

# Universidad de las Ciencias Informáticas

## Facultad 2



**Título:** Sistema de Gestión de Reportes de vulnerabilidades para el proyecto LABSI”.

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

**Autores:** Yanira Pantin Kindelán  
Humberto Arencibia Lorenzo

**Tutor:** Ing. Dairon Javier Soler González.

Ciudad de La Habana, 30 de Junio del 2010

“Año 52 de la Revolución”

*La felicidad consiste en sentirse a gusto en la vida, en ir realizando paso a paso nuestros ideales, en disfrutar los pequeños goces que lleva consigo la existencia, en cumplir nuestro deber y en realizar la vocación a la que uno ha sido llamado.*

*Padre Gregorio Mateo.*

# *Declaración de Autoría*

---

## **DECLARACIÓN DE AUTORÍA**

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste se firma la presente a los \_\_\_\_ días del mes de \_\_\_\_\_ del año \_\_\_\_\_.

Yanira Pantin Kindelán.

Humberto Arencibia Lorenzo.

\_\_\_\_\_  
Firma del Autor

\_\_\_\_\_  
Firma del Autor

Dairon Javier Soler González.

\_\_\_\_\_  
Firma del Tutor

*Agradezco:*

*A mi mamá, que ha estado ahí siempre, y me ha apoyado en todo momento.*

*A mi papá por la fuerza que me ha dado en todo.*

*A mi abuela Victoria, que ha sido un ejemplo para mí; es una gran persona.*

*A mi abuelo Manuel de Jesús, que siempre tiene los consejos exactos.*

*A mis tíos Marlene y Amilkar, por su amor tan especial.*

*A mis primos Jesús Abrahan y Arlen Aimee, por su alegría.*

*En general a toda mi familia, porque de una forma u otra han estado presentes en mi formación.*

*A mi compañero de tesis Humberto, por ayudarme a hacer su tesis.*

*A mis amigas Jessica, Laura y Arianna, con las que he compartido mucho.*

*A mis compañeros Yoannis Ordoñez y Adrian Yeja, por lo que me han enseñado.*

*Y a todos los que de una forma u otra me ayudaron en la realización de este trabajo y a lo largo de toda mi vida.*

***Yanira Pantin Kindelán***

## AGRADECIMIENTOS

- A mis abuelos chicha y nejo, la viejita más inteligente del mundo y el "sangabia".*
- A mi padre y mi hermano.*
- A mis abuelos Oreste y Maira Teresa, el viejo y la vieja.*
- A mi tía Elvis.*
- A toda mi familia en general.*
- A mis amigos Elvis y Yoandy, mis hermanos también.*
- A mi compañera de tesis Yanira por todo lo que me ayudo de forma desinteresada a hacer su tesis.*
- A mis amigos Marlon, Gustavo, Kike, Alie, Manuel, Lago, Pancho, Camilo, Pj, Omar y muchos otros con los que he compartido muy buenos momentos.*
- A todos mis socios del mundo (wow).*
- A mis senseis del PHP Adrian y Yosbany.*
- A nuestro tutor Dayron Javier Soler González.*
- A todas las personas que han ayudado de alguna manera en mi tesis.*

***Humberto Arencibia Lorenzo***

*A mi madre que es la luz que guía mi vida.*

***Yanira***

*A mi madre.*

*Humberto*

## RESUMEN

La Universidad de Ciencias Informáticas cuenta con un Laboratorio de Seguridad Informática (LABSI), el cual se encarga de realizar auditorías a los sistemas construidos en la universidad, para asegurarse de que no tienen brechas de seguridad, ya sea por un diseño deficiente o una implementación con elementos no seguros. El laboratorio también provee de una base de datos de vulnerabilidades para consulta de todos los usuarios de la UCI, y de sistemas para el análisis de código estático para los lenguajes de programación: PHP, Python y C++, entre otros servicios.

En LABSI se realizan auditorías a distintos sistemas, éstas se hacen con diferentes herramientas y por diferentes personas pertenecientes al proyecto, por lo que surgió la necesidad de crear un sistema de gestión de reportes de vulnerabilidades, que es una aplicación donde se gestionan (crear, insertar, modificar, eliminar, mostrar y listar), las auditorías, los usuarios del laboratorio de seguridad, las herramientas con la que se realizan las pruebas y los reportes arrojados por estas herramientas. También el sistema brinda la oportunidad de llevar las estadísticas del funcionamiento del laboratorio de seguridad.

Se hace una propuesta del uso de un livecd que contiene herramientas para hacer pruebas de penetración a sistemas Web, este fue el resultado de la investigación realizada para la construcción de la aplicación.

Para la realización del sistema se utilizó el lenguaje de programación **PHP**, la herramienta Case Visual **Paradigm**, el framework **KumbiaPHP**, el gestor de base de datos **PostgreSQL** y para guiar el desarrollo del sistema se utilizó **RUP**.

Palabras Claves: OWASP, vulnerabilidades, sistema de gestión, reporte.



## ÍNDICE

<b>DECLARACIÓN DE AUTORÍA</b> .....	II
<b>AGRADECIMIENTOS</b> .....	III
<b>AGRADECIMIENTOS</b> .....	IV
<b>DEDICATORIA</b> .....	V
<b>DEDICATORIA</b> .....	VI
<b>RESUMEN</b> .....	VII
<b>ÍNDICE</b> .....	VIII
<b>ÍNDICE DE ILUSTRACIONES</b> .....	XI
<b>ÍNDICE DE TABLAS</b> .....	XIV
<b>INTRODUCCIÓN</b> .....	1
<b>CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA</b> .....	5
1.1 Introducción. ....	5
1.2 Proyecto OWASP.....	5
1.3 Clasificación de los proyectos OWASP.....	5
1.3.1 Proyectos de Protección.....	5
1.3.2 Proyectos de Detección.....	6
1.3.3 Proyectos de Ciclo de Vida.....	7
1.4 OWASP-livecd-AustinTerrier-Feb2009 .....	8
1.4.1 Herramientas de OWASP-Livecd-AustinTerrier-Feb2009 .....	8
1.5 Sistemas que se dedican a gestionar reportes de seguridad .....	10
1.6 Necesidad del trabajo .....	11
1.7 Gestión de Reportes en el proyecto LABSI.....	11
1.8 Lenguajes de Programación. ....	11
1.8.1 Lenguajes de programación a utilizar para el desarrollo del sistema.....	12
1.9 Metodología de Desarrollo de Software. ....	13
1.9.1 Rational Unified Process (RUP). ....	13
1.9.2 Características de RUP. ....	13
1.10 Herramienta CASE a utilizar en el desarrollo del sistema. ....	15
1.10.1 Visual Paradigm para UML. ....	15

1.11 Gestor de Bases de Datos a usar en el sistema.....	15
1.11.1 PostgreSQL .....	16
1.12 Frameworks .....	16
1.12.1 KumbiaPHP .....	17
1.13 Conclusiones del Capítulo.....	17
<b>CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA.....</b>	<b>19</b>
2.1 Introducción. ....	19
2.2 Problema. ....	19
2.3 Objeto de automatización. ....	19
2.4 Información que se maneja. ....	20
2.5 Propuesta del sistema. ....	20
2.6 Modelo de dominio. ....	21
2.7 Especificación de los requisitos del software.....	22
2.7.1 Requisitos Funcionales.....	23
2.7.2 Requisitos no Funcionales .....	24
2.8 Modelo de Casos de Uso del Sistema. ....	26
2.8.1 Actores del Sistema.....	26
2.8.2 Listado de los Casos de Uso del Sistema. ....	27
2.8.3 Descripción textual de los Casos de Uso del Sistema .....	30
2.8.4 Prototipos de Interfaz de Usuario .....	30
2.9 Conclusión del Capítulo.....	30
<b>CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA.....</b>	<b>31</b>
3.1 Introducción. ....	31
3.2 Análisis. ....	31
3.2.1 Diagrama de clases del análisis .....	32
3.3 Diseño. ....	34
3.3.1 Patrones de Diseño. ....	35
3.3.2 Diagramas de clase del diseño Web. ....	38
3.3.3 Diagramas de secuencia del diseño. ....	41
3.4 Modelo Lógico de Datos. ....	41

3.5 Modelo Físico de Datos. ....	42
3.6 Conclusiones del Capítulo.....	43
<b>CAPÍTULO 4: IMPLEMENTACIÓN DEL SISTEMA.</b> .....	<b>44</b>
4.1 Introducción. ....	44
4.2 Diagrama de Despliegue.....	44
4.3 Diagrama de Componentes. ....	45
4.4 Conclusiones del Capítulo.....	48
<b>CAPÍTULO 5: ESTUDIO DE FACTIBILIDAD</b> .....	<b>49</b>
5.1 Introducción. ....	49
5.2 Método de estimación Por Puntos de Casos de Uso. ....	49
5.2.1 Cálculo de Puntos de Caso de Uso sin ajustar.....	49
5.2.2 Factor de Peso de los Actores sin ajustar. ....	50
5.2.3 Factor de Peso de los Casos de Uso sin ajustar. ....	51
5.2.4 Puntos de Casos de Uso ajustados.....	52
5.2.5 Factores de Complejidad Técnica. ....	52
5.2.6 Factores de Ambiente.....	54
5.2.7 Esfuerzo Horas-Hombres .....	55
5.3 Costos .....	56
5.4 Beneficios Tangibles e Intangibles .....	57
5.5 Análisis de Costos y Beneficios.....	57
5.6 Conclusiones del Capítulo.....	58
<b>CONCLUSIONES</b> .....	<b>59</b>
<b>RECOMENDACIONES</b> .....	<b>60</b>
<b>BIBLIOGRAFÍA</b> .....	<b>61</b>
<b>ANEXOS</b> .....	<b>63</b>
<b>GLOSARIO DE TÉRMINOS</b> .....	<b>99</b>

## ÍNDICE DE ILUSTRACIONES

<i>Figura 1: Logo de Visual Paradigm para UML.....</i>	<i>15</i>
<i>Figura 2: Logo de PostgreSQL.....</i>	<i>16</i>
<i>Figura 3: Modelo de Dominio.....</i>	<i>22</i>
<i>Figura 4: Diagrama de Casos de Uso del Sistema.....</i>	<i>26</i>
<i>Figura 5: Diagrama de Clases del Análisis-CUS Autenticar Usuario.....</i>	<i>32</i>
<i>Figura 6: Diagrama de Clases del Análisis-CUS Gestionar Usuario.....</i>	<i>32</i>
<i>Figura 7: Diagrama de Clases del Análisis-CUS Gestionar Reporte.....</i>	<i>33</i>
<i>Figura 8: Diagrama de Clases del Análisis-CUS Gestionar Herramientas.....</i>	<i>33</i>
<i>Figura 9: Diagrama de Clases del Análisis-CUS Gestionar Auditoría.....</i>	<i>34</i>
<i>Figura 10: Relación entre el Modelo, la Vista y el Controlador.....</i>	<i>36</i>
<i>Figura 11: Diagrama de clases del diseño Caso de Uso Autenticar Usuario.....</i>	<i>38</i>
<i>Figura 12: Diagrama de clases del diseño Caso de Uso Gestionar Usuario.....</i>	<i>39</i>
<i>Figura 13: Diagrama de clases del diseño Caso de Uso Gestionar Auditoría.....</i>	<i>39</i>
<i>Figura 14: Diagrama de clases del diseño Caso de Uso Gestionar Herramienta.....</i>	<i>40</i>
<i>Figura 15: Diagrama de clases del diseño Caso de Uso Gestionar Reporte.....</i>	<i>40</i>
<i>Figura 16: Diagrama de clases del diseño Caso de Uso Mostrar Estadística.....</i>	<i>41</i>
<i>Figura 17: Modelo Lógico de Datos.....</i>	<i>42</i>
<i>Figura 18: Modelo Físico de Datos.....</i>	<i>43</i>
<i>Figura 19: Diagrama de Despliegue.....</i>	<i>44</i>
<i>Figura 20: Diagrama de Componentes Base de Datos.....</i>	<i>45</i>
<i>Figura 21: Diagrama de Componentes Vistas, Controlador, Modelo.....</i>	<i>46</i>
<i>Figura 22: Diagrama de Componentes Código Fuente.....</i>	<i>47</i>
<i>Figura 23: Diagrama de Secuencia Caso de Uso Autenticar Usuario.....</i>	<i>78</i>
<i>Figura 24: Diagrama de Secuencia Caso de Uso Gestionar Usuario. Escenario Insertar Usuario.....</i>	<i>78</i>
<i>Figura 25: Diagrama de Secuencia Caso de Uso Gestionar Usuario. Escenario Modificar Usuario.....</i>	<i>79</i>
<i>Figura 26: Diagrama de Secuencia Caso de Uso Gestionar Usuario. Escenario Eliminar Usuario.....</i>	<i>79</i>
<i>Figura 27: Diagrama de Secuencia Caso de Uso Gestionar Herramienta. Escenario Adicionar Herramienta.....</i>	<i>80</i>

<i>Figura 28: Diagrama de Secuencia Caso de Uso Gestionar Herramienta. Escenario Modificar Herramienta.</i>	80
<i>Figura 29: Diagrama de Secuencia Caso de Uso Gestionar Herramienta. Escenario Eliminar Herramienta.</i>	81
<i>Figura 30: Diagrama de Secuencia Caso de Uso Listar Herramienta.</i>	81
<i>Figura 31: Diagrama de Secuencia Caso de Uso Gestionar Auditoría. Escenario Adicionar Auditoría.</i>	82
<i>Figura 32: Diagrama de Secuencia Caso de Uso Gestionar Auditoría. Escenario Modificar Auditoría.</i>	83
<i>Figura 33: Diagrama de Secuencia Caso de Uso Gestionar Auditoría. Escenario Eliminar Auditoría.</i>	83
<i>Figura 34: Diagrama de Secuencia Caso de Uso Listar Auditoría.</i>	84
<i>Figura 35: Diagrama de Secuencia Caso de Uso Gestionar Reporte. Escenario Crear Reporte.</i>	84
<i>Figura 36: Diagrama de Secuencia Caso de Uso Gestionar Reporte. Escenario Modificar Reporte.</i>	85
<i>Figura 37: Diagrama de Secuencia Caso de Uso Gestionar Reporte. Escenario Eliminar Reporte.</i>	85
<i>Figura 38: Diagrama de Secuencia Caso de Uso Listar Reporte.</i>	86
<i>Figura 39: Diagrama de Secuencia Caso de Uso Mostar Reporte.</i>	86
<i>Figura 40: Diagrama de Secuencia Caso de Uso Imprimir Reporte.</i>	87
<i>Figura 41: Diagrama de Secuencia Caso de Uso Mostar Estadística.</i>	87
<i>Figura 42: Prototipo Autenticar Usuario.</i>	88
<i>Figura 43: Prototipo Adicionar Auditoría.</i>	88
<i>Figura 44: Prototipo Adicionar Herramienta.</i>	89
<i>Figura 45: Prototipo Adicionar Reporte.</i>	89
<i>Figura 46: Prototipo Adicionar Usuario.</i>	90
<i>Figura 47: Prototipo Modificar Auditoría.</i>	90
<i>Figura 48: Prototipo Modificar Herramienta.</i>	91
<i>Figura 49: Prototipo Modificar Reporte.</i>	91
<i>Figura 50: Prototipo Modificar Usuario.</i>	92
<i>Figura 51: Prototipo Eliminar Auditoría.</i>	92
<i>Figura 52: Prototipo Eliminar Herramienta.</i>	93
<i>Figura 53: Prototipo Eliminar Reporte.</i>	93
<i>Figura 54: Prototipo Eliminar Usuario.</i>	94
<i>Figura 55: Prototipo Buscar Auditoría.</i>	94

<i>Figura 56: Prototipo Buscar Herramienta.</i> .....	95
<i>Figura 57: Prototipo Buscar Reporte.</i> .....	95
<i>Figura 58: Prototipo Buscar Usuario.</i> .....	96
<i>Figura 59: Prototipo Listar Auditoría.</i> .....	96
<i>Figura 60: Prototipo Listar Herramienta.</i> .....	97
<i>Figura 61: Prototipo Listar Reporte.</i> .....	97
<i>Figura 62: Prototipo Listar Usuario.</i> .....	98
<i>Figura 63: Prototipo Ver Estadísticas.</i> .....	98

## ÍNDICE DE TABLAS

<i>Tabla 1: Actores del Sistema.....</i>	<i>27</i>
<i>Tabla 2: Caso de Uso Autenticar Usuario. ....</i>	<i>27</i>
<i>Tabla 3: Caso de Uso Mostrar Reporte.....</i>	<i>27</i>
<i>Tabla 4: Caso de Uso Gestionar Reporte.....</i>	<i>28</i>
<i>Tabla 5: Caso de Uso Imprimir Reporte. ....</i>	<i>28</i>
<i>Tabla 6: Caso de Uso Listar Auditoría. ....</i>	<i>28</i>
<i>Tabla 7: Caso de Uso Listar Reporte. ....</i>	<i>28</i>
<i>Tabla 8: Caso de Uso Listar Herramientas.....</i>	<i>29</i>
<i>Tabla 9: Caso de Uso Mostrar Estadísticas.....</i>	<i>29</i>
<i>Tabla 10: Caso de Uso Gestionar Auditoría. ....</i>	<i>29</i>
<i>Tabla 11: Caso de Uso Gestionar Herramienta. ....</i>	<i>29</i>
<i>Tabla 12: Caso de Uso Gestionar Usuario. ....</i>	<i>30</i>
<i>Tabla 13: Clases del Análisis. ....</i>	<i>31</i>
<i>Tabla 14: Cálculo del factor de peso de los actores sin ajustar. ....</i>	<i>50</i>
<i>Tabla 15: Peso de las Transacciones. ....</i>	<i>51</i>
<i>Tabla 16: Peso de los Factores de Complejidad Técnica.....</i>	<i>53</i>
<i>Tabla 17: Cálculo del Valor de TFactor. ....</i>	<i>54</i>
<i>Tabla 18: Peso de los Factores Ambientales. ....</i>	<i>55</i>
<i>Tabla 19: Cálculo del Valor de EFactor.....</i>	<i>55</i>
<i>Tabla 20: Cantidad de Horas-Hombres según el valor.....</i>	<i>56</i>
<i>Tabla 21: Descripción textual del Caso de Uso Autenticar Usuario. ....</i>	<i>64</i>
<i>Tabla 22: Descripción textual del Caso de Uso Mostrar Reporte.....</i>	<i>64</i>
<i>Tabla 23: Descripción textual del Caso de Uso Gestionar Reporte.....</i>	<i>66</i>
<i>Tabla 24: Descripción textual del Caso de Uso Imprimir Reporte. ....</i>	<i>67</i>
<i>Tabla 25: Descripción textual del Caso de Uso Listar Auditoría ....</i>	<i>68</i>
<i>Tabla 26: Descripción textual del Caso de Uso Listar Reporte. ....</i>	<i>69</i>
<i>Tabla 27: Descripción textual del Caso de Uso Listar Herramientas.....</i>	<i>69</i>
<i>Tabla 28: Descripción textual del Caso de Uso Mostrar Estadística. ....</i>	<i>70</i>

<i>Tabla 29: Descripción textual del Caso de Uso Gestionar Auditoría. ....</i>	<i>72</i>
<i>Tabla 30: Descripción textual del Caso de Uso Gestionar Herramienta. ....</i>	<i>75</i>
<i>Tabla 31: Descripción textual del Caso de Uso Listar Reporte. ....</i>	<i>77</i>



## INTRODUCCIÓN

Con el desarrollo de las tecnologías de la informática y las comunicaciones la información se procesa de forma digital usando aplicaciones que facilitan el trabajo. Algunas de las tareas que se han automatizado son, transacciones bancarias, transferencia de datos, almacenamiento de grandes volúmenes de información, entre otras.

Sin embargo, en la creación de un software muchas veces los desarrolladores no se ocupan de la seguridad del mismo o no utilizan los recursos adecuados y en consecuencia el producto final queda con brechas que ponen en riesgo datos sensibles que son manejados por el sistema. Estas brechas son conocidas como vulnerabilidades, las mismas son debilidades de los sistemas que pueden ser explotadas para facilitar ataques como robo de identidades, ejecución de códigos malignos, apertura de puertos y obtención de permisos no autorizados.

Estas vulnerabilidades están presentes en casi todas las aplicaciones con las que se trabaja, en el caso de los sistemas Web es más necesario tener los mecanismos para mantener protegidos los datos que en ellos se manejan, ya que la información que se encuentra en estos sistemas se ve más amenazada porque está en un medio inseguro como es la red. Para que un sistema sea seguro debe cumplir con las siguientes características: **Integridad**, esta manifiesta que la información solo puede ser modificada por la persona con los permisos para hacerlo y de manera controlada, **Confidencialidad**, esta propone que la información solo puede ser leída por los usuarios autorizados, **Disponibilidad**, ella hace referencia a que la información debe estar disponible cuando se necesite e **Irrefutabilidad**, esta hace alusión a que el uso o modificación por parte de un usuario de la información debe ser irrefutable.

Para comprobar la seguridad de un sistema y con el objetivo de que los auditores puedan medir la eficacia de sus mecanismos de defensa contra ataques informáticos, se han creado métodos que ayudan a proteger la información manejada por estos. Uno de estos métodos es hacerle pruebas de seguridad al sistema, estas pruebas encuentran algunas deficiencias presentes e informan al propietario sobre los problemas identificados.

Entre estas pruebas se encuentran las pruebas de penetración. Una prueba de penetración es un examen independiente usado para auditar aplicaciones Web, donde se simulan posibles acciones de usuarios no

autorizados, con el objetivo de detectar si el sistema es seguro, previendo que se descarguen datos sensibles o se ponga en riesgo la información que se maneja en la aplicación.

Para hacer las pruebas anteriormente mencionadas y con el objetivo de que los sistemas Web sean cada vez más seguros se utilizan distintas herramientas y metodologías que indican cómo construir una aplicación Web y cómo comprobar su seguridad. En el proyecto Laboratorio de Seguridad (LABSI), se utilizan las herramientas y metodologías que propone el Proyecto Abierto de Seguridad en Aplicaciones Web (OWASP por sus siglas en inglés *Open Web Application Security Project*) un proyecto que brinda documentación y herramientas de forma libre y gratuita, más adelante se explica con detenimiento.

En el proyecto LABSI, los reportes arrojados por las pruebas que se realizan en las auditorías a los distintos sistemas informáticos, son almacenados de forma individual por la persona que realizó la prueba, también estos reportes están dados en distintos formatos, los cuales son en la mayoría de los casos muy difíciles de interpretar. Esto provoca que el proceso de revisión, por parte del cliente, de los resultados de las pruebas realizadas a su sistema sea muy complicado ya que requiere de la explicación de la persona que realizó la prueba. También es muy compleja la creación de un reporte general donde estén comprendidos todos los resultados de cada una de las pruebas y donde el cliente pueda ver en texto plano cuáles son las principales vulnerabilidades presentes en su sistema y de ser posible una posible solución. Otro problema es que en el proyecto LABSI no existe un lugar centralizado donde se almacenen todos los reportes de las pruebas efectuadas a los distintos sistemas Web, por lo cual tampoco se puede hacer un análisis estadístico de cuánto se ha avanzado de una auditoría a otra en un mismo sistema y de cuáles son los principales problemas que se repiten en las distintas aplicaciones sometidas a prueba.

A partir de la anterior situación problemática, el **problema a resolver** es: ¿Cómo gestionar los reportes de las pruebas realizadas en el proyecto LABSI?

Por lo tanto el **objeto de estudio** es la realización de las pruebas de seguridad y la gestión de los reportes arrojados por las mismas a los sistemas auditados por el proyecto LABSI y dentro de él el **campo de acción** es el proceso de gestión de los reportes generados por las pruebas de seguridad.

El **objetivo general** del presente trabajo es desarrollar una aplicación Web que gestione los reportes de las pruebas de seguridad realizadas a sistemas informáticos, por el proyecto LABSI.

Las **tareas investigativas** para cumplir dicho objetivo son:

- ✓ Estudio previo del estado del arte sobre las aplicaciones que se dedican a gestionar reportes de seguridad.
- ✓ Estudio del proyecto OWASP (Open Web Application Security Project).
- ✓ Estudio de las herramientas que propone OWASP para realizar pruebas de penetración a sistemas Web.
- ✓ Determinación de las herramientas a utilizar en el proyecto LABSI para hacer pruebas de penetración a sistemas Web.
- ✓ Estudio de la gestión de los reportes dados por las pruebas de seguridad que se hacen en el proyecto LABSI en la actualidad.
- ✓ Definición de las herramientas a usar para el desarrollo de la aplicación.
- ✓ Definición del diseño del sistema para realizar la gestión de los reportes de seguridad del proyecto LABSI.
- ✓ Implementación del sistema de gestión de reportes de las pruebas realizadas en el proyecto LABSI.

El **aporte práctico** de este trabajo es una aplicación Web que gestione los reportes de seguridad arrojados por las pruebas realizadas en el proyecto LABSI a sistemas informáticos.

El presente documento está estructurado en cinco capítulos.

En el **Capítulo 1** “Fundamentación Teórica”, se realiza un estudio sobre las aplicaciones que gestionan reportes de seguridad, el proyecto OWASP y las herramientas que propone para la ejecución de pruebas de penetración a sistemas Web y los reportes arrojados por las mismas. Se definen las herramientas a utilizar en el proyecto LABSI para hacer pruebas de penetración y además se determinan las herramientas para el desarrollo del sistema que se va a construir.

En el **Capítulo 2** “Características del Sistema”, se brinda una visión del sistema, se definen los requisitos funcionales y no funcionales y se ofrece una propuesta del sistema a construir.

En el **Capítulo 3** “Análisis y Diseño del Sistema”, se presentan los diagramas de clases del análisis, además el diagrama de clases del diseño, para mostrar el funcionamiento del sistema.

En el **Capítulo 4** “Implementación del Sistema”, se presentan el diagrama de despliegue y el diagrama de componentes como resultado de la implementación del sistema.

En el **Capítulo 5** “Estudio de Factibilidad”, se determina el tiempo que será necesario consumir y el costo para la realización del sistema.

# Capítulo 1: Fundamentación Teórica

---

## CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

### 1.1 Introducción.

En el presente capítulo se presenta el resultado de una investigación sobre el proyecto OWASP, las metodologías y herramientas que este propone, además de un análisis sobre las herramientas del livecd owasp-livecd-AustinTerrier-Feb2009 y los reportes que estas brindan como resultado. También una investigación sobre los sistemas que se dedican a gestionar reportes de seguridad y un análisis del lenguaje y las herramientas más adecuadas para desarrollar una aplicación informática de este tipo.

### 1.2 Proyecto OWASP.

OWASP es una fundación creada en el 2004, sin fines de lucro. El objetivo principal de OWASP es determinar y contrarrestar las causas que hacen a un sistema Web inseguro. Es un proyecto de software libre que brinda artículos, herramientas, metodologías, documentación y tecnologías que pueden ser usadas de forma gratuita (1).

### 1.3 Clasificación de los proyectos OWASP.

Los proyectos de OWASP son un conjunto de tareas que tiene un plan de trabajo definido, estos se clasifican en 3 categorías:

- ✓ Protección.
- ✓ Detección.
- ✓ Ciclo de Vida.

#### 1.3.1 Proyectos de Protección.

Son las herramientas y documentos que se pueden utilizar para protegerse contra vulnerabilidades resultantes de diseños ineficientes y aplicaciones defectuosas.

Las herramientas que son utilizadas para esta categoría son:

## Capítulo 1: Fundamentación Teórica

---

### **Proyecto OWASP AntiSamy:**

Es una API para asegurar que las entradas HTML/CSS del usuario cumplan con las reglas de la aplicación y asegurarse de que los clientes no manden código malicioso en el HTML que envían con su perfil, comentarios y otras acciones que realicen en el sistema.

### **Proyecto OWASP Enterprise Security API (ESAPI):**

Es un paquete de herramientas que proporciona ayuda a los desarrolladores de software para protegerse contra las vulnerabilidades resultantes de diseños ineficientes y aplicaciones defectuosas.

La documentación utilizada en esta categoría es:

### **Guía de desarrollo de OWASP:**

Dirigida a arquitectos, desarrolladores y auditores, es un manual completo para el diseño, desarrollo y despliegue de aplicaciones y servicios Web seguros.

### **Proyecto OWASP. NET:**

Este proyecto tiene como objetivo ofrecer un repositorio general con información y herramientas referentes a la seguridad para la comunidad de desarrolladores que usan el framework de .NET.

### **Guía de seguridad de OWASP V2 para Ruby on Rails:**

Es una guía orientada a ofrecer información relacionada con la seguridad para las aplicaciones Web desarrolladas con Ruby on Rails.

### **1.3.2 Proyectos de Detección.**

Son las herramientas y documentos que se pueden utilizar para detectar vulnerabilidades resultantes de diseños y aplicaciones defectuosos.

Las herramientas que se utilizan en esta categoría:

### **Proyecto OWASP JBroFuzz:**

# Capítulo 1: Fundamentación Teórica

---

Es una aplicación Web que tiene como propósito ofrecer una única aplicación portable con un protocolo Web estable.

## **Proyecto de Livecd de OWASP:**

Este proyecto tiene como objetivo brindar un paquete de herramientas de OWASP, que se utilizan para realizar pruebas de penetración y la documentación sobre las mismas.

## **Proyecto OWASP WebScarab**

Es un proyecto donde se desarrolla la herramienta WebScarab que es un entorno de análisis para las aplicaciones que se comunican usando los protocolos HTTP o HTTPS.

La documentación utilizada por esta categoría es:

## **OWASP Application Security Verification Standard Project:**

Define un estándar para realizar evaluaciones a la seguridad en aplicaciones Web.

## **OWASP Code Review Guide v 1.1:**

Es una guía de revisión de código que da a la comunidad de desarrolladores un punto de partida en relación con la creación de aplicaciones seguras.

## **OWASP Testing Guide:**

Brinda a los usuarios una guía de las técnicas más comunes para realizar pruebas de penetración a aplicaciones y servicios Web.

## **OWASP Top Ten Project:**

Es un documento en el que se presenta un amplio consenso sobre los fallos de seguridad más críticos en cuanto a aplicaciones Web.

### **1.3.3 Proyectos de Ciclo de Vida**

Son las herramientas y documentos que se pueden utilizar para dar seguridad durante el ciclo de vida del desarrollo de software.

# Capítulo 1: Fundamentación Teórica

---

Las herramientas que se utilizan en esta categoría son:

## **Proyecto OWASP WebGoat:**

Es una aplicación para dar lecciones sobre cómo diseñar una aplicación Web segura. Es un sistema que en cada lección los usuarios deben demostrar su comprensión de una vulnerabilidad real en la aplicación WebGoat.

La documentación utilizada para esta categoría es:

## **OWASP AppSec FAQ Project:**

Este proyecto brinda la posibilidad de que el usuario conozca las preguntas más frecuentes que se hacen con respecto al proyecto OWASP y a la seguridad en aplicaciones Web.

## **OWASP Legal Project.**

Es una ayuda que le permite a los desarrolladores de software y clientes establecer los principales términos estructurales sobre el desarrollo y entrega de un software seguro.

## **OWASP Source Code Review for OWASP-Projects:**

Tiene como objetivo desarrollar y documentar un flujo de trabajo para el desarrollo de proyectos de software libre.

## **1.4 OWASP-livecd-AustinTerrier-Feb2009**

Es un proyecto que se encarga de reunir las principales herramientas de OWASP y su documentación, para la detección de vulnerabilidades. Es una mejora de OWASP Live CD 2007 y de Portugal release Dic 2008. Tiene entre sus objetivos asegurarse de que las herramientas de OWASP sean tan fáciles de usar como sea posible y facilitar a los usuarios el soporte, mantenimiento y actualización de las herramientas y documentación de OWASP.

### **1.4.1 Herramientas de OWASP-Livecd-AustinTerrier-Feb2009**

- ✓ OWASP WebScarab



# Capítulo 1: Fundamentación Teórica

---

- ✓ OWASP WebGoat
- ✓ OWASP CAL9000
- ✓ OWASP JBroFuzz
- ✓ Paros Proxy
- ✓ Nmap & Zenmap
- ✓ Wireshark
- ✓ Tcpdump
- ✓ Firefox 3
- ✓ Burp Suite
- ✓ Grendel-Scan
- ✓ OWASP DirBuster
- ✓ OWASP SQLiX
- ✓ OWASP WSFuzzer
- ✓ Metasploit 3
- ✓ W3af & GTK GUI for w3af
- ✓ Netcats collection
- ✓ OWASP Wapiti
- ✓ Nikto
- ✓ Fierce Domain Scanner

- ✓ Maltego CE
- ✓ Httpprint
- ✓ SQLBrute
- ✓ Spike Proxy
- ✓ Rat Proxy
- ✓ Webshag
- ✓ The next module

### 1.5 Sistemas que se dedican a gestionar reportes de seguridad

La gestión, almacenamiento y organización de los reportes de pruebas de seguridad y las vulnerabilidades que estos contienen es una tarea de gran envergadura para la comunidad de seguridad informática a nivel mundial, ya que permite que los interesados en el tema tengan como referencia para su trabajo los reportes y vulnerabilidades que ya han sido detectadas por otros usuarios y sirve de base en el desarrollo de nuevos sistemas más seguros y estables.

En Internet existen muchas aplicaciones que se dedican a la realización de pruebas de seguridad a diferentes sistemas, pero estos no brindan información detallada sobre como realizan estas pruebas, es decir, con que herramientas, quién realizó la prueba, y como gestionan los reportes que son arrojados como resultado de estas pruebas, entre otras informaciones de interés para los clientes.

Algunos de estos sistemas son:

- ✓ [www.cgsi.com.ve/index.htm](http://www.cgsi.com.ve/index.htm)
- ✓ [www.consultoresgt.com/Default.aspx](http://www.consultoresgt.com/Default.aspx)
- ✓ [www.softwebasesores.com/index2.php](http://www.softwebasesores.com/index2.php)
- ✓ [www.accurateqa.com/ES/ManualTest.aspx](http://www.accurateqa.com/ES/ManualTest.aspx)

### **1.6 Necesidad del trabajo**

Es necesario un sistema que gestione las vulnerabilidades encontradas y los reportes dados por las pruebas que se realicen en el proyecto LABSI; este sistema permitirá gestionar de forma particular los resultados arrojados por las auditorías a las aplicaciones probadas por el proyecto y establecerá un análisis estadístico de cuánto se ha avanzado con respecto a anteriores auditorías realizadas a un mismo sistema, incluyendo cuáles son las principales vulnerabilidades presentes en las distintas aplicaciones evaluadas. Este trabajo también servirá como un repositorio de reportes, lo que permitirá a los clientes conocer y consultar las deficiencias encontradas en sus aplicaciones, desde sus ordenadores y sin necesidad de acudir a LABSI.

### **1.7 Gestión de Reportes en el proyecto LABSI**

Con el objetivo de realizar una adecuada gestión de los reportes de las pruebas pertenecientes a las auditorías que se realicen a sistemas informáticos por el proyecto LABSI, en el presente trabajo se propone una organización de los mismos atendiendo a la herramienta que generó cada reporte y la información relacionada con la auditoría a la cual pertenece.

### **1.8 Lenguajes de Programación.**

Un lenguaje de programación es un idioma artificial diseñado para expresar acciones que deben ser ejecutadas por un ordenador, es el medio de comunicación entre el programador y la máquina, como todo lenguaje está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y significado (2).

Los lenguajes de programación se clasifican en dos grupos fundamentales en cuanto a la manera en que pueden ser ejecutados: Primero los programas compilados, son aquellos donde el código fuente del programa se traduce totalmente a lenguaje máquina y se guarda para ser ejecutado posteriormente, el otro tipo de lenguaje es el interpretado donde el intérprete tiene que ir traduciendo en cada instrucción el significado del código fuente a código máquina, por lo cual los programas interpretados son más lentos en tiempo de ejecución pero más rápido en tiempo de compilación, ya que no es necesario hacer una

# Capítulo 1: Fundamentación Teórica

---

traducción completa del código fuente sino una interpretación del segmento de código que se necesita traducir. Algunos ejemplos de lenguajes de programación son: C, Pascal, Delphi, Python, PHP, Java, entre otros.

## 1.8.1 Lenguajes de programación a utilizar para el desarrollo del sistema

Para el desarrollo de la aplicación se necesita un lenguaje de programación que tenga dentro de sus características: Que soporte la programación orientada a objetos, que sea multiplataforma, que permita la conexión con distintos gestores de bases de datos y que sea libre. Existen varios lenguajes de programación que cumplen con estas características como son: Python, PHP, Java, entre otros.

### 1.8.1.1 ¿Por qué PHP?

Se ha escogido PHP para el desarrollo de la aplicación porque es un lenguaje de programación que cumple con los requisitos necesarios para la construcción del sistema a desarrollar. PHP Hipertext Preprocesor es un lenguaje del lado del servidor con una amplia librería de funciones y mucha documentación, es rápido, gratuito e independiente de la plataforma, lo que significa que puede ser ejecutado en diferentes sistemas operativos como son: Windows, Linux, Mac OS X, entre otros. Es un lenguaje de programación interpretado, de código abierto y de alto nivel, ampliamente utilizado en el mundo por los desarrolladores debido a su simplicidad y potencia (3).

Usando PHP es posible conectarse a distintos gestores de bases de datos como son: MySQL, PostgreSQL, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite, lo que permite flexibilidad y robustez en el desarrollo de aplicaciones.

### 1.8.1.2 ¿Por qué JavaScript?

Es un lenguaje de programación interpretado del lado del cliente que posibilita la interacción entre el usuario y el sistema mediante un navegador Web, es un lenguaje bastante sencillo y ligero que permite a usuarios sin mucha experiencia utilizarlo en la creación de aplicaciones. Se pueden realizar operaciones como la validación de campos en los formularios.

## 1.9 Metodología de Desarrollo de Software.

Las metodologías de desarrollo de software definen el conjunto de acciones que guían los esfuerzos de las personas implicadas en el proyecto con el objetivo de lograr que se llegue a completar la construcción del software y que éste cumpla los requerimientos del usuario. Las metodologías de desarrollo de software se pueden clasificar en:

### Ágiles

Exigen poca documentación y sirven para trabajar en proyectos donde los requisitos no se conocen con exactitud. Ejemplos de estas son: XP (Extreme Programming), SCRUM, entre otras.

### Robustas o Tradicionales

Están hechas para dar solución a proyectos, con soluciones complejas y de gran tamaño, donde es esencial una documentación amplia y detallada de cada paso en el desarrollo del software. Ejemplos de estas son: Microsoft Solution Framework (MSF), Rational Unified Process (RUP), entre otras.

#### 1.9.1 Rational Unified Process (RUP).

Es una metodología de desarrollo de software robusta, utiliza UML como lenguaje de modelado para la descripción del sistema, ayuda a guiar a un equipo de trabajo mejorando la productividad del mismo y orientándolo hacia las mejores prácticas de desarrollo, con el objetivo de lograr un software de calidad que cumpla con los requisitos del cliente.

#### 1.9.2 Características de RUP.

##### Dirigido por casos de uso

Los casos de uso son el hilo conductor de todo el proceso, ya que se basan en dar respuestas a las necesidades de los usuarios finales del sistema y después de captar estos requisitos en el modelamiento del negocio, todo el proceso queda guiado por los mismos y los modelos que se obtienen como resultado de los diferentes flujos de trabajo representan la realización de estos casos de uso.

##### Iterativo Incremental

# Capítulo 1: Fundamentación Teórica

---

RUP propone que cada fase se desarrolle en iteraciones donde cada iteración tiene tareas para cada flujo de trabajo y en cada una se divide el proyecto en mini proyectos que deben dar un resultado de cada iteración que constituye un paso más en el desarrollo del sistema.

## **Centrado en la arquitectura**

En la arquitectura se abarcan las diferentes vistas del sistema, lo que permite tener una visión completa del mismo, se describen los casos de uso más significativos y por cuál debe comenzar la implementación del software.

RUP cuenta con cuatro fases:

- ✓ Inicio.
- ✓ Elaboración.
- ✓ Construcción.
- ✓ Transición.

En cada fase se ejecutan una o varias iteraciones de tamaño variable y se definen tareas para cada uno de los flujos de trabajo, estos flujos son:

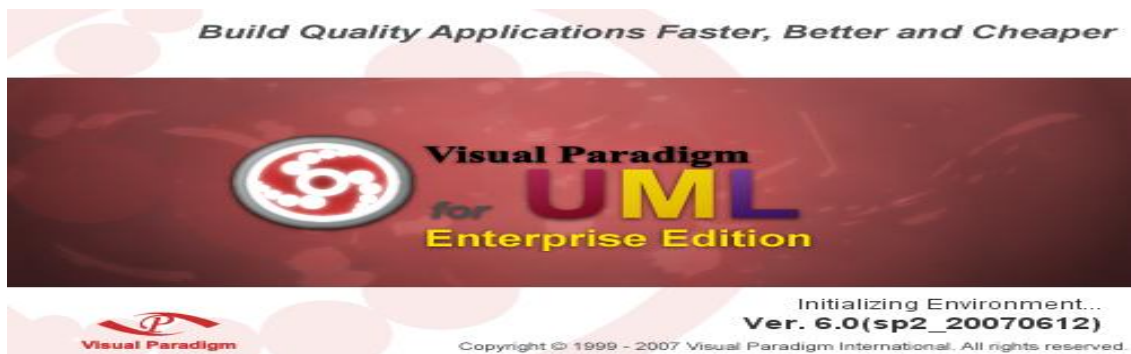
- ✓ Modelado del negocio
- ✓ Requerimientos
- ✓ Análisis y diseño
- ✓ Implementación
- ✓ Prueba
- ✓ Instalación
- ✓ Administración del proyecto
- ✓ Administración de configuración y cambios
- ✓ Ambiente

# Capítulo 1: Fundamentación Teórica

## 1.10 Herramienta CASE a utilizar en el desarrollo del sistema.

Computer Aided Software Engineering (CASE), son un grupo de herramientas que tienen como objetivo brindar ayuda a los analistas, ingenieros de software y desarrolladores, en el proceso de construcción de un software. Se puede definir también como la unión de las herramientas automáticas y las metodologías de desarrollo de software formales. Algunos ejemplos de herramientas CASE son: ERwin, Rational Rose, Visual Paradigm, EasyCASE, entre otros (4).

### 1.10.1 Visual Paradigm para UML.



**Figura 1: Logo de Visual Paradigm para UML.**

Visual Paradigm para UML es una herramienta que soporta el ciclo de vida completo de un software, ofrece una agradable interfaz gráfica y facilita el trabajo en equipo, por lo que permite desarrollar un software con mayor rapidez y calidad. Brinda la posibilidad de hacer diseños orientados a objeto y es gratis. Está diseñado para una amplia gama de usuarios que incluye a los Ingenieros de Software, Analistas del Sistema, Analistas del Negocio, Arquitectos, entre otros (5).

## 1.11 Gestor de Bases de Datos a usar en el sistema

Los sistemas de gestión de base de datos (SGBD) ó DataBase Management System (DBMS) son programas especializados en servir como interfaz entre las bases de datos y las aplicaciones que las utilizan tienen como objetivo general manejar de forma clara y sencilla los datos que se gestionan en el sistema. Algunos gestores de bases de datos son: PostgreSQL, MySQL, Oracle, MaxDb, SQLite, entre otros.

## 1.11.1 PostgreSQL



*Figura 2: Logo de PostgreSQL.*

PostgreSQL es un sistema de bases de datos objeto-relacional. Es libre y su código fuente completo está disponible. El desarrollo de PostgreSQL es realizado por un equipo de desarrolladores en su mayoría voluntarios extendido por todo el mundo, que se comunican vía Internet. Se trata de un proyecto comunitario y no está controlado por compañía alguna.

Sus características técnicas la hacen una de las bases de datos más potentes y robustas del mercado. Estabilidad, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema.

## 1.12 Frameworks

Es un entorno de trabajo que facilita el proceso de desarrollo de un software y tiene como características fundamentales:

- ✓ Definir una filosofía de trabajo.
- ✓ Proporcionar librerías y funciones.
- ✓ Ahorrar trabajo y tiempo.
- ✓ Permitir la producción de aplicaciones más fáciles de mantener.
- ✓ Evitar código duplicado.
- ✓ Crear Aplicaciones Multi-Capa.



# Capítulo 1: Fundamentación Teórica

---

## 1.12.1 KumbiaPHP

KumbiaPHP es un Web framework libre escrito en PHP5. Basado en las mejores prácticas de desarrollo Web, usado en software comercial y educativo, KumbiaPHP fomenta la velocidad y eficiencia en la creación y mantenimiento de aplicaciones Web, reemplazando tareas de codificación repetitivas por poder, control y placer (6).

Está basado en el modelo MVC (Modelo Vista Controlador). Permite la separación de las reglas de negocio, lógica de aplicación y vistas de presentación de una aplicación Web. Además posee otras herramientas y clases que ayuden a acortar el tiempo de desarrollo de una aplicación Web.

Las principales ventajas que brinda KumbiaPHP son:

- ✓ Implementa los mejores patrones de programación orientados a la Web.
- ✓ Fomenta la utilización de características Web 2.0 en el software.
- ✓ Hace la mayor parte del trabajo y se ocupa de los “detalles”.
- ✓ Es más fácil mantener una aplicación.
- ✓ Es software libre, por lo tanto se obtienen todas las ventajas que este proporciona.
- ✓ Su documentación está principalmente en español.

## 1.13 Conclusiones del Capítulo

Se ha presentado la investigación sobre el estado del arte de los laboratorios de seguridad, así como la realización de las pruebas de seguridad en estos; también se realizó un estudio sobre OWASP, sobre sus proyectos y se hizo énfasis en los proyectos de detección, de ahí se hizo una propuesta de un livecd que contiene herramientas para hacer pruebas de penetración, a sistemas Web. Asimismo se llegó a la conclusión de que los laboratorios de seguridad encontrados no tienen un sistema para gestionar sus reportes, es decir, solo realizan las pruebas a un sistema determinado, pero no llegan a dar los datos específicos, como las herramientas utilizadas, los datos del personal que hizo cada prueba, etc. Luego de esta investigación se decidió que el proyecto LABSI necesitaba un Sistema de Gestión de Reportes, para manejar los datos pertinentes a cada una de las pruebas realizadas por el mismo. Para hacer dicho sistema se decidieron las herramientas a utilizar, como: lenguajes de programación PHP y Java Script;

## *Capítulo 1: Fundamentación Teórica*

---

metodología de desarrollo de software RUP; la herramienta CASE escogida es Visual Paradigm; el gestor de base de datos es PostgreSQL y el framework KumbiaPHP.

# Capítulo 2: Características del Sistema

---

## CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

### 2.1 Introducción.

En este capítulo se presentan las características que debe cumplir el sistema, se describen los requisitos funcionales y no funcionales, así como un análisis detallado del problema a resolver y a partir de esto se hace una propuesta de la aplicación, también se muestran los actores y trabajadores del sistema, además el modelo de dominio, el diagrama de caso de uso del sistema, la descripción textual de estos casos de uso y los prototipos de interfaz de usuario.

### 2.2 Problema.

En el proyecto LABSI, los reportes arrojados por las pruebas que se realizan en las auditorías a sistemas Web son almacenados de forma individual por la persona que realizó la prueba. Esto provoca que el proceso de la revisión, por parte del cliente, de los resultados arrojados por las pruebas realizadas a su sistema, sea muy complicado, ya que requiere de la explicación de la persona que realizó la prueba. Otro es que en este proyecto no existe un lugar centralizado donde se almacenen todos los reportes de las pruebas realizadas a los distintos sistemas informáticos y tampoco se puede hacer un análisis estadístico de cuánto se ha avanzado de una auditoría a otra.

Esta situación provoca que el proceso de gestión de los reportes de las pruebas de seguridad, realizadas por el proyecto LABSI a sistemas informáticos, sea ineficiente e inadecuado, por lo cual se necesita una aplicación informática que se dedique a esta tarea.

### 2.3 Objeto de automatización.

El objeto a automatizar es el proceso de gestión de los reportes de las pruebas de seguridad que realizan los integrantes del proyecto LABSI, a los sistemas informáticos auditados por el laboratorio, así como el proceso de revisión de dichos reportes por parte del cliente.

Los clientes y los integrantes del proyecto LABSI tendrán diferencias en cuanto a la forma en que interactúan con el sistema. El sistema permitirá crear nuevas auditorías, así como guardar todos los datos

## *Capítulo 2: Características del Sistema*

---

referentes a las pruebas de seguridad realizadas al sistema, que esté en proceso de análisis. Igualmente brindará a los clientes la posibilidad de consultar los resultados de las pruebas realizadas a su sistema y una descripción de las mismas.

### **2.4 Información que se maneja.**

La información que se maneja en el sistema es la referente a las pruebas de seguridad que se le realicen a un sistema informático, por el proyecto LABSI, así como las herramientas utilizadas para las pruebas, los reportes arrojados por ellas y el usuario que las realizó.

### **2.5 Propuesta del sistema.**

En el presente trabajo, y con el propósito de dar solución al problema científico planteado, teniendo en cuenta los estudios e investigaciones realizadas y dada las necesidades del proyecto, se propone la creación de un sistema Web que permita la gestión de los reportes de las pruebas de seguridad hechas por los integrantes del proyecto LABSI.

El sistema será creado y administrado usando el framework KumbiaPHP y para la implementación se usará el lenguaje de programación PHP 5. La aplicación debe permitir a los usuarios que interactúan con el sistema en todo el proceso de realización de una auditoría, hacer una gestión eficiente de los reportes generados por las distintas pruebas efectuadas. También ofrecerá datos estadísticos sobre las auditorías hechas por el proyecto. Para garantizar el correcto funcionamiento de la aplicación se han creado roles que tributan al nivel de acceso que tiene cada usuario al sistema, estos son: Administrador, Auditor, Cliente y Supervisor.

El Administrador del sistema puede crear, eliminar, listar y modificar, los usuarios, las auditorías y las herramientas con las que se realizan las pruebas de seguridad. Establecer los permisos según el nivel de acceso que tiene cada usuario en el sistema. También puede consultar datos estadísticos con respecto a las auditorías hechas por el proyecto LABSI.

## *Capítulo 2: Características del Sistema*

---

El Auditor es el encargado de suministrar el reporte que ha obtenido como resultado de la prueba de seguridad y los datos de la misma. También debe dar una descripción de dicho reporte, que contenga, de ser posible, la solución de los problemas detectados.

El Cliente es el usuario que solicitó la auditoría a su sistema y tiene la posibilidad de ver todos los datos relacionados con las pruebas que le fueron realizadas, por los auditores del proyecto LABSI.

El Supervisor tiene la posibilidad de acceder a todos los datos de las auditorías realizadas por el proyecto LABSI, así como consultar datos estadísticos con respecto a las mismas.

### **2.6 Modelo de dominio.**

El proceso de realización de las pruebas de penetración por parte de los auditores del proyecto LABSI no está bien definido, lo que provoca que no exista un flujo estable de actividades que describa de forma exacta cómo se ejecutan las auditorías. Por lo cual se propone la realización de un modelo de dominio para desarrollar el negocio.

El objetivo del modelo de dominio es contribuir a la comprensión del contexto del sistema, y por lo tanto también contribuir a la comprensión de los requisitos del sistema que se desprenden de este contexto. En otras palabras el modelado del dominio debería contribuir a la comprensión del problema que se supone que el sistema resuelve en relación a su contexto (7).

## Capítulo 2: Características del Sistema

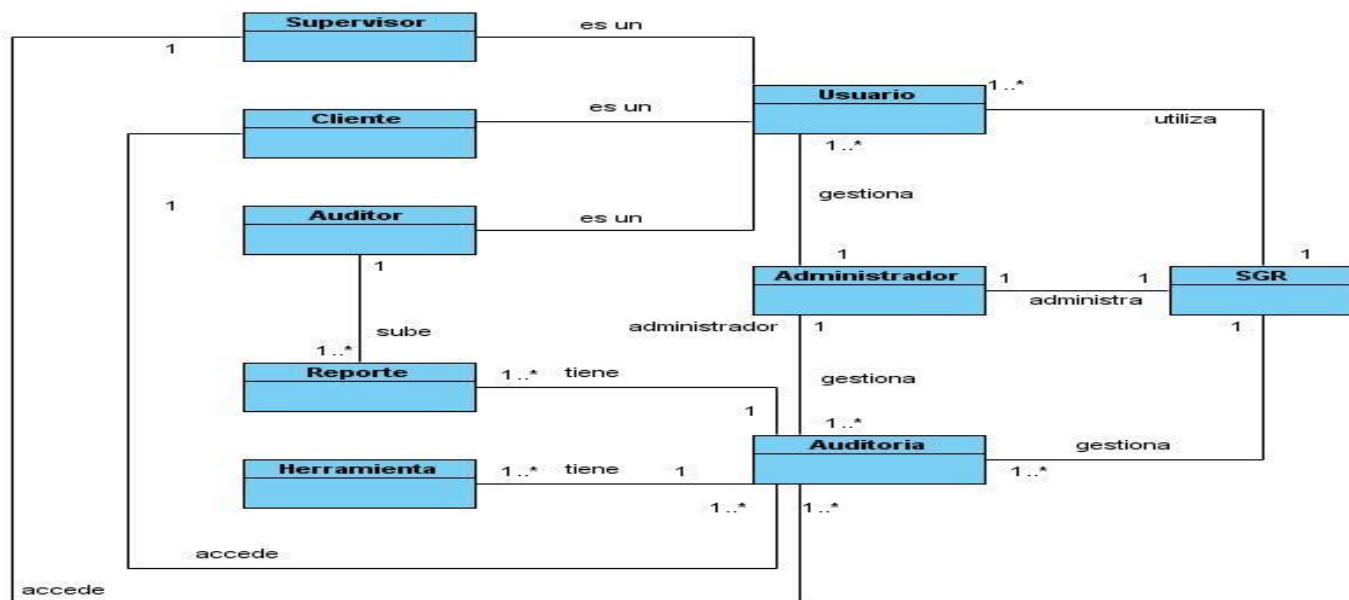


Figura 3: Modelo de Dominio.

A continuación se identifican en un glosario, los conceptos que forman parte del diagrama, para favorecer el entendimiento del modelo de dominio y el negocio a desarrollar.

**SGR:** Servicio de Gestión de Reportes de vulnerabilidades en el proyecto LABSI.

**Auditoría:** Conjunto de pruebas que se le realiza a un sistema informático.

**Reporte:** Resultado de una prueba de seguridad.

**Herramienta:** Software que se usa para hacer una prueba de penetración.

**Usuario:** Rol genérico para agrupar otros roles.

**Administrador:** Encargado de administrar el SGR.

**Supervisor:** Encargado de supervisar las auditorías realizadas en el proyecto LABSI.

**Cliente:** Usuario cuyo sistema ha sido auditado.

**Auditor:** Encargado de dar los resultados de la prueba de seguridad realizada.

### 2.7 Especificación de los requisitos del software.

La IEEE (Institute of Electrical and Electronics Engineers) Standard Glossary of Software Engineering Terminology (1990), define los requerimientos de software como condiciones o capacidades que tienen

## *Capítulo 2: Características del Sistema*

---

que ser alcanzadas o poseídas por un sistema o componente de un sistema para satisfacer un contrato, estándar u otro documento impuesto formalmente.

Los requisitos del software definen qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen. Es una característica que el sistema debe tener para cubrir alguna de las necesidades de los usuarios que lo motivan para resolver un problema o lograr un objetivo.

### **2.7.1 Requisitos Funcionales.**

Los requerimientos funcionales son capacidades o condiciones que un sistema determinado debe cumplir. A continuación se listan las funciones que el sistema debe ser capaz de realizar:

RF1: Gestionar Usuario

RF1.1: Insertar Usuario

RF1.2: Modificar Usuario

RF1.3: Eliminar Usuario

RF1.4: Listar Usuario

RF1.5: Mostrar Usuario

RF2: Gestionar Auditoría

RF2.1: Crear Auditoría

RF2.2: Modificar Auditoría

RF2.3: Eliminar Auditoría

RF2.4: Listar Auditoría

RF2.5: Mostrar Auditoría

RF3: Gestionar Herramienta

RF3.1: Insertar Herramienta

RF3.2: Modificar Herramienta

RF3.3: Eliminar Herramienta

RF3.4: Listar Herramienta

## *Capítulo 2: Características del Sistema*

---

RF3.5: Mostrar Herramienta

RF4: Gestionar Reporte

RF4.1: Crear Reporte

RF4.2: Modificar Reporte

RF4.3: Eliminar Reporte

RF4.4: Listar Reporte

RF4.5: Mostrar Reportes

RF5: Mostrar Estadística

RF6: Autenticar Usuario

RF7: Imprimir Reporte

RF8: Confeccionar Estadística

### **2.7.2 Requisitos no Funcionales**

Los requerimientos no funcionales detallan las propiedades o cualidades que el producto debe tener.

Estos requerimientos están dados en las siguientes categorías:

#### **Usabilidad:**

El sistema brindará la posibilidad de gestionar, dígase, crear, insertar, modificar, eliminar, mostrar y listar, los reportes, herramientas, usuarios y auditorías, que se realizan en el Laboratorio de Seguridad (LABSI). Así como dar datos estadísticos sobre el funcionamiento de LABSI.

#### **Apariencia o interfaz externa:**

Diseño sencillo, por lo que no será necesario mucho entrenamiento para utilizar el sistema.

No debe tener animaciones ni imágenes pesadas que comprometan la rapidez de la aplicación.

El diseño deberá ser atractivo con colores que no resulten agotadores a la vista.

#### **Rendimiento:**

El sistema no presenta un elevado procesamiento de información por lo que la respuesta a las peticiones debe ser de 1 segundo aproximadamente. El tiempo de respuesta del sistema a la hora de subir un



## *Capítulo 2: Características del Sistema*

---

reporte estará determinado por el tamaño del mismo, pero no debe excederse de 60 segundos. Podrán estar conectados simultáneamente 50 usuarios.

### **Confiabilidad:**

Deben ser exactas y seguras las salidas del software. Deben realizarse salvadas de seguridad para no perder la información en caso de algún fallo.

### **Seguridad:**

Garantizar que la información esté disponible únicamente para las personas que estén en el dominio UCI, y se hallan autenticado en el LDAP antes de acceder al sistema. Protección contra acciones no autorizadas o que puedan afectar la integridad de la información presente en el sistema, para esto los usuarios solo podrán conocer los datos a los que tienen permiso, éste otorgado solamente por el administrador del sistema.

### **Soporte:**

El sistema deberá estar bien documentado para cuando exista la necesidad de realizar cambios, mejoras o arreglar algún problema.

### **Legales:**

El producto fue desarrollado con herramientas libres por lo que su uso es autorizado. El sistema es propiedad de la Universidad de las Ciencias Informáticas.

## Capítulo 2: Características del Sistema

### 2.8 Modelo de Casos de Uso del Sistema.

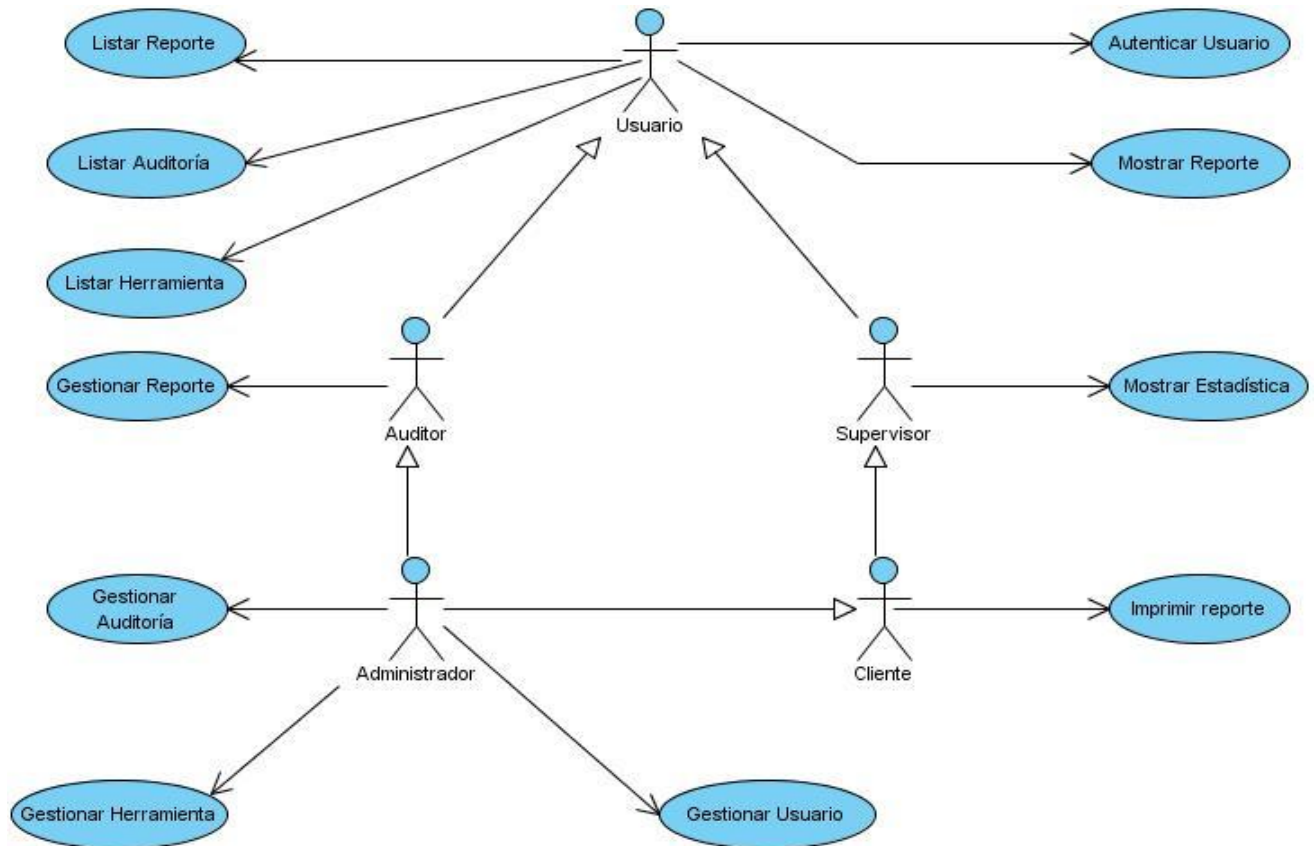


Figura 4: Diagrama de Casos de Uso del Sistema.

#### 2.8.1 Actores del Sistema.

Nombres	Justificación
<b>Administrador</b>	Persona encargada de gestionar los usuarios, auditorías, reportes y herramientas del SGR.
<b>Supervisor</b>	Persona con permisos para revisar todos los datos de todas las auditorías realizadas por el proyecto LABSI.

## Capítulo 2: Características del Sistema

<b>Auditor</b>	Persona con permisos para gestionar los reportes arrojados por las pruebas de seguridad.
<b>Cliente</b>	Persona que tiene permiso para revisar los datos de las auditorías realizadas a su sistema. Así como imprimir los reportes que formen parte de las auditorías.
<b>Usuario</b>	Actor genérico que inicia los Casos de Uso “Autenticar Usuario” y “Mostrar Reporte”, común entre Auditor y Supervisor.

*Tabla 1: Actores del Sistema.*

### 2.8.2 Listado de los Casos de Uso del Sistema.

Caso de Uso 1	Autenticar Usuario
<b>Actor</b>	Usuario
<b>Descripción</b>	Permite establecer los permisos necesarios para restringir el acceso a la información manejada por el sistema, de acuerdo con el rol de cada actor.
<b>Referencia</b>	RF6.

*Tabla 2: Caso de Uso Autenticar Usuario.*

Caso de Uso 2	Mostrar Reporte
<b>Actor</b>	Usuario
<b>Descripción</b>	Permite que los actores según sus permisos, vean los datos de los reportes almacenados en el sistema.
<b>Referencia</b>	RF4.5.

*Tabla 3: Caso de Uso Mostrar Reporte.*

## Capítulo 2: Características del Sistema

Caso de Uso 3	Gestionar Reporte
<b>Actor</b>	Auditor
<b>Descripción</b>	Permite crear, modificar y eliminar los datos relacionados con los reportes de las pruebas realizadas.
<b>Referencia</b>	RF4, RF4.1, RF4.2, RF4.3, RF4.4, RF4.5.

*Tabla 4: Caso de Uso Gestionar Reporte.*

Caso de Uso 4	Imprimir Reporte
<b>Actor</b>	Cliente
<b>Descripción</b>	Permite imprimir los datos relacionados con los reportes almacenados en el sistema.
<b>Referencia</b>	RF7, RF4.5.

*Tabla 5: Caso de Uso Imprimir Reporte.*

Caso de Uso 5	Listar Auditoría
<b>Actor</b>	Supervisor
<b>Descripción</b>	Permite obtener un listado de las auditorías almacenadas en el sistema.
<b>Referencia</b>	RF2.4, RF2.5.

*Tabla 6: Caso de Uso Listar Auditoría.*

Caso de Uso 6	Listar Reporte
<b>Actor</b>	Usuario
<b>Descripción</b>	Permite obtener un listado de los reportes almacenados en el sistema
<b>Referencia</b>	RF4.4.

*Tabla 7: Caso de Uso Listar Reporte.*

## Capítulo 2: Características del Sistema

Caso de Uso 7	Listar Herramientas
<b>Actor</b>	Supervisor
<b>Descripción</b>	Permite obtener un listado de las herramientas almacenadas en el sistema.
<b>Referencia</b>	RF3.4, RF3.5.

*Tabla 8: Caso de Uso Listar Herramientas.*

Caso de Uso 8	Mostrar Estadísticas
<b>Actor</b>	Supervisor
<b>Descripción</b>	Permite obtener datos estadísticos relacionados con los datos de las auditorías realizadas por el proyecto LABSI.
<b>Referencia</b>	RF5, RF8.

*Tabla 9: Caso de Uso Mostrar Estadísticas.*

Caso de Uso 9	Gestionar Auditoría
<b>Actor</b>	Administrador
<b>Descripción</b>	Permite crear, modificar y eliminar las auditorías que son manejadas por el sistema.
<b>Referencia</b>	RF2, RF2.1, RF2.2, RF2.3, RF2.4, RF2.5.

*Tabla 10: Caso de Uso Gestionar Auditoría.*

Caso de Uso 10	Gestionar Herramienta
<b>Actor</b>	Administrador
<b>Descripción</b>	Permite insertar, modificar y eliminar los datos de las herramientas que se usan en las auditorías manejadas por el sistema.
<b>Referencia</b>	RF3, RF3.1, RF3.2, RF3.3, RF3.4, RF3.5

*Tabla 11: Caso de Uso Gestionar Herramienta.*

## Capítulo 2: Características del Sistema

---

Caso de Uso 11	Gestionar Usuario
<b>Actor</b>	Administrador
<b>Descripción</b>	Permite insertar, modificar y eliminar los usuarios que interactúan con los datos manejados por el sistema.
<b>Referencia</b>	RF1, RF1.1, RF1.2, RF1.3, RF1.4, RF1.5.

Tabla 12: Caso de Uso Gestionar Usuario.

### 2.8.3 Descripción textual de los Casos de Uso del Sistema

Ver anexo 1.

### 2.8.4 Prototipos de Interfaz de Usuario

Ver anexo 3.

## 2.9 Conclusión del Capítulo

En el presente capítulo se hace una propuesta para el desarrollo del sistema. Se expusieron las características que debe cumplir el sistema. Se realizó una descripción de los requisitos funcionales y no funcionales del sistema. También se presenta el modelo de dominio y el diagrama de casos de uso del sistema. Además se presentaron los actores del sistema y la función que cumplirán dentro del mismo. Se listan todos los casos de uso y se describen textualmente cada uno de ellos.

A partir de este punto se puede continuar con las siguientes fases en el desarrollo del sistema atendiendo a que se cumplan los requerimientos funcionales y no funcionales del sistema que se propone implementar.

# Capítulo 3: Análisis y Diseño del sistema

## CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

### 3.1 Introducción.

En este capítulo se introducen los conceptos de Análisis y Diseño. Al realizar el análisis se garantiza un buen diseño y posteriormente una implementación correcta. Se mostrarán los diagramas de clases del análisis, también los diagramas de clases del diseño con estereotipos Web. Se realizan los diagramas de secuencia del diseño, que ayudarán a la implementación del sistema. También se da el modelo lógico (diagrama de clases persistentes) y el modelo físico de los datos.

### 3.2 Análisis.

“Durante el análisis, se analizan los requisitos que se describen en la captura de requerimientos, refinándolos y estructurándolos. El objetivo de hacerlo es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que ayude a estructurar el sistema entero, incluyendo su arquitectura” (7).

El artefacto más importante que genera el análisis es el Modelo de Análisis, éste contiene las clases del análisis, éstas se centran en los requisitos funcionales. Las clases del análisis se pueden clasificar en:

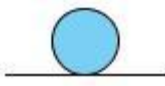
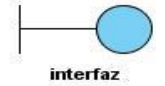

Nombre	Características	Representación
Entidad	Modelan información que posee larga vida y que es a menudo persistente.	 entidad
Interfaz	Modelan la interacción del sistema y sus actores.	 interfaz
Control	Coordinan la relación entre uno u otros casos de uso, coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso.	 control

Tabla 13: Clases del Análisis.

## Capítulo 3: Análisis y Diseño del sistema

### 3.2.1 Diagrama de clases del análisis

Es un artefacto del análisis, que representa las clases del análisis y la relación entre ellas. Este diagrama da una vista estática del sistema. Representa el mundo real, no la implementación.

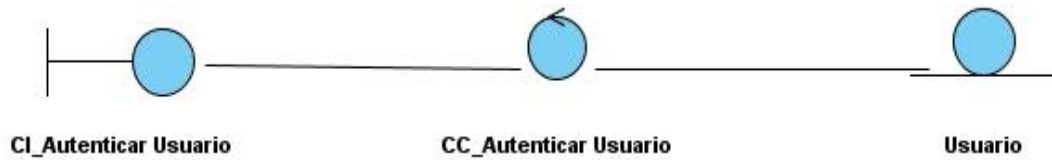


Figura 5: Diagrama de Clases del Análisis-CUS Autenticar Usuario.

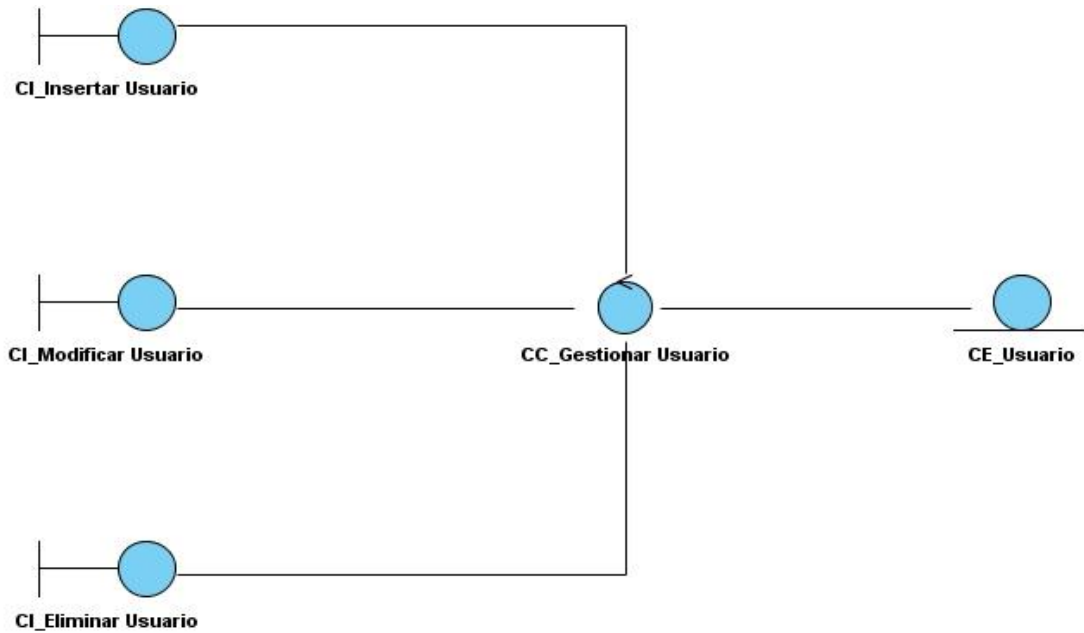


Figura 6: Diagrama de Clases del Análisis-CUS Gestionar Usuario.



## Capítulo 3: Análisis y Diseño del sistema

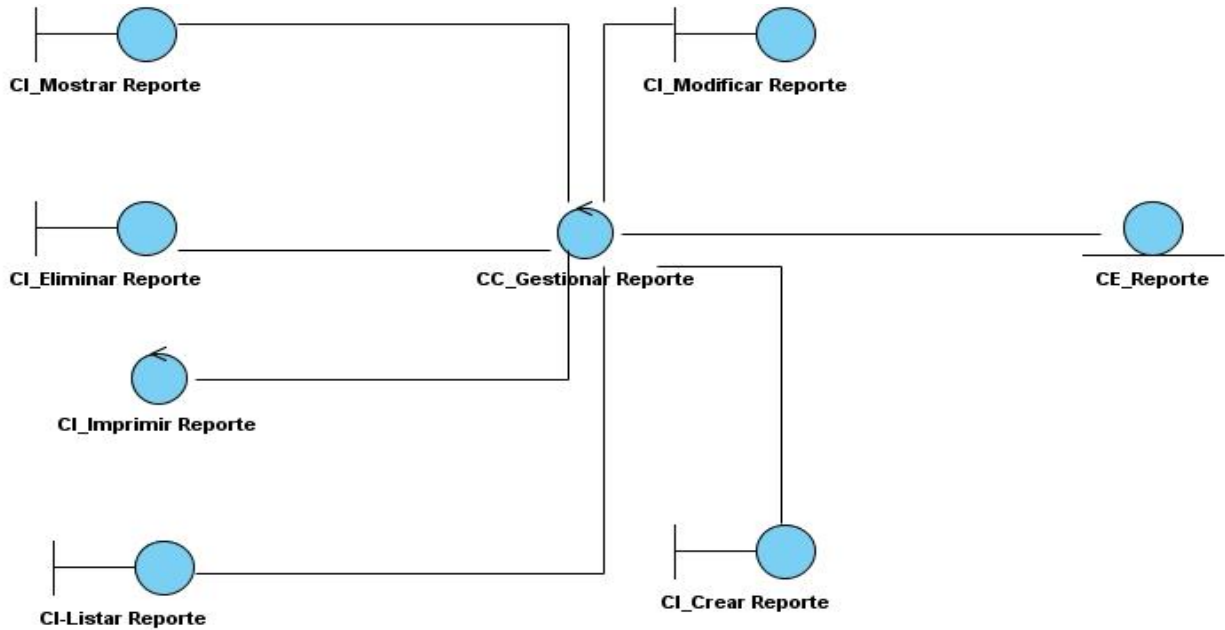


Figura 7: Diagrama de Clases del Análisis-CUS Gestionar Reporte.

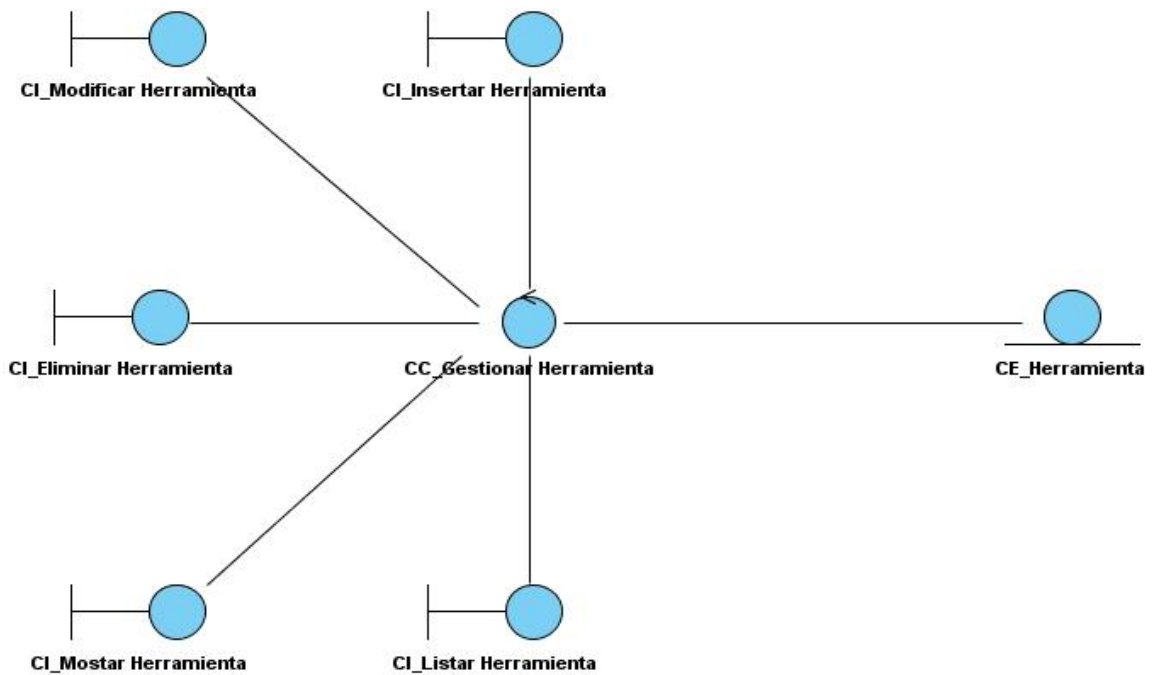


Figura 8: Diagrama de Clases del Análisis-CUS Gestionar Herramientas.

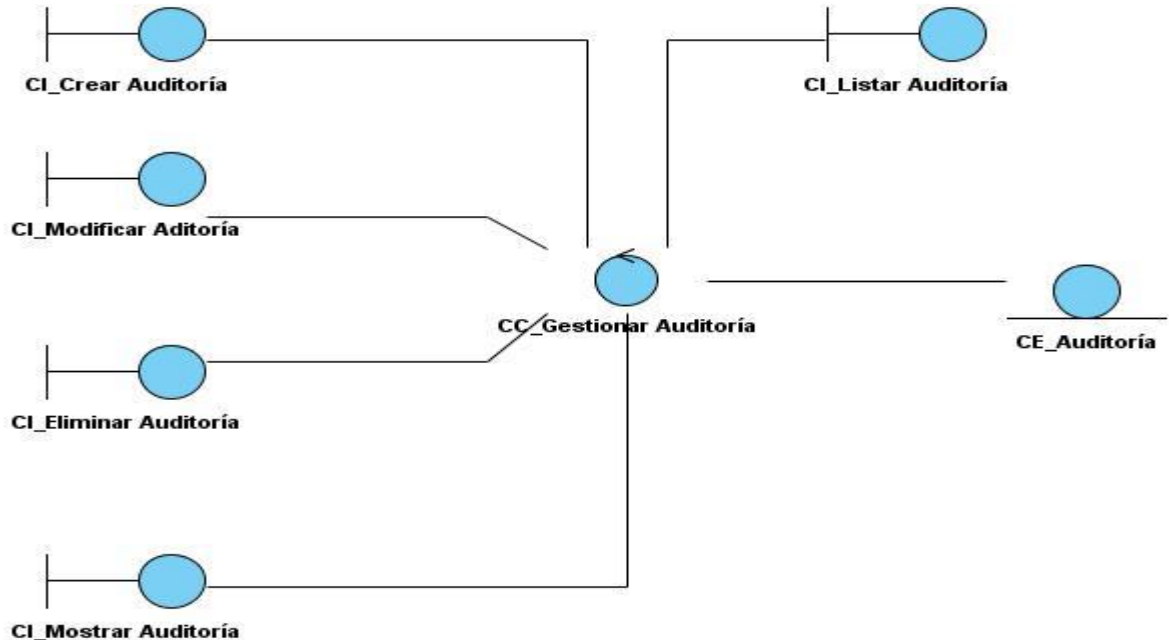


Figura 9: Diagrama de Clases del Análisis-CUS Gestionar Auditoría.

### 3.3 Diseño.

El propósito del diseño es adquirir una comprensión profunda de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia, tecnología de interfaz de usuario, tecnología de gestión de transacciones (7).

El artefacto más importante es el Modelo de Diseño, el cual es un modelo de objetos que describe la realización física de los casos de uso, centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen un impacto en el sistema a considerar (7).

Para el diseño de la aplicación se tendrán en cuenta el patrón de arquitectura MCV (Modelo-Vista-Controlador), el framework KumbiaPHP y como lenguaje de programación PHP.

### 3.3.1 Patrones de Diseño.

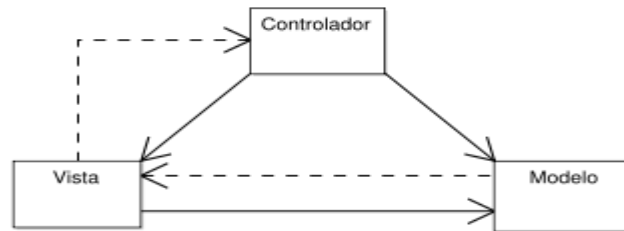
Los patrones son una solución a un problema en un determinado contexto o situación; éste codifica conocimiento específico en un dominio. Se puede asegurar que un sistema para que quede bien estructurado debe estar lleno de patrones. Cada patrón hace una descripción de la solución de cada problema, y de esa forma la solución se puede utilizar una y otra vez, sin repetir las mismas acciones.

Para el desarrollo de la aplicación se tiene en cuenta el Modelo-Vista-Controlador (MCV), que es el patrón arquitectónico que implementa el framework KumbiaPHP. MVC es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones Web, donde la vista es la página HTML y el código, que provee de datos dinámicos a la página. El modelo es el sistema de gestión de base de datos y la lógica del negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista (6).

Para una mayor comprensión se brinda una descripción a continuación.

- ✓ **Modelo:** Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de éstos y permite derivar nuevos datos; por ejemplo, no admitiendo comprar un número negativo de unidades, calculando si hoy es el cumpleaños del usuario y/o los totales, impuestos o importes en un carrito de compras.
- ✓ **Vista:** Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- ✓ **Controlador:** Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

A continuación se brinda una ilustración que muestra las relaciones entre el modelo, la vista y el controlador:



**Figura 10: Relación entre el Modelo, la Vista y el Controlador**

Los patrones de diseño brindan esquemas para definir las diferentes estructuras del diseño, con las que se construye el sistema de software. Para el diseño del sistema se aplicaron los patrones de asignación de responsabilidades GRASP y los patrones GOF.

### **Patrones GRASP:**

Patrones de Software para la Asignación General de Responsabilidad (GRASP), éstos describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. A continuación se explican el uso de cada uno en el sistema.

**Creador:** Es el responsable de la creación de una nueva instancia de alguna clase. En KumbiaPHP existe un único script PHP, que se encarga de instanciar al controlador frontal (Dispatcher), este último es el encargado de instanciar las clases controladoras y éstas, a su vez, instancian objetos del modelo que extienden de la clase ActiveRecord. El uso de este patrón permite crear las dependencias mínimas necesarias entre las clases, lo que favorece al mantenimiento del sistema.

**Experto:** El uso de este patrón posibilita la asignación de responsabilidad al experto en información, la clase que tiene la información necesaria para la realización de la asignación. KumbiaPHP, permite que las clases controladoras manejen las peticiones del cliente y las clases del modelo son las encargadas del acceso a datos, esto proporciona que se conserve el encapsulamiento y además se dé soporte a un bajo acoplamiento y una alta cohesión.

## Capítulo 3: Análisis y Diseño del sistema

---

**Bajo acoplamiento:** Permite soportar bajas dependencias, un bajo impacto del cambio e incremento de la reutilización. Este patrón se tiene en cuenta por la importancia de realizar un diseño de clases independiente que puedan soportar los cambios de una manera fácil y permitan la reutilización. El uso de los patrones Experto y Creador favorecen al bajo acoplamiento entre las clases del sistema.

**Controlador:** Es el responsable de gestionar un evento de entrada al sistema. KumbiaPHP colabora en la utilización de este patrón, ya que este define un Controlador Frontal (Dispatcher) esto conlleva a que todas las solicitudes son dirigidas a un único script PHP que se encarga de instanciar al controlador frontal y redirigir las llamadas.

**Alta cohesión:** Es el responsable de mantener la complejidad manejable, este patrón se caracteriza por asignar una responsabilidad de manera que la cohesión permanezca alta.

### Patrones GOF:

Existen 3 tipos de patrones GOF:

- ✓ De Creación: abstraen el proceso de creación de instancias.
- ✓ Estructurales: se ocupan de cómo clases y objetos son utilizados para componer estructuras de mayor tamaño.
- ✓ De Comportamiento: atañen a los algoritmos y a la asignación de responsabilidades entre objetos.

Los patrones GOF usado para el diseño del sistema son:

**Virtual Proxy:** Este patrón se utiliza en objetos de la clase Auth, a no ser que sea solicitado por el usuario o se cumplan determinadas condiciones, no es necesario instanciarlos.

**Abstract Factory:** Se utiliza cuando es necesario crear diferentes objetos, todos pertenecientes a la misma familia. Este permite configurar en tiempo de ejecución un sistema con una familia u otra de objetos. También proporciona que un conjunto de clases se usen a la vez.

**Singleton:** Este patrón asegura que exista una única instancia de una clase.

## 3.3.2 Diagramas de clase del diseño Web.

Una clase del diseño es una abstracción de una clase; y hay que tener algunas consideraciones con dichas clases, por ejemplo que deben ser especificadas en el mismo lenguaje en el que se va a implementar.

En los diagramas de clases del diseño se muestran las clases del diseño, los subsistemas y sus relaciones. Este permite coordinar todos los requisitos para la realización de los diferentes casos de uso.

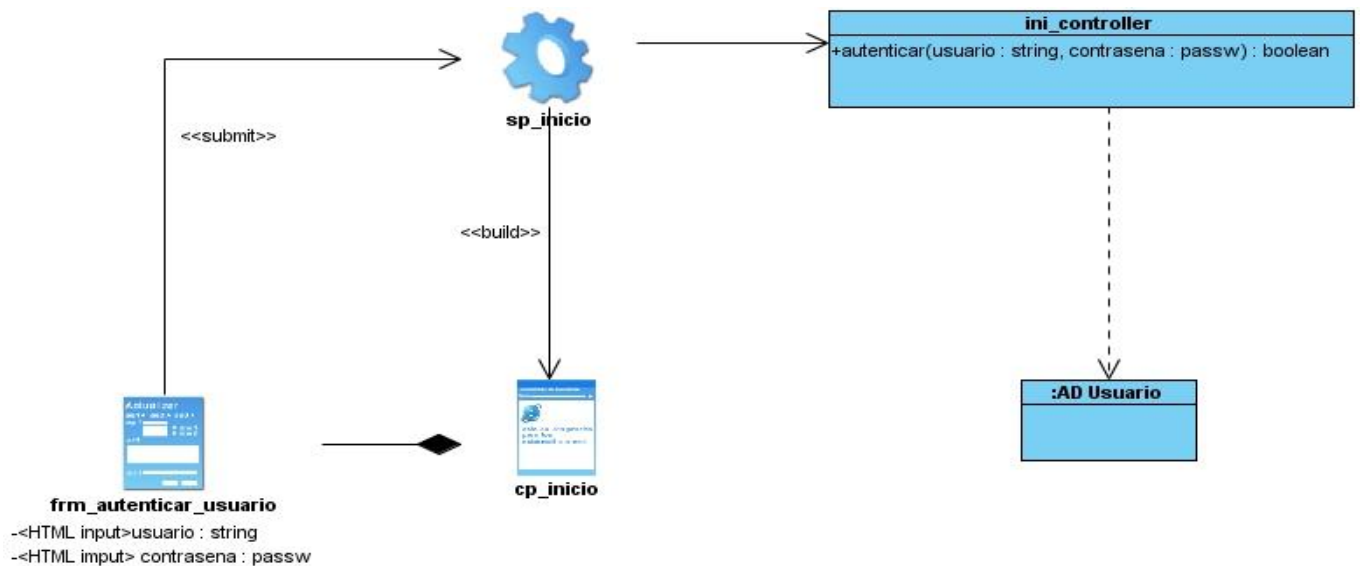


Figura 11: Diagrama de clases del diseño Caso de Uso Autenticar Usuario.

# Capítulo 3: Análisis y Diseño del sistema

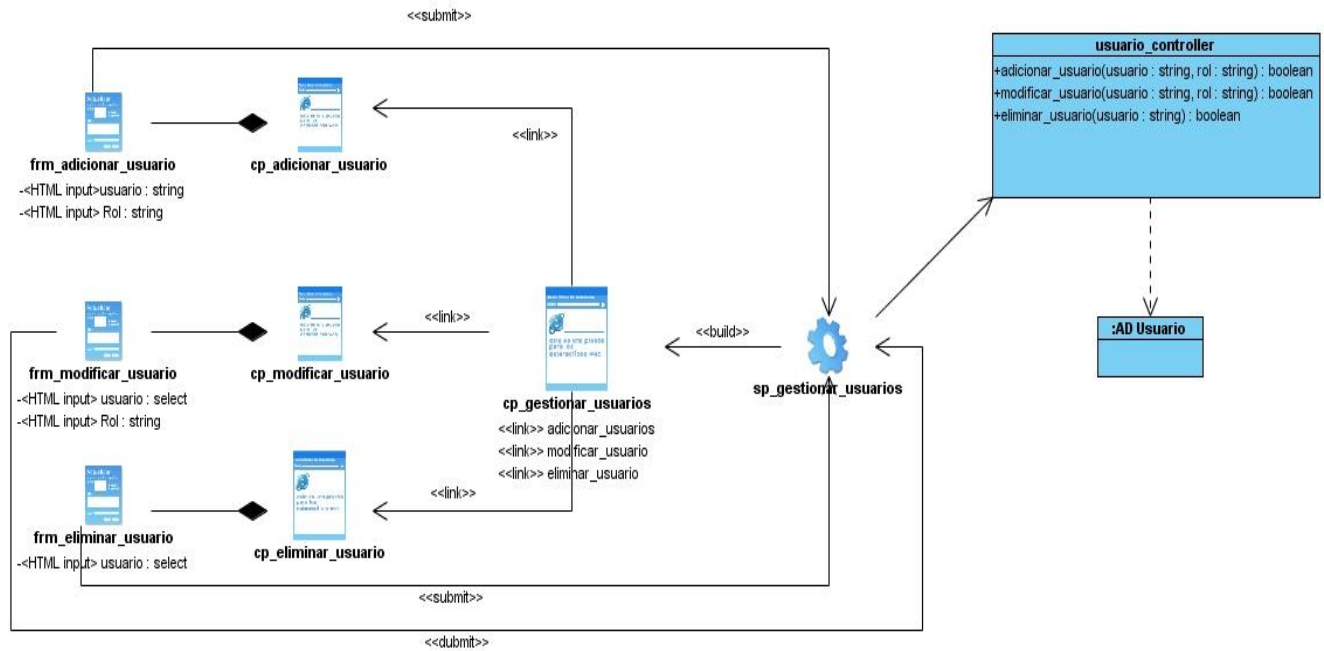


Figura 12: Diagrama de clases del diseño Caso de Uso Gestionar Usuario.

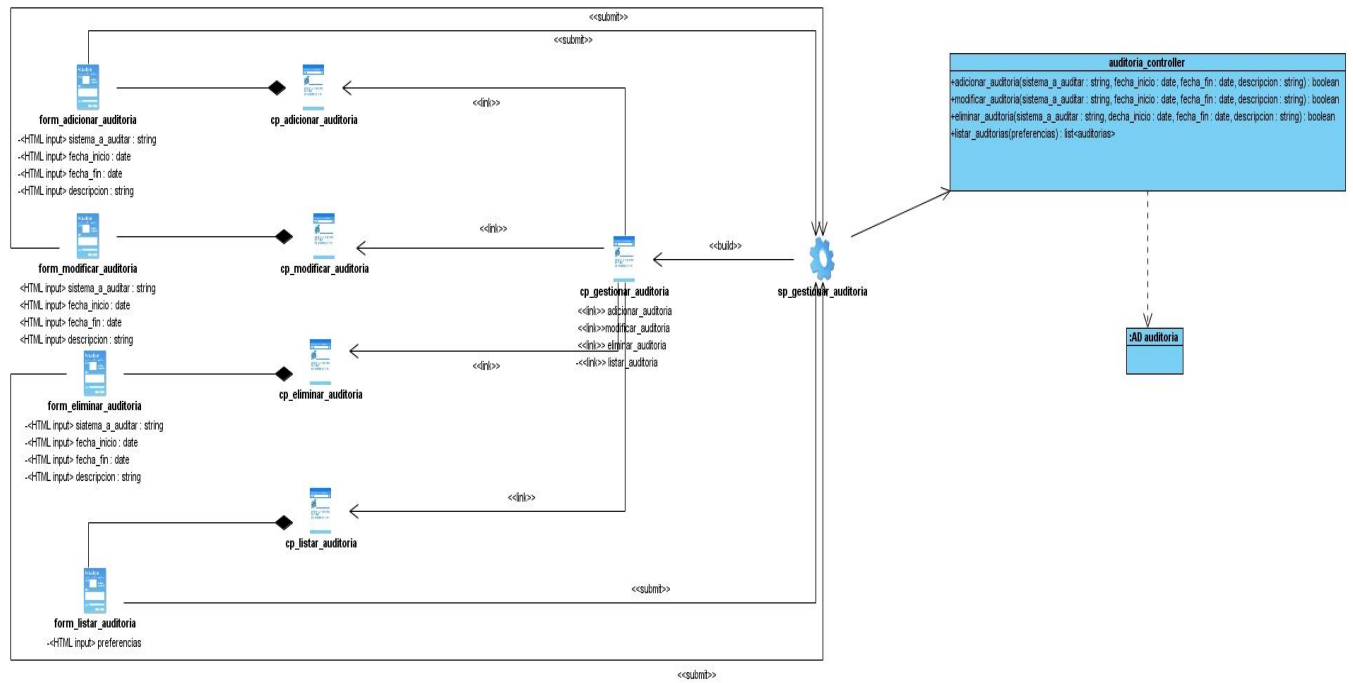


Figura 13: Diagrama de clases del diseño Caso de Uso Gestionar Auditoría.

# Capítulo 3: Análisis y Diseño del sistema

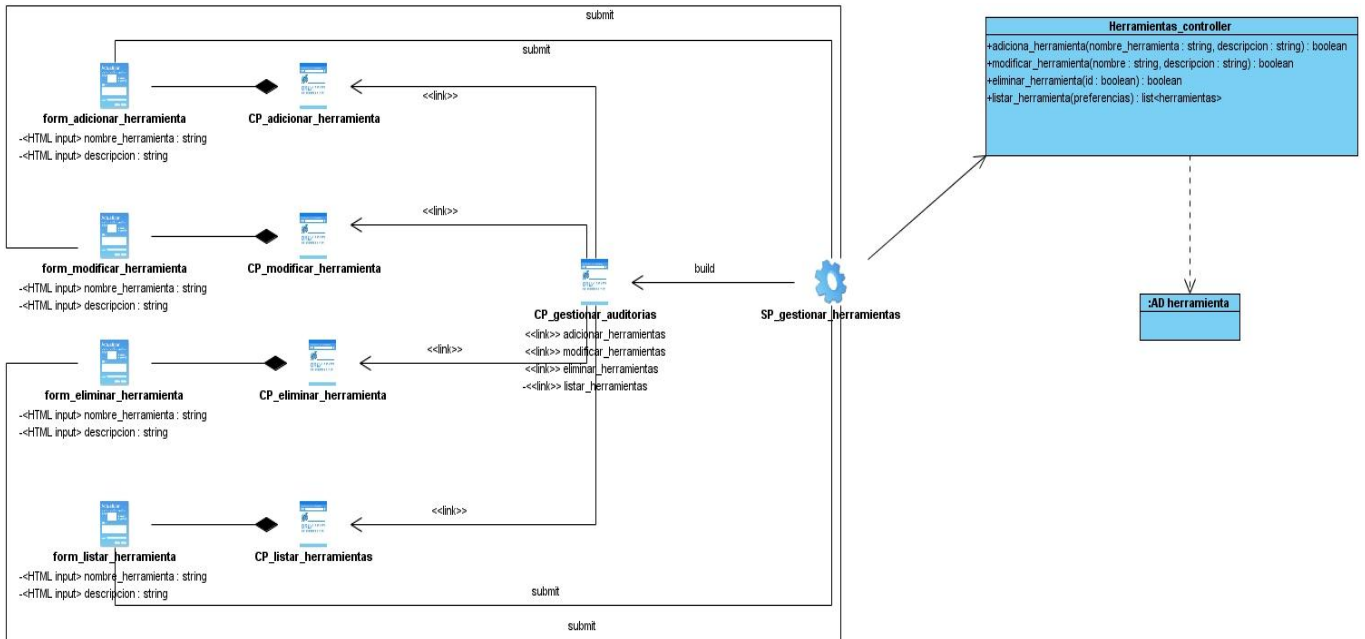


Figura 14: Diagrama de clases del diseño Caso de Uso Gestionar Herramienta.

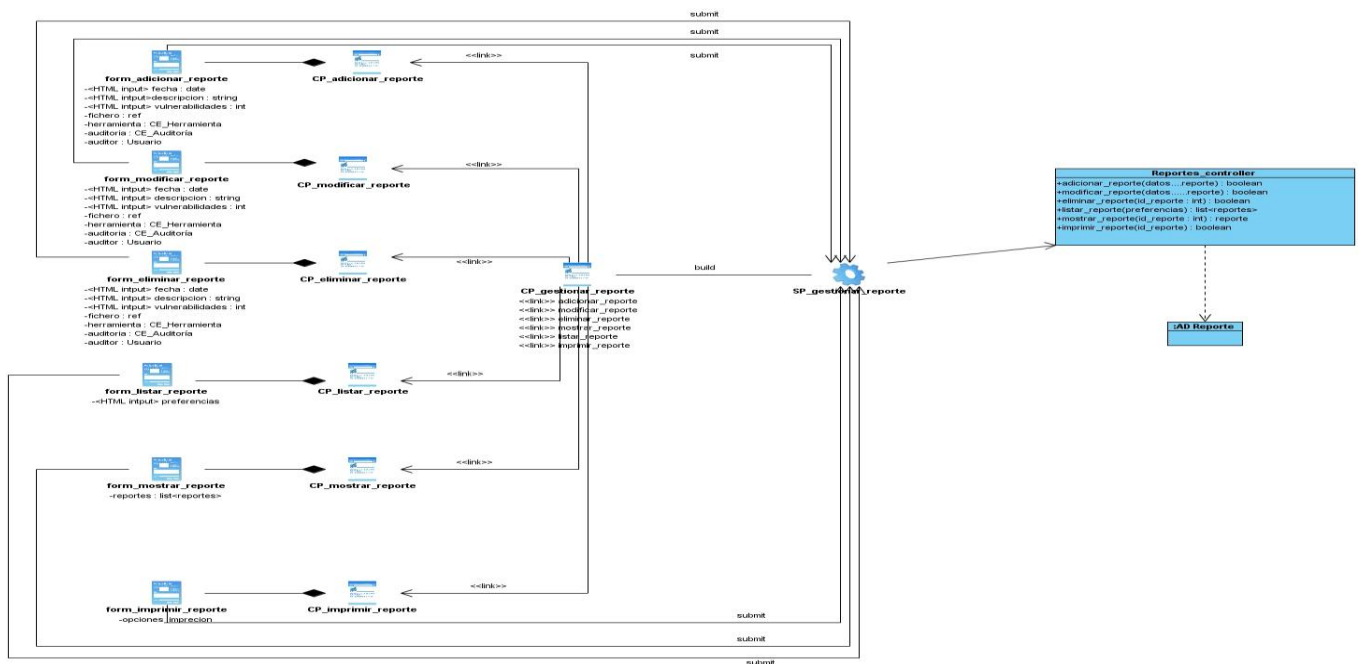


Figura 15: Diagrama de clases del diseño Caso de Uso Gestionar Reporte.



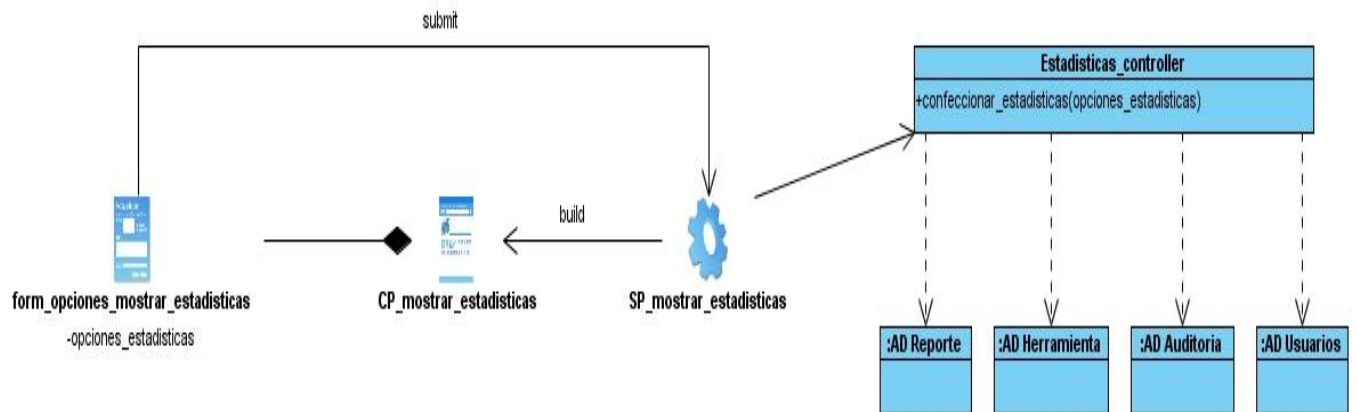


Figura 16: Diagrama de clases del diseño Caso de Uso Mostrar Estadística.

### 3.3.3 Diagramas de secuencia del diseño.

Ver anexo 2.

### 3.4 Modelo Lógico de Datos.

Se trata de una representación gráfica, mediante símbolos y signos normalizados, de la base de datos. Su objetivo es representar la estructura de los datos y las dependencias de los mismos, garantizando la consistencia y evitando la duplicidad (8).

El objetivo del diseño lógico es convertir los esquemas conceptuales locales en un esquema lógico global que se ajuste al modelo de SGBD sobre el que se vaya a implementar el sistema.

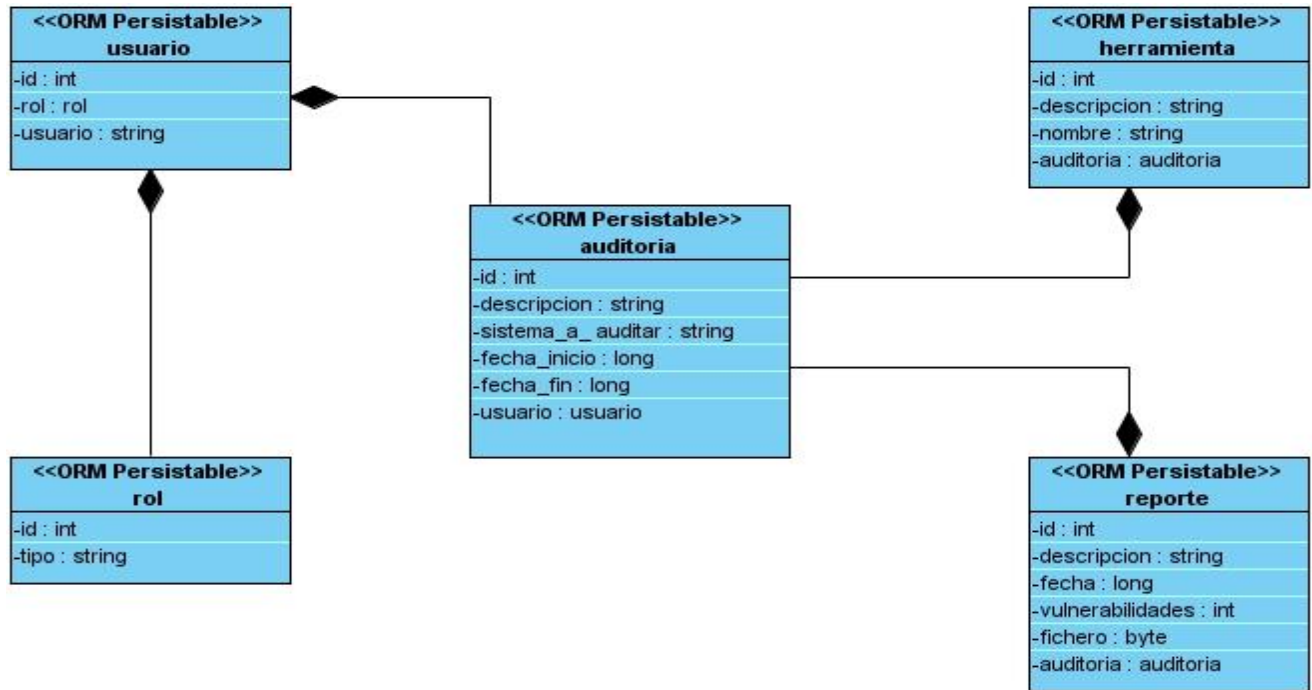


Figura 17: Modelo Lógico de Datos.

### 3.5 Modelo Físico de Datos.

Describen cómo se almacenan los datos en el ordenador: el formato de los registros, la estructura de los ficheros y los métodos de acceso utilizados (9).

## Capítulo 3: Análisis y Diseño del sistema

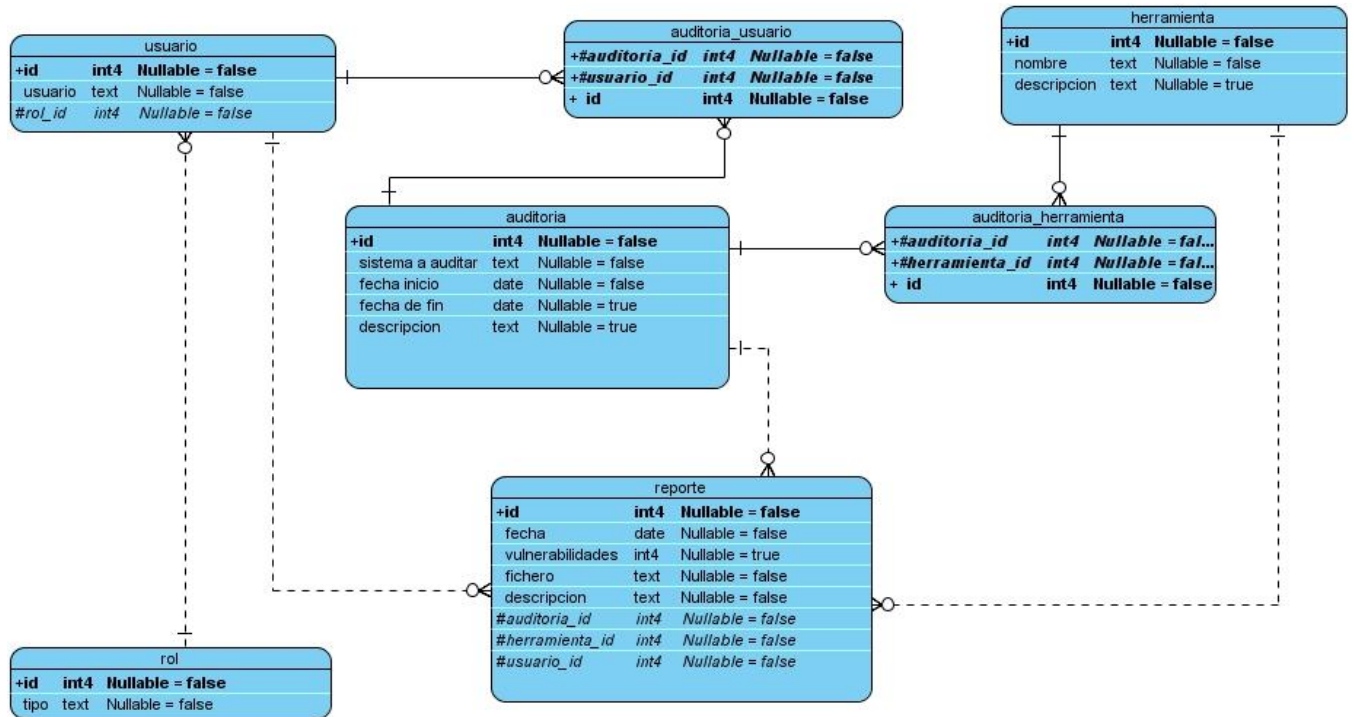


Figura 18: Modelo Físico de Datos.

### 3.6 Conclusiones del Capítulo.

En este capítulo se dio la introducción al análisis y el diseño de la aplicación. Se brindaron los diagramas de clases del análisis, los de clases del diseño con estereotipos Web y los de secuencia del diseño. Se describe la arquitectura Modelo-Vista-Controlador (MCV) que es la utilizada en el desarrollo del sistema así como los patrones de diseño que ayudan a que la fase posterior quede bien realizada. También se da el diseño de la base de datos mediante el modelo físico con todas las entidades que se manejan en el sistema y el modelo lógico de los datos que es conocido también como diagrama de clases persistentes.

# Capítulo 4: Implementación del Sistema

## CAPÍTULO 4: IMPLEMENTACIÓN DEL SISTEMA.

### 4.1 Introducción.

Teniendo los resultados del análisis y del diseño, la implementación se simplifica a implementar el sistema en término de componentes, es decir, ejecutables, código fuente y scripts. Aquí se desarrolla la arquitectura y el sistema como un todo. Además se implementan las clases del diseño como componentes que contienen código fuente.

### 4.2 Diagrama de Despliegue.

Se puede decir que un diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación, donde un nodo puede contener instancias de componentes. En general, un nodo se entiende como una unidad de computación de algún tipo como es el caso de impresoras o la misma computadora (7).

En un diagrama de despliegue se muestran las relaciones físicas entre los componentes de hardware y de software, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes de software, que no son más, que los procesos que se ejecutan en ellos.

A continuación se muestra el diagrama de despliegue:

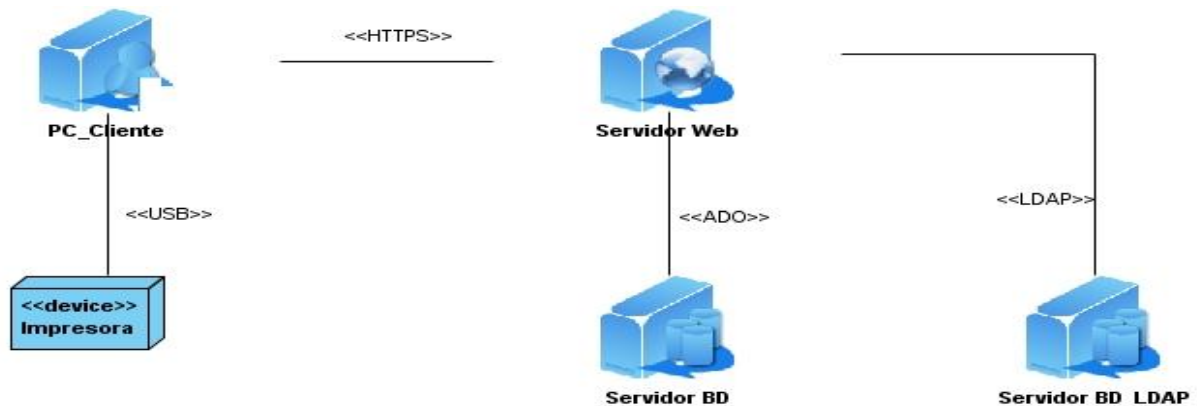


Figura 19: Diagrama de Despliegue.

# Capítulo 4: Implementación del Sistema

## 4.3 Diagrama de Componentes.

Un componente es la parte modular de un sistema, desplegable y reemplazable que encapsula implementación y un conjunto de interfaces y proporciona la realización de los mismos. Un componente típicamente contiene clases y puede ser implementado por uno o más artefactos.

Los diagramas de componentes estructuran el modelo de implementación en términos de subsistemas de implementación y muestra las relaciones entre los elementos de implementación. Estos diagramas muestran la estructura de alto nivel del modelo de implementación.

Un diagrama de componentes representa las dependencias entre componentes de software, incluyendo componentes de código fuente, de código binario y ejecutable. Algunos componentes existen en tiempo de compilación, algunos en tiempo de enlace y algunos en tiempo de ejecución, otros en varias de éstas (7).

A continuación se muestran los diagramas de componentes:

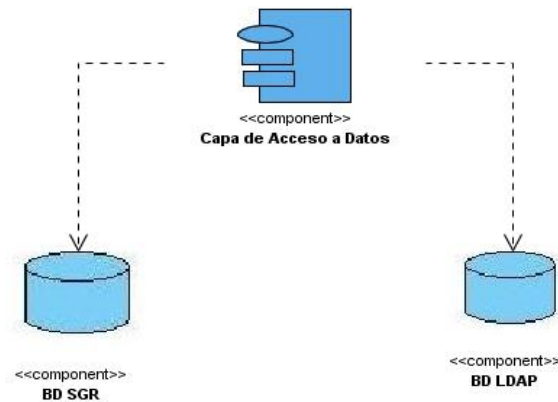


Figura 20: Diagrama de Componentes Base de Datos.

# Capítulo 4: Implementación del Sistema

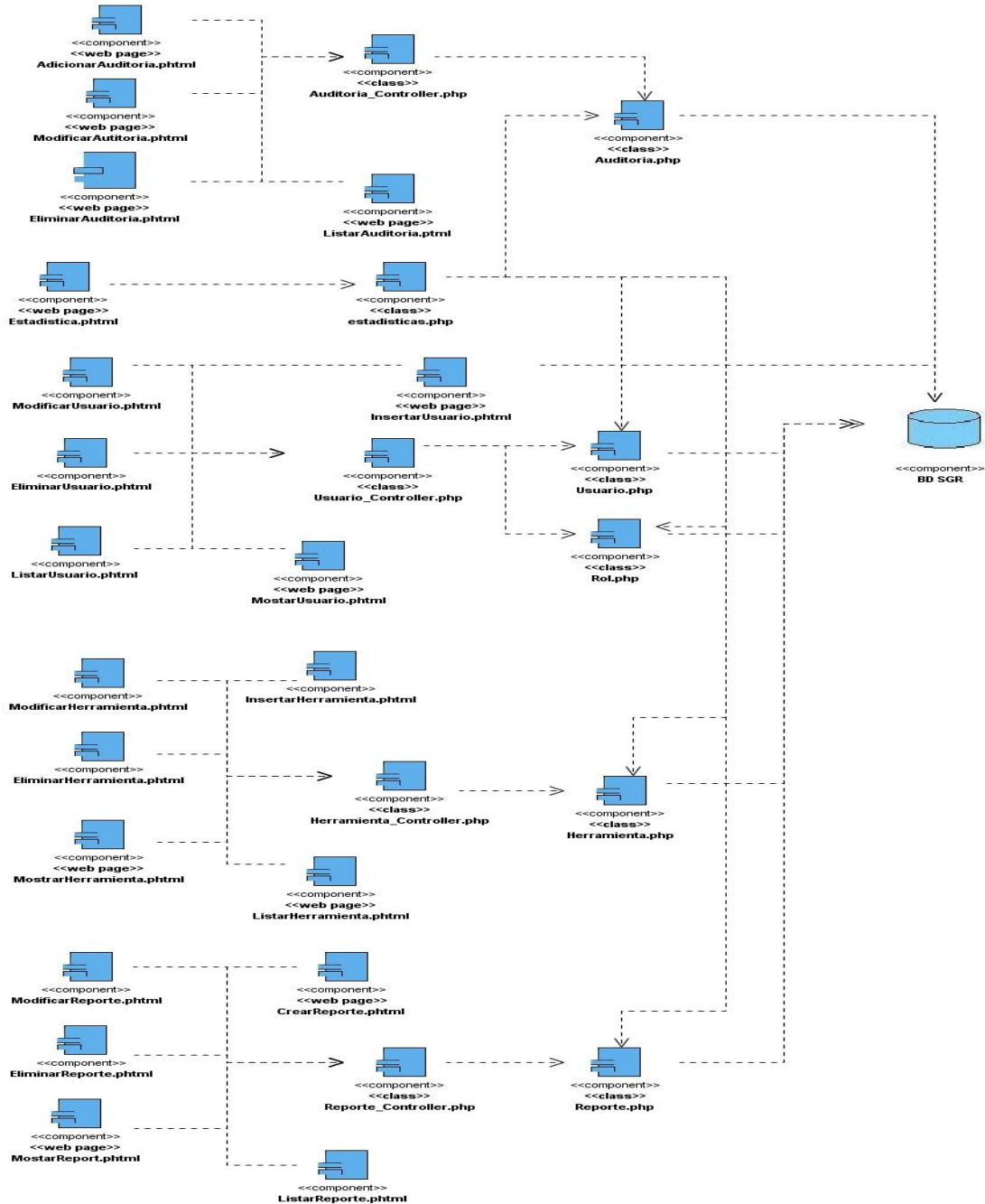


Figura 21: Diagrama de Componentes Vistas, Controlador, Modelo.

# Capítulo 4: Implementación del Sistema

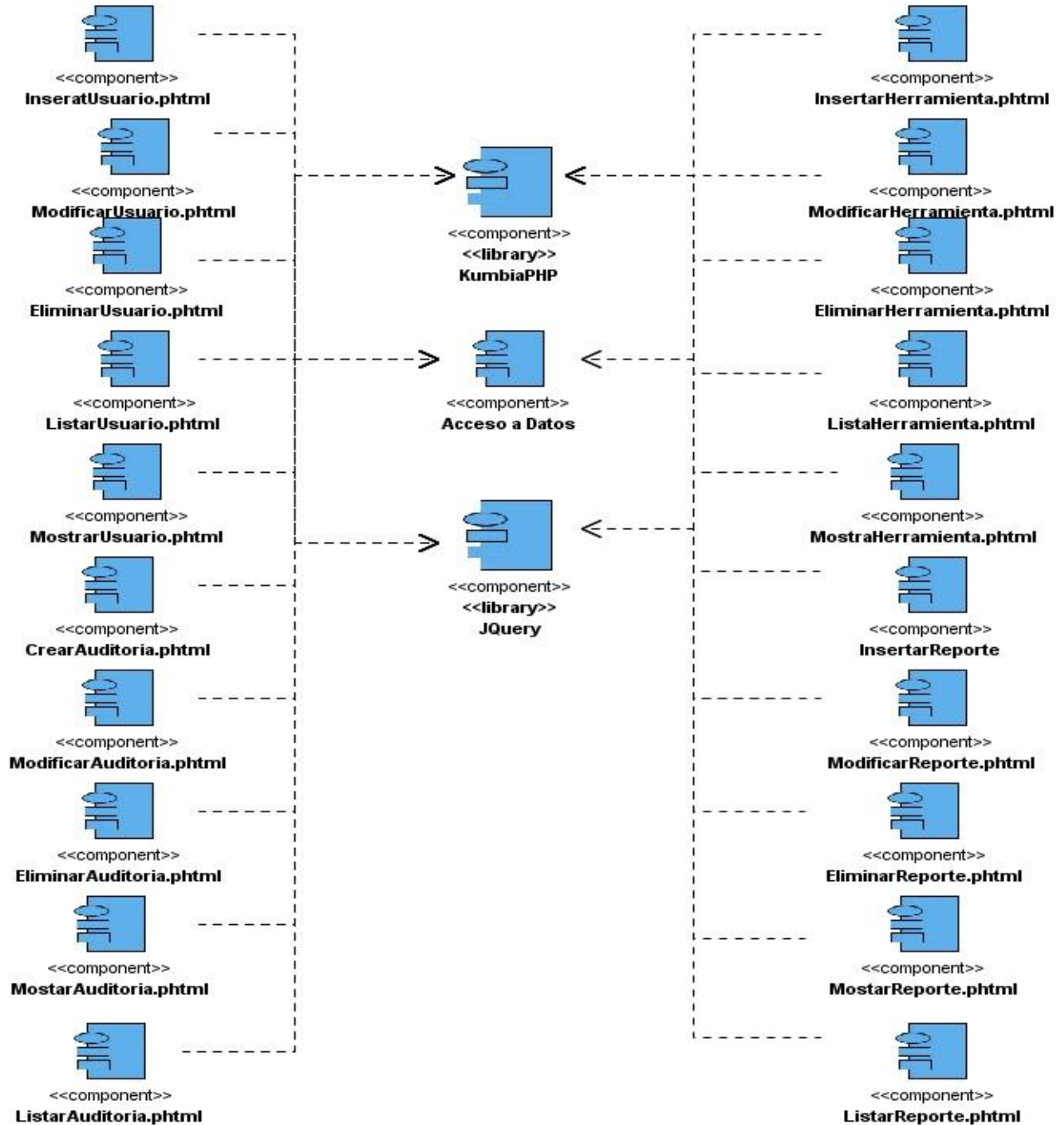


Figura 22: Diagrama de Componentes Código Fuente.

## *Capítulo 4: Implementación del Sistema*

---

### **4.4 Conclusiones del Capítulo.**

En el presente capítulo se abordó el tema de la implementación del sistema. Se muestra como está construido el sistema. Se brinda el diagrama de despliegue que muestra los nodos que se utilizan para la implantación de la aplicación. También los diagramas de componentes como estructura fundamental de la implementación en término de subsistemas de implementación. Esta fase representa la forma en que está construido el sistema para mayor comprensión del mismo.



## CAPÍTULO 5: ESTUDIO DE FACTIBILIDAD

### 5.1 Introducción.

Con el objetivo de determinar el tiempo que será necesario consumir y el costo para la realización del sistema, en el presente capítulo, se realiza un estudio sobre la factibilidad del proyecto. El proceso de desarrollo de un software suele ser altamente costoso en cuanto a dinero, equipos y tiempo de trabajo, de ahí la necesidad de un análisis que permita hacer una estimación sobre la factibilidad del mismo.

Para el cálculo de la complejidad de un proyecto de software existen varios métodos, por ejemplo, el método de Puntos de Función, método COCOMO, Por Puntos de Caso de Uso y Model Driven Architecture. Para el caso de este proyecto en el cual se utiliza RUP como metodología de desarrollo de software el método más adecuado es el de Por Puntos de Casos de Uso, ya que esta requiere de una descripción abundante de cada caso de uso, lo que facilita la ejecución del método y minimiza el margen de error del mismo.

### 5.2 Método de estimación Por Puntos de Casos de Uso.

Este método usa los casos de uso y actores identificados para calcular de forma aproximada el costo y el esfuerzo que será necesario, para el desarrollo del sistema, está basado en asignarle un peso a los casos de uso según su complejidad y a los actores según el tipo de actor que sea, también se utilizan factores de entorno y de complejidad, técnica para afinar el resultado. Lo primero es calcular los puntos por casos de uso no ajustados o UUCP, el factor de complejidad técnica o TCF y el factor de entorno o EF, una vez calculados estos datos se puede proceder a calcular los puntos de casos de uso o UCP, los cuales finalmente se traducen a esfuerzo en horas-hombre requeridos para el desarrollo de la aplicación.

#### 5.2.1 Cálculo de Puntos de Caso de Uso sin ajustar.

Para obtener puntos de casos de uso sin ajustar se suman el factor de peso de los actores sin ajustar o UAW y el factor de peso de los casos de uso sin ajustar o UUCW.

$$UUCP = UAW + UUCW$$

## Capítulo 5: Estudio de Factibilidad

Significado de estas siglas:

UUCP: puntos de caso de uso sin ajustar.

UAW: factor de peso de los actores sin ajustar.

UUCW: factor de peso de los caso de uso sin ajustar.

Una vez calculado tanto el UAW como el UUCW, el resultado sería:

$$UUCP = UAW + UUCW$$

$$UUCP = 12 + 80$$

$$UUCP = 92$$

### 5.2.2 Factor de Peso de los Actores sin ajustar.

Para calcular en UAW se hace una evaluación de la complejidad de los actores con los que tendrá que interactuar el sistema y se calcula como la sumatoria de la multiplicación la cantidad de actores de cada tipo, por un factor de peso asociado a ellos.

$$UAW = \text{Sum}(\text{cantidadDeUnTipoDeActor} * \text{Factor})$$

Tipo	Descripción	Peso	Cantidad	Cant * Peso
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación API: Application Programming Interface.	1	0	0 * 1
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o interfaz basado en texto.	2	0	0 * 2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica.	3	4	4 * 3
<b>Total:</b>				12

Tabla 14: Cálculo del factor de peso de los actores sin ajustar.

## *Capítulo 5: Estudio de Factibilidad*

### 5.2.3 Factor de Peso de los Casos de Uso sin ajustar.

Para calcular el Factor de peso de los casos de uso sin ajustar existen dos métodos, basado en transacciones y en clases del análisis. El primer método toma en cuenta el número de transacciones que se realizan en un caso de uso y lo evalúa teniendo en cuenta que una transacción es un conjunto de actividades atómicas, lo que quiere decir que se ejecutan todas o no se ejecuta ninguna, el segundo método toma en cuenta el número de clases que tiene un caso de uso y lo evalúa. Independientemente del método que se escoja para clasificar a un caso de uso la fórmula para obtener el UUCW es:

$$\text{UUCW} = \text{Sum}(\text{cantidadDeUnTipoDeCasoUso} * \text{Factor})$$

Tipo de Caso de Uso	Descripción	Factor
Simple	3 transacciones o menos.	5
Medio	4 a 7 transacciones.	10
Complejo	Más de 7 transacciones	15

*Tabla 15: Peso de las Transacciones.*

No.	Nombre de Caso de Uso	Cantidad de Transacciones	Tipo	Peso	Cant * Peso
1	Gestionar Usuarios	4	Medio	10	6*5
2	Autenticar Usuario	3	Simple	5	
3	Gestionar Herramienta	4	Medio	10	
4	Gestionar Auditoría	4	Medio	10	
5	Gestionar Reporte	4	Medio	10	5*10
6	Imprimir Reporte	3	Simple	5	
7	Mostrar Reporte	3	Simple	5	
8	Listar Reporte	3	Simple	5	
9	Listar Auditoría	3	Simple	5	0*15
10	Listar Herramienta	3	Simple	5	
11	Mostrar Estadísticas	4	Medio	10	

<b>TOTAL:</b>	80
---------------	----

### 5.2.4 Puntos de Casos de Uso ajustados

Para calcular los UCP se usa la siguiente fórmula:

$$UCP = UUCP * TCF * EF$$

Significado de estas siglas:

UCP: Puntos de Casos de Uso ajustados.

UUCP: Puntos de Casos de Uso sin ajustar.

TFC: Factores Técnicos.

EF: Factores de Ambiente.

Una vez conocidos los valores de UUCP, TCF y EF, el valor de los puntos de Casos de Uso ajustados sería:

$$UCP = UUCP * TCF * EF$$

$$UCP = 92 * 1.025 * 0.8$$

$$UCP = 75.44$$

### 5.2.5 Factores de Complejidad Técnica.

Para calcular el factor de complejidad técnica se utilizan las siguientes fórmulas:

$$TFactor = \text{Sum}(\text{Valor} * \text{Peso})$$

$$TCF = 0.6 + (0.01 * TFactor)$$

Para realizar este cálculo, se debe evaluar cada factor, asignándole un valor como se menciona anteriormente, después se multiplican y se suman los productos para obtener el TFactor. Luego, se debe seguir la segunda fórmula multiplicando el TFactor por 0.01 y sumar el resultado a 0.6 obteniendo como resultado el TCF.

## Capítulo 5: Estudio de Factibilidad

Factor	Descripción.	Peso
T1	Sistema distribuido.	2
T2	Objetivos de performance o tiempo de respuesta.	1
T3	Eficiencia del usuario final.	1
T4	Procesamiento interno complejo.	1
T5	Código reutilizable.	1
T6	Facilidad de instalación.	0.5
T7	Facilidad de uso.	0.5
T8	Portabilidad.	2
T9	Facilidad de cambio.	1
T10	Concurrencia	1
T11	Incluye objetivos especiales de seguridad.	1
T12	Provee acceso directo a terceras partes.	1
T13	Requiere facilidades especiales de entrenamiento a usuarios.	1

**Tabla 16: Peso de los Factores de Complejidad Técnica.**

Cada uno de estos factores de complejidad técnica se evalúa según la siguiente escala:

Irrelevante: De 0 a 2

Medio: De 3 a 4

Esencial: 5

Factor	Peso	Valor	Comentario	Valor * Peso
T1	2	5	El sistema es distribuido.	10
T2	1	4	El tiempo de respuesta del sistema debe ser óptimo.	4
T3	1	0	No hay restricciones de eficiencia del usuario final.	0
T4	1	2	El procesamiento interno no tiene gran complejidad	2
T5	1	4	El código debe ser reutilizable.	4
T6	0.5	2	El sistema debe ser fácil de montar y configurar	1

## Capítulo 5: Estudio de Factibilidad

T7	0.5	5	Debe ser fácil de usar.	2.5
T8	2	0	El sistema debe ser portable.	0
T9	1	5	El sistema deberá ser flexible a cambio mejorar y extender su funcionamiento.	5
T10	1	4	Debe presentar concurrencia.	4
T11	1	5	El sistema gestiona información sensible	5
T12	1	4	Provee acceso directo a terceras partes	4
T13	1	1	No se requieren facilidades especiales de entrenamiento de usuarios.	1
<b>Total:</b>				42.5

*Tabla 17: Cálculo del Valor de TFactor.*

Calculado el TFactor, el resultado del Factor de Complejidad Técnica sería:

$$TCF = 0.6 + (0.01 * 42.5)$$

$$TCF = 1.025$$

### 5.2.6 Factores de Ambiente

Los factores sobre los cuales se realiza la evaluación son 8, que están relacionados con las habilidades y experiencias del grupo de personas involucradas con el desarrollo del proyecto. Estos factores se muestran a continuación:

Factor	Descripción	Peso
E1	Familiaridad con el modelo de proyecto utilizado.	1.5
E2	Experiencia en la aplicación.	0.5
E3	Experiencia en orientación a objetos.	1
E4	Capacidad del analista líder.	0.5
E5	Motivación.	1
E6	Estabilidad de los requerimientos	2
E7	Personal part-time	-1

## Capítulo 5: Estudio de Factibilidad

E8	Dificultad del lenguaje de programación	-1
----	---	----

**Tabla 18: Peso de los Factores Ambientales.**

Cada uno de estos factores se debe calificar con un valor de 0 a 5. Las fórmulas para este punto son:

$$E\text{Factor} = \text{Sum}(\text{Valor} * \text{Peso})$$

$$EF = 1.4 + (-0.03 * E\text{Factor})$$

Factor	Peso	Valor	Comentario	Valor * Peso
E1	1.5	3	El grupo está familiarizado con el modelo de proyecto.	4.5
E2	0.5	2	No hay experiencia en la aplicación.	1
E3	1	5	El grupo tiene experiencia en la programación orientado a objetos.	5
E4	0.5	3	Capacidad del analista líder.	1.5
E5	1	4	El grupo está motivado.	4
E6	2	3	Estabilidad de los requerimientos	6
E7	-1	0	Personal part-time	0
E8	-1	2	Se utilizará PHP como lenguaje de programación de baja complejidad.	-2
<b>Total:</b>				20

**Tabla 19: Cálculo del Valor de EFactor.**

Calculado el EFactor, el resultado del Factor de Ambiente sería:

$$EF = 1.4 + (-0.03 * 20)$$

$$EF = 0.8$$

### 5.2.7 Esfuerzo Horas-Hombres

Este cálculo se realiza con el fin de tener una aproximación del esfuerzo, pensando solo en el desarrollo según las funcionalidades de los casos de uso. Está basado en los factores ambientales y se calcula de la siguiente manera: Primero se debe contar la cantidad de factores de ambientes del E1 al E6 que tienen

## Capítulo 5: Estudio de Factibilidad

una puntuación menor a 3 y la cantidad de factores de ambiente del E7 al E8 que son mayores que 3. Este resultado se evalúa según la siguiente tabla:

Horas-Hombres	Descripción
20	Si el valor es $\leq 2$
28	Si el valor es $\leq 4$
36	Si el valor es $\geq 5$

*Tabla 20: Cantidad de Horas-Hombres según el valor.*

El esfuerzo en Horas-Hombres viene dado por:

$$E = UCP * CF$$

Significado de estas siglas:

E: Esfuerzo estimado en Horas-Hombres.

UCP: Puntos de Casos de Uso ajustados.

CF: Horas-Hombres.

Por tanto el esfuerzo en Horas-Hombres sería:

$$E = 75.44 * 28$$

$$E = 2112.32$$

### 5.3 Costos

Para el cálculo del costo estimado del proyecto se establece la relación entre el salario de los trabajadores del proyecto y el tiempo que se necesitará para desarrollar el mismo.

Salario medio de la fuerza de trabajo: \$100.00

Cantidad de trabajadores: 2

Teniendo en cuenta que cada trabajador trabaja 6 horas diarias y que el resultado del esfuerzo necesario en horas-hombre para terminar el proyecto es de 2112.32, se puede afirmar que se necesitan 176.02 días



## Capítulo 5: Estudio de Factibilidad

---

de trabajo por cada integrante del proyecto lo que significa que el proyecto debe durar un total de 7 meses y 8 días, por lo cual se puede afirmar que el costo total del proyecto es:

$$CTM = SM * CT * DP$$

Significado de las siglas:

CTM: Costo Total del módulo.

SM: Salario Medio.

CT: Cantidad de Trabajadores.

DP: Duración del Proyecto (Meses).

Por lo tanto el costo total del módulo sería:

$$CTM = 100 * 2 * 7.3$$

$$CTM = 1460$$

### 5.4 Beneficios Tangibles e Intangibles

La aplicación permitirá a los usuarios que interactúan con el sistema en todo el proceso de realización de una auditoría a un sistema Web por parte del proyecto LABSI, hacer una gestión eficiente de los reportes generados por las distintas pruebas de penetración efectuadas. Además ofrecerá datos estadísticos sobre las auditorías realizadas por el proyecto.

### 5.5 Análisis de Costos y Beneficios

Teniendo en cuenta que todas la herramientas usadas en el proceso de desarrollo del proyecto son libres por lo que no requieren del pago de licencias y que el soporte a la aplicación final será brindado por trabajadores de la organización que lo utilizará, se puede afirmar que es factible el desarrollo del Sistema de Gestión de Reportes de Vulnerabilidades en el proyecto LABSI.

## *Capítulo 5: Estudio de Factibilidad*

---

### **5.6 Conclusiones del Capítulo.**

En el presente capítulo se realizó un estudio de la factibilidad del proyecto, así como la determinación del tiempo y costo para la realización del sistema. Además se utilizó el método de estimación Por Puntos de Caso de Uso, que resultó ser el más adecuado para el desarrollo con RUP, que fue la metodología usada en este trabajo. El esfuerzo horas- hombres fue de: 2112.32. El costo estimado del sistema fue de 1460. Y se puede afirmar que la creación del sistema es factible.

### **CONCLUSIONES**

Al finalizar la investigación y desarrollo del sistema que se proponía, se puede concluir que se logró almacenar los reportes arrojados por las pruebas, y crear un sistema que permite gestionar las auditorías, herramientas, usuarios y reporte que maneja el laboratorio de seguridad de la UCI, brindando así un lugar centralizado que da resultados estadísticos importantes para conocer el estado del funcionamiento del laboratorio.

Las herramientas escogidas para el desarrollo fueron las adecuadas ya que permitieron el buen desarrollo de la aplicación.

El análisis y diseño de la aplicación permitieron que la misma quedara con la calidad requerida.

Se realizó la implementación del software según las metas planteadas, brindando todas las funcionalidades propuestas al inicio.

Los clientes quedaron complacidos con la aplicación que se brindó.

### **RECOMENDACIONES**

- ✓ Mejorar la interfaz visual que interactúa con el cliente.
- ✓ La aplicación se puede mejorar implementando la funcionalidad de evaluar los sistemas auditados, ésta debe ser capaz de confeccionar una calificación, teniendo como criterio los datos almacenados en el sistema.
- ✓ La utilización de la metodología OWASP para el proyecto LABSI ya que sus herramientas y documentación están capacitadas para lograr que un sistema Web sea seguro.

## BIBLIOGRAFÍA

1. **OWASP**. Sitio Oficial de OWASP. [En línea] 2006. [Citado el: 8 de febrero de 2010.] [www.owasp.org](http://www.owasp.org).
2. Lenguajes de Programación. [En línea] 2009. [Citado el: 15 de febrero de 2010.] [www.lenguajes-de-programacion.com](http://www.lenguajes-de-programacion.com).
3. Programando Ideas. *Blog de programación*. [En línea] 2009. [Citado el: 18 de marzo de 2010.] <http://programandoideas.com>.
4. INSTITUTO NACIONAL DE ESTADISTICA E INFORMATICA. [En línea] [Citado el: 5 de enero de 2010.] <http://jhoel-jp.tripod.com>.
5. Página Oficial de Visual Paradigm. [En línea] 1999. [Citado el: 12 de febrero de 2010.] <http://www.visual-paradigm.com/product/vpum/>.
6. KumbiaPHP Framework. [En línea] 2007. [Citado el: 20 de enero de 2010.] <http://www.kumbiaphp.com>.
7. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James**. El Proceso Unificado del Software. [En línea] [Citado el: 10 de enero de 2010.] <http://bibliodoc.uci.cu/pdf/8478290362.pdf>.
8. **SYSTEMS, SPARX**. SPARX SYSTEMS. [En línea] 2000. [Citado el: 19 de abril de 2010.] [http://www.sparxsystems.com.ar/resources/tutorial/logical\\_model.html](http://www.sparxsystems.com.ar/resources/tutorial/logical_model.html).
9. Programación en Castellano. [En línea] 1998. [Citado el: 2 de mayo de 2010.] [http://www.programacion.com/articulo/modelo\\_de\\_datos\\_178#moddatos\\_logico](http://www.programacion.com/articulo/modelo_de_datos_178#moddatos_logico).
10. PostgreSQL. *Comunidad de usuarios de PostgreSQL*. [En línea] [Citado el: 20 de enero de 2010.] <http://www.postgresql.org>.
11. NetBeans. *Comunidad de usuarios de NetBeans*. [En línea] [Citado el: 15 de marzo de 2010.] <http://netbeans.org>.

12. **Quiñones, Ernesto A.** Introducción a PostgreSQL. [En línea] [Citado el: 14 de febrero de 2010.] [http://www.postgresql.org.pe/articles/introduccion\\_a\\_postgresql.pdf](http://www.postgresql.org.pe/articles/introduccion_a_postgresql.pdf).
13. **Larman, Graig.** UML y Patrones. *Introducción al análisis y diseño de la programación orientada a objetos*. [En línea] [Citado el: 22 de febrero de 2010.] <http://bibliodoc.uci.cu/pdf/reg00061.pdf>.
14. Modelado del sistema con UML. *Popkin Software and Systems*. [En línea] [Citado el: 22 de marzo de 2010.] <http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/multiple-html/>.
15. Manual de PHP. [En línea] [Citado el: 25 de febrero de 2010.] <http://www.webestilo.com/php/>.
16. Sitio Oficial de JQuery. [En línea] septiembre de 2009. [Citado el: 12 de abril de 2010.] <http://jquery.org>.
17. Jesús García Molina, M. José Ortín, Begoña Moros, Joaquín Nicolás, Ambrosio Toval. De los Procesos del Negocio a los Casos de Uso. [En línea] [http://www.willydev.net/descargas/willydev\\_modeladodenegocio.pdf](http://www.willydev.net/descargas/willydev_modeladodenegocio.pdf).
18. WAMP SERVER. *Para usuarios del servidor wamp*. [En línea] [Citado el: 4 de febrero de 2010.] <http://www.wampserver.com/en/>.
19. FPDF Library. *Librería de fpdf*. [En línea] [Citado el: 2 de mayo de 2010.] <http://www.fpdf.org/>.
20. UTF-8 and Unicode Standards. [En línea] [Citado el: 10 de mayo de 2010.] <http://www.utf8.com/>.

Anexo 1: Descripción textual de los casos de uso.

Caso de Uso	
<b>Caso de Uso 1</b>	Autenticar Usuario
<b>Propósito</b>	Establece los permisos necesarios para restringir el acceso a la información manejada por el sistema, de acuerdo con el rol de cada actor.
<b>Actores</b>	Usuario
<b>Resumen</b>	El caso de uso se inicia cuando un usuario trata de acceder al sistema.
<b>Referencias</b>	RF6.
<b>Precondiciones</b>	El usuario debe estar conectado a la red.
<b>Poscondiciones</b>	El usuario queda autenticado.
Flujo normal de eventos.	
Acción del actor	Respuesta del sistema
1. Accede al sistema.	2. Muestra la página de inicio que contiene el formulario para la entrada de los datos del usuario.
3. Llena el formulario con los datos requeridos.	4. Muestra la interfaz según los privilegios del actor.
Flujo Alternativo	
Acción del actor	Respuesta del sistema
	4.1 Muestra mensaje indicando que se deben llenar todos los campos. 4.1 Muestra mensaje indicando que los datos

	son incorrectos.
--	------------------

**Tabla 21: Descripción textual del Caso de Uso Autenticar Usuario.**

Caso de Uso	
<b>Caso de Uso 2</b>	Mostar Reporte
<b>Propósito</b>	Permite que los actores según sus permisos, vean los datos de los reportes almacenados en el sistema.
<b>Actores</b>	Usuario
<b>Resumen</b>	El caso de uso se inicia cuando un usuario solicita ver los datos de un reporte.
<b>Referencias</b>	RF1.5.
<b>Precondiciones</b>	El usuario debe estar autenticado en el sistema y deben estar listados los reportes a los que el usuario tiene acceso.
<b>Poscondiciones</b>	El usuario ve los datos del reporte solicitado.
Flujo normal de eventos	
Acción del actor	Respuesta del sistema
1. Selecciona el reporte que desea ver.	2. Muestra los datos del reporte solicitado.

**Tabla 22: Descripción textual del Caso de Uso Mostrar Reporte.**

Caso de Uso	
<b>Caso de Uso 3</b>	Gestionar Reporte
<b>Propósito</b>	Permite crear, modificar y eliminar los datos relacionados con los reportes de las pruebas de penetración realizadas.
<b>Actores</b>	Auditor (inicia), Administrador.
<b>Resumen</b>	El caso de uso se inicia cuando un auditor



	solicita crear, modificar o eliminar un reporte.
<b>Referencias</b>	RF4, RF4.1, RF4.2, RF4.3, RF4.4, RF4.5.
<b>Precondiciones</b>	El usuario debe estar autenticado en el sistema y tener los privilegios requeridos.
<b>Poscondiciones</b>	Queda creado, modificado o eliminado un reporte.
<b>Flujo normal de eventos</b>	
<b>Acción del actor</b>	<b>Respuesta del sistema</b>
1. Selecciona la opción Gestionar Reporte.	2. Muestra una interfaz con las opciones posibles.
3. Escoge la opción a realizar.	4. Si escoge la opción Crear Reporte ver sección "Crear Reporte". 5. Si escoge la opción Modificar Reporte ver sección "Modificar Reporte". 6. Si escoge la opción Eliminar Reporte ver sección "Eliminar Reporte".
<b>Sección "Crear Reporte"</b>	
<b>Acción del actor</b>	<b>Respuesta del sistema</b>
1. El auditor accede a la interfaz Crear Reporte	2. Muestra el formulario con los datos requeridos para crear un nuevo reporte.
3. Inserta los datos requeridos	4. Crea el reporte con los datos insertados y muestra un mensaje de confirmación.
<b>Flujo Alterno</b>	
<b>Acción del actor</b>	<b>Respuesta del sistema</b>
	4.1 Muestra mensaje indicando que se deben llenar todos los campos. 4.1 Muestra mensaje indicando que los datos

	son incorrectos.
<b>Sección “Modificar Reporte”</b>	
Acción del actor	Respuesta del sistema
1. El auditor accede a la interfaz Modificar Reporte.	2. Muestra una interfaz con un listado de los reportes a los que el usuario tiene acceso.
3. Selecciona el reporte a modificar	4. Muestra un formulario con los datos requeridos para modificar el reporte seleccionado.
5. Inserta los nuevos datos	6. Modifica el reporte con los datos entrados por el auditor y muestra un mensaje de confirmación de la acción Modificar Reporte.
<b>Flujo Alterno</b>	
Acción del actor	Respuesta del sistema
	6.1 Muestra mensaje indicando que se deben llenar todos los campos. 6.1 Muestra mensaje indicando que los datos son incorrectos.
<b>Sección “Eliminar Reporte”</b>	
Acción del actor	Respuesta del sistema
1. El auditor accede a la interfaz Eliminar Reporte.	2. Muestra un listado de los reportes a los cuales tiene acceso el actor
3. Selecciona el reporte que desea eliminar.	4. Elimina el reporte seleccionado por el actor y muestra un mensaje de confirmación de la acción Eliminar Reporte.

*Tabla 23: Descripción textual del Caso de Uso Gestionar Reporte.*

<b>Caso de Uso</b>	
<b>Caso de Uso 4</b>	Imprimir Reporte

<b>Propósito</b>	Permite imprimir los datos relacionados con los reportes almacenados en el sistema.
<b>Actores</b>	Cliente
<b>Resumen</b>	El caso de uso se inicia cuando el cliente solicita imprimir un reporte.
<b>Referencias</b>	RF7, RF4.5.
<b>Precondiciones</b>	El cliente debe estar autenticado en el sistema y debe tener una impresora conectada a su ordenador.
<b>Poscondiciones</b>	El sistema manda a imprimir el documento seleccionado.
<b>Flujo normal de eventos.</b>	
<b>Acción del actor</b>	<b>Respuesta del sistema</b>
1. Selecciona la opción Imprimir Reporte.	2. Muestra la página con un listado de los reportes a los que tiene acceso el cliente.
3. Selecciona el reporte que desea imprimir	4. Manda a imprimir el reporte y muestra un mensaje de confirmación.
<b>Flujo Alterno</b>	
<b>Acción del actor</b>	<b>Respuesta del sistema</b>
	4.1 Muestra mensaje indicando que se debe marcar al menos un reporte. 4.1 Muestra mensaje indicando que no se encontró la impresora para realizar la acción.

**Tabla 24: Descripción textual del Caso de Uso Imprimir Reporte.**

<b>Caso de Uso</b>	
<b>Caso de Uso 5</b>	Listar Auditoría.
<b>Propósito</b>	Permite obtener un listado de las auditorías

	almacenadas en el sistema.
<b>Actores</b>	Supervisor
<b>Resumen</b>	El caso de uso se inicia cuando el supervisor solicita listar las auditorías.
<b>Referencias</b>	RF2.4, RF2.5.
<b>Precondiciones</b>	El cliente debe estar autenticado en el sistema.
<b>Poscondiciones</b>	El sistema lista las auditorías.
Flujo normal de eventos	
Acción del actor	Respuesta del sistema
1. Selecciona la opción Listar Auditoría.	2. Muestra la página con un listado de los datos de las auditorías a las que tiene acceso el supervisor y brinda la posibilidad de seleccionar una auditoría para ver sus datos.

*Tabla 25: Descripción textual del Caso de Uso Listar Auditoría*

Caso de Uso	
<b>Caso de Uso 6</b>	Listar Reporte
<b>Propósito</b>	Permite obtener un listado de los reportes almacenados en el sistema
<b>Actores</b>	Usuario.
<b>Resumen</b>	El caso de uso se inicia cuando el usuario solicita listar los reportes.
<b>Referencias</b>	RF4.4, RF4.5.
<b>Precondiciones</b>	El cliente debe estar autenticado en el sistema.
<b>Pos-condiciones</b>	El sistema lista todos los reportes a los que el usuario autenticado tiene acceso.

Flujo normal de eventos	
Acción del actor	Respuesta del sistema
1. Selecciona la opción Listar Reporte.	2. Muestra la página con un listado del reporte a los que tiene acceso el usuario y brinda la opción de seleccionar un reporte para ver sus datos.

Tabla 26: Descripción textual del Caso de Uso Listar Reporte.

Caso de Uso	
<b>Caso de Uso 7</b>	Listar Herramienta
<b>Propósito</b>	Permite obtener un listado de las herramientas almacenadas en el sistema.
<b>Actores</b>	Supervisor
<b>Resumen</b>	El caso de uso se inicia cuando el supervisor solicita listar herramientas.
<b>Referencias</b>	RF3.4, RF3.5.
<b>Precondiciones</b>	El cliente debe estar autenticado en el sistema.
<b>Poscondiciones</b>	El sistema lista las herramientas a las que el supervisor tiene acceso.
Flujo normal de eventos	
Acción del actor	Respuesta del sistema
1. Selecciona la opción Listar Herramienta.	2. Muestra la página con un listado de las herramientas a las que tiene acceso el supervisor y da la posibilidad de seleccionar una para ver sus datos.

Tabla 27: Descripción textual del Caso de Uso Listar Herramientas.

Caso de Uso	
<b>Caso de Uso 8</b>	Mostrar Estadística

<b>Propósito</b>	Permite obtener datos estadísticos relacionados con los datos de las auditorías realizadas por el proyecto LABSI.
<b>Actores</b>	Supervisor
<b>Resumen</b>	El caso de uso se inicia cuando el supervisor solicita mostrar estadísticas
<b>Referencias</b>	RF5, RF8.
<b>Precondiciones</b>	El usuario debe estar autenticado en el sistema.
<b>Poscondiciones</b>	El sistema muestra las estadísticas del proyecto.
<b>Flujo normal de eventos.</b>	
<b>Acción del actor</b>	<b>Respuesta del sistema</b>
1. Selecciona la opción Mostrar Estadística.	2. Confecciona las estadísticas del sistema y muestra una página con las estadísticas del proyecto LABSI.

*Tabla 28: Descripción textual del Caso de Uso Mostrar Estadística.*

Caso de Uso	
<b>Caso de Uso 9</b>	Gestionar Auditoría
<b>Propósito</b>	Permite crear, modificar y eliminar las auditorías que son manejadas por el sistema.
<b>Actores</b>	Administrador.
<b>Resumen</b>	El caso de uso se inicia cuando un administrador selecciona la opción gestionar auditoría.
<b>Referencias</b>	RF2, RF2.1, RF2.2, RF2.3, RF2.4, RF2.5.
<b>Precondiciones</b>	El usuario debe estar autenticado en el sistema

	y tener los privilegios requeridos.
<b>Poscondiciones</b>	Queda creada, modificada o eliminada una auditoría.
<b>Flujo normal de eventos</b>	
<b>Acción del actor</b>	<b>Respuesta del sistema</b>
1. Selecciona la opción Gestionar Auditoría.	2. Muestra una interfaz con las opciones posibles.
3. Escoge la opción a realizar.	4. Si escoge la opción Crear Auditoría ver sección "Crear Auditoría". 5. Si escoge la opción Modificar Auditoría ver sección "Modificar Auditoría". 6. Si escoge la opción Eliminar Auditoría ver sección "Eliminar Auditoría".
<b>Sección "Crear Auditoría"</b>	
<b>Acción del actor</b>	<b>Respuesta del sistema</b>
1. El administrador accede a la interfaz Crear Auditoría	2. Muestra el formulario con los datos requeridos para crear una nueva auditoría.
3. Inserta los datos requeridos.	4. Crea la auditoría con los datos insertados y muestra un mensaje de confirmación.
<b>Flujo Alterno</b>	
<b>Acción del actor</b>	<b>Respuesta del sistema</b>
	4.1 Muestra mensaje indicando que se deben llenar todos los campos. 4.1 Muestra mensaje indicando que los datos son incorrectos.
<b>Sección "Modificar Auditoría"</b>	
<b>Acción del actor</b>	<b>Respuesta del sistema</b>

1. El administrador accede a la interfaz Modificar Auditoría.	2. Muestra una interfaz con un listado de las auditorías.
3. Selecciona la auditoría a modificar	4. Muestra un formulario con los datos requeridos para modificar la auditoría seleccionada.
5. Inserta los nuevos datos	6. Modifica la auditoría con los datos entrados por el administrador y muestra un mensaje de confirmación de la acción Modificar Auditoría.
Flujo Alternativo	
Acción del actor	Respuesta del sistema
	6.1 Muestra mensaje indicando que se deben llenar todos los campos. 6.1 Muestra mensaje indicando que los datos son incorrectos.
Sección "Eliminar Auditoría"	
Acción del actor	Respuesta del sistema
1. El administrador accede a la interfaz Eliminar Auditoría.	2. Muestra un listado de las auditorías.
3. Selecciona la auditoría que desea eliminar.	4. Elimina la auditoría seleccionada por el actor y muestra un mensaje de confirmación de la acción Eliminar Auditoría.

Tabla 29: Descripción textual del Caso de Uso Gestionar Auditoría.

Caso de Uso	
<b>Caso de Uso 10</b>	Gestionar Herramienta
<b>Propósito</b>	Permite insertar, modificar y eliminar los datos de las herramientas que se usan en las auditorías manejadas por el sistema.



<b>Actores</b>	Administrador.
<b>Resumen</b>	El caso de uso se inicia cuando un administrador selecciona la opción gestionar herramienta.
<b>Referencias</b>	RF3, RF3.1, RF3.2, RF3.3, RF3.4, RF3.5.
<b>Precondiciones</b>	El usuario debe estar autenticado en el sistema y tener los privilegios requeridos.
<b>Poscondiciones</b>	Queda insertada, modificada o eliminada una herramienta.
<b>Flujo normal de eventos</b>	
<b>Acción del actor</b>	<b>Respuesta del sistema</b>
1. Selecciona la opción Gestionar Herramienta.	2. Muestra una interfaz con las opciones posibles.
3. Escoge la opción a realizar.	4. Si escoge la opción Insertar Herramienta ver sección "Insertar Herramienta". 5. Si escoge la opción Modificar Herramienta ver sección "Modificar Herramienta". 6. Si escoge la opción Eliminar Herramienta ver sección "Eliminar Herramienta".
<b>Sección "Insertar Herramienta"</b>	
<b>Acción del actor</b>	<b>Respuesta del sistema</b>
1. El administrador accede a la interfaz Insertar Herramienta.	2. Muestra el formulario con los datos requeridos para insertar una nueva herramienta.
3. Inserta los datos requeridos	4. Inserta la herramienta con los datos entrados por el administrador y muestra un mensaje de confirmación.
<b>Flujo Alternativo</b>	

Acción del actor	Respuesta del sistema
	4.1 Muestra mensaje indicando que se deben llenar todos los campos. 4.1 Muestra mensaje indicando que los datos son incorrectos.
<b>Sección “Modificar Herramienta”</b>	
Acción del actor	Respuesta del sistema
1. El administrador accede a la interfaz Modificar Herramienta.	2. Muestra una interfaz con un listado de las herramientas.
3. Selecciona la herramienta a modificar	4. Muestra un formulario con los datos requeridos para modificar la herramienta seleccionada.
5. Inserta los nuevos datos	6. Modifica la herramienta con los datos entrados por el administrador y muestra un mensaje de confirmación de la acción Modificar Herramienta.
<b>Flujo Alterno</b>	
Acción del actor	Respuesta del sistema
	6.1 Muestra mensaje indicando que se deben llenar todos los campos. 6.1 Muestra mensaje indicando que los datos son incorrectos.
<b>Sección “Eliminar Herramienta”</b>	
Acción del actor	Respuesta del sistema
1. El administrador accede a la interfaz Eliminar Herramienta.	2. Muestra un listado de las herramientas.

3. Selecciona la herramienta que desea eliminar.	4. Elimina la herramienta seleccionada por el actor y muestra un mensaje de confirmación de la acción Eliminar Herramienta.
--	---

Tabla 30: Descripción textual del Caso de Uso Gestionar Herramienta.

Caso de Uso	
<b>Caso de Uso 11</b>	Gestionar Usuario
<b>Propósito</b>	Permite insertar, modificar y eliminar los usuarios que podrán interactuar con los datos manejados por el sistema.
<b>Actores</b>	Administrador.
<b>Resumen</b>	El caso de uso se inicia cuando un administrador selecciona la opción gestionar usuario.
<b>Referencias</b>	RF1, RF1.1, RF1.2, RF1.3, RF1.4, RF1.5.
<b>Precondiciones</b>	El usuario debe estar autenticado en el sistema y tener los privilegios requeridos.
<b>Poscondiciones</b>	Queda insertado, modificado o eliminado un usuario.
Flujo normal de eventos	
Acción del actor	Respuesta del sistema
1. Selecciona la opción Gestionar Usuario.	2. Muestra una interfaz con las opciones posibles.
3. Escoge la opción a realizar.	4. Si escoge la opción Insertar Usuario ver sección "Insertar Usuario". 5. Si escoge la opción Modificar Usuario ver sección "Modificar Usuario". 6. Si escoge la opción Eliminar Usuario ver

	sección “Eliminar Usuario”.
<b>Sección “Insertar Usuario”</b>	
<b>Acción del actor</b>	<b>Respuesta del sistema</b>
1. El administrador accede a la interfaz Insertar Usuario.	2. Muestra el formulario con los datos requeridos para insertar un nuevo usuario.
3. Inserta los datos requeridos	4. Inserta el usuario con los datos entrados por el administrador y muestra un mensaje de confirmación.
<b>Flujo Alterno</b>	
<b>Acción del actor</b>	<b>Respuesta del sistema</b>
	4.1 Muestra mensaje indicando que se deben llenar todos los campos. 4.1 Muestra mensaje indicando que los datos son incorrectos.
<b>Sección “Modificar Usuario”</b>	
<b>Acción del actor</b>	<b>Respuesta del sistema</b>
1. El administrador accede a la interfaz Modificar Usuario.	2. Muestra una interfaz con un listado de los usuarios.
3. Selecciona un usuario a modificar	4. Muestra un formulario con los datos del usuario y la posibilidad de modificarlos.
5. Inserta los nuevos datos	6. Modifica el usuario con los datos entrados por el administrador y muestra un mensaje de confirmación de la acción Modificar Usuario.
<b>Flujo Alterno</b>	
<b>Acción del actor</b>	<b>Respuesta del sistema</b>
	6.1 Muestra mensaje indicando que se deben llenar todos los campos.

	6.1 Muestra mensaje indicando que los datos son incorrectos.
<b>Sección "Eliminar Usuario"</b>	
Acción del actor	Respuesta del sistema
1. El administrador accede a la interfaz Eliminar Usuario.	2. Muestra un listado de los usuarios.
3. Selecciona el usuario que desea eliminar.	4. Elimina el usuario seleccionado por el actor y muestra un mensaje de confirmación de la acción Eliminar Usuario.

**Tabla 31: Descripción textual del Caso de Uso Listar Reporte.**

Anexo 2: Diagramas de secuencia del diseño.

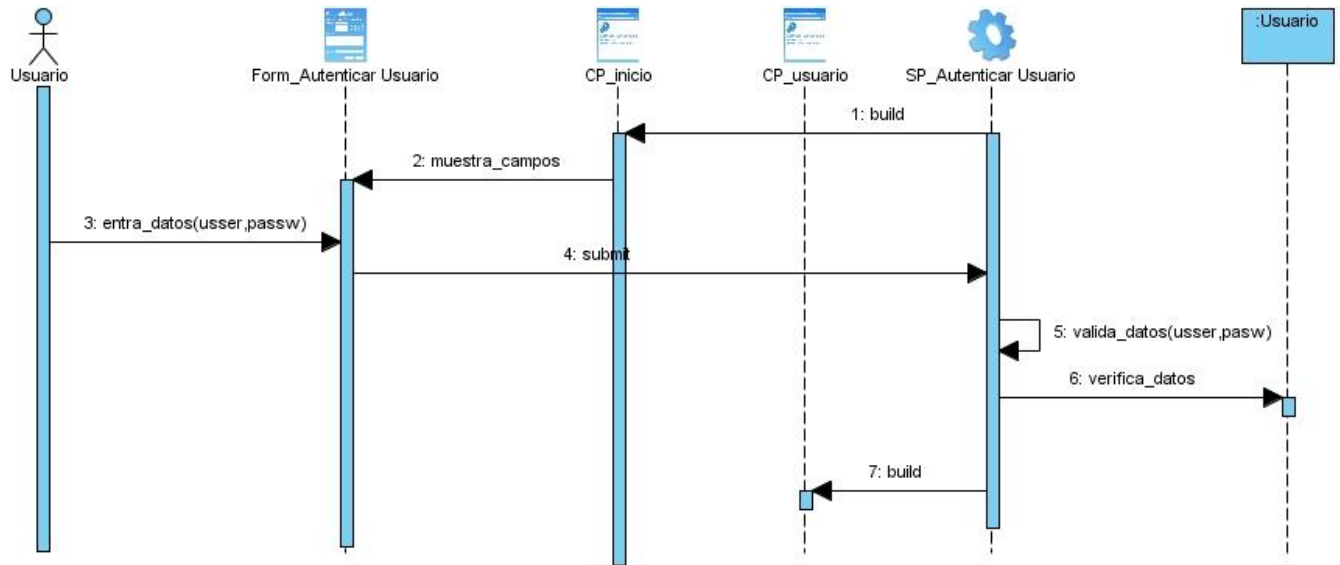


Figura 23: Diagrama de Secuencia Caso de Uso Autenticar Usuario.

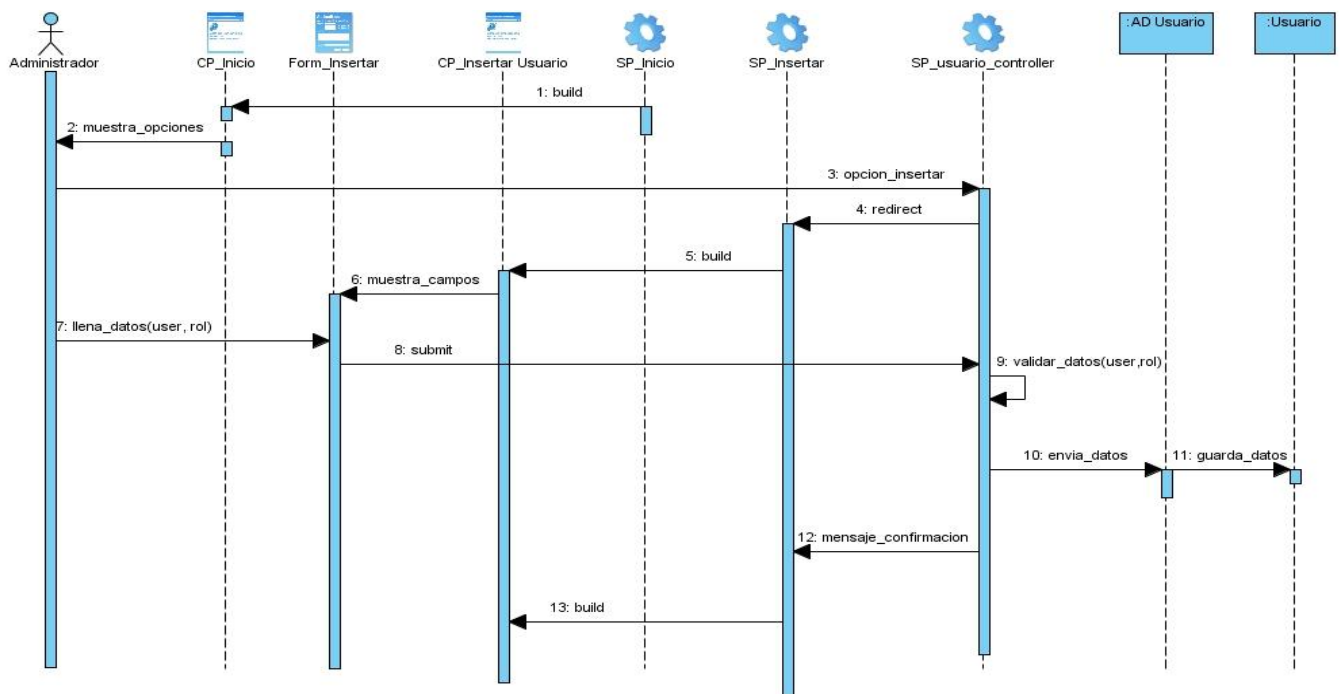


Figura 24: Diagrama de Secuencia Caso de Uso Gestionar Usuario. Escenario Insertar Usuario.

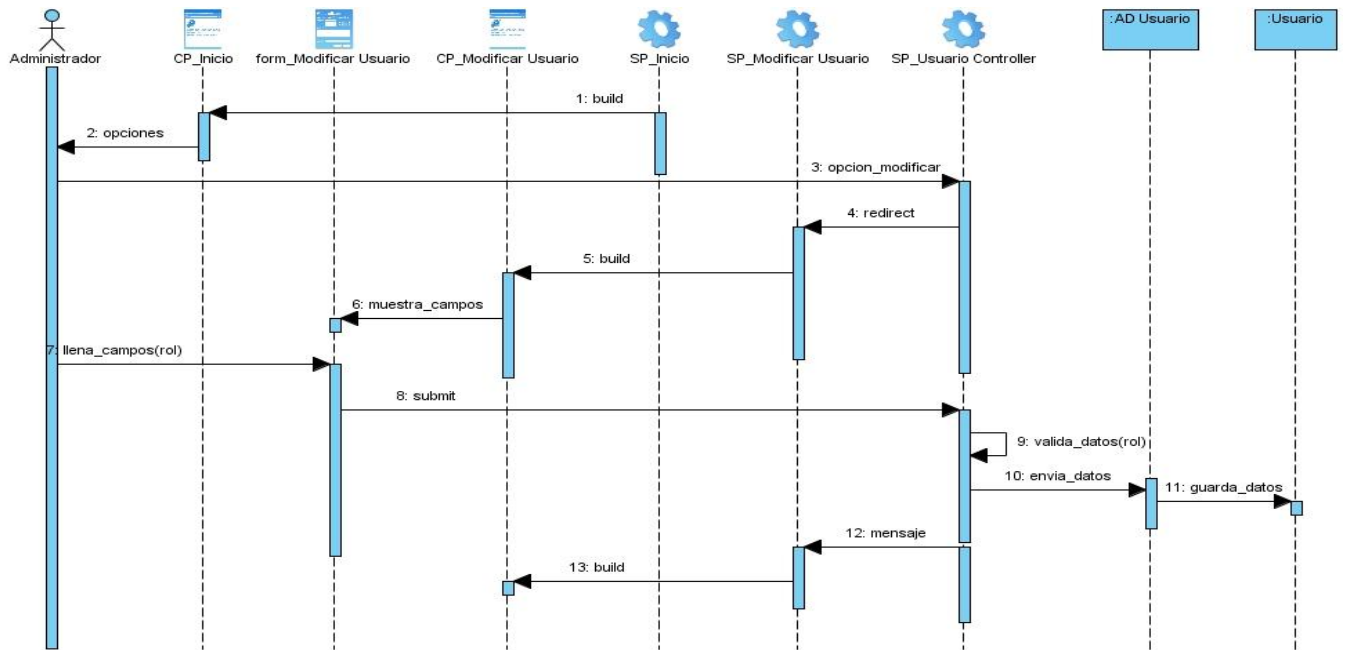


Figura 25: Diagrama de Secuencia Caso de Uso Gestionar Usuario. Escenario Modificar Usuario.

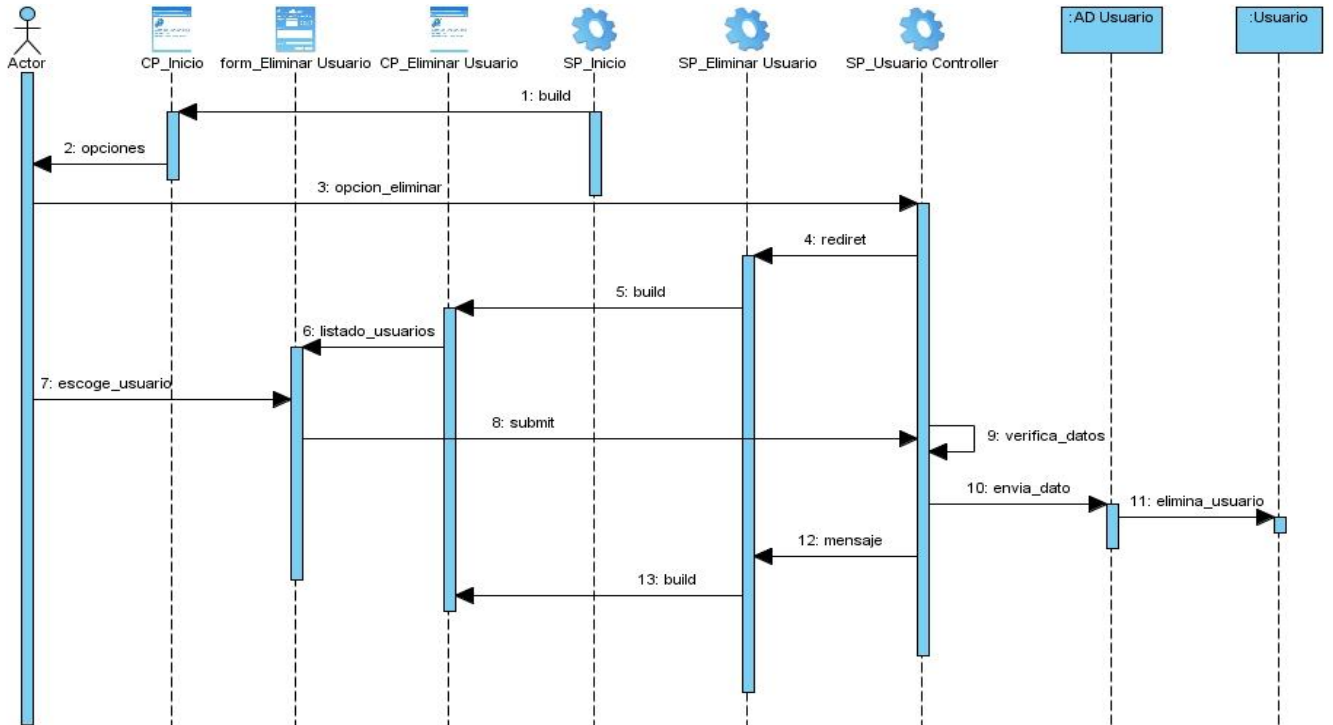


Figura 26: Diagrama de Secuencia Caso de Uso Gestionar Usuario. Escenario Eliminar Usuario.

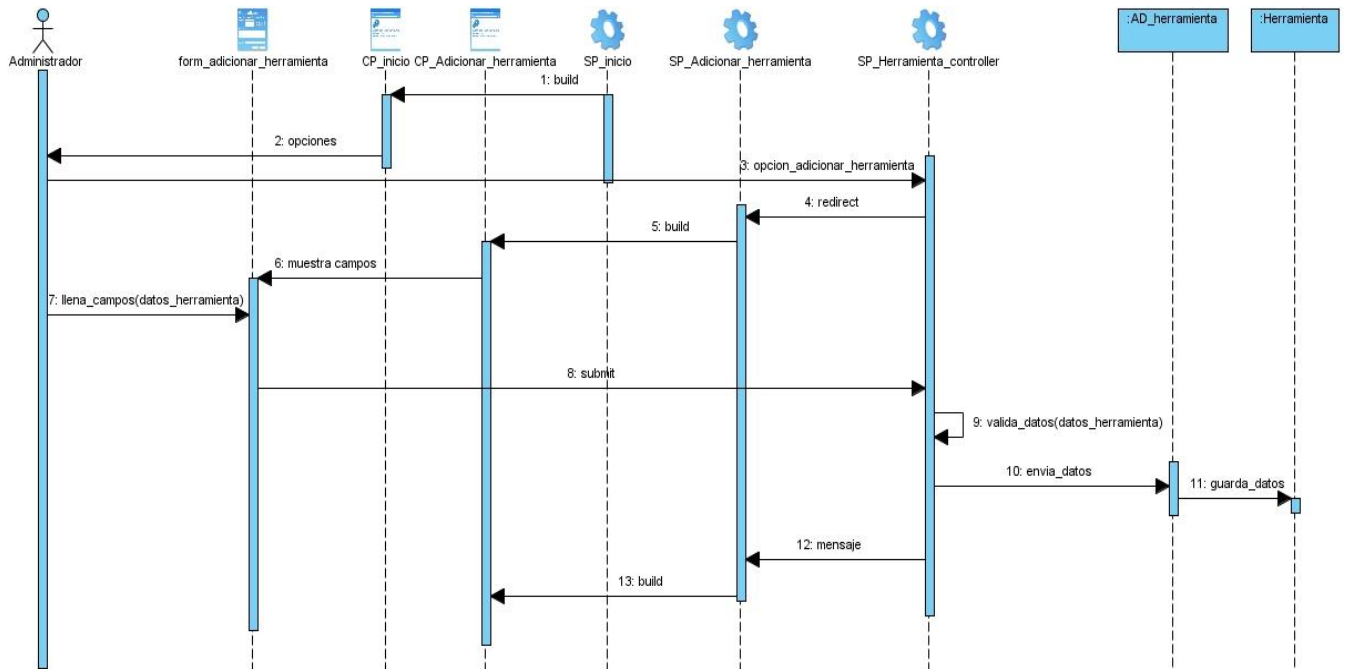


Figura 27: Diagrama de Secuencia Caso de Uso Gestionar Herramienta. Escenario Adicionar Herramienta.

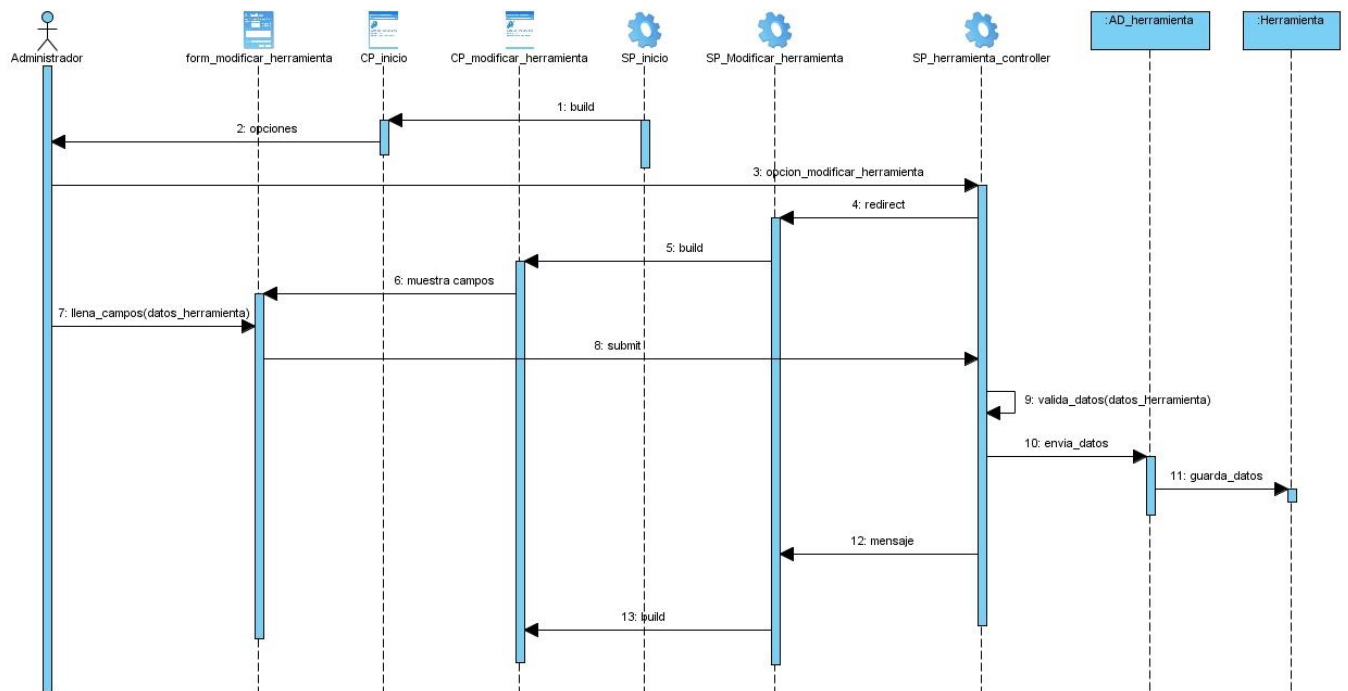


Figura 28: Diagrama de Secuencia Caso de Uso Gestionar Herramienta. Escenario Modificar Herramienta.



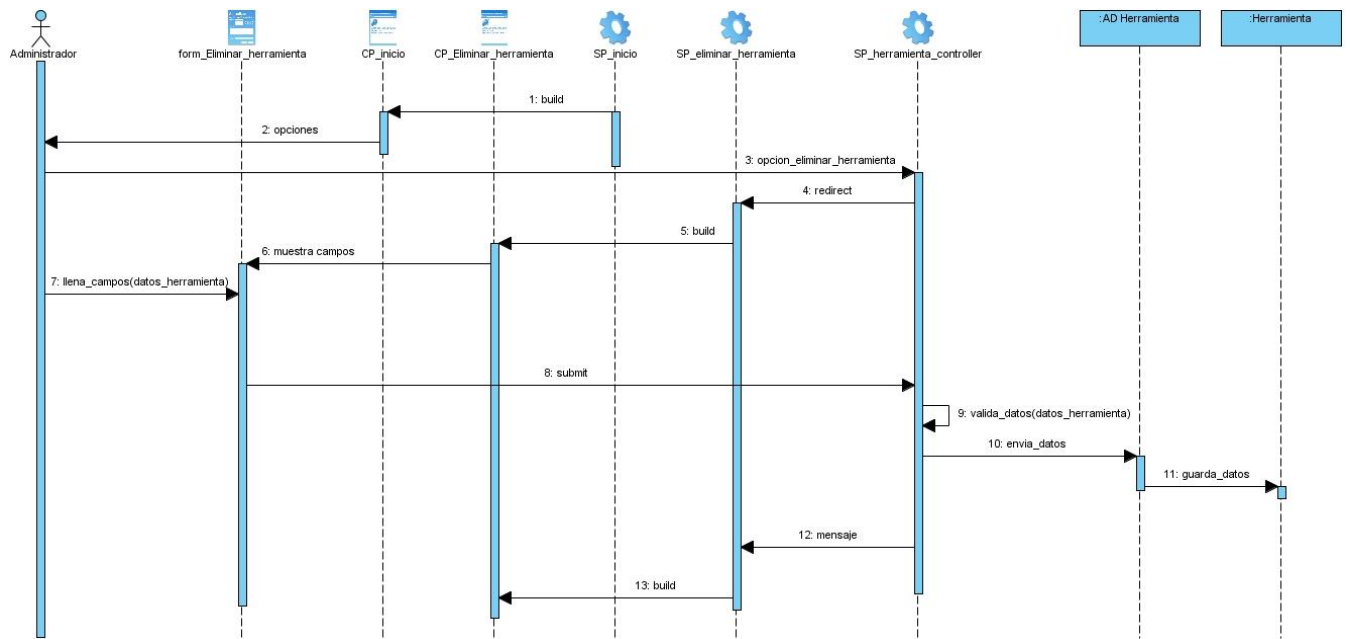


Figura 29: Diagrama de Secuencia Caso de Uso Gestionar Herramienta. Escenario Eliminar Herramienta.

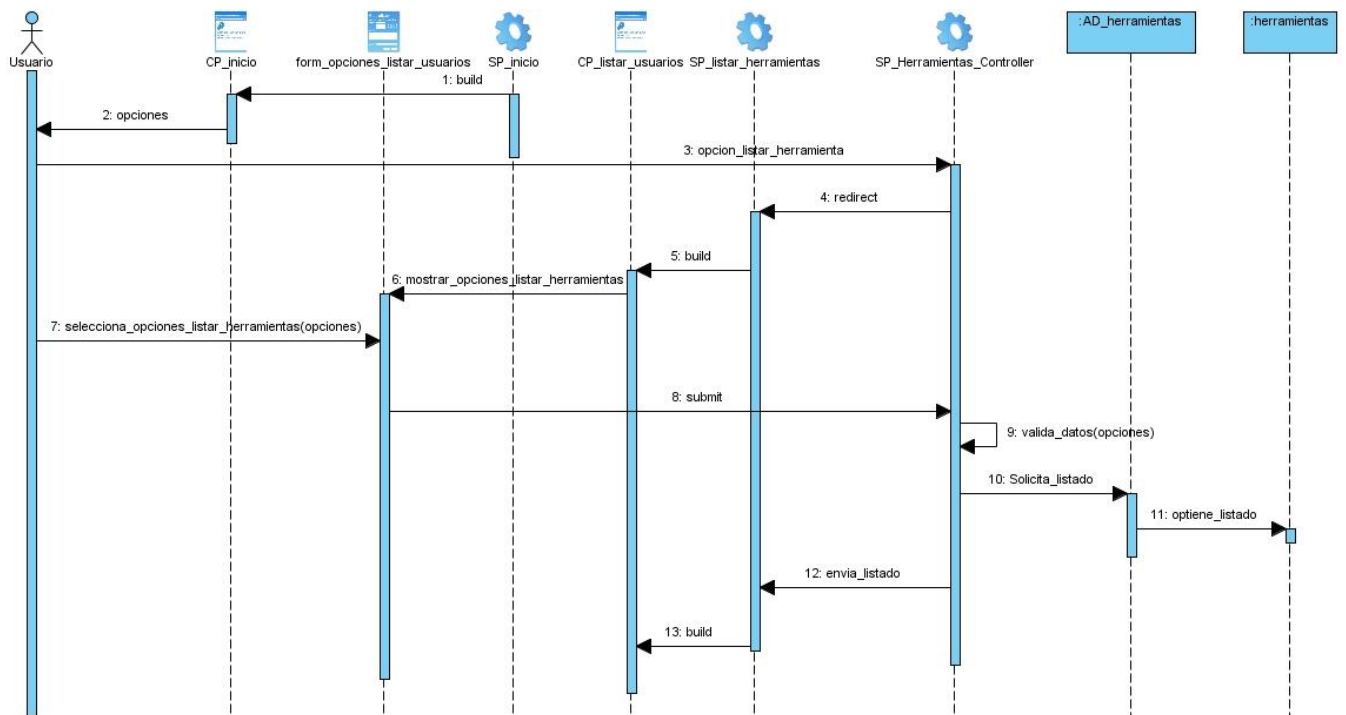


Figura 30: Diagrama de Secuencia Caso de Uso Listar Herramienta.

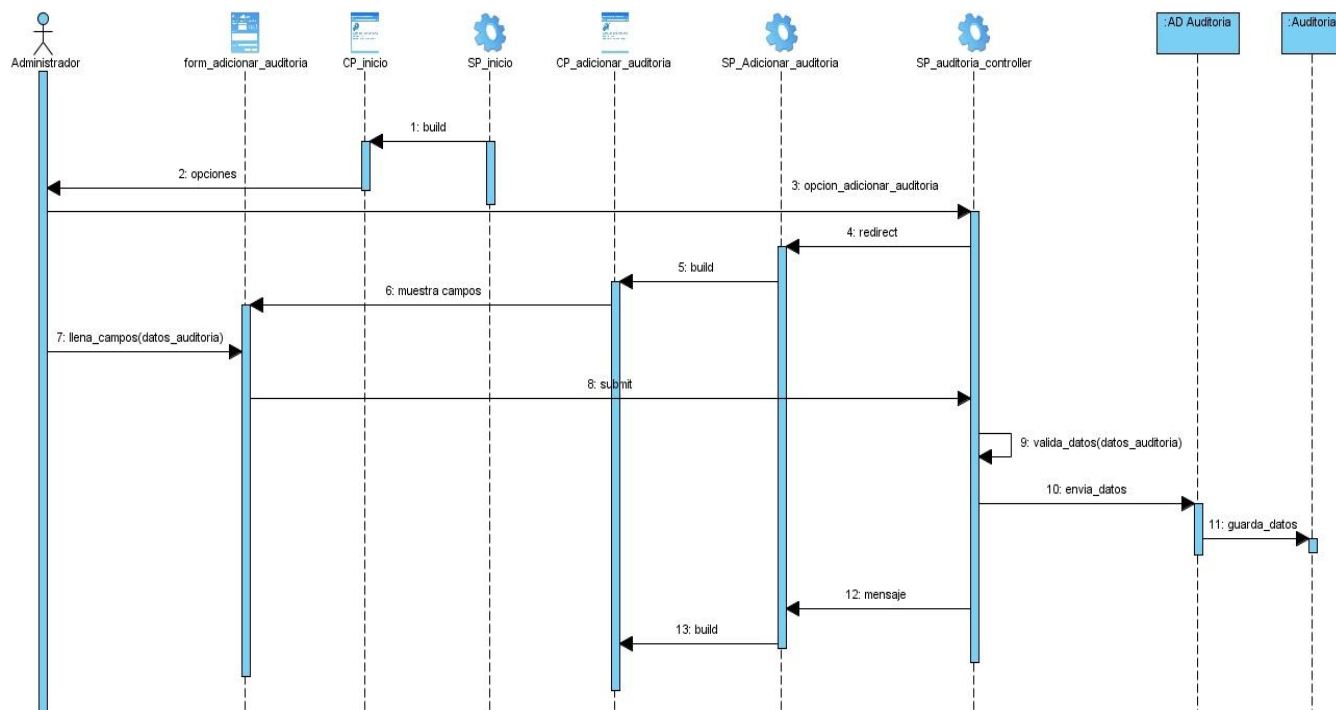


Figura 31: Diagrama de Secuencia Caso de Uso Gestionar Auditoría. Escenario Adicionar Auditoría.

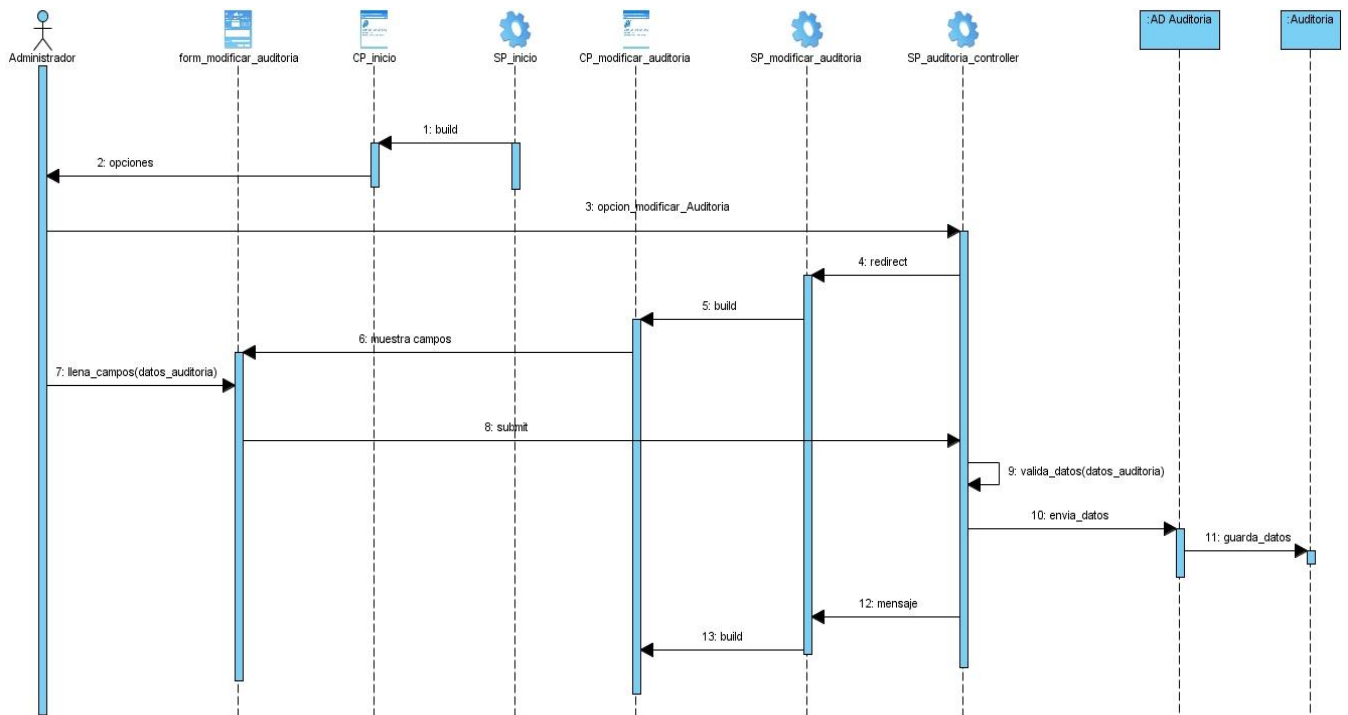


Figura 32: Diagrama de Secuencia Caso de Uso Gestionar Auditoría. Escenario Modificar Auditoría.

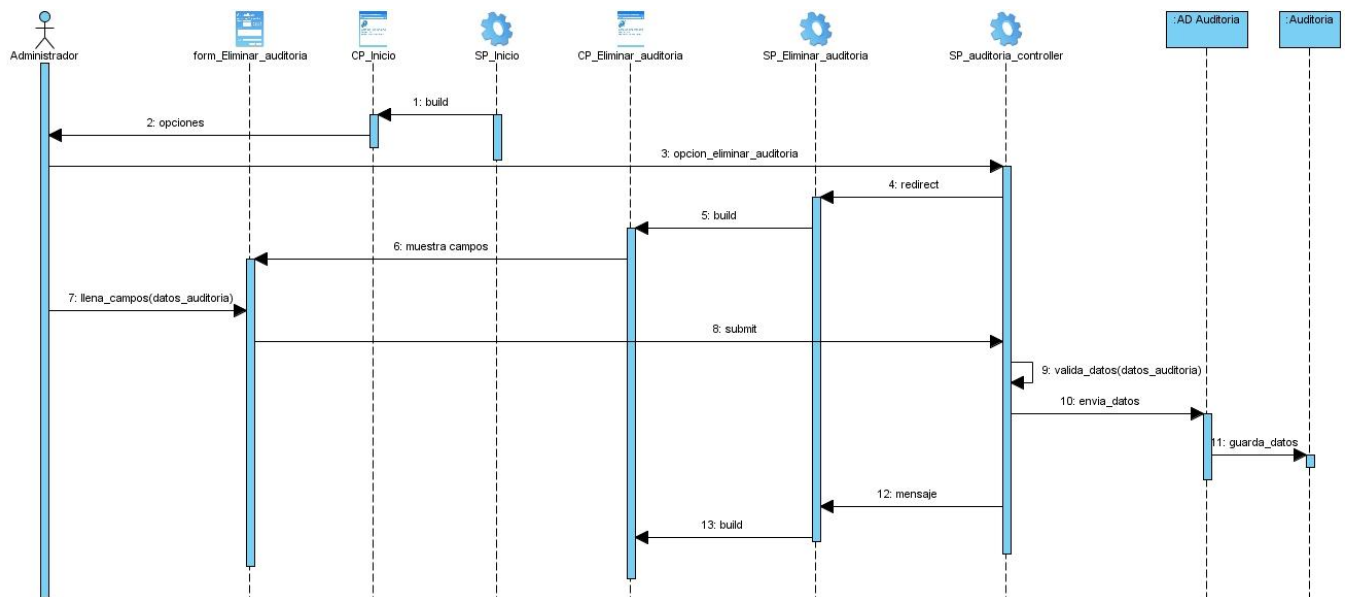


Figura 33: Diagrama de Secuencia Caso de Uso Gestionar Auditoría. Escenario Eliminar Auditoría.

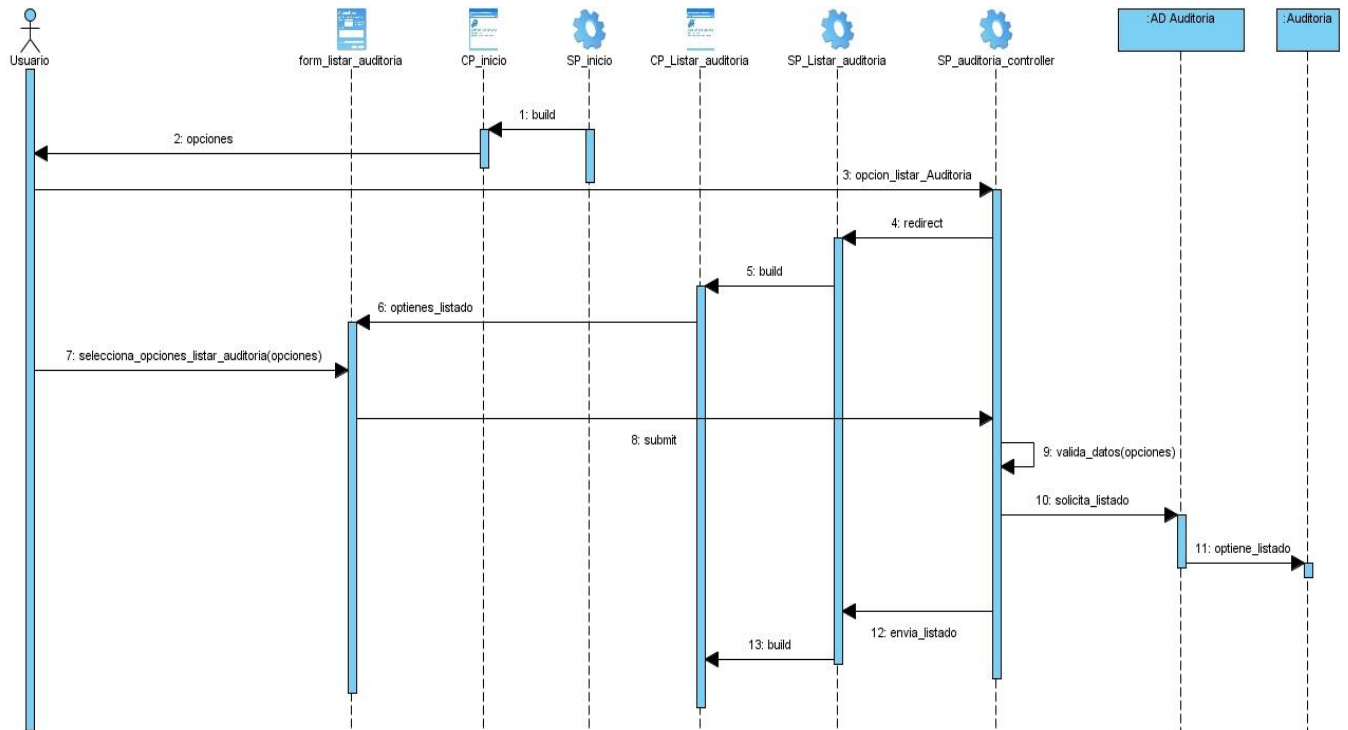


Figura 34: Diagrama de Secuencia Caso de Uso Listar Auditoría.

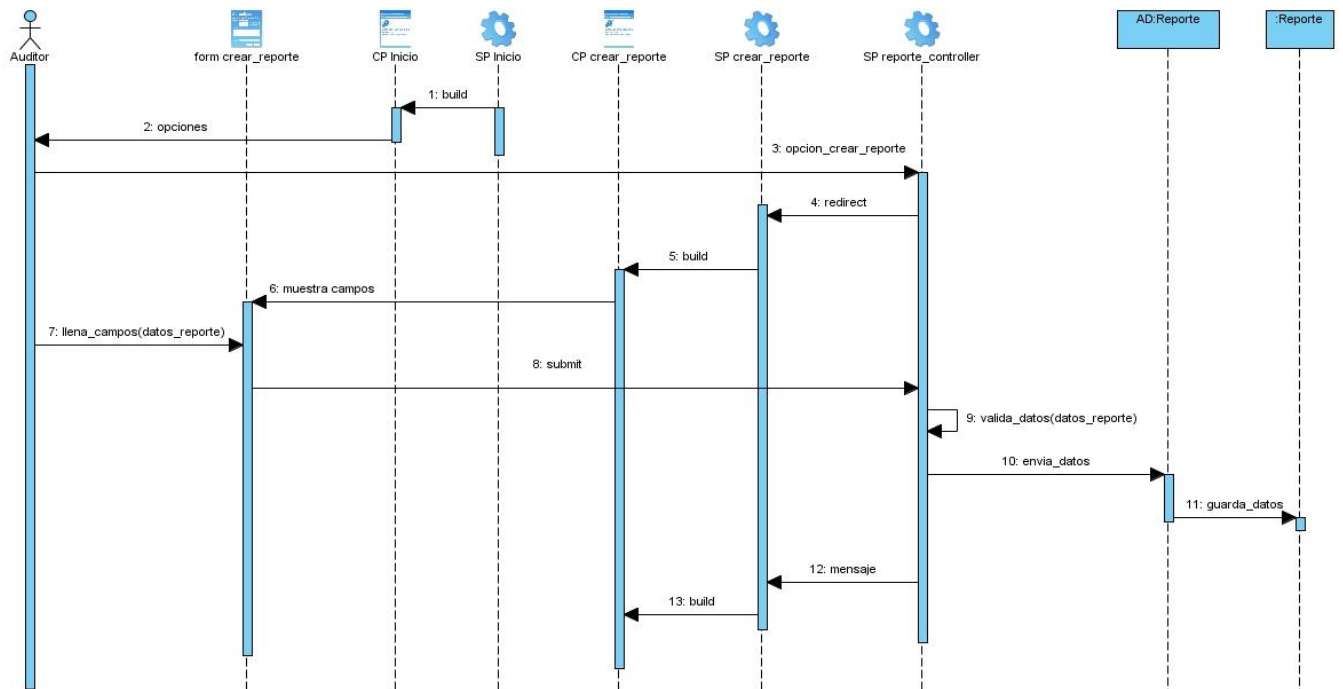


Figura 35: Diagrama de Secuencia Caso de Uso Gestionar Reporte. Escenario Crear Reporte.

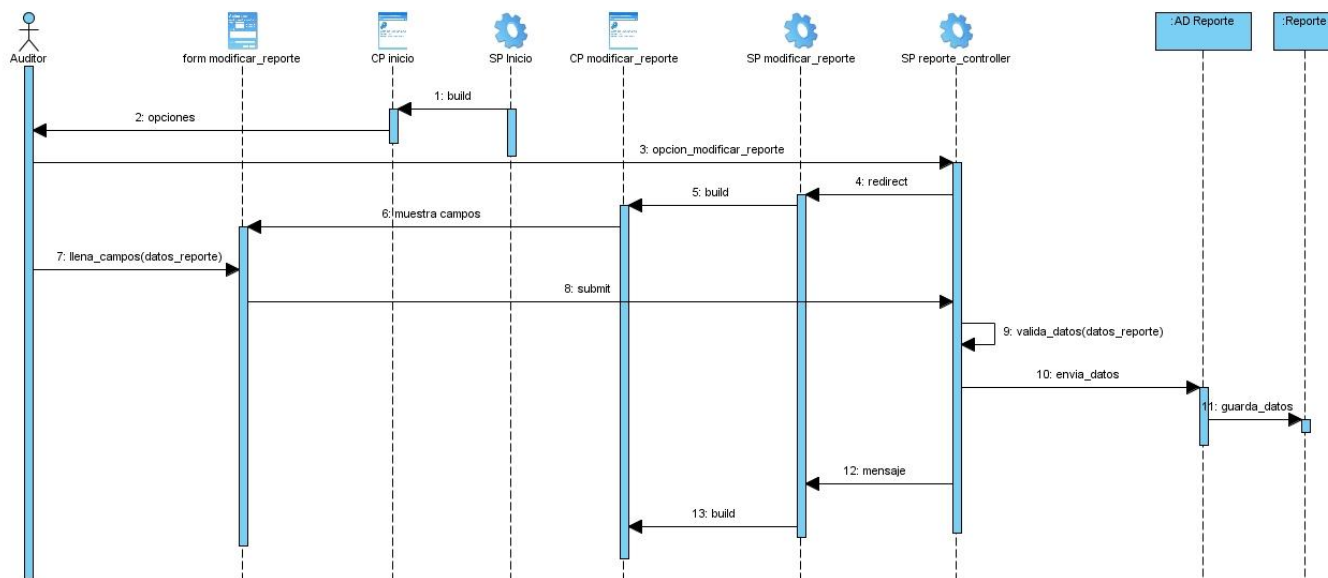


Figura 36: Diagrama de Secuencia Caso de Uso Gestionar Reporte. Escenario Modificar Reporte.

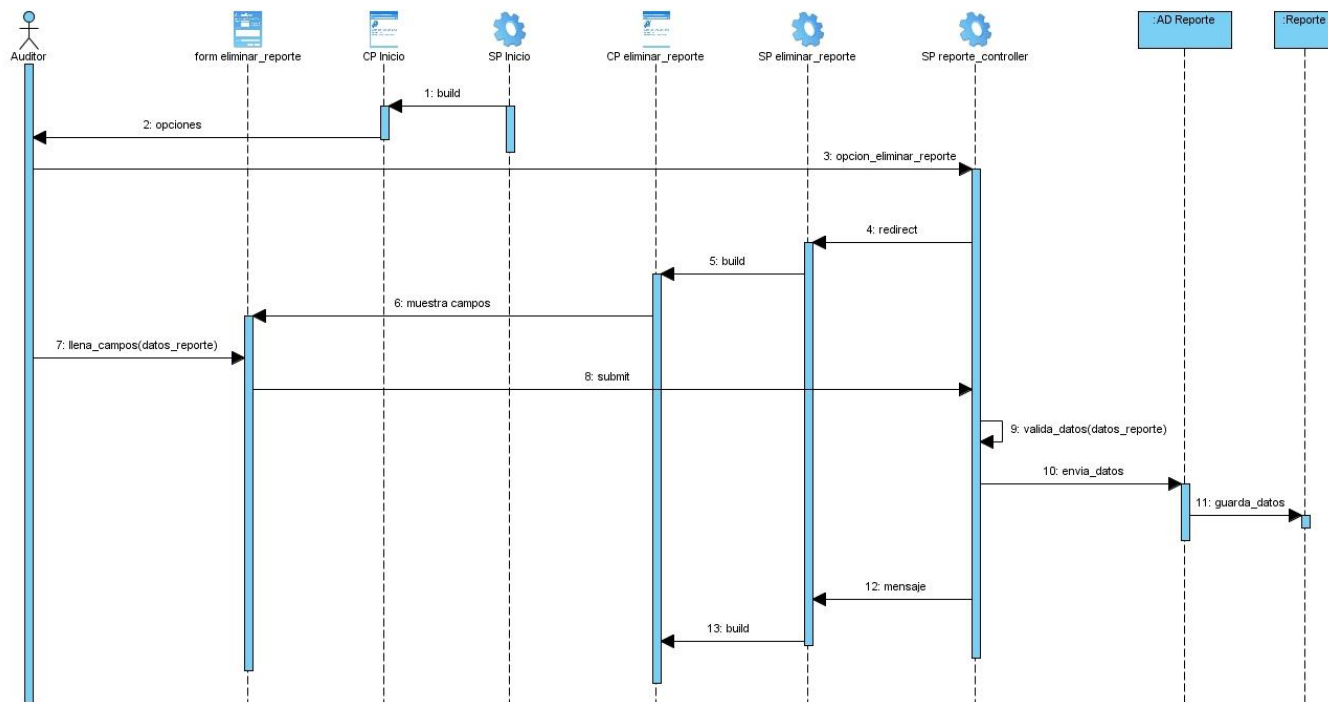


Figura 37: Diagrama de Secuencia Caso de Uso Gestionar Reporte. Escenario Eliminar Reporte.

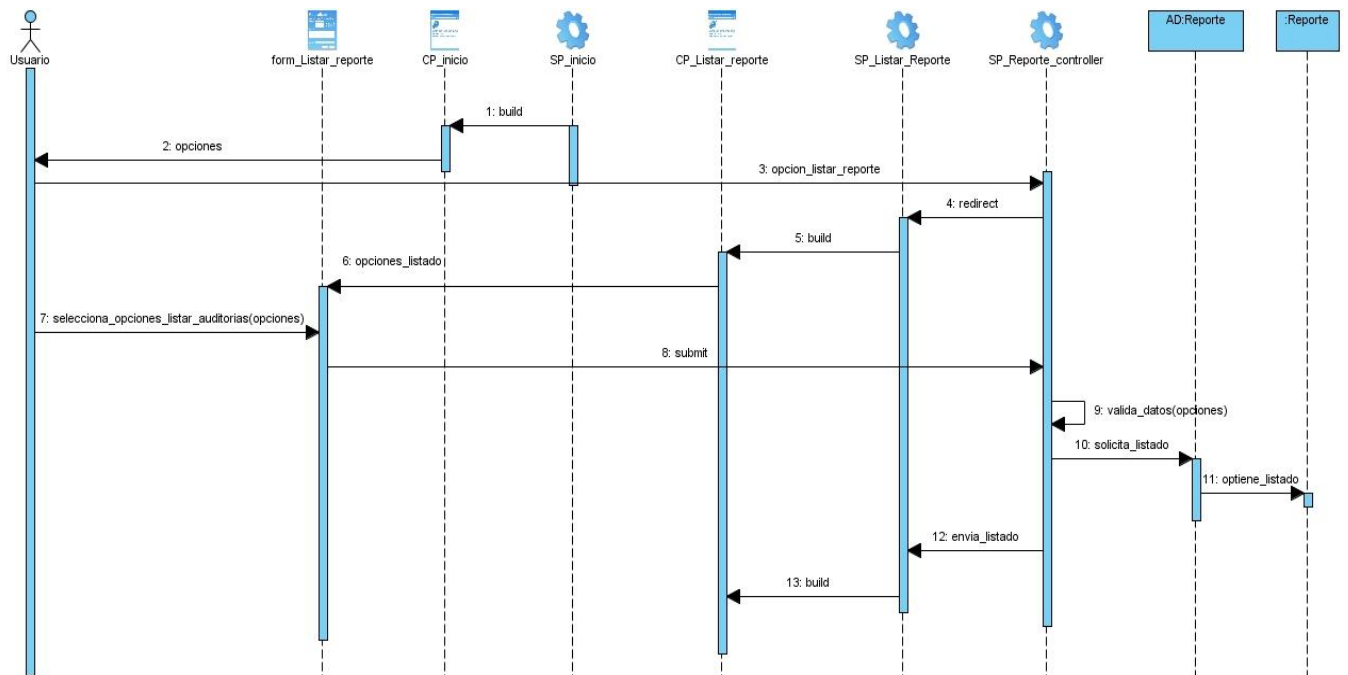


Figura 38: Diagrama de Secuencia Caso de Uso Listar Reporte.

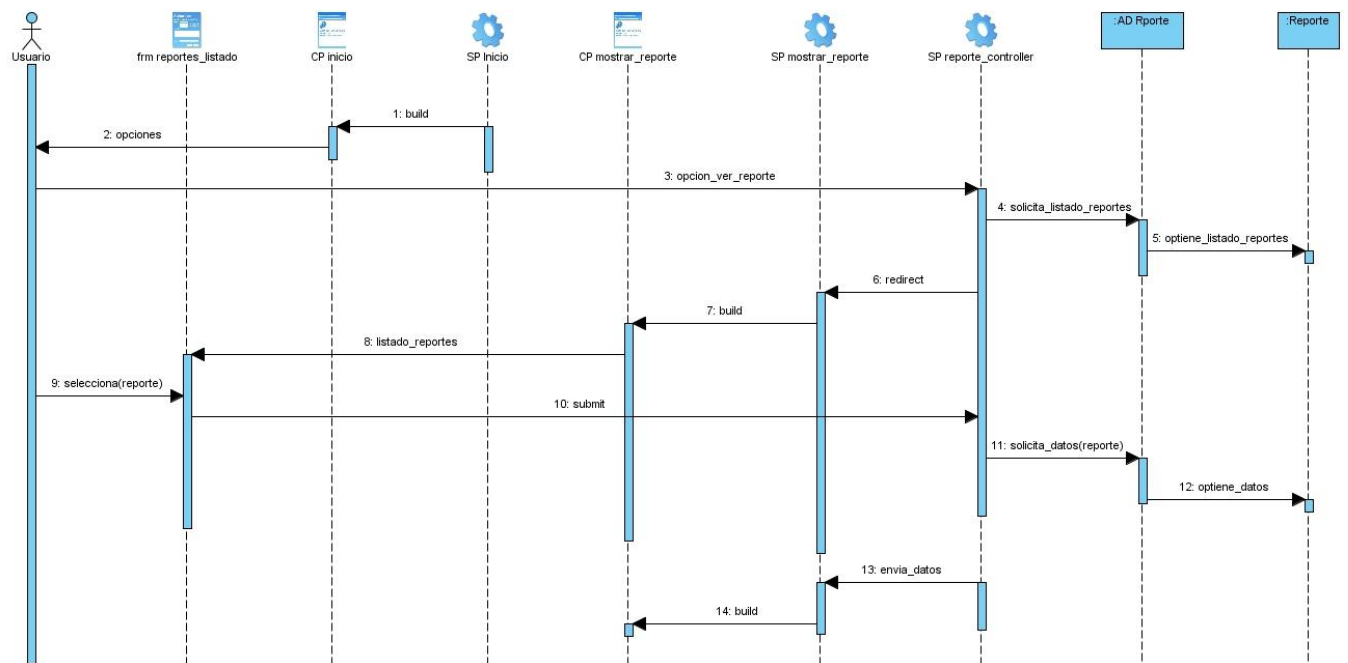


Figura 39: Diagrama de Secuencia Caso de Uso Mostar Reporte.

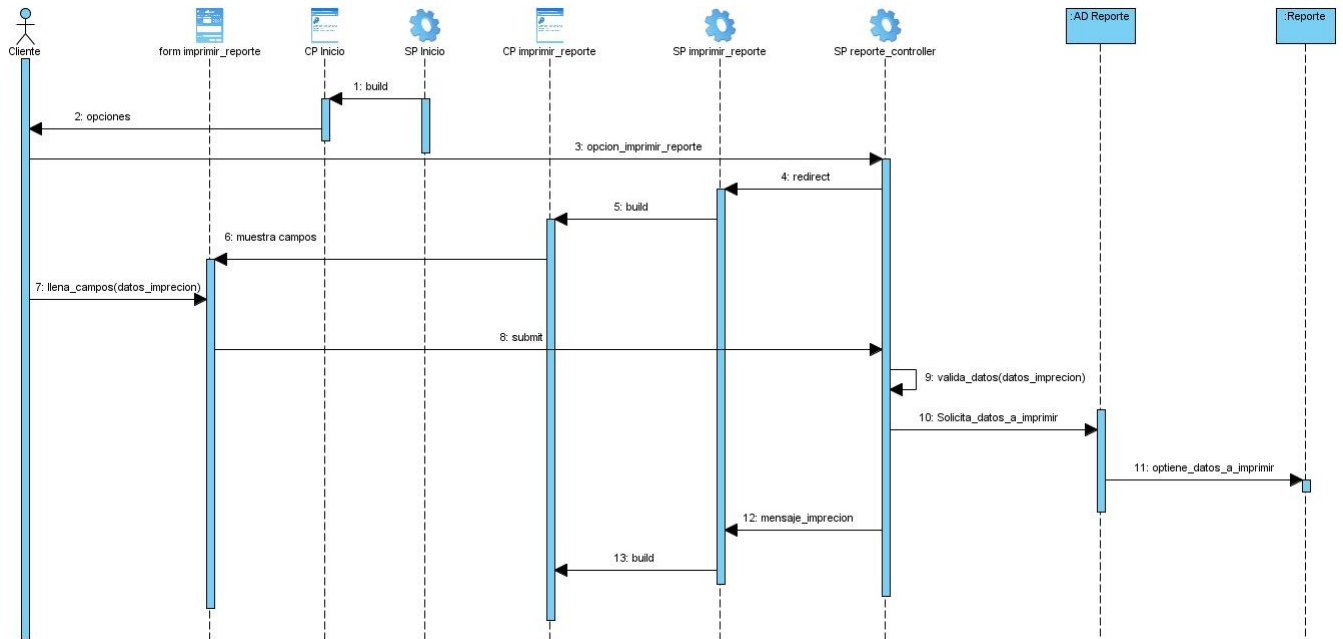


Figura 40: Diagrama de Secuencia Caso de Uso Imprimir Reporte.

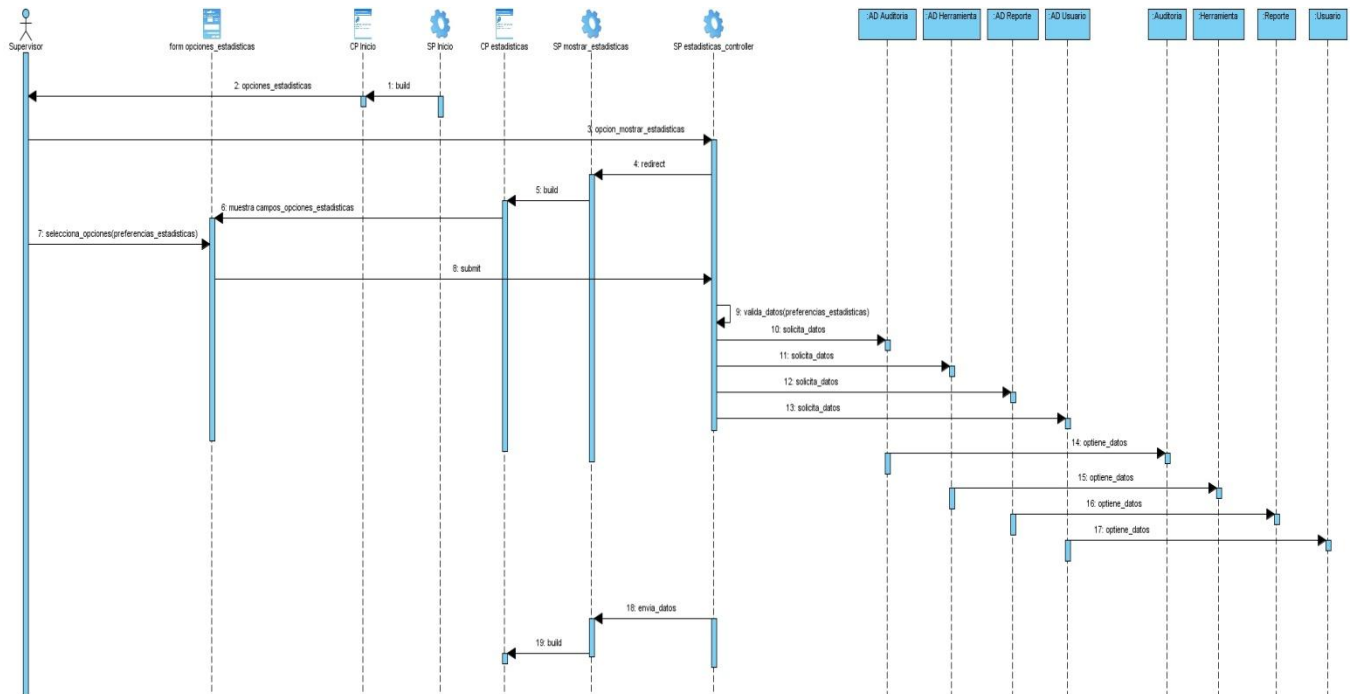


Figura 41: Diagrama de Secuencia Caso de Uso Mostrar Estadística.

Anexo 3: Prototipos de interfaz de usuario

The screenshot shows the top header of the application with the text "Sistema de Gestión de Reportes" and a logo for "LABSI". Below the header is a central login form titled "Acceso". The form contains two input fields: "Usuario" and "Contraseña", followed by an "Aceptar" button. At the bottom of the page, there is a footer with the text "UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS".

Figura 42: Prototipo Autenticar Usuario.

The screenshot displays the "Adicionar Auditoría" form within the "Sistema de Gestión de Reportes" application. On the left, there is a navigation menu for an "ADMINISTRADOR" role, with "Auditorías" selected. The main form area is titled "Adicionar Auditoría" and includes the following fields: "Sistema Cliente" (text input), "Fecha de Inicio" (calendar icon and text input), "Fecha de Fin" (calendar icon and text input), "Descripción:" (text area), and "Cliente" (dropdown menu with "seleccione un cliente" as the selected option). An "Aceptar" button is located at the bottom right of the form.

Figura 43: Prototipo Adicionar Auditoría.



**Sistema de Gestión de Reportes**

ADMINISTRADOR

- Auditorías
  - Adicionar**
  - Modificar
  - Eliminar
  - ADD-Clientes
  - Listar
  - Buscar
- Herramientas
- Reportes
- Usuarios
- Estadísticas

**Adicionar Herramientas**

Nombre de la Herramienta:

Descripción:

Aceptar

Figura 44: Prototipo Adicionar Herramienta.

**Sistema de Gestión de Reportes**

ADMINISTRADOR

- Auditorías
  - Adicionar**
  - Modificar
  - Eliminar
  - ADD-Clientes
  - Listar
  - Buscar
- Herramientas
- Reportes
- Usuarios
- Estadísticas

**Adicionar Reporte**

Fecha:

Auditoría:

Herramienta:

Descripción:

Archivo:  Browse...

Vulnerabilidades:

Aceptar

Figura 45: Prototipo Adicionar Reporte.

**Sistema de Gestión de Reportes**

ADMINISTRADOR

- Auditorias
- Herramientas
- Reportes
- Usuarios**
  - Adicionar
  - Modificar
  - Eliminar
  - Listar
  - Buscar
- Estadísticas

Adicionar usuarios

**Adicionar usuarios**

Usuario

Rol

Figura 46: Prototipo Adicionar Usuario.

**Sistema de Gestión de Reportes**

ADMINISTRADOR

- Auditorias**
  - Adicionar
  - Modificar
  - Eliminar
  - ADD Clientes
  - Listar
  - Buscar
- Herramientas
- Reportes
- Usuarios
- Estadísticas

Modificar Auditoria

Seleccione una Auditoria a modificar

Sistema Cliente

Fecha de Inicio

Fecha de Fin

Descripción:

Cliente

Figura 47: Prototipo Modificar Auditoría.

**Sistema de Gestión de Reportes**

ADMINISTRADOR

- Auditorías
  - Adicionar
  - Modificar
  - Eliminar
  - ADD-Clientes
  - Listar
  - Buscar
- Herramientas
- Reportes
- Usuarios
- Estadísticas

**Modificar Herramientas**

Nombre de la Herramienta: --Seleccione la Herramienta a Modificar--

Descripción:

Aceptar

Figura 48: Prototipo Modificar Herramienta.

**Sistema de Gestión de Reportes**

ADMINISTRADOR

- Auditorías
  - Adicionar
  - Modificar
  - Eliminar
  - ADD-Clientes
  - Listar
  - Buscar
- Herramientas
- Reportes
- Usuarios
- Estadísticas

**Modificar Reporte**

seleccione un reporte

Fecha:

Auditoría: selec--auditorias--

Herramienta: --Herramientas--

Descripción:

Archivo:  Browse...

Vulnerabilidades:

Aceptar

Figura 49: Prototipo Modificar Reporte.

**Sistema de Gestión de Reportes**

ADMINISTRADOR

- Auditorías
- Herramientas
- Reportes
- Usuarios
- Estadísticas

Adicionar  
 Modificar  
 Eliminar  
 Listar  
 Buscar

Modificar usuarios

**Modificar usuarios**

Usuario

Rol

Modificar

Figura 50: Prototipo Modificar Usuario.

**Sistema de Gestión de Reportes**

ADMINISTRADOR

- Auditorías
- Herramientas
- Reportes
- Usuarios
- Estadísticas

Adicionar  
 Modificar  
 Eliminar  
 ADD-Clientes  
 Listar  
 Buscar

Eliminar Auditoría

**Eliminar Auditoría**

Seleccione una Auditoría a ELIMINAR

Aceptar

Figura 51: Prototipo Eliminar Auditoría.



Figura 52: Prototipo Eliminar Herramienta.



Figura 53: Prototipo Eliminar Reporte.

**Sistema de Gestión de Reportes**

ADMINISTRADOR

- Auditorías
- Herramientas
- Reportes
- Usuarios**
  - Adicionar
  - Modificar
  - Eliminar
  - Listar
  - Buscar
- Estadísticas

Eliminar usuarios

**Eliminar usuarios**

Usuario:

Figura 54: Prototipo Eliminar Usuario.

**Sistema de Gestión de Reportes**

ADMINISTRADOR

- Auditorías**
  - Adicionar
  - Modificar
  - Eliminar
  - ADD-Cientes
  - Listar
  - Buscar
- Herramientas
- Reportes
- Usuarios
- Estadísticas

Buscar Auditoría

**Buscar Auditoría**

Seleccione una Auditoría

selecione un cliente

Figura 55: Prototipo Buscar Auditoría.

**Sistema de Gestión de Reportes**

ADMINISTRADOR

- Auditorías
  - Adicionar
  - Modificar
  - Eliminar
  - ADD-Clientes
  - Listar
  - Buscar
- Herramientas
- Reportes
- Usuarios
- Estadísticas

Buscar Herramienta

**Buscar Herramienta**

--Seleccione la Herramienta--

Aceptar

Figura 56: Prototipo Buscar Herramienta.

**Sistema de Gestión de Reportes**

ADMINISTRADOR

- Auditorías
  - Adicionar
  - Modificar
  - Eliminar
  - ADD-Clientes
  - Listar
  - Buscar
- Herramientas
- Reportes
- Usuarios
- Estadísticas

Buscar Reporte

**Buscar Reporte**

seleccione un reporte

Seleccione una Auditoría

seleccione un auditor

seleccione una herramienta

Aceptar

Figura 57: Prototipo Buscar Reporte.



Figura 58: Prototipo Buscar Usuario.



Figura 59: Prototipo Listar Auditoría.



*Sistema de Gestión de Reportes*



ADMINISTRADOR

- Auditorías
- Herramientas
- Reportes
- Usuarios
- Estadísticas

Nombre de la Herramienta	Detalles
WebScarb	<a href="#">Ver</a>
NmaP	<a href="#">Ver</a>
xx	<a href="#">Ver</a>
ff	<a href="#">Ver</a>
gg	<a href="#">Ver</a>
qq	<a href="#">Ver</a>

[Página:] 1 [Total de páginas:] 1 [Elementos:] 6

Figura 60: Prototipo Listar Herramienta.

*Sistema de Gestión de Reportes*



ADMINISTRADOR

- Auditorías
- Herramientas
- Reportes
- Usuarios
- Estadísticas

Fecha	# Vuln.	Auditor	Sistema	Fichero	Detalles
01-05-2010	3	harencibia	ERP	<a href="#">Archivo</a>	<a href="#">Ver</a>
04-05-2010	5	ypantin	HHHH	<a href="#">Archivo</a>	<a href="#">Ver</a>
05-05-2010	5	ypantin	Siglab	<a href="#">Archivo</a>	<a href="#">Ver</a>
05-05-2010	2	ypantin	xx	<a href="#">Archivo</a>	<a href="#">Ver</a>
14-05-2010	4	harencibia	Siglab	<a href="#">Archivo</a>	<a href="#">Ver</a>
16-05-2010	5	ypantin	HHHH	<a href="#">Archivo</a>	<a href="#">Ver</a>

[Página:] 1 [Total de páginas:] 2 [Elementos:] 6

[Siguiente](#)

Figura 61: Prototipo Listar Reporte.



Figura 62: Prototipo Listar Usuario.



Figura 63: Prototipo Ver Estadísticas.

## **GLOSARIO DE TÉRMINOS**

### **A**

**API:** Application Programming Interface, en español interfaz de programación de aplicaciones, se define como el conjunto de procedimientos y funciones que ofrecen una biblioteca para ser utilizada por otra aplicación como una capa de abstracción. Son usados generalmente en las bibliotecas.

**Aplicación:** Programa preparado para una utilización específica, como el pago de nóminas, formación de un banco de términos léxicos, etc.

**Aplicación WEB:** Es una aplicación de software que se codifica en un lenguaje soportado por los navegadores Web.

**Ataques informáticos:** Acción de atacar, perjudicar o destruir, mediante un sistema informático, a una red, una computadora, etc.

### **B**

**Brechas de seguridad:** Vulnerabilidad presente que deja la cobertura a posibles ataques.

### **C**

**CSS:** Cascading Style Sheets, en español, hojas de estilo en cascada, se define como un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML.

**COCOMO:** Constructive Cost Model, en español, Modelo constructivo de costes. Se considera un modelo matemático que se utiliza para la estimación de costes de proyectos de software.

### **F**

**Framework:** Generadores de Aplicación que se relacionan directamente con un dominio específico, es decir, con una familia de problemas relacionados. Además tienen la capacidad para promover la reutilización del código del diseño y el código fuente.

### **G**

**Gestor de base de datos (SGBD):** Es un conjunto de programas que permiten crear y mantener una Base de datos, asegurando su integridad, confidencialidad y seguridad.

### **H**

**Herramienta Case:** CASE (Computer Aided Software Engineering). Conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software.

**HTML:** Hyper Text Markup Language, en español, Lenguaje de marcación de Hipertexto, se puede definir como el lenguaje que se utiliza para la elaboración de páginas Web.

**HTTP:** Hypertext Transfer Protocol, en español, protocolo de transferencia de hipertexto. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor.

**HTTPS:** Hypertext Transfer Protocol Secure, en español Protocolo Seguro de Transferencia de Hipertexto. Es la versión de HTTP, pero segura, es un protocolo de red basado en el protocolo HTTP, destinado a la transferencia segura de datos.

**Hardware:** Corresponde a todas las partes físicas y tangibles de una computadora: sus componentes eléctricos, electrónicos, electromecánicos y mecánicos; sus cables, gabinetes o cajas, periféricos de todo tipo y cualquier otro elemento físico involucrado.

### **I**

**IEEE:** Instituto de Ingenieros Electricistas y Electrónicos, se definen la asociación profesional más grande del mundo dedicada al avance de la innovación tecnológica y excelencia en beneficio de la humanidad.

### **K**

**KumbiaPHP:** Es un framework para aplicaciones Web libre escrito en PHP5. Basado en las prácticas de desarrollo Web para software comercial y educativo. Kumbiaphp fomenta la velocidad y eficiencia en la

creación y mantenimiento de aplicaciones Web, reemplazando tareas de codificación repetitivas por poder, control y placer.

### **L**

**LABSI:** Laboratorio de Seguridad Informática

**Livecd:** Es un sistema operativo que normalmente contiene un conjunto de aplicaciones almacenado en un medio extraíble, que puede ser un CD o un DVD, que puede ejecutarse desde éste sin necesidad de instalarlo en el disco duro de una computadora, para lo cual usa la memoria RAM como disco duro virtual y el propio medio como sistema de archivos.

### **M**

**MVC:** Modelo Vista Controlador es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos

### **P**

**PostgresSQL:** Es un sistema de gestión de base de datos relacional orientada a objetos y libre.

**PHP:** Es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas Web dinámicas.

### **R**

**RUP:** Rational Unified Process, en español, Proceso Unificado de Rational, es el proceso de desarrollo del un software, que utiliza UML como lenguaje de modelado.

**Ruby on Rails:** Es un framework o un entorno de desarrollo Web de código abierto.

**Requisitos funcionales:** Son las capacidades o condiciones que un sistema determinado debe cumplir.

**Requisitos no funcionales:** Son propiedades o cualidades que el producto debe tener.

### **S**

**Sistemas Web:** Complejos sistemas con interfaces de usuario cada vez más parecidas a las aplicaciones de escritorio, que dan servicio a procesos de negocio de considerable envergadura y estableciéndose sobre ellas requisitos estrictos de accesibilidad y respuesta.

**Software:** Se refiere al equipamiento lógico o soporte lógico de una computadora digital, y comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de tareas específicas.

**Sistema Operativo:** Es el software básico de una computadora que provee una interfaz entre el resto de programas del ordenador, los dispositivos hardware y el usuario. Las funciones básicas del Sistema Operativo son administrar los recursos de la máquina, coordinar el hardware y organizar archivos y directorios en dispositivos de almacenamiento.

### **U**

**UML:** Unified Modeling Language. El Lenguaje Unificado de Modelado prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real.