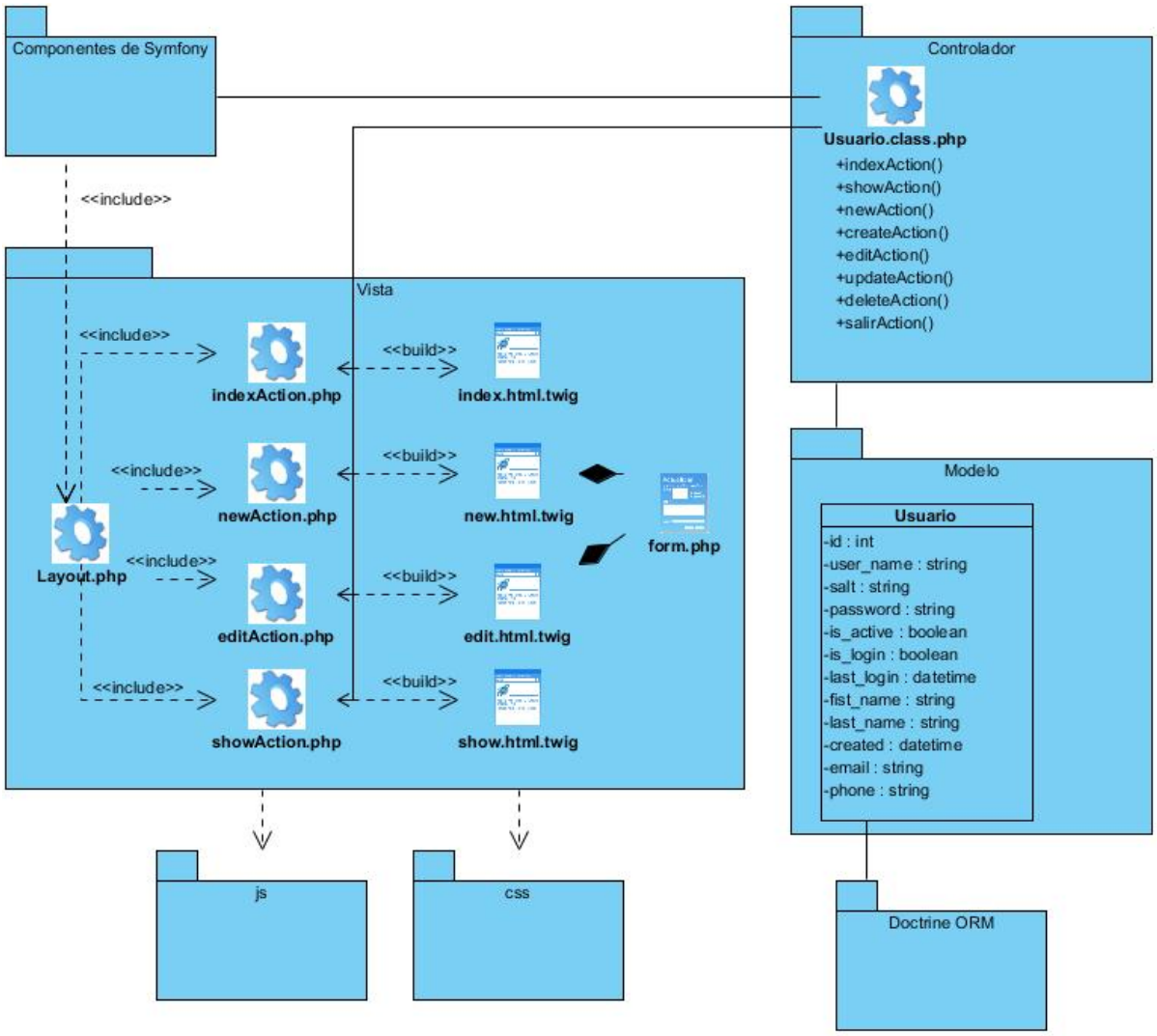

Figura 19: Gestionar Usuario

Anexo 4. Descripciones de las tablas de la base de datos

n_rol		
Descripción: Esta tabla se encarga de guardar los principales datos del rol.		
Atributo	**Tipo**	**Descripción**
Id	integer	Identificador del rol.
name	varchar	Nombre especial que identifica al rol.
description	text	Describe el rol.

Tabla 16: Descripción de la tabla n_rol.

tb_user_n_rol		
Descripción: Esta tabla se encarga de guardarlas relaciones existentes entre usuario y rol.		
Atributo	**Tipo**	**Descripción**
Id	integer	Identificador del rol.
tb_userid	varchar	Identificador del usuario
tb_rolid	text	Identificador del rol.

Tabla 17: Descripción de la tabla tb_user_n_rol.

tb_agreement_study_program		
Descripción: Esta tabla se encarga de guardar los datos variables del programa de estudio relacionados con un contrato.		
Atributo	**Tipo**	**Descripción**
id	integer	Identificador del programa de estudio.
description	text	Describe el programa de estudio.

gain_percentage	float	Porciento que gana el cliente por programa de estudio.
name_study_program	varchar	Nombre del programa de estudio.
cant_teacher	integer	Cantidad de licencias de tipo docente que tiene el programa de estudio.
charge_teacher	Float	Precio que tiene el paquete de licencias de tipo docentes
cant_student	integer	Cantidad de licencias de tipo estudiantes que tiene el programa de estudio.
charger_student	Float	Precio que tiene el paquete de licencias de tipo estudiantes.
cant_courtesy	integer	Cantidad de licencias de tipo cortesía que tiene el programa de estudio
charger_courtesy	Float	Precio que tiene el paquete de licencias de tipo cortesía.
crated_date	datatime	Muestra la fecha en que se activará el programa de estudio.
expiry_date	datatime	Muestra la fecha de vencimiento del programa de estudio.

Tabla 18: Descripción de la tabla tb_agreement_study_program.

tb_file		
Descripción: Esta tabla se encarga de guardar los datos donde se guardan los contratos.		
Atributo	**Tipo**	**Descripción**
Id	integer	Identificador del documento.
file_dir	varchar	Dirección de archivo donde se guarda el documento.

Tabla 19: Descripción de la tabla tb_file.

Anexo 5. Diseños de casos de prueba

SC 2: Ver datos del Rol

Escenario	Descripción		Flujo central
EC 2.1 “Ver datos”	Selecciona la opción de ver los datos del Rol.		Página principal/Ver
EC 2.2 “Selecciona cancelar”	Selecciona la opción de Cancelar.		Página principal/ Ver/ Cancelar
EC 2.3 “Modificar datos”	Selecciona la opción de Modificar los Roles.		Página principal/ edit
EC 2.4 “Eliminar datos”	Selecciona la opción de Eliminar los datos del Rol.		Página principal/ delete

SC 3: Modificar datos del Rol.

Escenario	Descripción	Nombre	Descripción	Respuesta del sistema	Flujo central

EC 3.1 "Selecciona modificar"	Selecciona la opción de modificar los datos del Rol.			Muestra los datos del Rol seleccionado y permite modificar sus valores: • Nombre. • Descripción Además permite: • Actualizar los datos. • Cancelar la operación en cualquier momento.	Página principal/ Editar
EC 3.2 "Modifica datos"	Modifica los datos que necesite y selecciona la opción de Actualizar.	*V*	*N/A*	Valida los datos. Actualiza los datos del Rol. Muestra los datos del Rol incluido. Ver SC 2: "Ver datos del Rol " Muestra un mensaje de confirmación.	Página principal/ Editar/ Actualizar
EC 3.3 "Selecciona cancelar"	Selecciona la opción de Cancelar.			Regresa al listado de Rol Muestra un mensaje de confirmación.	Página principal/ Editar/ Actualizar
EC 3.4 "Campos vacíos"	Existen datos incompletos.	*I*	*N/A*	Muestra un mensaje de información. Resalta en rojo el campo vacío. Regresa al EC 3,2	Página principal/ Editar/ Actualizar
		I	*N/A*		
		(Vacío)			
		V	*N/A*		
		V	*N/A*		
EC 3.5 "Campos incorrectos"	Existen datos incorrectos.	*I*	*N/A*	Muestra un mensaje de información. Muestra un indicador sobre cada uno de los	Principal/ edit
		I	*N/A*		

				campos incorrectos. Regresa al EC 3,2	
		V	N/A		
		V	N/A		
EC 3.6 "Ya existe"	El Rol ya existe.	I	V	Muestra un mensaje de información. Muestra un indicador sobre el campo. Regresa al EC 3,2	Principal/ Rol

Tabla 20: Descripción de Caso de Prueba Rol