



**ESCUELA SUPERIOR POLITECNICA DE CHIMBORAZO
FACULTAD DE INFORMATICA Y ELECTRONICA
ESCUELA DE INGENIERIA EN SISTEMAS**

**“ANÁLISIS DE COMPONENTES DE SEGURIDAD A NIVEL DE INTERFAZ DE
USUARIO EN JSF PRIMEMOBILE PARA DESARROLLO DE APLICACIONES
MÓVILES, CASO PRÁCTICO DEPARTAMENTO FINANCIERO-ESPOCH.”**

TESIS DE GRADO

**Previa a la obtención del título de:
INGENIERIA EN SISTEMAS INFORMATICOS**

**Presentado por:
JENNY ALICIA PEREZ ZAVALA
GUSTAVO EFRAIN PARCO SANCHEZ**

RIOBAMBA- ECUADOR

2013

AGRADECIMIENTO

Agradezco a mi Jesús Amado ya que sin él nada de este proceso importante de mi vida podría haberse realizado, a mis padres por su trabajo y su sacrificio diario para que pueda estudiar, a mi hermano por apoyarme y motivarme, a mi gran amiga y a Efraín por compartir bellos momentos que se graban en el corazón.

Jenny Perez Zavala

Agradezco a Dios por concederme la vida, a mis padres por su amor, apoyo incondicional y sus consejos para que sea una persona de bien, a mis hermanos por motivarme a crecer personal e intelectualmente, y toda mi familia por ser ayuda idónea en mi vida, y a Jenny por brindarme su apoyo sin condiciones.

Gustavo Parco Sanchez

DEDICATORIA

A Dios por guiar mi vida en todo momento, a mis padres porque sus consejos han sido vitales en mi diario vivir, a mi hermano por su amor y compañía diario, a Elisa por ser una mujer valiente y enseñarme a vivir, a Lulú por su amistad y complicidades compartidas.

Jenny Perez Zavala

A Dios por ayudarme siempre, a mis padres por sus valores y enseñanzas, a mis hermanos por ser base fundamental en mi vida, a mis profesores por ser una guía durante toda la etapa de formación universitaria.

Gustavo Parco Sanchez

NOMBRE	FIRMAS	FECHA
ING. IVÁN MENES CAMEJO DECANO DE LA FACULTAD DE INFORMÁTICA Y ELECTRÓNICA
ING. RAÚL ROSERO DIRECTOR DE LA ESCUELA DE INGENIERÍA EN SISTEMAS
ING. GLORIA ARCOS MEDINA DIRECTORA DE TESIS
ING. IVONNE RODRIGUEZ FLORES MIEMBRO DE TESIS
TLGO. CARLOS RODRÍGUEZ DIRECTOR DEL CENTRO DE DOCUMENTACIÓN
NOTA DE LA TESIS	

RESPONSABILIDAD DEL AUTOR

“Nosotros, Jenny Perez Zavala y Gustavo Parco Sanchez, somos los responsables de las ideas, doctrinas y resultados expuestos en esta Tesis de Grado, y el patrimonio intelectual de la misma pertenece a la Escuela Superior Politécnica de Chimborazo.”

Jenny Perez Zavala

Gustavo Parco Sanchez

ÍNDICE DE ABREVIATURAS

A

API: Application Programming Interface

C

CSRF: Cross-Site Request Forgery

D

DOM: Document Object Model

E

ESAPI: Enterprise Security API

J

JCA: Java Cryptography Architecture

JPA: Java Persistence API

JSF: Java Server Faces

JSP: Java server Pages

M

MD: Message-Digest Algorithm

O

ORM: Object Relational Mapping

OWASP: Open Web Application Security Project

R

RAD: Rapid Application Development

S

SCMTPM: Sistema de Consultas y Monitoreo de Trámites de Pago Móvil

SHA: Secure Hash Algorithm

SOAP: Simple Object Access Protocol

W

WML: Wireless Markup Language

WSDL: Web Service Description Language

X

XHTML: Extensible HiperText Markup Language

XML: Extensible Markup Language

XP: Xtreme Programming

XSS: Cross-site Scripting

INDICE GENERAL

CAPITULO I.....	22
MARCO REFERENCIAL	22
1.1 Antecedentes	22
1.2. Justificación Teórica.....	24
1.3. Justificación Práctica.....	25
1.4. Ámbito de la Solución	27
1.5. Objetivos.....	28
1.5.1. Objetivo General.....	28
1.5.2. Objetivos Específicos.....	28
1.6. Marco hipotético	28
1.6.1. Hipótesis.....	28
CAPITULO II.....	30
CONCEPTOS GENERALES	30
1.1. Introducción.....	30
2.2. Servicios Web Java	31
2.3. Dispositivos Móviles	31
2.4. Sistemas Operativos para Móviles	32
2.5. Principios generales sobre seguridad.....	32

2.6.	Criptografía	33
2.6.1.	Criptografía Java	33
2.6.1.1.	Java Cryptography Architecture (JCA)	34
2.7.	Seguridad en las aplicaciones móviles	35
2.8.	Seguridad en aplicaciones web Java	36
2.9.	Fundamentos de Java Server Faces	37
2.9.1.	Ventajas de Java Server Faces.....	37
2.10.	Prime Faces.....	39
2.10.1.	Propiedades	39
2.10.2.	Pros y contras	40
2.11.	Prime Faces Mobile.....	40
2.12.	OWASP – Open Web Application Security Project	40
2.12.1.	Las vulnerabilidades más comunes de OWASP.....	41
2.12.2.	Descripción de las Vulnerabilidades.....	41
2.12.2.1.	Fallas de Inyección.....	41
2.12.2.1.1.	Protección y tratamiento para Fallas de Inyección	43
2.12.2.2.	Cross- Site scripting (XSS).....	44
2.12.2.2.1.	Protección y tratamiento contra Cross Site Scripting (XSS)	46
2.12.2.3.	Falsificación de Peticiones en Sitios Cruzados	47

2.12.2.3.1.	Protección contra Falsificación de Peticiones en Sitios Cruzados	48
2.12.2.4.	Exposición de Datos Sensibles.....	49
2.12.2.4.1.	Protección para Exposición de Datos Sensibles	51
2.12.3.	OWASP ESAPI (Enterprise Security API)	51
2.12.4.	OWASP y Java Server Faces	52
2.13.	Descripción de Componentes de Seguridad de JSF.....	54
2.13.1.	Sistema de Conversión de JSF.....	54
2.13.2.	Sistema de Validación de JSF	58
2.13.3.	Filtrado de salida con componentes JSF	59
2.14.	Prueba para filtrar la salida y escape de caracteres peligrosos con componentes JSF	60
2.15.	Seguridad de JSF para CSRF	63
2.16.	Exposición de Vulnerabilidades en JSF + PrimeFaces Mobile	63
2.16.1.	Exposición de vulnerabilidad en JSF + PrimeFaces Mobile ante XSS	63
2.16.2.	Exposición de vulnerabilidad en JSF + PrimeFaces Mobile ante Sql Injection	65
2.16.3.	Exposición de Datos de Datos Sensibles.....	66
2.17.	Creación de componentes de entrada personalizados con seguridad para evitar Sql Injection, XSS	67
2.17.1.	El Componente Personalizado	68
2.17.1.1.	Renderer	69

2.17.1.2.	Personalización del archivo TLD	70
2.17.1.3.	Configuración del archivo web.xml	70
2.18.	Incorporando seguridad al componente personalizado para evitar posibles ataques de Sql Injection y XSS	71
2.19.	Creación de componentes personalizados para evitar la falta de confidencialidad de datos	72
2.20.	Pruebas de Vulnerabilidad a los componentes creados	73
2.20.1.	Pruebas de vulnerabilidad a los componentes creados para evitar XSS	73
2.20.2.	Pruebas de vulnerabilidad a los componentes creados para evitar Sql Injection.....	73
2.20.3.	Pruebas de vulnerabilidad a los componentes creados para evitar la exposición de datos sensibles	75
CAPITULO III.....		76
MATERIALES Y MÉTODOS		76
3.1.	Diseño de la Investigación.....	76
3.2.	Métodos	76
3.3.	Técnicas	77
3.4.	Planteamiento de la hipótesis.....	77
3.5.	Variables	77
3.6.	Operacionalización de Variables	78
3.6.1.	Operacionalización Conceptual.....	78

3.6.2.	Operacionalización Metodológica	78
3.7.	Población y Muestra	79
3.8.	Instrumentos de Recolección de Datos	80
3.9.	Ambientes de Prueba	82
3.9.1.	Prototipos de Prueba	83
CAPITULO IV		85
ANÁLISIS DE RESULTADOS		85
3.10.	Variable Independiente	88
3.11.	Variable Dependiente	88
3.12.	Resumen General de Equivalencias.....	95
3.13.	Prueba de Hipótesis	97
CAPITULO V		99
IMPLEMENTACIÓN DEL SISTEMA DE CONSULTAS Y MONITOREO DE PROCESOS DE TRÁMITE DE PAGO PARA EL DEPARTAMENTO FINANCIERO (ESPOCH).....		99
4.1.	Metodología XP (Xtreme Programming)	99
4.1.1.	Fase de Planificación	100
4.1.1.1.	Planificación Inicial	100
4.1.1.2.	Historias de Usuario.....	101
4.1.1.3.	Planificación de Publicaciones:	103
4.1.1.3.1.	Iteraciones	103

4.1.2.	Fase de Diseño	108
4.1.2.1.	Descripción del Diseño de Base de Datos	108
4.1.2.2.	Diseño de Interfaces	110
4.1.2.3.	Diagramas de Caso de Uso	111
4.1.2.4.	Diagrama de Procesos	114
4.1.2.5.	Arquitectura	116
4.1.2.6.	Tareas de las Historias de Usuario	116
4.1.3.	Fase de Codificación	117
4.1.3.1.	Paquetes	117
4.1.3.2.	Diagrama de Despliegue.....	118
4.1.3.3.	Diagrama de Componentes	118
4.1.4.	Fase de Pruebas	119
4.1.4.1.	Pruebas Funcionales	120
4.1.5.	Pruebas de Stress y Rendimiento.....	121
	CONCLUSIONES	129
	RECOMENDACIONES.....	130
	RESUMEN.....	131
	SUMMARY	132
	GLOSARIO DE TERMINOS	133

ANEXOS 135

BIBLIOGRAFIA..... 147

ÍNDICE DE FIGURAS

Figural.1: Ámbito de la solución.....	27
Figura II.1: Ciclo de Vida de un componente JSF	38
Figura II.2: Fallas de Inyección [13].....	41
Figura II.3: Secuencia de Comandos en Sitios Cruzados [15]	44
Figura II.4: Falsificación de Peticiones en Sitios Cruzados	47
Figura II.5: Exposición de Datos Sensibles	50
Figura II.6: Métodos de Interfaz javax.faces.convert.Converter.	54
Figura II.7: Uso de etiqueta “<f:convertDateTime>”.....	56
Figura II.8: Uso de etiqueta “<f:convertNumber>”	56
Figura II.9: uso de etiqueta “<f:converter>”	57
Figura II.10: Anidación de converter en la etiqueta	57
Figura II.11. Ingreso de script Malicioso.	61
Figura II.12. Resultados del script ingresado.....	61
Figura II.13 Desactivación del filtro de escape.	62
Figura II.14. Ejecución del script por la desactivación del filtro de escape.....	62
Figura II.15: Visualización del “javax.faces.ViewState” y el código generado.....	63
Figura II.16. Visualización del Script inyectado	64
Figura II.17. Prueba en la aplicación con el plugin SQL Inject Me	65
Figura II.18. Sql Inyección que han pasado y son peligrosos para las demás capas.....	66
Figura II.19. Deducción de tabla con una simple revisión del código fuente xhtml.	67
Figura II.20: Componente personalizado.....	71
Figura II.21: componentes creados para evitar XSS	73
Figura II.22: Ingreso de inyecciones SQL.....	74
Figura II.23: Ejecución de inyecciones SQL	74
Figura II.24: Inspección del código HTML	75

Figura III.1: Scanner Wapiti.....	80
Figura III.2: Herramienta acunetix WVS.....	80
Figura III.3: Herramienta XSS ME.....	81
Figura III.4: Herramienta SQL Inject ME.....	81
Figura III.5: Herramienta CSRFTester.....	81
Figura III.6: Prototipo de Prueba 1.....	83
Figura III.7: Prototipo de Prueba 2.....	84
Figura IV.1: Escaneo de vulnerabilidades con software Acunetix.....	89
Figura IV.2: Comparación de la Confidencialidad.....	91
Figura IV.3: Escaneo de vulnerabilidad con XSS Me.....	91
Figura IV.4: Comparación de la Disponibilidad.....	93
Figura IV.5: Escaneo de vulnerabilidades con Wapiti.....	93
Figura IV.6: Comparación de la Integridad.....	95
Figura IV.7 Síntesis de Indicadores.....	96
Figura IV.8: Comparación de la Seguridad.....	98
Figura V.1: Fases de la Metodología XP.....	100
Figura V.2: Plan de entrega Iteración 1.....	104
Figura V.3: Plan de entrega Iteración 2.....	105
Figura V.4: Plan de entrega Iteración 3.....	106
Figura V.5: Diseño de base de Datos “Financiero”.....	109
Figura V.6: Modelo de Datos para Autenticación.....	110
Figura V.7: Interfaz Gráfica del Menú principal.....	111
Figura V.8: Diagrama de caso de uso para el rol Usuario Titular.....	112
Figura V.9: Flujo de Procesos para realizar consultas con los Comprobantes de Pago.....	115
Figura V.10: Arquitectura de la aplicación web móvil.....	116
Figura V.11: Diagrama de despliegue.....	118
Figura V.12: Diagrama de Componentes.....	119

Figura V.13: Captura de variables con la herramienta badBoy	122
Figura V.14: Propiedades de Hilos en JMeter	123
Figura V.15: Resultados obtenidos en JMeter.....	125
Figura V.16: Grafico de Resultados	126
Figura V.17: Árbol de Resultados en JMeter	127
Figura V.18: Respuesta exitosa	128
Figura V.19: Respuesta fallida	128

ÍNDICE DE TABLAS

Tabla II.1 Descripción de Fallas de Inyección	42
Tabla II.2: Descripción de Secuencia de Comandos en Sitios Cruzados.....	44
Tabla II.3: Descripción de Falsificación de Peticiones en Sitios Cruzados	47
Tabla II.4: Descripción de Exposición de Datos Sensibles	50
Tabla II.5: Convertidores estándares	55
Tabla II.6: Validadores estándares en JSF	58
Tabla III.1: Operacionalización Conceptual	78
Tabla III.2: Operacionalización Metodológica	78
Tabla III.3: Scanners de Vulnerabilidades para aplicaciones web	82
Tabla III.4: Hardware utilizado para las pruebas	82
Tabla III.5: Software utilizado para las pruebas	82
Tabla IV.1: Ponderación de la Severidad.....	85
Tabla IV.2: Distribución del Porcentaje del Nivel de Seguridad	86
Tabla IV.3: Cuadro guía para el escaneo de vulnerabilidades	87
Tabla IV.4: Escaneo de vulnerabilidades de Confidencialidad	89
Tabla IV.5. Aplicación de fórmula 2 en Confidencialidad.....	90
Tabla IV.6: Resumen de escaneo de vulnerabilidades de Disponibilidad	92
Tabla IV.7: Aplicación de fórmula 3 para Disponibilidad.....	92
Tabla IV.8: Resumen de escaneo de vulnerabilidades de Integridad	94
Tabla IV.9: Aplicación de fórmula 3 para Integridad	94
Tabla IV.10: Síntesis general de porcentajes	96
Tabla IV.11: Valoración del Nivel de Seguridad	97
Tabla V.1: Equipo de Trabajo	100
Tabla V.2: Historias de usuario	101
Tabla V.3: Plan de entrega iteración 1	103
Tabla V.4: Plan de entrega Iteración 2.....	104

Tabla V.5: Plan de entrega Iteración 3.....	106
Tabla V.6: Historia de usuario Consulta Datos Personales	107
Tabla V.7: Historia de usuario Comprobante de pago.....	107
Tabla V.8: Historia de usuario Notificaciones de Comprobantes por Cobrar	108
Tabla V.9: Caso de uso Detallado Autenticación de Usuarios	113
Tabla V.10: Tarea de implementación de clases, funciones, controladores	116

INTRODUCCIÓN

El mundo tecnológico se ha ido transformando desde sus inicios tanto el hardware como el software, día a día se observa que los usuarios se van acoplando a esta transformación y es que ellos son los autores intelectuales de este proceso.

Hemos sido testigos de cómo poco a poco se va cambiando la predilección de un habitual computador de escritorio a dispositivos móviles como: Smartphone, iPhone, Tablet, etc. y en el uso cotidiano de estos, muchas de los usuarios los utilizan ya que permiten la fácil conexión en redes móviles y la permanente conexión a internet.

A menudo se observa como por medio de estos dispositivos se trabaja con información de gran magnitud de confidencialidad como: consultas bancarias, transacciones, etc., pero en los últimos tiempos se ha visto como el robo de información mediante el uso de la internet ha ido creciendo, las vulnerabilidades en las aplicaciones han hecho parte de ello debido a un defecto de diseño o errores en las aplicaciones, los cuales han sido perjudiciales causando daños como: sustracciones de identidades, desfalcos, etc.

Los desarrolladores de software también hacen parte de todo este proceso de transformación ya que al desarrollar aplicaciones ya sea para el desktop o la web es de gran responsabilidad proporcionar seguridad para preservar la información, evitando que usuarios ajenos o con privilegios de usar las aplicaciones realicen acciones no permitidas y perjudiquen en parte o en su totalidad.

Y es así que en el Departamento Financiero lo más importante es que sus usuarios conozcan el proceso de sus trámites de pago los cuales deben ser tratados celosamente ya que poseen información monetaria así como cuenta bancaria todo ello de suma importancia, y no cualquier usuario debe tener acceso a ella sino tan solo la parte interesada.

Por tal razón es importante brindar seguridad a los usuarios protegiendo su información evitando que vulnerabilidades, ataques informáticos, errores de desarrollo perjudiquen a las aplicaciones y todos

los involucrados con la misma; no solo porque el usuario haga uso de dispositivos móviles sino más bien porque hoy por hoy lo más esencial de las empresas es la información con la que estas trabajan.

El presente trabajo de investigación se ha dividido en capítulos donde el primero contiene el anteproyecto de tesis el cual ha servido como referencia para iniciar en este trabajo; el segundo capítulo trata acerca de los conceptos generales de los términos que han sido de gran relevancia y poder aplicarlos en la creación de los componentes personalizados; el tercer capítulo trata acerca de los materiales y métodos empleados para la realización del trabajo de investigación como: identificación de variables, definición de indicadores, planteamiento de hipótesis, definición de herramientas software y hardware; el cuarto capítulo trata acerca del análisis de resultados obtenidos y la comprobación de la hipótesis del trabajo de investigación y finalmente el quinto capítulo trata acerca de la metodología utilizada para el desarrollo e implementación del sistema de Consulta y Monitoreo de Trámite de pago Móvil.

Para la comprobación de la hipótesis se ha utilizado el método científico mediante el cual se ha avalado este trabajo de investigación, mediante este se ha obtenido que al utilizar los JSF+ PrimeMobile y los componentes personalizados con la seguridad se ha obtenido un nivel de seguridad adecuado para la realización de consultas y la exposición de datos sensibles en el sistema de Consultas y Monitoreo de Tramites de Pago Móvil.

CAPITULO I

MARCO REFERENCIAL

1.1 Antecedentes

La predilección de los usuarios está cambiando del desktop con acceso a la Web al Web móvil. Se predice que en los próximos 5 años habrá más usuarios conectados a Internet a través de dispositivos móviles que con PCs tradicionales.

Es importante garantizar la seguridad al usuario al momento que se desenvuelve en una aplicación y sobremanera móvil. Desde el inicio de la tecnología Java, ha sido fuerte y creciente interés en torno a la seguridad de la plataforma Java, así como nuevos problemas de seguridad planteados por el despliegue de la tecnología Java como son:

- Secuencia de Comandos en Sitios Cruzados.
- Fallas de Inyección.
- Falsificación de Petición en Sitios Cruzados.
- Falta de Confidencialidad de los Datos.

Desde el punto de vista de un proveedor de tecnología la seguridad de Java incluye dos aspectos importantes: proporcionar la plataforma Java como un sistema seguro, y también proporcionar herramientas y servicios de seguridad implementadas en el lenguaje de programación Java que permiten una gama más amplia de la seguridad de las aplicaciones sensibles, por ejemplo, en el mundo empresarial.

Es así que la plataforma Java hace especial hincapié en la seguridad, incluida la seguridad de idioma, la criptografía, infraestructura de clave pública, la autenticación, la comunicación segura y control de acceso.

Además se ve el uso de frameworks como JSF que permite desarrollar rápidamente aplicaciones de negocio dinámicas en las que toda la lógica de negocio se implementa en java, o es llamada desde java, creando páginas para las vistas muy sencillas.

Todo esto permite utilizar Java Server Faces el cual es una tecnología y framework para aplicaciones Java basados en web que simplifica el desarrollo de interfaces de usuario en aplicaciones JEE, ya que este provee un conjunto de componentes de interfaz de usuario predefinidos listos para usar; pero como Faces provee pocos mecanismos nativos para dotar de seguridad a las aplicaciones web, se verá la forma de usar estos mecanismos nativos y complementarlos mediante la personalización de componentes los cuales sean seguros, de tal forma q estos satisfagan las necesidades de este trabajo de investigación. También se hará uso de componentes como PrimeFaces Mobile que no es más que un kit de interfaz de usuario para implementar páginas JSF que están optimizadas para dispositivos móviles con un aspecto y comportamiento nativo que cuenta con las siguientes características:

- Mobile RenderKit para el estándar JSF y componentes básicos PrimeFaces.
- JSF componentes móviles.
- El mismo modelo de servidor para web de escritorio y la web móvil.
- Amplia gama de soporte de plataforma.

- Integración con el modelo de programación de JSF.
- Cuenta con Ajax para llevar la experiencia de las aplicaciones nativas.
- Nada de lo necesario para instalar en el dispositivo.
- Desarrollado por jQuery Mobile.
- Soporta diversas plataformas, tales como iPhone, Android, Windows Mobile.

El Departamento Financiero de la ESPOCH no cuenta en la actualidad con un sistema web móvil por medio del cual los clientes puedan averiguar el proceso de los trámites que realizan, es decir no hay consultas automáticas y la exposición de información para conocer el estado en el cual se encuentra el trámite. Cada trámite que se lleva a cabo se efectúa tradicionalmente presentando la documentación necesaria, cada uno de estos trámites pasa por diferentes departamentos, y si el cliente desea conocer el estado de su trámite debe acercarse físicamente al departamento.

Mediante el uso de las herramientas de seguridad que nos proporciona Java y JSF, el libre uso de componentes como PrimeFaces Mobile orientado a dispositivos móviles, el Departamento Financiero de la ESPOCH obtendrá mayor seguridad en las consultas que realicen los clientes, evitando que se atente contra la integridad y confiabilidad de la institución; y también brindar comodidad y eficiencia a los usuarios para averiguar a cerca del progreso de sus trámites realizados.

1.2. Justificación Teórica

La necesidad de implantar componentes de seguridad debido al gran crecimiento de los usuarios en la utilización de aplicaciones móviles, esto hace ver que es un requerimiento crítico; ya que algunas aplicaciones manejan información personal del usuario tal como: cuentas bancarias, itinerarios; lo cual es importante que estos componentes sean estudiados, especialmente en JSF-Prime Mobile.

Java Server Faces es una tecnología y framework, producto de las tendencias de desarrollo de software aplicativo basado en web, enfocado para aplicaciones en Java, la cual simplifica el

desarrollo de interfaces de usuario en aplicaciones Java EE.

JSF nos ofrece una serie de ventajas:

- El código JSF con el que se crea las vistas (etiquetas jsp) es muy parecido al HTML estándar. Lo pueden utilizar fácilmente desarrolladores y diseñadores web.
- JSF se integra dentro de la página JSP y se encarga de la recogida y generación de los valores de los elementos de la página.
- JSF resuelve validaciones, conversiones, mensajes de error e internacionalización (i18n).
- JSF permite introducir javascript en la página para acelerar la respuesta de la interfaz en el cliente (navegador del usuario).
- JSF es extensible, por lo que se pueden desarrollar nuevos componentes a medida. También se puede modificar el comportamiento del framework mediante APIs que controlan su funcionamiento.

Prime Faces es una librería open source para Java Server Faces, el objetivo principal de esta es ofrecer un conjunto de componentes ricos para facilitar la creación de aplicaciones web.

1.3. Justificación Práctica

En el Departamento Financiero se realizan pagos de trámite como: órdenes de pago, solicitudes de viáticos, pago a los coordinadores de Programas de Postgrado, facturas por servicios profesionales por dictado de cursos, seminarios y afines, etc.; la recepción de los documentos se ha venido realizando de forma manual, es decir el usuario o cliente presenta sus documentos en el departamento, cada uno de los trámites sigue un orden establecido y pasa por diferentes departamentos, si el usuario o cliente desea saber cómo se va desarrollando su trámite debe acercarse al departamento y preguntar acerca de ello; pero con la implementación del sistema web

móvil el cliente podrá conocer el estado de su trámite accediendo a través de la web haciendo uso de un dispositivo móvil, ya que el futuro del internet se ve enfocado en la movilidad del mismo.

- **Prototipos:** Para analizar el framework JSF + PrimeFaces Mobile y la seguridad a nivel de interfaz de usuario que se definirá en el desarrollo de la tesis presente, se realizarán prototipos los cuales servirán principalmente para dar una solución informática que permita al personal del Departamento Financiero y a los usuarios realizar las consultas de trámite de pago de forma más amigable, versátil y confiable.

El ámbito de la solución será:

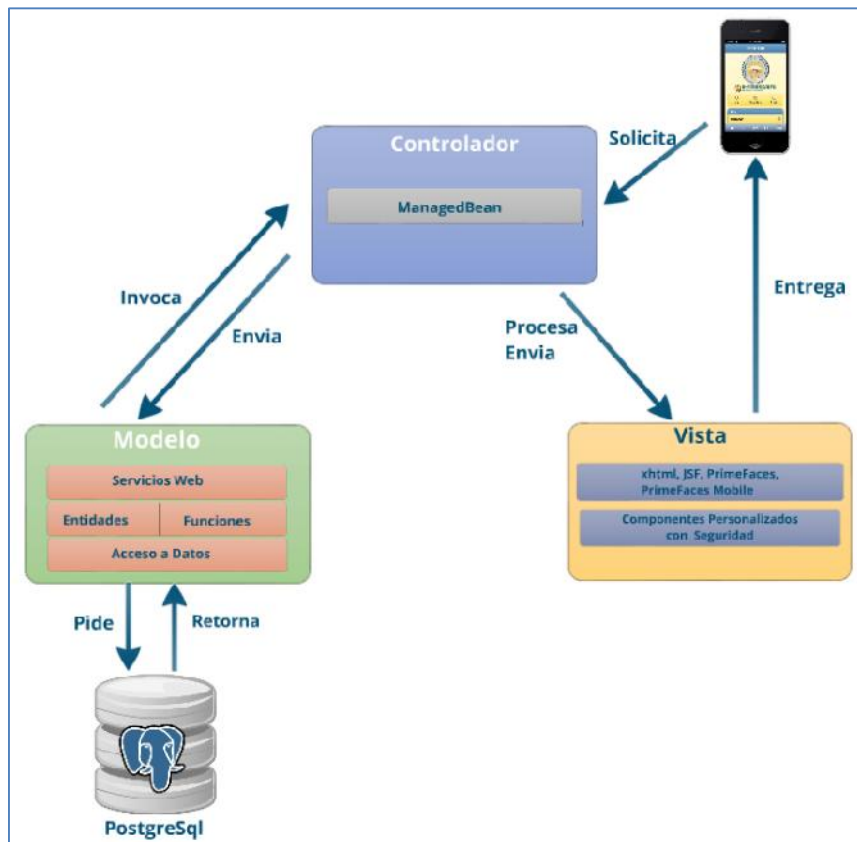
- Crear prototipos de la aplicación móvil utilizando el framework JSF+ PrimeFaceMobile y los componentes de seguridad construidos para poder sanear las vulnerabilidades más comunes como se ha mencionado anteriormente.
- Implementar los Servicios Web con la tecnología Java que permitirá la conexión entre las demás capas y la base de datos que están construidas o están en desarrollo por el DESITEL.
- Implementar una aplicación web móvil la misma que contendrá los siguientes módulos:
 - **Módulo de Autenticación:** Permitirá autenticar a los distintos usuarios que realizan trámites en el Departamento Financiero.
 - **Módulo de Consultas:** Este módulo permitirá realizar consultas avanzadas como: comprobantes por titular y representante legal, comprobantes en un rango de fechas, comprobantes anulados, etc.
 - **Módulo de Seguimiento o Estado del Trámite:** Este módulo permitirá dar seguimiento y conocer el estado de avance del comprobante de pago.
 - **Módulo de Ayuda:** Este módulo permitirá obtener información en formato legible de la aplicación móvil.

El sistema de consultas y monitoreo de procesos de trámites de pago, permitirá conocer el estado en que se encuentra un trámite realizado por un cliente-usuario, ingresando un código al sistema

web móvil el cuál será emitido al momento de la emisión de la factura; también permitirá realizar consultas; implementando seguridad en los procesos realizados por el cliente.

1.4. **Ámbito de la Solución**

En la Figura I.1 se visualiza el ámbito de la solución propuesta para el sistema de consultas y monitoreo de trámites de pago:



Figural.1: Ámbito de la solución

1.5. Objetivos

1.5.1. Objetivo General

- Analizar los componentes de seguridad a nivel de interfaz de usuario en JSF PrimeMobile para desarrollo de aplicaciones móviles para el Departamento Financiero de la ESPOCH.

1.5.2. Objetivos Específicos

- Describir los componentes de seguridad a utilizarse en la implementación de la aplicación móvil.
- Diseñar prototipos para determinar las potencialidades de los componentes de seguridad a nivel de interfaz de usuario.
- Evaluar los componentes construidos mediante parámetros definidos y de acuerdo a los requerimientos planteados por el personal del departamento Financiero de la ESPOCH.
- Implementar una aplicación móvil para el sistema de consultas y monitoreo de procesos de trámite de pago del departamento Financiero de la ESPOCH.

1.6. Marco hipotético

1.6.1. Hipótesis

- El framework JSF + PrimeFaces Mobile y los componentes de seguridad permitirán desarrollar una aplicación web móvil con un nivel de seguridad adecuado.

1.7. Metodología Programación Extrema.- Es una metodología que se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios, y es adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

Se ha escogido seguir este proceso debido a que aplica los últimos avances en Ingeniería de Software; ya que adopta un enfoque eminentemente práctico. Esta metodología contiene fases con sus respectivas actividades que se mencionarán cuando el caso lo amerite.

CAPITULO II

CONCEPTOS GENERALES

1.1. Introducción

Para poder iniciar con una investigación de cualquier índole es importante conocer cada uno de los términos con los cuales se va a trabajar, ya que estos en el lenguaje diario no son habituales y sobre todo realizar un uso adecuado de estos términos.

Para saber en qué consiste la seguridad a nivel de interfaz de usuario es importante comprender varios términos como: en qué consisten las vulnerabilidades de igual manera comprender la seguridad que brinda JSF así como la de PrimeFaces y cómo combinar cada uno de estos términos, así se adquiere ampliar los conocimientos tanto de los autores como del lector y lo más esencial aplicarlos en la investigación.

Mediante este capítulo se pretende dar a conocer los conceptos de los términos con los cuales se trabaja para el desarrollo de la investigación.

2.2. Servicios Web Java

Un Servicio Web es un componente software con las siguientes características:

- Es accesible a través del interface SOAP (Simple Object Access Protocol).
- Su interface se describe en un documento WSDL (Web Service Description Language).

SOAP es un protocolo de mensajería XML extensible que forma la base de los Servicios Web. SOAP proporciona un mecanismo simple y consistente que permite a una aplicación enviar mensajes XML a otra aplicación. WSDL es un documento XML que contiene un conjunto de definiciones (qué hace el Servicio Web, cómo se comunica, y dónde reside) que describen un Servicio Web. Proporciona toda la información necesaria para acceder y utilizar un Servicio Web. [1]

Los servicios Web tienen como principal objetivo proporcionar una forma estándar de interoperabilidad entre diferentes aplicaciones de software que se ejecuta en una variedad de plataformas.

2.3. Dispositivos Móviles

El mundo móvil está revolucionando, está en constante cambio para adaptar los hábitos del PC a las plataformas móviles. Los dispositivos móviles se han convertido en repositorio para almacenar información sensible y los fabricantes de dispositivos móviles amplían rápidamente sus prestaciones técnicas. Hasta hace poco los dispositivos móviles resultaban poco atractivos para las compañías de software. Los procesadores tenían escasas capacidades y no daban para muchas fantasías. Al ofrecer mucha más potencia y memoria, estos procesadores van ahora alojados en dispositivos que caben en la palma de la mano y que incorporan capacidades de comunicaciones móviles. En la actualidad, los dispositivos móviles aportan diferentes ventajas: sirven como medio de comunicación,

son portables, con mayores recursos, con acceso a la red y por último es posible desarrollar aplicaciones de acuerdo a las necesidades de cada usuario y cargarlas en estos dispositivos.

Gran relevancia en el futuro seguirán ganando los dispositivos móviles, que no acabarán con los ordenadores o con tabletas como el iPad; porque el futuro será mixto, aunque con tendencia al uso de los primeros. [2]

2.4. Sistemas Operativos para Móviles

Un sistema operativo para móvil es aquel que controla un dispositivo móvil, los sistemas operativos móviles son mucho más simples y están más orientados a la conectividad inalámbrica, los cuales a través de diversas funcionalidades y aplicaciones permiten la usabilidad de los dispositivos, los formatos multimedia para móviles y las diferentes maneras de introducir información en ellos.

La amplia, pero poco conocida brecha entre el hardware de los dispositivos móviles y el usuario, denominada como 'Capa' o 'máquina virtual', facilita al cibernauta básico o experto las herramientas e interfaces adecuadas para realizar sus tareas informáticas, ahorrándole trabajo y tiempo, a fin de evitarles complicados procesos. Los sistemas más populares del mercado son: Android, iOS, WindowsPhone, Symbian, BlackberryOS.

2.5. Principios generales sobre seguridad

En términos generales, la seguridad puede entenderse como aquellas reglas técnicas y/o actividades destinadas a prevenir, proteger y resguardar lo que es considerado como susceptible de robo, pérdida o daño, ya sea de manera personal, grupal o empresarial. [3]

En este sentido, es la información el elemento principal a proteger, resguardar y recuperar. El número de virus y programas espía para dispositivos móviles es todavía muy pequeño comparado con el que afecta a sistemas Windows de sobremesa. Amenazas como malware (software malicioso) en

aplicaciones de terceros, o vulnerabilidades en los sistemas operativos pueden originar robo de contraseñas, o facturas telefónicas enormes por envíos ocultos de SMS Premium. Se tiene un control físico total sobre los teléfonos móviles y esto hace pensar que son menos accesibles a intrusos. Esta falsa sensación de seguridad, junto con el uso de aplicaciones de estilo de correo electrónico, redes sociales y contenido multimedia privado, lleva a que se almacenen datos personales y confidenciales en el terminal, muchas veces de forma inadvertida para el propio usuario; esta sensación de seguridad conduce a que los usuarios descuiden las precauciones básicas tales como cambiar la configuración por defecto de Seguridad. [4]

2.6. Criptografía

Saber poco de criptografía es más peligroso que no saber nada, los errores frecuentes que se comenten debido a ello son: no identificar la necesidad y aplicarla mal. Con la criptografía se intenta garantizar las siguientes propiedades deseables en la comunicación de información de forma segura:

- **Confidencialidad:** solamente los usuarios autorizados tienen acceso a la información.
- **Autenticación del destinatario:** es el proceso que permite garantizar la identidad del usuario destinatario.
- **Integridad de la información:** garantía ofrecida a los usuarios de que la información original no será alterada, ni intencional ni accidentalmente. [5]

2.6.1. Criptografía Java

La plataforma Java hace especial hincapié en la seguridad, incluyendo la seguridad del idioma, la criptografía, la infraestructura de clave pública, la autenticación, la comunicación segura, y control de acceso.

2.6.1.1. Java Cryptography Architecture (JCA)

La JCA es una pieza importante de la plataforma Java, y contiene una arquitectura de "proveedor" y un conjunto de APIs para la firma digital, resúmenes de mensajes (hashs), certificados y validación de certificados, encriptación (cifrado de bloques / corriente simétrica / asimétrica), generación de claves y la gestión, y la generación segura de números aleatorios, para nombrar unos pocos. Estas API permiten a los desarrolladores integrar fácilmente la seguridad en su código de solicitud.

- **Clases Motores y Algoritmos:** En el contexto del JCA se utiliza el término *motor (engine)* para referirse a una representación abstracta de un servicio criptográfico que no tiene una implementación concreta. Un servicio criptográfico siempre está asociado con un algoritmo o tipo de algoritmo y puede tener alguna de las siguientes funciones:
 - Proporcionar operaciones criptográficas (como las empleadas en el firmado y el resumen de mensajes).
 - Generar o proporcionar el material criptográfico (claves o parámetros) necesario para realizar las operaciones.
 - Generar objetos (almacenes de claves o certificados) que agrupen claves criptográficas de modo seguro.
- **Algoritmo:** Es una implementación de un motor. Por ejemplo: el algoritmo MD5 es una implementación del motor de algoritmos de resumen de mensajes. La implementación interna puede variar dependiendo del código que proporcione la clase MD5.
- **Proveedor:** Es el encargado de proporcionar la implementación de uno o varios algoritmos al programador (es decir, darle acceso a una implementación interna concreta de los algoritmos).

[6]

Los algoritmos MD5 y SHA-1 son vulnerables a ataques, sobre todo MD5, pero SHA-2(SHA-256, 384 y 512) no lo son, de momento, y se espera que se mantengan así por un buen tiempo.

La JCA permite los siguientes algoritmos en sus clases motor:

- MD2
- MD5
- SHA-1
- SHA-256
- SHA-384
- SHA-512

Tanto como si quiere usar un hash para comprobar como para almacenar datos, en Java se lo puede hacer utilizando lo que se conoce como “*Clases Motor*”, como por ejemplo la clase “*MessageDigest*”. Esto significa que JCA ofrece unas clases que proporcionan la funcionalidad de servicios criptográficos. [7]

En resumen, Java no proporciona las funciones MD5 o SHA-256 de forma directa, sino que hay que tomar una clase que ofrece servicios e implementar las funciones o métodos usando los servicios de encriptación que ofrece esa clase.

2.7. Seguridad en las aplicaciones móviles

Las aplicaciones móviles se presentan como una importante línea de negocios, estos han ganado gran popularidad con la masificación de dispositivos móviles inteligentes, constantemente se mejoran o renuevan las tecnologías de transmisión de datos. La información sensible que se almacena o se transmite a través de aplicaciones móviles se expone a riesgos de seguridad, los ambientes móviles se han convertido en el nuevo foco de atención de los atacantes informáticos.

Entre las vías de infección se incluyen el email, MMS, tarjetas de memoria intercambiables, sincronización con el ordenador e incluso Bluetooth. La seguridad de las aplicaciones móviles no sólo hay que verla desde el punto de vista del usuario final, su alcance comprende desde el núcleo del sistema operativo de cada dispositivo móvil hasta el modelo de distribución de sus aplicaciones,

pasando por los entornos de desarrollo de cada plataforma. Sin duda el control de las aplicaciones disponibles para estos dispositivos será clave para mantener la seguridad del usuario final. [8]

2.8. Seguridad en aplicaciones web Java

Las aplicaciones Web Java al igual que las aplicaciones de escritorio que aspiran a ser seguras deberían resolver los siguientes ítems:

- Integridad.
- Autenticación.
- Autorización.
- Confidencialidad.

La Integridad de los datos se refiere a que si el usuario hace algún tipo de operación dentro del sistema, este no debe ser saboteado o cambiado. La Autenticación se refiere al ya conocido LOGIN (usuario y contraseña) y la Autorización, se refiere a lo que el usuario puede hacer en el sistema luego de haber sido autenticado.

La Confidencialidad se refiere a que sólo el usuario indicado debe acceder a la información sensible. La diferencia con Autorización es que la confidencialidad asegura que incluso si la información cae en malos manos, esta sea inutilizable.

Todas las instituciones que expongan sus servicios a Internet tienen que proteger muy cuidadosamente su información y recursos. Desde luego que existen sistemas más críticos que otros, por ejemplo, los bancos tendrán que prestar más atención en estos temas de la seguridad porque la información que maneja es monetaria, en general, todas las aplicaciones Web deben estar resguardadas y aseguradas ante cualquier tipo de ataque. [9]

2.9. Fundamentos de Java Server Faces

Java Server Faces es una tecnología y framework, producto de las tendencias de desarrollo de software aplicativa basada en web, enfocada para aplicaciones en Java, la cual simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE, usando JSP (Java server pages) mediante el lenguaje de programación JAVA. JSF fue desarrollado por la JAVA Community Process (JSR 344 - JSF 2.2). La idea básica de Faces es escribir aplicaciones Web de la misma manera que se escriben aplicaciones de escritorio.

2.9.1. Ventajas de Java Server Faces

- Una de las ventajas más importantes que posee JSF es desarrollar aplicaciones Web al estilo Rapid Application Development (RAD). RAD permite construir rápidamente aplicaciones a partir de un conjunto de componentes de interfaz de usuario reusables. Así se obtienen desarrollos más productivos en un lapso de tiempo menor, en conjunción con la ventaja de poder construir interfaces de usuario complejas (UIs) de forma razonablemente sencilla.
- Provee un conjunto de componentes de interfaz de usuario predefinidos (botones, hipervínculos, checkboxes, etc.), un modelo para la creación de componentes de interfaz de usuario customizados y una manera para procesar en el servidor los eventos generados en el cliente.
- JSF permite introducir javascript en la página, para acelerar la respuesta de la interfaz en el cliente (navegador del usuario).

Todas las aplicaciones Java Server Faces son construidas sobre la API de Servlet. Se comunican a través del protocolo HTTP y usan tecnologías de rendering como por ejemplo JSP. El framework no restringe a JSP la tecnología de presentación, también se podrían utilizar tecnologías como XML/XSLT, motores de templetizado, HTML, XHTML, WML o algún otro protocolo que entiendan tanto el cliente como el servidor.

JSF es considerado como un framework de aplicaciones Web debido a que resuelve la mayoría de las tareas tediosas y repetitivas del desarrollo Web, así el desarrollador se puede enfocar en tareas más desafiantes, como por ejemplo la construcción de la lógica de negocio. Una de las funcionalidades claves es soportar el patrón de diseño Modelo 2, el cual separa el código de la presentación del de la lógica de negocio; aunque Faces se focaliza en los componentes de interfaz de usuario y el manejo de eventos. Faces se integra muy bien con otros frameworks como Struts y así permite agregar funcionalidades a frameworks de alto nivel.

Una página JSF utiliza la extensión *.xhtml, es decir, una combinación de XML con HTML y puede incluir componentes como CSS, JavaScript, entre otros.

La especificación de JSF define seis fases distintas en su ciclo de vida de un componente como se muestra en la Figura II.1:

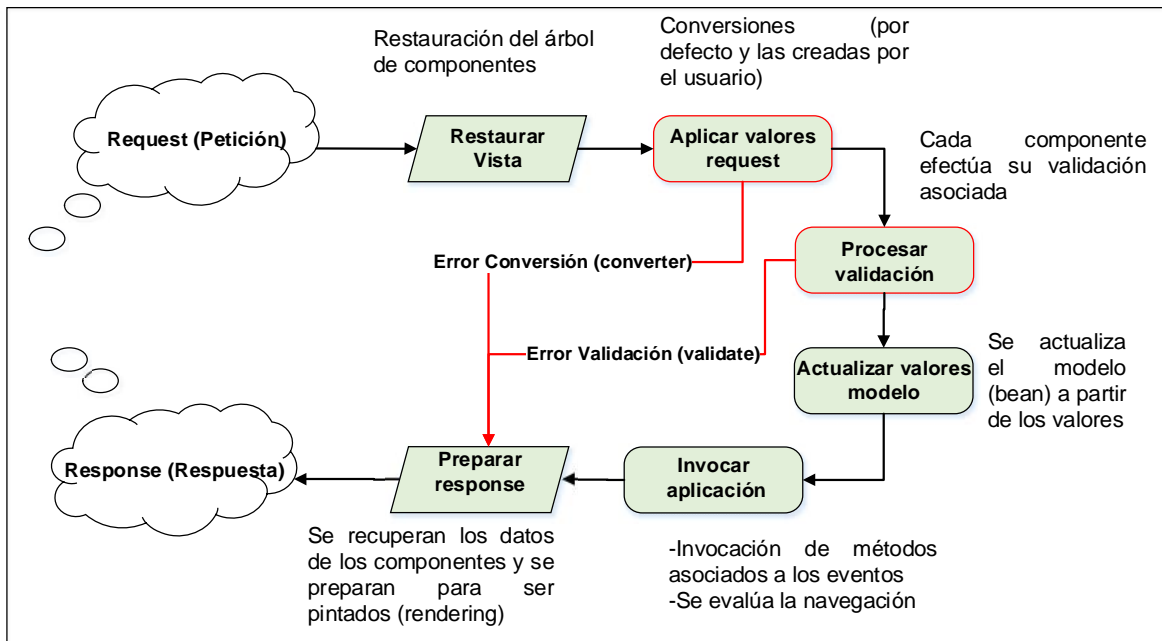


Figura II.1: Ciclo de Vida de un componente JSF

1. **Restauración vista (Restore View):** Crea un árbol de componentes en el servidor para representar la información de un cliente.

2. **Aplicar valores a la petición (Apply Requests Values):** Actualiza los valores del servidor con datos del cliente.
3. **Procesar validación (Process Validations):** Valida los datos del usuario y hace la conversión.
4. **Actualizar valores del modelo (Update Model Values):** Actualiza el modelo del servidor con nuevos datos.
5. **Invocar la aplicación (Invoke Application):** Ejecutar cualquier lógica de aplicación para cumplir con la solicitud.
6. **Producir la respuesta (Render Response):** Guarda un estado y da una respuesta al cliente.

2.10. Prime Faces

PrimeFaces es un componente para Java Server Faces (JSF) de código abierto que cuenta con un conjunto de componentes enriquecidos que facilitan la creación de las aplicaciones web. Primefaces está bajo la licencia de Apache License V2. Una de las ventajas de utilizar Primefaces, es que permite la integración con otros componentes como por ejemplo RichFaces.

2.10.1. Propiedades

- Conjunto de componentes ricos (Editor de HTML, autocompletar, cartas, gráficas o paneles, entre otros).
- Soporte de ajax con despliegue parcial.
- Cuenta con 25 temas prediseñados.
- Componente para desarrollar aplicaciones web para móviles-celulares, especialmente para iPhone, Palm, Android y Windows Phone.

2.10.2. Pros y contras

En cuanto a la experiencia de los usuarios finales los componentes de Primefaces son amigables al usuario además que cuentan con un diseño innovador. Pero en lo que respecta al programador, cada nueva liberación de una nueva versión por parte de los desarrolladores de Primefaces está plagada de errores, además, lo más crítico es que sus desarrolladores no respetan un principio básico del desarrollo de componentes: la compatibilidad hacia atrás, es decir, un componente de una nueva versión de Primefaces por lo general no es compatible al 100% con una aplicación desarrollada con la versión previa a la misma. [10]

2.11. Prime Faces Mobile

PrimeFaces Mobile es un kit de interfaz de usuario para crear aplicaciones JSF optimizados para dispositivos móviles. Funciona con jQuery Mobile con extensiones de PrimeFaces, es compatible con diversas plataformas como iPhone, Android, Palm, Blackberry, Windows Mobile. [11]

2.12. OWASP – Open Web Application Security Project

El software inseguro está debilitando actualmente la infraestructura crítica financiera, de salud, defensa, energía, y otras. A medida que la infraestructura digital es cada vez más compleja e interconectada, la dificultad de lograr que una aplicación sea segura incrementa exponencialmente. Ya no se puede dar el lujo de tolerar los problemas de seguridad relativamente simples. Para la tarea de identificación de las posibles vulnerabilidades en las que podría incurrir una aplicación web, se utilizó como referencia la información que se encuentra en el sitio del proyecto OWASP.

El proyecto abierto de seguridad en aplicaciones Web (OWASP siglas en inglés) es una comunidad abierta dedicada a habilitar a las organizaciones para desarrollar, comprar y mantener aplicaciones confiables. [12]

2.12.1. Las vulnerabilidades más comunes de OWASP

El OWASP Top 10 fue lanzado por primera vez en 2003, se hicieron actualizaciones menores en 2004, 2007 y 2010, y está en la versión de 2013 la cual se ha utilizado seleccionando las vulnerabilidades de interés para la realización de este trabajo de investigación:

- A1 – Inyección Sql
- A3 – Secuencia de Comandos en Sitios Cruzados
- A7 – Exposición de Datos Sensibles
- A8 – Falsificación de Peticiones en Sitios Cruzados

2.12.2. Descripción de las Vulnerabilidades

Las vulnerabilidades anteriormente citadas y tomadas de OWASP se describirán en qué consisten.

2.12.2.1. Fallas de Inyección

Las fallas de inyección, en particular la inyección SQL, son comunes en aplicaciones Web. Existen distintos tipos de inyecciones: SQL, LDAP, XPath, XSLT, HTML, XML, inyección de órdenes en el sistema operativo y otras muchas.

Mediante la Figura II.2 se ilustra cuan dañino llega a ser las fallas de inyección en las aplicaciones web:

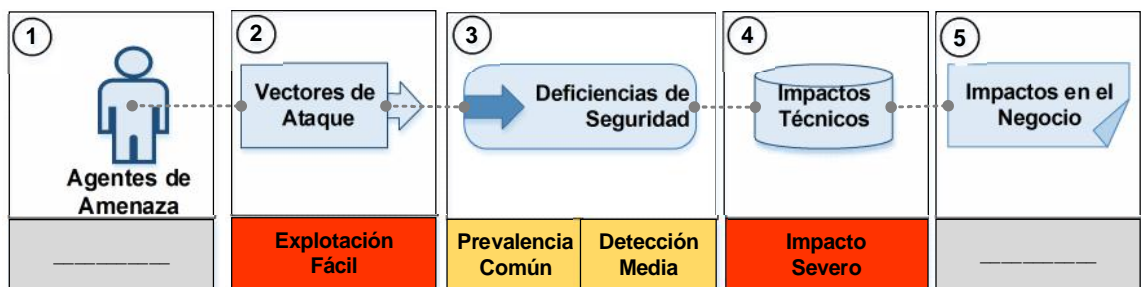


Figura II.2: Fallas de Inyección [13]

Tabla II.1 Descripción de Fallas de Inyección

Factores	Descripción
Agentes de amenaza (1)	Considerar cualquier persona que pueda enviar datos no confiables al sistema, incluyendo usuarios externos, internos y administradores.
Vectores de ataque (2)	El atacante envía simples cadenas de texto que explotan la sintaxis del intérprete atacado. Casi cualquier fuente de datos puede ser un vector de inyección, incluyendo fuentes internas.
Deficiencias de seguridad (3)	Las fallas de inyección ocurren cuando una aplicación envía datos no confiables a un intérprete. Las fallas de inyección son muy prevalentes, particularmente en código enviado, el cual es frecuentemente encontrado en consultas SQL, LDAP, XPath, comandos de SO, argumentos de programa, etc. Las fallas de inyección son fácil de descubrir cuando se examina el código, pero más difícil a través de testeos. Los scanners y fuzzers pueden ayudar a los atacantes a descubrir estas fallas.
Impactos Técnico (4)	Una falla de inyección puede resultar en pérdida o corrupción de datos, falta de integridad, o negación de acceso. Una falla de inyección puede algunas veces llevar a la toma de posesión completa del servidor.
Impactos en el negocio (5)	Considerar el valor para el negocio de los datos afectados y la plataforma corriendo el intérprete. Todos los datos pueden ser robados, modificados, o eliminados

Fuente: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Todas las plataformas de aplicación Web que usen intérpretes o invoquen otros procesos son vulnerables a las fallas de inyección. Esto incluye cualquier componente del marco de trabajo, que pueda usar intérpretes en la capa de bases de datos.

2.12.2.1.1. Protección y tratamiento para Fallas de Inyección

Es recomendable evitar el uso de intérpretes cuando sea posible. Si se tiene que invocar un intérprete, el mecanismo clave para evitar inyecciones es el uso de APIs seguras, como consultas con parámetros de asignación rigurosa y bibliotecas de mapeo relacional de objetos (ORM). Estas interfaces manejan todo el escape de datos, o no requieren escaparlos. Se debe tener en cuenta que aunque las interfaces seguras resuelven el problema, se recomienda usar la validación de todas maneras para detectar ataques.

El uso de intérpretes es peligroso, se debe tomar en cuenta casos como:

- Validación de entrada: Usar un mecanismo estándar de validación de entradas para validar todas las entradas contra el tamaño, el tipo, la sintaxis y las reglas de negocio antes de aceptar los datos que se van a mostrar o almacenar. Usar una estrategia de validación para aceptar las entradas reconocidas como buenas. Rechazar las entradas inválidas en lugar de intentar desinfectar datos potencialmente hostiles. No olvidar que los mensajes de error pueden incluir datos inválidos.
- Usar APIs de consultas con parámetros de asignación rigurosa que reemplacen los parámetros de sustitución, incluso cuando se llame a procedimientos almacenados.
- Conceder los mínimos privilegios cuando se conecte a bases de datos y otros sistemas de bases de datos.
- Evitar los mensajes de error detallados que son útiles para un atacante.
- Usar procedimientos almacenados ya que generalmente no les afectan las inyecciones SQL. Sin embargo, tener cuidado ya que estos pueden ser inyectables (como a través del uso de `exec()` o concatenando argumentos a el procedimiento almacenado).
- Cuando se usen mecanismos de escape simples, se debe tener en cuenta que las funciones simples de escape no pueden escapar los nombres de tabla. Los nombres de tabla deben ser sentencias SQL legales, y por lo tanto son completamente inapropiadas como datos de entrada del usuario.

- Tomar en cuenta errores de canonización: Las entradas deben ser decodificadas y formalizadas en la representación interna de la aplicación actual antes de ser validadas. Asegurarse de que la aplicación no decodifica la misma entrada dos veces. Estos errores pueden ser utilizados para evadir esquemas de listas blancas introduciendo entradas peligrosas después de que hayan sido comprobadas.

2.12.2.2. Cross- Site scripting (XSS)

La secuencia de comandos en sitios cruzados, es un subconjunto de inyección HTML. XSS es la más prevalente y perniciosa problemática de seguridad en aplicaciones Web. Las fallas de XSS ocurren cuando una aplicación toma información originada por un usuario y la envía a un navegador Web sin primero validarla o codificando el contenido.

Mediante la Figura II.3 se ilustra cuan dañino es XSS en las aplicaciones web:

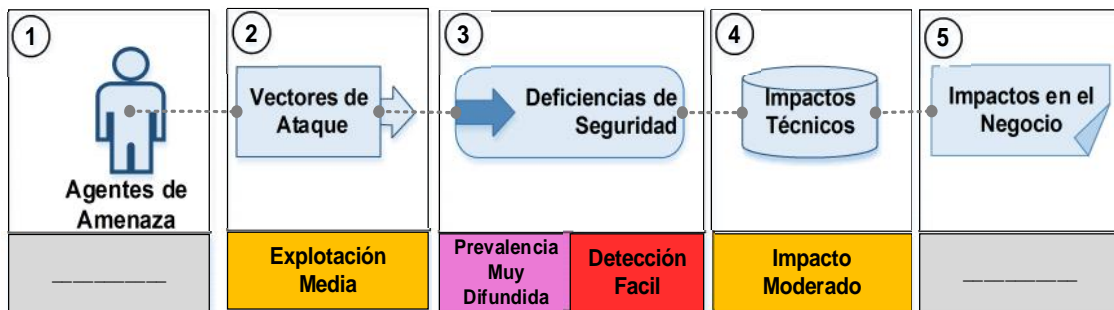


Figura II.3: Secuencia de Comandos en Sitios Cruzados [15]

Tabla II.2: Descripción de Secuencia de Comandos en Sitios Cruzados

Factores	Descripción
Agentes de Amenaza (1)	Considerar cualquier persona que pueda enviar datos no confiables al sistema, incluyendo usuarios externos, internos y administradores.
Vectores de Ataque (2)	El atacante envía simples cadenas de texto que explotan la sintaxis del intérprete atacado. Casi

Factores	Descripción
	cualquier fuente de datos puede ser un vector de inyección, incluyendo fuentes internas tales como datos de la base de datos.
Deficiencias de Seguridad (3)	XSS es la falla de seguridad más prevalente en aplicaciones web. Las fallas XSS ocurren cuando una aplicación incluye datos suministrados por el usuario en una página enviada al navegador sin ser el contenido apropiadamente validado o escapado. La detección de la mayoría de las fallas XSS es relativamente fácil a través de pruebas análisis de código.
Impactos Técnico (4)	Los atacantes pueden ejecutar secuencias de comandos en el navegador de una víctima para secuestrar las sesiones de usuario, destruir sitios web, insertar código hostil, redirigir usuarios, instalar código malicioso en el navegador de la víctima, etc.
Impactos en el negocio (5)	Considerar el valor de negocio de los datos afectados o funciones de la aplicación. También considere el impacto en el negocio la exposición pública de la vulnerabilidad.

Fuente: [https://www.owasp.org/index.php/Top_10_2013-Cross-Site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Top_10_2013-Cross-Site_Scripting_(XSS))

Existen tipos de Cross-Site Scripting:

- **XSS Reflejado:** Es aquel en el que el código inyectado se refleja en el servidor web, por ejemplo, en un mensaje de error, resultado de búsqueda, o cualquier otra respuesta que incluye algunos o todos de la entrada enviada al servidor como parte de la solicitud. Los ataques reflejados se entregan a las víctimas a través de otra ruta, tal como en un mensaje de correo electrónico, o en algún otro servidor web. Cuando un usuario es engañado para hacer clic en un enlace malicioso o enviar un formulario especialmente diseñado, el código inyectado viaja al servidor web vulnerable, lo que refleja el ataque al navegador del usuario. El navegador ejecuta

el código, ya que provenía de un servidor de "confianza". A continuación hay un ejemplo de esto en un servlet:

Texto plano: `out.println("You searched for: "+request.getParameter("query"));`

- **XSS Almacenado:** Son aquellos en los que el código inyectado se almacena de forma permanente en los servidores de destino, como en una base de datos, en un foro de mensajes, registro de visitantes, campo de comentarios, etc. La víctima recupera el script malicioso desde el servidor cuando se solicita la información almacenada. A continuación un ejemplo en un servlet que es pedido desde la base de datos y devuelto en la página HTML sin validación alguna: [16]

Texto Plano: `out.println("<tr><td>" + guest.name + "<td>" + guest.comment);`

2.12.2.2.1. Protección y tratamiento contra Cross Site Scripting (XSS)

Para protegerse contra XSS todos los parámetros de la aplicación deberían estar validados y/o codificados antes de darles salida en una página HTML.

- La opción preferida es escapar todos los datos no confiables basados en el contexto HTML (cuerpo, atributo, JavaScript, CSS o URL) donde los mismos serán ubicados. Los desarrolladores necesitan incluir esta técnica en sus aplicaciones al menos que el marco UI lo realice por ellos.
- Una validación de entradas positiva o "whitelist" (lo que se permite) con apropiada codificación y decodificación es también recomendable ya que ayuda a proteger contra XSS, pero no es una defensa completa ya que muchas aplicaciones requieren caracteres especiales en sus entradas. Tal validación debería, tanto como sea posible, decodificar cualquier entrada codificada, y luego validar la longitud, caracteres, formato, y cualquier regla de negocio en dichos datos antes de aceptar la entrada. [19]

2.12.2.3. Falsificación de Peticiones en Sitios Cruzados

Las vulnerabilidades de falsificación de petición en sitios cruzados no es un ataque nuevo, pero son simples y devastadores. Un ataque CSRF (Cross Site Reference Forgery) fuerza el navegador validado de una víctima a enviar una petición a una aplicación Web vulnerable, la cual entonces realiza la acción elegida a través de la víctima.

Mediante la Figura II.4 se ilustra cuan dañino es CSRF en las aplicaciones web:

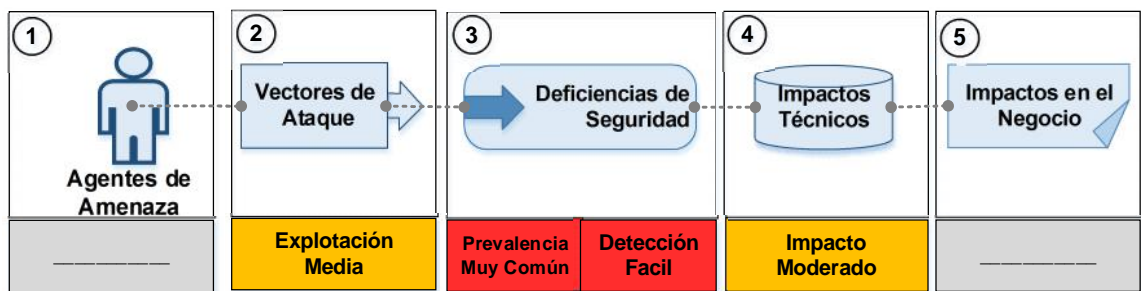


Figura II.4: Falsificación de Peticiones en Sitios Cruzados

Tabla II.3: Descripción de Falsificación de Peticiones en Sitios Cruzados

Factores	Descripción
Agentes de Amenaza (1)	Cualquiera que pueda suplantar a usuarios al momento de enviar peticiones a un sitio web. Cualquier sitio web, u otros canales HTML, a los cuales accedan los usuarios de un determinado sitio web.
Vectores de Ataque (2)	Los atacantes crean peticiones HTTP falsas. Engañan a la víctima al enviarlas a través de etiquetas de imágenes, XSS, o muchas otras técnicas. Si el usuario está autenticado entonces el ataque será exitoso.
Deficiencias de Seguridad (3)	La CSRF aprovecha aplicaciones web que permiten a los atacantes predecir todos los detalles de una acción en particular. Cuando los navegadores envían credenciales de autenticación automáticamente, como en el caso de las cookies de sesión, los

Factores	Descripción
	atacantes pueden crear páginas web maliciosas las cuales generan peticiones falsas que son indistinguibles de las auténticas. Los fallos debidos a CSRF son fácilmente detectables a través de código, o pruebas de penetración.
Impactos Técnicos (4)	Los atacantes pueden cambiar cualquier dato que la víctima esté autorizado a cambiar, o acceder a cualquier funcionalidad que la víctima esté autorizada a utilizar.
Impactos en el Negocio (5)	Considerar el valor de negocio asociado a los datos o funciones afectados. Tener en cuenta lo que representa no estar seguro si los usuarios en realidad desean realizar dichas acciones. Considerar el impacto que tiene en la reputación del negocio.

Fuente: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Desgraciadamente, hoy, la mayoría de las aplicaciones Web confían únicamente en las credenciales presentadas automáticamente tales como cookies de sesión, credenciales de autenticación básica, dirección IP de origen, certificados SSL, o credenciales de dominio Windows.

2.12.2.3.1. Protección contra Falsificación de Peticiones en Sitios Cruzados

Las siguientes estrategias deberían ser inherentes en todas las aplicaciones Web:

- Las aplicaciones deben asegurarse de que no dependen de aplicaciones o testigos suministrados automáticamente por los navegadores. La única solución es utilizar un testigo a medida que el navegador no pueda "recordar" y por lo tanto incluir automáticamente en un ataque CSRF.
- Asegurarse que no existen vulnerabilidades XSS en su aplicación
- Introducir testigos aleatorios "a la medida" en cada formulario y URL que no sean automáticamente presentados por el navegador; por ejemplo:

```
<form action="/transfer.do" method="post">
```



```
<input type="hidden" name="8438927730" value="43847384383">  
...  
</form>
```

y entonces verificar que el testigo suministrado es correcto para el usuario actual. Tales testigos pueden ser únicos para una función particular o página para ese usuario, o simplemente único para la sesión global. Cuanto más enfocado esté un testigo a una función particular y/o a un conjunto particular de datos, más fuerte será la protección, pero también más complicada de construir y mantener.

- Para datos delicados o transacciones de gran valor, re-autenticar o utilizar firmado de transacción para asegurar que la petición es verdadera. Configurar mecanismos externos como e-mail o contacto telefónico para verificar las peticiones o notificar al usuario de la petición.
- No utilizar peticiones GET (URLs) para datos delicados o realizar transacciones de valor. Utilizar únicamente métodos POST cuando se procese datos delicados del usuario. Sin embargo, la URL puede contener el testigo aleatorio que crea una única URL, la cual hace que el CSRF sea casi imposible de realizar.
- El método POST aislado es una protección insuficiente. Se debe también combinarlo con testigos aleatorios, autenticación externa o re-autenticación para proteger apropiadamente contra CSRF. [20]

2.12.2.4. Exposición de Datos Sensibles

Muchas aplicaciones web no protegen adecuadamente datos sensibles como: tarjetas de crédito, credenciales de autenticación. Los atacantes pueden robar o modificar los datos protegidos débilmente y conducir a robo de identidad, fraude de tarjetas de crédito u otros crímenes. Los datos sensibles merecen protección extra como encriptación ya sea que estos se encuentren en reposo o en transmisión, así como precaución especial cuando se los intercambia con el navegador. [21]

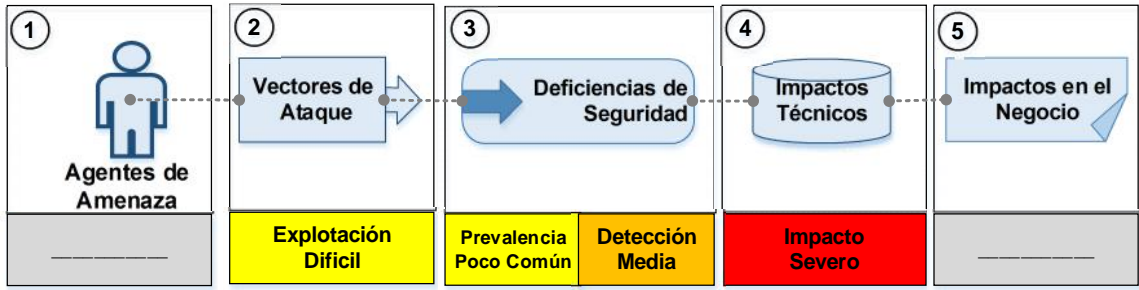


Figura II.5: Exposición de Datos Sensibles

Tabla II.4: Descripción de Exposición de Datos Sensibles

Factor	Descripción
Agentes de amenaza (1)	Considerar a quien pueda obtener acceso a nuestros datos sensibles y cualquier copia de seguridad de estos datos. Esto incluye datos en reposo, en transmisión e incluso en los navegadores de sus usuarios. Incluye tanto a amenazas externas e internas.
Vectores de ataque (2)	Los atacantes generalmente no rompen la criptografía directamente. Ellos rompen alguna otra cosa como: robar claves, hacen ataque de hombre en el medio, roban datos de texto del servidor, los roban en transmisión o directamente desde el navegador.
Deficiencias de Seguridad (3)	La falla más común no es simplemente encriptación de datos sensibles. Cuando la criptografía es empleada, generación y manejo débil de claves, y uso de algoritmos débiles es común, particularmente las soluciones de hashes débiles para proteger las contraseñas. Las deficiencias de navegadores son muy comunes y fácil de detectar, pero difícil de explotar. Los atacantes tienen dificultad detectando más de estos tipos de fallos debido al acceso limitado y suelen ser difíciles de explotar
Impactos Técnicos (4)	Las fallas frecuentemente comprometen todos los datos que deben haber sido protegidos. Típicamente esta información incluye datos sensibles como datos personales, tarjetas de crédito, credenciales, registros de salud, etc.

Factor	Descripción
Impactos en el negocio (5)	Considerar el valor del negocio de la pérdida de datos y el impacto para su reputación. Considerar el impacto que tiene en la reputación del negocio.

Fuente: https://www.owasp.org/index.php/Top_10_2013-A6

2.12.2.4.1. Protección para Exposición de Datos Sensibles

Para todo dato sensible, hacer todo lo siguiente, como mínimo:

- Considerar las amenazas que se planea proteger esos datos de: atacantes internos, usuarios externos, asegurarse de cifrar todos los datos sensibles en reposo y en transmisión de manera que se protege contra estas amenazas.
- No almacenar datos sensibles innecesariamente. Descartarlos lo antes posible. Los datos que no se tiene no pueden ser robados.
- Asegurarse que se usa algoritmos estándares fuertes y claves fuertes y manejo adecuado de claves.
- Asegurarse que las contraseñas están almacenadas con un algoritmo específicamente diseñado para protección de contraseñas.
- Deshabilitar el autocompletado en formularios que recogen datos sensibles y deshabilitar el almacenamiento en cache para páginas que despliegan datos sensibles. [22]

2.12.3. OWASP ESAPI (Enterprise Security API)

ESAPI es una colección de métodos de seguridad para aplicaciones web, esta es libre, de código abierto, que facilita a los programadores escribir aplicaciones de menor riesgo. Las bibliotecas ESAPI están diseñados para hacer más fácil a los programadores el reacondicionamiento de seguridad en

las aplicaciones existentes. Las bibliotecas ESAPI también sirven como una base sólida para un nuevo desarrollo.

Teniendo en cuenta las diferencias específicas del idioma, todas las versiones de OWASP ESAPI tienen el mismo diseño básico:

- Hay un conjunto de interfaces de control de seguridad. Ellas definen, por ejemplo, los tipos de parámetros que se pasan a los tipos de controles de seguridad.
- Hay una implementación de referencia para cada control de seguridad. La lógica no es específica de la organización y la lógica no es específica de la aplicación. Un ejemplo: la validación de entrada basada en cadenas.
- Existe sus propias implementaciones para cada control de seguridad. Puede haber lógica de la aplicación contenida en estas clases que pueden ser desarrolladas por o para su organización. Un ejemplo: la autenticación de la empresa.

Se puede utilizar o modificar ESAPI como se desee, incluso incluirlo en productos comerciales. [23]

2.12.4. OWASP y Java Server Faces

En el sitio del proyecto OWASP se puede encontrar una sección dedicada al análisis de la seguridad en el framework JSF. En ella se describe el funcionamiento de Faces según el estándar JSF y cómo los atacantes tienen conocimiento sobre la arquitectura y el ciclo de vida de cualquier aplicación Faces. Otro dato relevante que abordan es, la carencia de modelo de seguridad del framework, pero que éste puede ser usado en conjunción con algún otro modelo de seguridad JEE sin mayores inconvenientes de integración.

A continuación se describe los hitos más importantes considerados para aplicaciones JSF, según OWASP:

- **Almacenando estado en el lado del cliente:** Las aplicaciones Faces podrían almacenar los estados de sus componentes de UI en la respuesta al cliente o en el servidor. En ocasiones el guardado del estado de los componentes requiere que se haga en el cliente, ya sea por motivos de configuración o por performance. Esta condición puede llevar a que un atacante pueda manipular los datos del estado de algunos componentes para intentar lograr un escalamiento de privilegios. Los datos sin encriptar son fácilmente manipulables y la mayoría de los datos guardados en el cliente son objetos JAVA serializados, los cuales contienen una gran cantidad de información sensible. En caso de utilizar almacenamiento de estado de los componentes de UI en el cliente, es importante analizar qué datos son esenciales que se almacenen en el cliente y el resto de la información de estado almacenarla en el servidor a fin de minimizar los riesgos.
- **Convertidores:** Los convertidores tienen como función convertir *strings* en objetos JAVA. Un atacante podría inyectar código malicioso para lograr que el convertidor exponga información sensible. El sistema de conversión de JSF no es inseguro en sí mismo, pero la conversión, desde una visión de seguridad para la lógica de negocio o los datos persistentes debe ser manejada con mucho cuidado. Toda entrada de datos debe ser validada antes de ser convertida.
- **Validadores:** Los validadores verifican que lo ingresado sea como se espera. Esto significa que si una entrada representa un string cuyo dominio es de 10 a 254 caracteres de longitud, el mismo puede ser validado para prevenir un ataque de XSS y una inyección de SQL.
- **ManagedBeans:** El servidor JSF actualiza el estado del *ManagedBean* asociado a una página con cada requerimiento de la misma. El estado se actualiza con los valores que vienen en el *request*, previo de ser validados de acuerdo a las reglas de validación que definió el desarrollador. Pero el *framework* permite a los desarrolladores que puedan obtener estáticamente al *FacesContext* y con él acceder a los parámetros de *request* exactamente como los que ingresó el usuario. Este atajo en JSF podría ser el camino para explotar alguna

vulnerabilidad, ya que se saltaría la validación y se accedería directamente a los parámetros del requerimiento. [24]

2.13. Descripción de Componentes de Seguridad de JSF

Java Server Faces cuenta con una gran suite de componentes y además numerosos puntos de extensión, pero consta de pocos componentes de seguridad nativa como son: validadores, convertidores y los filtros de salida o escape de caracteres peligrosos, estos componentes se describen a continuación:

2.13.1. Sistema de Conversión de JSF

En la fase denominada “*Apply Request Value*” (ciclo de vida) es justamente donde se ejecuta la conversión de los valores que llegan desde el formulario hacia el tipo de dato apropiado. Si la conversión es exitosa, los valores pasan a ser validados.

Todos los objetos que cumplan la función de “*converters*” deben implementar a la interfaz “*javax.faces.convert.Converter*”.

Esta interfaz tiene dos métodos como se visualiza en la figura II.6:

```
public Object getAsObject(FacesContext context,
                          UIComponent component,
                          String value)

public String getAsString(FacesContext context,
                          UIComponent component,
                          Object value)
```

Figura II.6: Métodos de Interfaz javax.faces.convert.Converter.

Donde:

- El parámetro “context” es la instancia “FacesContext” del request.
- El parámetro “component” es el componente cuyo valor será convertido.

- El parámetro “value” es el valor a ser convertido.

El método “*getAsObject*” es invocado durante el procesamiento de entrada, para convertir los valores String que llegan en el request, al tipo de dato deseado.

El método “*getAsString*” es invocado durante la fase de salida para mostrar los valores en formato de String en cualquiera de las tecnologías de rendering que soporta el cliente.

Si se lanza una excepción de tipo “*ConverterException*”, el componente se marca como inválido y se coloca un mensaje en el FacesContext. Los convertidores estándares son:

Tabla II.5: Convertidores estándares

Class in javax.faces.convert	Converts Values of Type	Converter ID
BigDecimalConverter	java.math.BigDecimal	javax.faces.BigDecimal
BigIntegerConverter	java.math. BigInteger	javax.faces.BigInteger
BooleanConverter	java.lang.Boolean Boolean	javax.faces. Boolean
ByteConverter	java.lang.Byte byte	javax.faces.Byte
CharacterConverter	java.lang.Character char	javax.faces.Character
DateTimeConverter	java.util. Date java.sql.Date	javax.faces. DateTime
DoubleConverter	java.lang.Doubledouble	javax.faces. Double
FloatConverter	java.lang.Float float	javax.faces. Float
IntegerConverter	java.lang.Integer int	javax.faces. Integer
LongConverter	java.lang.Long long	javax.faces. Long
NumberConverter	java.lang.Number (para las monedas, porcentajes, etc	javax.faces. Number
ShortConverter	java.lang.Short short	javax.faces.Short

Elaborado por: Jenny Perez, Gustavo Parco

Todos los “converters” se pueden llamar usando la etiqueta “<f:converter>” y especificando el ID, a excepción de *DateTimeConverter* y *NumberConverter* que tienen sus propias etiquetas: “<f:convertDateTime>” y “<f:convertNumber>”.

Conversión Implícita: Se realiza cuando JSF conoce el tipo de dato del valor. Por ejemplo, asumiendo que “edad” es un atributo de tipo Integer del managed bean “usuario”, en el siguiente caso la conversión es implícita:

```
<h:inputText id="edad" value="#{usuario.edad}" />
```

Conversión Explícita: Es cuando se usan las tres etiquetas que proporciona la librería core de JSF.

- **Etiqueta #1:** “<f:convertDateTime>” (Figura II.7)

```
<f:convertDateTime dateStyle="dateStyle"
  locale="locale" pattern="formatPattern" timeZone="timeZone"
  type="type"
  binding="valueExpression" />
```

Figura II.7: Uso de etiqueta “<f:convertDateTime>”

Donde:

- El “dateStyle” puede ser: short, medium, long, full o default.
- El valor de “type” puede ser: date, time o both (default).
- El valor de “binding” es una expresión que apunta a un método de una clase que implementa la interfaz `javax.faces.convert.Converter`.

- **Etiqueta #2:** “<f:convertNumber>” (Figura II.8)

```
<f:convertNumber currencyCode="code"
  currencySymbol="symbol"
  groupingUsed="boolean" locale="locale"
  pattern="formatPattern" type="type"
  binding="valueExpression" />
```

Figura II.8: Uso de etiqueta “<f:convertNumber>”

Donde:

- “currencyCode” es el código ISO 4217 de las monedas. Por ejemplo, “USD” es para los dólares americanos y “EUR” para el euro. Para la moneda peruana se tiene el código “PEN”.
- “currencySymbol” se puede especificar para indicar el símbolo de la moneda a utilizar (pero se debe tener en cuenta que esto es válido con el JDK 1.4 o superior). El atributo currencyCode toma preferencia sobre este.
- “groupingUsed” es un flag boolean que sirve para indicar cuando se debe usar un delimitador.
- type” puede ser: number, currency o percentage.
- El valor de “binding” es una expresión que apunta a un método de una clase que implementa la interfaz javax.faces.convert.Converter.

Etiqueta #3: “<f:converter>” (Figura II.9)

```
<f:converter converterId="converter-id"
  binding="valueExpression" />
```

Figura II.9: uso de etiqueta “<f:converter>”

Donde:

- “converter-id” es el valor registrado en el archivo faces-config.xml de la aplicación.
- El valor de “binding” es una expresión que apunta a un método de una clase que implementa la interfaz javax.faces.convert.Converter.

Para vincular un Componente UI con un converter, simplemente hay que anidar el converter dentro de la etiqueta (Figura II.10):

```
<h:outputText id="date" value="#{transaction.date}">
  <f:convertDateTime dateStyle="short" />
</h:inputText>
```

Figura II.10: Anidación de converter en la etiqueta

2.13.2. Sistema de Validación de JSF

La interfaz “*javax.faces.validator.Validator*” es el núcleo del sistema de validaciones de JSF. Todos los objetos que cumplan la función de validadores deben implementar esta interfaz.

Dicha interfaz tiene un único método:

“public void validate (FacesContext context, UIComponent component, Object value)”

Este método es invocado durante la fase de “Apply Request Values” (ciclo de vida) si el componente tiene el atributo “immediate” con su valor activo o también durante la fase de “Process Validations” en caso contrario. Antes de invocar al método, JSF marca al componente como “inválido”.

- El método lanza la excepción “*ValidatorException*” si ocurre un error durante la validación: se almacena un mensaje en el FacesContext.
- En caso de no ocurrir errores, el componente recién se marca como “válido”.

A diferencia de los “converters”, los “validators” se registran por *validator-id* debido a que el concepto de validación no está asociado al tipo del objeto.

Adicionalmente, JSF proporciona la facilidad de usar el atributo “*required*” en las etiquetas. Si esta propiedad está presente, se ejecutan las validaciones.

Los validadores estándares son (Tabla II.6):

Tabla II.6: Validadores estándares en JSF

<i>Validator-ID</i>	<i>Tag Handler</i>	<i>Descripción</i>
javax.faces.DoubleRange	f:validateDoubleRange	Valida que el tipo de dato sea java.lang.Double y se encuentre dentro del rango especificado.
javax.faces.Length	f:validateLength	Valida que el tipo de dato sea String y que tenga la longitud mínima especificada en el parámetro.

Validator-ID	Tag Handler	Descripción
javax.faces.LongRange	f:validateLongRange	Valida que el tipo de dato sea java.lang.Long y se encuentre dentro del rango especificado.
javax.faces.Bean	f:validateBean	Sirve para indicar que el valor sea validado por el framework de validación EE.
avax.faces.RegularExpression	f:validateRegex	Permite el uso de expresiones regulares
javax.faces.Required	f:validateRequired	Tiene la misma funcionalidad que el atributo "required": campo obligatorio.

Elaborado por: Gustavo Parco

2.13.3. Filtrado de salida con componentes JSF

Los componentes JSF más comúnmente utilizados para este propósito es "`<h:outputText/>`". En el siguiente ejemplo se imprime el valor del parámetro de petición de nombre:

```
<h:outputText value="#{param.name}"/>
```

El componente JSF filtrará la salida y escapará caracteres peligrosos como entidades XHTML. Por ejemplo, el carácter "`<`" escapó "`<`" automáticamente. En el código fuente JSF para este componente, se puede encontrar la siguiente rutina:

```
switch (c) {  
    case '<': htmlEntity = "&lt;"; break;  
    case '>': htmlEntity = "&gt;"; break;  
    case '&': htmlEntity = "&amp;"; break;  
    case '"': htmlEntity = "&quot;"; break;  
}
```

El componente "`outputText`" también tiene un interruptor para desactivar "`escape`" de caracteres peligrosos. El siguiente ejemplo abre la solicitud de vulnerabilidad de seguridad XSS:

```
<h:outputText value="#{param.name}" escape="false"/>  
<- NO HACER ESTO! XSS agujero de seguridad! ->
```

Sólo se debe utilizar de esta manera el atributo “*escape = false*” y cambiar cuando se esté seguro de que el contenido que se va a imprimir en HTML no contiene ningún carácter peligroso. Por ejemplo, si el contenido de texto ya es filtrado y almacenado escapado en la base de datos, quizá se quiera evitar doble escape del símbolo “&”:

```
<h:outputText value="#{myBean.myTextContent}" escape="false"/>
```

<- Contenido contiene entity; y ya está a salvo! ->

La siguiente plantilla XHTML implícitamente creará un componente `<h:outputText/>` y también es seguro:

```
<p>Este es mi nombre:#{param.name} </p>
```

Otros componentes JSF como `<h:outputLink/>` y `<h:message/>` se derivan del mismo componente padre como `<h:outputText/>` y también son por lo tanto para filtrar valores automáticamente.

2.14. Prueba para filtrar la salida y escape de caracteres peligrosos con componentes JSF

Para demostrar la funcionalidad del componente y su atributo, se realiza con el script: “`<script>alert(document.cookie); </script>`”, el cual obtendrá toda la información de la cookie almacenada en el navegador.

Una de las deficiencias que tienen los componentes de entrada de JSF es que dejan ingresar todos los caracteres con sentencias HTML y javascript (Figura II.11), pero al momento de ser mostrada la información se filtra por el atributo de escape el cual está activo por defecto y no permite la ejecución del script además muestra la información al momento de solicitarlo (Figura II.12), lo que demuestra

la inconsistencia en las demás capas, esto se da por la inexperiencia de programadores que no filtran los datos en las capas inferiores.

Datos Personales

Nombre:
Juan

Apellido:
Perez

Edad:
15

Direccion:
Av. Leopoldo Freire

Password:
<script>alert(document.cookie);</script>

Ingresar

Figura II.11. Ingreso de script Malicioso.

Atras Mis Datos

Nombre: Av. Leopoldo Freire
Apellido: Perez
Password: <script>alert(document.cookie);</script>
Edad: 15
Direccion: Av. Leopoldo Freire

Figura II.12. Resultados del script ingresado.

Para demostrar la vulnerabilidad y obtener la cookie mediante la ejecución del script se desactiva el atributo de escape como se visualiza en la Figura II.13 y se observa los resultados devastadores que esto provocará como se visualiza en la Figura II.14.

```
<h:outputText id="otApellido" value="#{usuarioC.usuario.apellido}"/><br/><br/><h:outputText id="otPassword" value="Password: #{usuarioC.usuario.password}" escape="false"/><br/><br/><h:outputText id="otEdad" value="#{usuarioC.usuario.edad}"/><br/><br/>
```

Figura II.13 Desactivación del filtro de escape.



Figura II.14. Ejecución del script por la desactivación del filtro de escape.

Una vez obtenida la cookie la cual almacena: el nombre de usuario y el password, ya se puede hacer mal uso de ella.

2.15. Seguridad de JSF para CSRF

JSF 2.x tiene ya incorporado prevención para CSRF en el campo de entrada oculto en el formulario denominado “*javax.faces.ViewState*” (Figura II.15), este mantiene el estado de la vista porque HTTP no tiene estado. El estado de los componentes necesitan ser mantenidos de una manera u otra a través de peticiones. Ya sea guardando el estado en el servidor y enlazarlo a la sesión, o serializar / deserializar en la petición / respuesta cada vez.

En JSF 1.x este valor fue muy débil y demasiado fácil predecible (realmente nunca fue pensado como prevención de CSRF). En JSF 2.0 esto ha mejorado usando un valor largo y fuerte generado automáticamente en lugar de un valor de secuencia bastante predecible y convirtiéndola en una robusta prevención de CSRF [25]. En JSF 2.2 esto será incluso aún mejor por lo que es una parte necesaria de la especificación JSF 2.

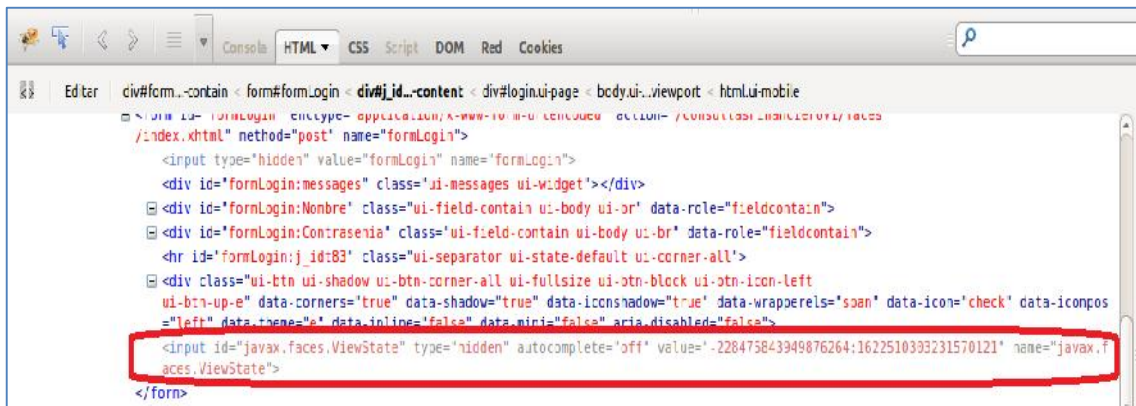


Figura II.15: Visualización del “*javax.faces.ViewState*” y el código generado.

2.16. Exposición de Vulnerabilidades en JSF + PrimeFaces Mobile

2.16.1. Exposición de vulnerabilidad en JSF + PrimeFaces Mobile ante XSS

El prototipo creado para la demostración no tiene ningún tipo de validación sintáctica de la entrada por defecto con lo cual es particularmente vulnerable a un ataque de XSS. Al analizar cómo se muestra la información en el sistema, queda de manifiesto que los parámetros que se ingresan se

muestran en la página de respuesta tal cual fueron ingresados por fuentes externas o por la misma aplicación. Con lo cual si se inyecta código a través de los mismos, este se visualizará tal y como es o se ejecutará algunas sentencias indeseables que pueden afectar a la seguridad de la aplicación:



Figura II.16. Visualización del Script inyectado

Como se visualiza en la Figura II.16 el script es inyectado por otra fuente de ingreso, en este caso es mostrado pero no es ejecutado, esto se da por falta de validación en la entrada.

Como se demuestra en la Figura II.11 y la Figura II.12 el script que ha sido ingresado por fuentes distintas o por la misma no es validado dando lugar a un posible agujero de seguridad, además el script no se ejecuta por la seguridad por defecto que incorpora JSF como lo es el atributo “escape” descrito anteriormente.

Por este motivo se propone la validación de los campos de entrada usando librerías de apoyo como ESAPI [23] .

2.16.2. Exposición de vulnerabilidad en JSF + PrimeFaces Mobile ante Sql Injection

Es posible también realizar de manera exitosa un ataque de inyección SQL en JSF + PrimeFaces Mobile, ya que no valida los parámetros de entrada, los cuales en ciertos casos son utilizados de manera directa para generar un script SQL.

La prevención de ataques Sql Injection no es responsabilidad de JSF. Ya que esto se puede manejar en capas inferiores como el acceso a datos o capas de persistencias como por ejemplo JPA y/o Hibernate, además de no utilizar sentencias SQL concatenadas con entradas del usuario. Hay varios métodos de combatir los SQL inyección en las distintas capas que no se tratarán ya que no es el objetivo de esta tesis.

Para realizar la prueba de vulnerabilidad se utiliza un plugin de Firefox llamado “SQL Inject Me” el cual inyecta sentencias SQL (Figura II.17). Para ver todas las sentencias SQL que pasaron se verifica en el Log del servidor de aplicación allí se verá las sentencias SQL que han pasado.

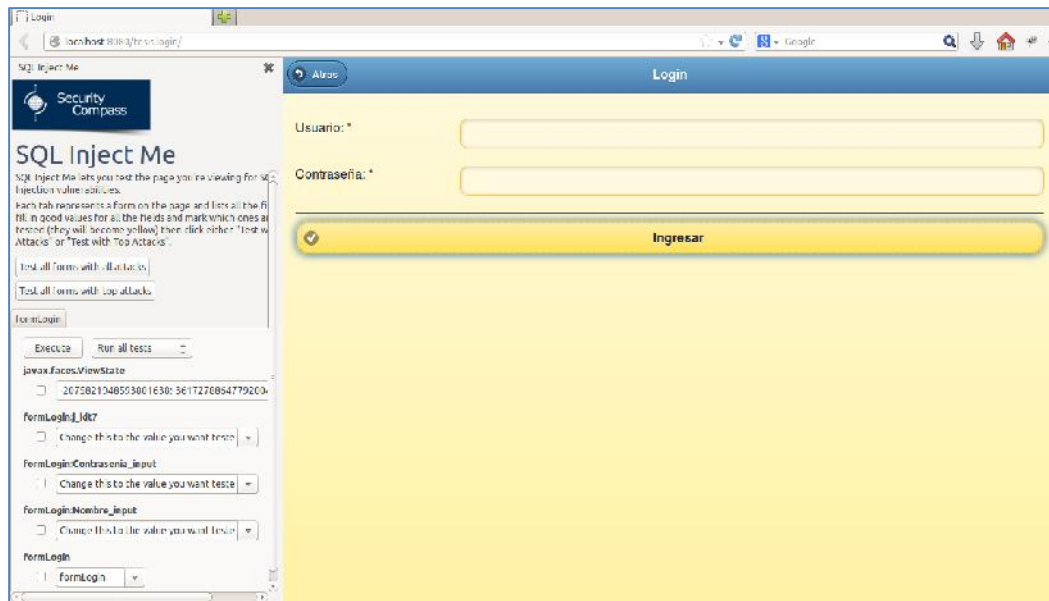


Figura II.17. Prueba en la aplicación con el plugin SQL Inject Me

Una vez realizada la prueba con la herramienta se observa las instrucciones SQL que han pasado como se visualiza en la Figura II.18. Por lo cual se deduciría que JSF + PrimeFaces mobile es altamente vulnerable a SQL Injection.

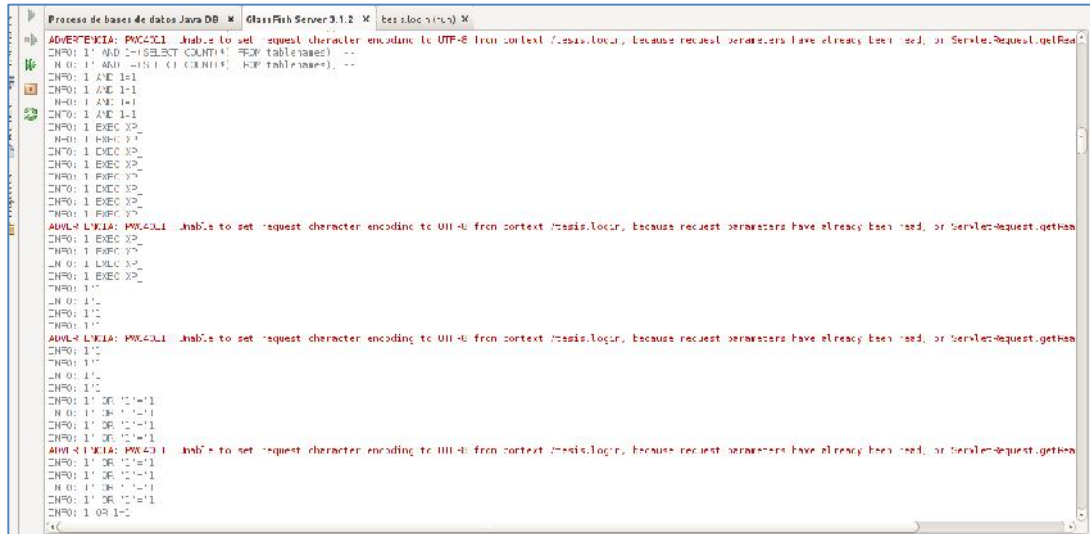


Figura II.18. Sql Injecion que han pasado y son peligrosos para las demás capas

2.16.3. Exposición de Datos de Datos Sensibles

En las aplicaciones web al no realizar ninguna acción en cuanto a la protección de la información, ni a la exposición de las estructuras de las tablas de la base de datos que utilizan para su funcionamiento, y al generarse un requerimiento como por ejemplo visualizar datos personales del usuario, toda la información sensible que se utiliza para concretar dicha acción, queda expuesta en el código retornado por el servidor como respuesta a la petición realizada por el cliente. Así con sólo inspeccionar la página resultado, se obtiene fácilmente la totalidad de la estructura de la tabla involucrada del usuario, con lo cual constituye una grave falla en cuanto a la confidencialidad de los datos respecta propiciando acceso a información que no debería ser pública.

La Figura II.19 muestra que con solo inspeccionar el HTML que retornó el servidor es inmediata la deducción de la estructura de la tabla teniendo acceso a información que no debería ser pública.

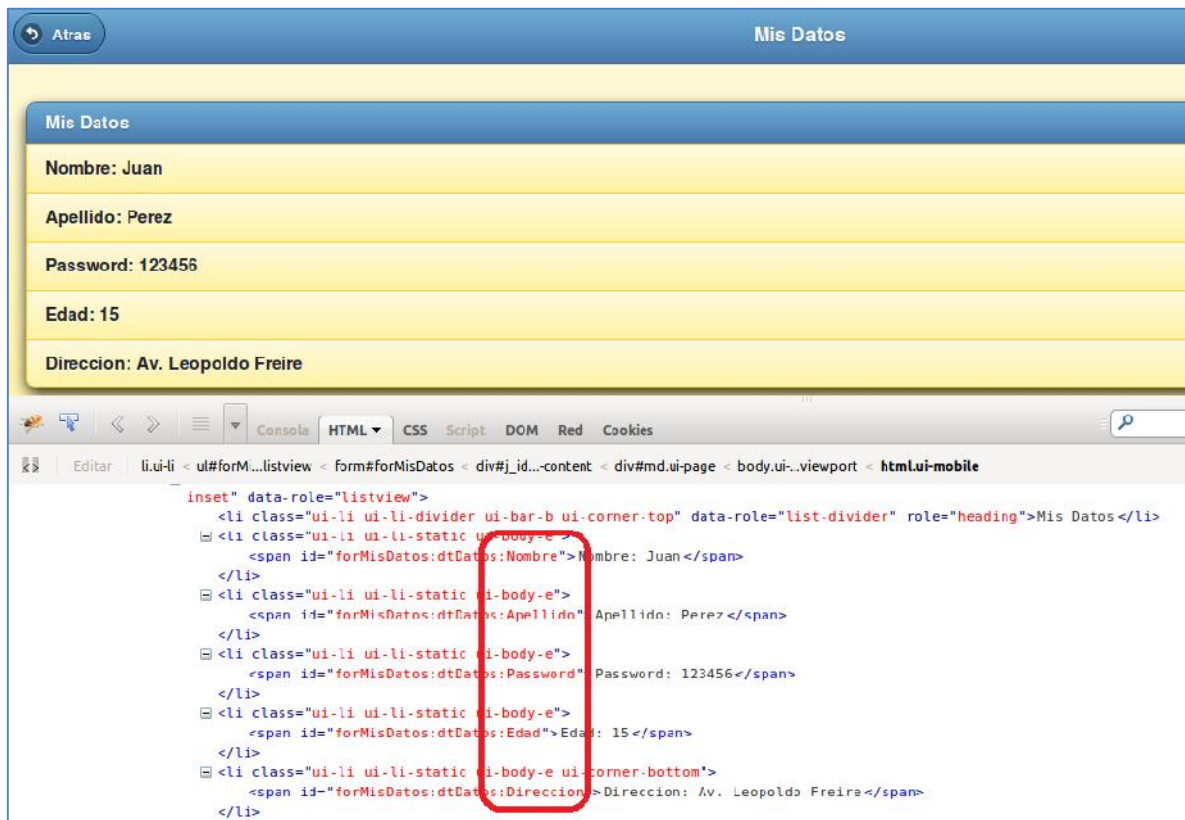


Figura II.19. Deducción de tabla con una simple revisión del código fuente xhtml.

2.17. Creación de componentes de entrada personalizados con seguridad para evitar Sql Injection, XSS

Particularmente en cuanto al desarrollo de los componentes JSF personalizados, OWASP expresa que la seguridad de éstos depende enteramente de cómo han sido desarrollados. Los componentes desarrollados por terceras partes deberían ser distribuidos en conjunción con reportes de seguridad y soporte a través de alguna comunidad o vendedor especial.

OWASP, en su sitio recomienda que si se desarrolla una librería de componentes JSF personalizados, se debe tener en cuenta las mismas cuestiones de seguridad de cualquier desarrollo Web. Dichas cuestiones son:

- No confiar en los parámetros de los requerimientos.
- No confiar en las cookies.

- Siempre considerar que la sesión pudo haber sido capturada.

Para agregar la seguridad en el componente personalizado hay que tener en cuenta los posibles riesgos descritos anteriormente que JSF ha pasado por alto como es el saneamiento de la entrada para evitar posibles ataques como SqlInjection, XSS y la falta de confidencialidad.

A continuación se muestra cómo desarrollar un componente que combina la funcionalidad de los tres componentes estándar necesarios para aceptar la entrada de un usuario: una etiqueta que explica lo que se espera, el campo de texto para aceptar la entrada, y un mensaje para informar de errores de entrada. En otras palabras, se muestra cómo reemplazar el código de JSF siguiente:

```
<h:outputText value="Nombre de Contacto"/>
<h:inputText id = "nombre" required = "true" value = "#{Usuario.nombre}" requiredMessage = "El
valor es necesario!"/>
<h:message for="name" styleClass="error"/>
```

Con este componente personalizado:

```
<mc:InputtextSeguro label="Nombre de Contacto" required = "true" value = "#{Usuario.nombre}"
errorStyleClass = "error" requiredMessage = "El valor es necesario!"/>
```

Además se mostrará cómo este nuevo componente *“InputtextSeguro”* imprime un asterisco al lado de la etiqueta si es necesario *required= "true"*.

2.17.1. El Componente Personalizado

Su única tarea es registrar con el componente de la cadena que identifica el renderer. La única propiedad del componente que se define es la etiqueta label. Esto se debe a que esta ampliación de la *“UIInput”*, que se encarga de definir todo lo que tiene que ver con el campo de entrada.

Se utiliza el método *“getFamily”* para encontrar todos los renderers asociados con este componente. El estado del componente consiste en el estado de *“UIInput”* más la propiedad de la etiqueta. Por lo tanto, define su estado como un arreglo de dos objetos. El método *SaveState* forma una matriz y lo

devuelve, para que JSF puede salvarlo. El método *“restoreState”* recibe el estado, lo descomprime, y lo almacena localmente.

Entonces se observa cómo las operaciones que tienen que ver con *“UllInput”* son siempre delegadas. Si desea visualizar el código del componente personalizado ir a anexo 1.

Una vez creado el componente se debe registrarlo mediante el uso de la notación *“@FacesComponent”* o la inserción de las siguientes líneas en el archivo faces-config.xml

```
<component>
  <component-type>inputEntry</component-type>
  <component-class>tesis.componente.components.InputEntryComponent</component-class>
</component>
```

2.17.1.1. Renderer

El renderer es un poco más complejo que el componente. Para implementarlo, se define una clase que extiende *“javax.faces.render.Renderer”*. Primero se sobrescribe tres métodos: *“decode”*, *“encodeBegin”*, y *“getConvertedValue”*.

La única propiedad que se agrega a UllInput es la etiqueta, que el usuario puede modificar. Por lo tanto, sólo es necesario decodificar el campo de entrada.

El proceso de decodificación ocurre en la fase *Apply Request Values* (fase 2 del ciclo de vida de jsf).

El proceso de codificación requiere más trabajo que el proceso de decodificación, porque tiene que enviar a la respuesta HTTP los tres componentes que se combinan para formar *“InputEntryComponent”*. Esto tiene lugar durante la fase *“Render Response”* (fase 6 del ciclo de jsf).

Hay que crear un método para codificar la etiqueta. Se abre el elemento HTML con la etiqueta con el método *“startElement”*, escribe la etiqueta con un simple método de escritura, escribe un asterisco, pero sólo si se requiere el componente, y se cierra el elemento de la etiqueta con el método *“endElement”*. El resultado es algo así como:

```
“<label>Nombre de contacto * </label>”.
```

Creamos otro método para codificar el campo de entrada. Se abre el elemento de entrada HTML *“input”*, añade los atributos con el método *“writeAttribute”*, y cierra el elemento. Los tres parámetros de *“writeAttribute”* son: el nombre y el valor del atributo HTML y el nombre de la propiedad del componente. El resultado es el siguiente elemento: Anexo

```
“<input type=“text” id=“form:nameEntry” name=“form:nameEntry” value=“” />”
```

El método *“encodeMessage”* es para codificar el mensaje de error. Se obtiene la lista de todos los mensajes de la cola para el componente, pero sólo muestra el primero. Si se desea mostrar todos ellos, sólo se tiene que sustituir la palabra clave *“if”* con un *“while”*.

Para mostrar el mensaje, el método abre el elemento *“SPAN HTML”*, agrega el atributo de clase para mostrar el mensaje con el estilo correcto, muestra el mensaje, y se cierra el elemento. El resultado es algo como el siguiente elemento:

```
“<span class=“error”>El valor es necesario!</span>”
```

Además se incluye la encriptación del *“id”* para suplir la falta de confidencialidad en el método del *“encodeLabel”*. Si desea visualizar el código del render personalizado por favor diríjase al anexo 2.

2.17.1.2. Personalización del archivo TLD

Antes de poder utilizar la etiqueta personalizada en la página xhtml se necesita crear el equivalente a un *“TLD (Tag Library Descriptor)”* que es un archivo xml y situarlo en *“>/WEB-INF/tags.tld.xml”* y configurarlo en el archivo *“web.xml”* para que el faces lo reconozca. Si desea visualizar el código del archivo personalizado diríjase al anexo 3.

2.17.1.3. Configuración del archivo web.xml

A continuación se visualiza el código de configuración del archivo web.xml para el componente personalizado:

```
<context-param>
```

```
<param-name>javax.faces.FACELETS_LIBRARIES</param-name>  
<param-value>/WEB-INF/tags.tld.xml</param-value>  
</context-param>
```

Seguidamente se crea la página XHTML y se hace el llamado al componente que se ha creado. Entonces el resultado será la siguiente interfaz optimizada para móviles (figura II.20).

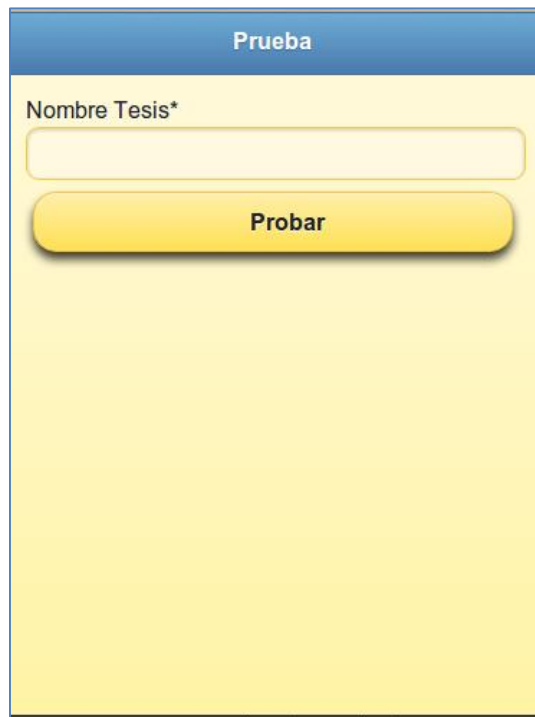
The image shows a mobile-optimized web form. At the top, there is a blue header bar with the text "Prueba" centered. Below the header, the main content area has a yellow background. It contains a text input field with the label "Nombre Tesis*" above it. Below the input field is a yellow button with the text "Probar" centered on it. The entire form is enclosed in a thin blue border.

Figura II.20: Componente personalizado

2.18. Incorporando seguridad al componente personalizado para evitar posibles ataques de Sql Injection y XSS

Para poder incorporar la seguridad en los componentes de interfaz de usuario se va a intervenir en el ciclo de vida de los componentes de JSF en la fase de "validación", se trabaja con el apoyo de la librería que sugiere OWASP denominada "esapi.2.0.1.jar". Si desea visualizar el código del validador diríjase al anexo 5.

Para que la clase “*ValidadorXSS_SQLInyection*” se reconozca a nivel de interfaz de usuario se agrega las siguientes líneas de código en el archivo “*faces-config.xml*”.

```
<validator>
    <validator-id>tesis.validadores.ValidadorXSS_SQLInyection</validator-id>
    <validator-class>tesis.validadores.ValidadorXSS_SQLInyection</validator-class>
</validator>
```

Ahora se debe llamar al validador creado y al componente creado. Si desea ver el código diríjase al anexo 6.

2.19. Creación de componentes personalizados para evitar la falta de confidencialidad de datos

Como se describió anteriormente en la creación del componente personalizado, se ha agregado la criptografía a los ids y nombres de los componentes utilizando los métodos de encriptación mencionados anteriormente, utilizando la más fuerte criptografía como “SHA-512”, si desea visualizar el código de encriptación diríjase al anexo 7.

Seguidamente se muestra una parte de la encriptación en el componente creado anteriormente. Una vez encriptado el “*id*” y el “*nombre*” se podrá visualizar la encriptación de los atributos al ver el código fuente HTML renderizado.

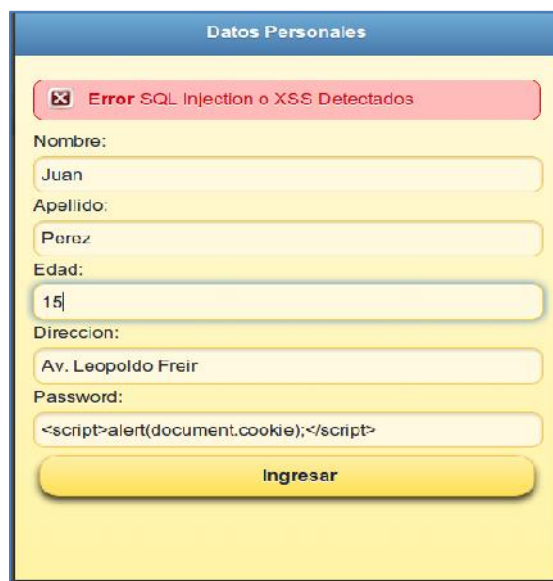
```
private void encodeLabel(ResponseWriter respWr, InputEntryComponent cmp)
    throws IOException {
    respWr.startElement("label", cmp);
        respWr.writeAttribute("id",
            StringEncryptor.getStringMessageDigest(cmp.getClientId(),
                StringEncryptor.SHA512), "id");
        respWr.writeAttribute("name",
            StringEncryptor.getStringMessageDigest(cmp.getClientId(), StringEncryptor.SHA512),
            "name");
        respWr.write(cmp.getLabel());
        if (cmp.isRequired()) {
            respWr.write("*");
        }
        respWr.endElement("label");
    }
```


2.20. Pruebas de Vulnerabilidad a los componentes creados

2.20.1. Pruebas de vulnerabilidad a los componentes creados para evitar XSS

Como ya se demostró anteriormente los componentes nativos de entrada son vulnerables aunque tienen muy poca seguridad al momento de validar los datos de entrada por esta razón se creó el componente y se agregó la seguridad que de cierta forma mitigará los ataques XSS.

Demostración (Figura II.21):

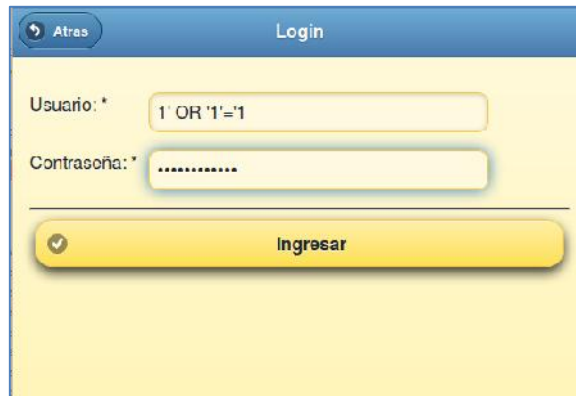


The image shows a web form titled "Datos Personales" with a blue header. Below the header is a red error message box that says "Error SQL Injection o XSS Detectados". The form contains several input fields: "Nombre:" with the value "Juan", "Apellido:" with the value "Perez", "Edad:" with the value "15", "Direccion:" with the value "Av. Leopoldo Freir", and "Password:" with the value "<script>alert(document.cookie);</script>". At the bottom of the form is a yellow button labeled "Ingresar".

Figura II.21: componentes creados para evitar XSS

2.20.2. Pruebas de vulnerabilidad a los componentes creados para evitar Sql Injection

Como ya se demostró los componentes nativos de JSF no evita este ataque ya que no le corresponde mitigar este ataque, porque suele darse más en capas inferiores de la arquitectura, por esta razón se ha implementado para que no permita pasar las inyecciones SQL y la mitigación de este ataque, así poder hacer uso de este. La demostración se visualiza en la figura II.22 y figura II.23:



A screenshot of a web application's login page. The page has a blue header with a back button labeled 'Atras' and the title 'Login'. Below the header, there are two input fields: 'Usuario: *' and 'Contraseña: *'. The 'Usuario: *' field contains the SQL injection payload '1' OR '1'=1'. The 'Contraseña: *' field contains a series of dots representing a masked password. Below the input fields is a yellow button with a checkmark icon and the text 'Ingresar'.

Figura II.22: Ingreso de inyecciones SQL



A screenshot of the same login page as in Figure II.22, but with an error message displayed. The error message is a red box with a white 'X' icon and the text 'Error SQL Injection o XSS Detectados'. The 'Usuario: *' field still contains the SQL injection payload '1' OR '1'=1', and the 'Contraseña: *' field contains dots. The 'Ingresar' button is still present below the fields.

Figura II.23: Ejecución de inyecciones SQL

2.20.3. Pruebas de vulnerabilidad a los componentes creados para evitar la exposición de datos sensibles

Como se lo ha demostrado anteriormente JSF hace un manejo deficitario de la confidencialidad de los datos sensibles. Ahora al tratar de recabar información mediante la inspección del código HTML devuelto por las aplicaciones que hacen uso de la librería de extensión presentada en esta tesis, se obtiene el siguiente código fuente (Figura II.24):

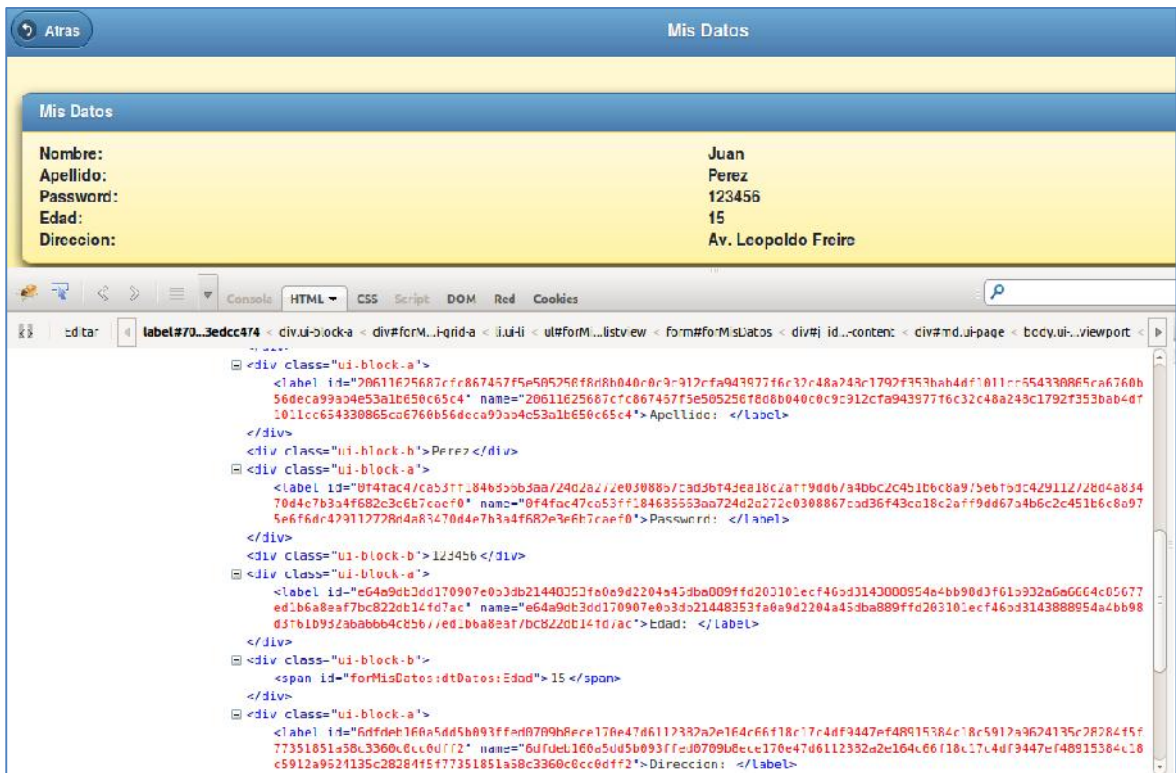


Figura II.24: Inspección del código HTML

CAPITULO III

MATERIALES Y MÉTODOS

3.1. Diseño de la Investigación

El presente trabajo de investigación es Cuasi Experimental ya que para llevarlo a cabo se procederá a recopilar información acerca de las vulnerabilidades que sean acorde a nivel de interfaz de usuario. Consecutivamente se desarrollará la aplicación para realizar las consultas y monitorear los trámites de pago para el Departamento Financiero (ESPOCH); mediante la utilización de prototipos se procederá a aplicar los componentes personalizados y también los que proporciona JSF+ PrimeMobile. Finalmente se valorará cuantitativamente y de esta manera se obtendrá las conclusiones y recomendaciones para poder determinar que componentes proporcionan seguridad más adecuada para la aplicación web móvil implementada.

3.2. Métodos

El método que utiliza este trabajo de investigación es:

Método Científico: Mediante el uso de este se realiza la recopilación de la información para determinar los componentes de seguridad de JSF+ PrimeFaces Mobile y como personalizar los componentes y aplicarlo en el sistema de consultas y monitoreo de trámites de pago al cual se pretende aplicar un nivel de seguridad adecuado. Este método comprende las siguientes fases:

- Planteamiento del Problema
- Formulación de la hipótesis
- Levantamiento de la información
- Análisis e interpretación de resultados
- Comprobación de la hipótesis
- Difusión de resultados

3.3. Técnicas

Las técnicas que se utiliza este trabajo de investigación son:

- Recopilación de Información
- Razonamiento
- Pruebas
- Metodología XP para el desarrollo de software

3.4. Planteamiento de la hipótesis

El framework JSF + PrimeFaces Mobile y los componentes de seguridad permitirán desarrollar una aplicación web móvil con un nivel de seguridad adecuado.

3.5. Variables

De acuerdo a la hipótesis se tiene las siguientes variables:

- **Variable Independiente**
 - Framework JSF + PrimeFaces Mobile y los componentes de seguridad.
- **Variable Dependiente**
 - Aplicación web móvil con un nivel de seguridad adecuado

3.6. Operacionalización de Variables

3.6.1. Operacionalización Conceptual

Tabla III.1: Operacionalización Conceptual

Variable	Tipo	Concepto
Framework JSF + PrimeFaces Mobile y los componentes de seguridad	Independiente	Conjunto de herramientas personalizados para evitar vulnerabilidades en aplicaciones web móviles.
Aplicación web móvil con un nivel de seguridad adecuado	Dependiente	Software el cual evita acciones inapropiadas e inadvertidas realizadas por el usuario sin tomar en cuenta los privilegios que este posea.

Elaborado por: Gustavo Parco, Jenny Perez

3.6.2. Operacionalización Metodológica

Tabla III.2: Operacionalización Metodológica

Hipótesis	Variables	Indicadores	Instrumentos
El framework JSF + PrimeFaces Mobile y los componentes de seguridad	Framework JSF + PrimeFaces Mobile y los componentes de seguridad	Vulnerabilidades aplicadas	Software Código Fuente Pruebas
permitirán desarrollar una aplicación web móvil con un nivel de seguridad adecuado.		Confidencialidad	Software de Monitoreo (Wapiti, XSS Me, Acunetix, CSRFTester) Revisión de código fuente.

Hipótesis	Variables	Indicadores	Instrumentos
	Aplicación web móvil con un nivel de seguridad adecuado	Disponibilidad	Software de Monitoreo (Wapiti, Acunetix, XSS Me)
		Integridad	Software de Monitoreo (Wapiti, Acunetix, XSS Me, CSRFTester, SQL Inject Me), Scripts.

Elaborado por: Jenny Perez, Gustavo Parco

- **Confidencialidad:** La confidencialidad se refiere al acceso a la información únicamente por personas que cuenten con la debida autorización.
- **Disponibilidad:** Se refiere al acceso a la información y a los sistemas por personas autorizadas en el momento que así lo requieran.
- **Integridad:** Se refiere al mantener con exactitud la información tal cual fue generada, sin ser manipulada o alterada por personas o procesos no autorizados.

3.7. Población y Muestra

Población: Es el conjunto de elementos de referencia sobre el que se realizan unas de las observaciones sobre el que estamos interesados en obtener conclusiones, constituye el objeto de investigación de donde se extrae la información requerida para el estudio, para nuestro objetivo la población son todas las vulnerabilidades que afectan a la seguridad de las aplicaciones web.

Muestra: Es una parte de la población que se selecciona para realizar el estudio. Una muestra debe ser representativa, es decir, deba reflejar las características esenciales de la población que se desea estudiar, en este caso la muestra que se seleccionará es No Aleatoria ya que el estudio se enfoca en las vulnerabilidades como: Injection SQL, CSRF, XSS y la falta de confidencialidad de los datos ya que estas son las que se adaptan a la seguridad a nivel de interfaz de usuario. Así se las aplicará en el Sistema de Consultas y Monitoreo de Trámites de Pago del Departamento Financiero.

3.8. Instrumentos de Recolección de Datos

Para realizar el escaneo de vulnerabilidades se utilizará las siguientes herramientas software:

- **Wapiti:** Permite auditar la seguridad de sus aplicaciones web. Realiza "recuadro negro", es decir, no estudia el código fuente de la aplicación, pero va a escanear las páginas de la aplicación. [28]

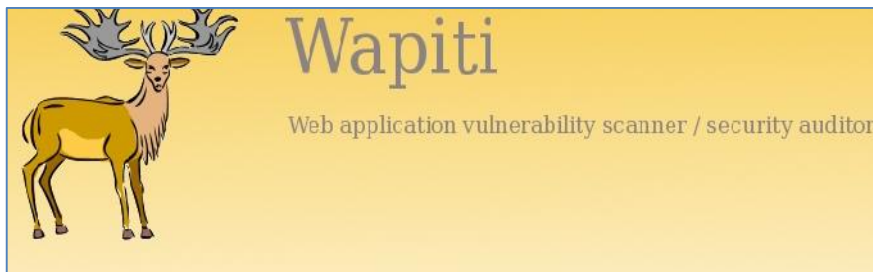


Figura III.1: Scanner Wapiti

- **Acunetix Web Vulnerability Scanner:** Comprueba automáticamente las aplicaciones web para inyecciones SQL, cross site scripting y otras vulnerabilidades. También incluye una serie de herramientas avanzadas de pruebas de penetración para facilitar los procesos de auditoría de seguridad manuales, y también tiene la capacidad de crear auditoría de seguridad profesional y los informes de cumplimiento normativo. [29]



Figura III.2: Herramienta acunetix WVS

- **XSS Me:** Esta herramienta busca los puntos de entrada posibles para ataques contra el sistema, ayuda a detectar las vulnerabilidades XSS para proteger las aplicaciones de errores innecesarios. [30]

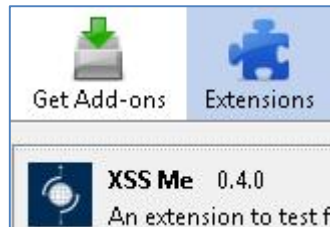


Figura III.3: Herramienta XSS ME

- **SQL Inject-Me:** es un complemento de Firefox usado para evaluar las vulnerabilidades de SQL Injection, utilizado para la prueba de vulnerabilidades de inyección SQL. [31]

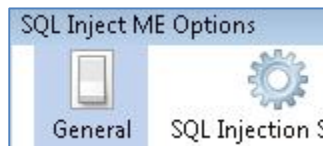


Figura III.4: Herramienta SQL Inject ME

- **CSRFTester:** Básicamente, registra una sesión de usuario legítima y después la utiliza para construir sitios web que intentan realizar las mismas acciones nuevamente. [33]



Figura III.5: Herramienta CSRFTester

Las herramientas software antes mencionadas se han clasificado de acuerdo al tipo de vulnerabilidades que estas escanean como se menciona en la tabla III.3.

Tabla III.3: Scanners de Vulnerabilidades para aplicaciones web

Tipo de vulnerabilidad	Herramienta Software
Confidencialidad	Wapiti, Acunetix, XSS Me, CSRFTester
Integridad	Wapiti, Acunetix, XSS Me, SQL Inject Me, CSRFTester
Disponibilidad	Wapiti, Acunetix, XSS Me

Elaborado por: Gustavo Parco: Jenny Perez

3.9. Ambientes de Prueba

Mediante la utilización de computadores portátiles los cuales contiene software para virtualización se procederá a realizar el escaneo de la aplicación con los componentes de JSF+ PrimeMobile y también con componentes de JSF+PrimeMobile y componentes personalizados. Ambos escenarios cuentan con las herramientas de software y hardware como visualiza a continuación:

Tabla III.4: Hardware utilizado para las pruebas

Cantidad	Equipo	Marca	Modelo	Especificaciones
1	Ordenador Portátil	HP	dv6-6153cl	Procesador Core i7 8GB Memoria RAM 750 Gb Disco Duro
1	Ordenador Portátil	DELL	Inspiron N5110	Procesador Core i7 8GB Memoria RAM 750 Gb Disco Duro

Elaborado por: Gustavo Parco, Jenny Perez

Tabla III.5: Software utilizado para las pruebas


Nombre	Descripción
Suite Backtrack	Sistema operativo con herramientas de escaneo pre instaladas
Ubuntu 13.04	Sistema operativo enfocado en la facilidad de uso e instalación, la libertad de los usuarios.
Glassfish 3.1.2	Servidor de Aplicaciones JAVA
Postgresql	Servidor de base de datos

Elaborado por: Jenny Perez, Gustavo Parco

3.9.1. Prototipos de Prueba

Para aplicar las vulnerabilidades se tendrán los siguientes prototipos:

- **Prototipo de prueba 1:** Mediante el uso de un proceso de ingreso de datos se aplicará las vulnerabilidades CSRF, SQL Injection, XSS, Falta de Confidencialidad de los Datos y se trabajará con los siguientes elementos:
 - JSF+PrimeMobile sin componentes personalizados.
 - Wapiti, Acunetix, SQL Injection ME, XSS Me, CSRFTester.



El formulario, titulado "Datos Personales", contiene los siguientes campos de texto y un botón:

- Nombre: Juan
- Apellido: Perez
- Edad: 15
- Direccion: Av. Leopoldo Freire
- Password: 123456
- Botón: Ingresar

Figura III.6: Prototipo de Prueba 1

- **Prototipo de prueba 2:** Mediante el uso de un proceso de ingreso de datos se aplicará las vulnerabilidades CSRF, SQL Injection, XSS y se trabajará con los siguientes elementos:
 - JSF+PrimeMobile y el uso de componentes personalizados con la respectiva seguridad.
 - Wapiti, Acunetix, SQL Injection ME, XSS Me, CSRFTester



Formulario de Datos Personales con los siguientes campos:

- Nombre: Juan
- Apellido: Perez
- Edad: 15
- Direccion: Av. Leopoldo Freire
- Password: 123456

Botón de Ingresar

Figura III.7: Prototipo de Prueba 2

CAPITULO IV

ANÁLISIS DE RESULTADOS

Para el desarrollo de este capítulo se toma como referencia el trabajo de investigación del Ing. Paul Paguay [2], en base al cual se ha definido la tabla IV.1 en la que se indica los criterios para *“la ponderación de la severidad de la vulnerabilidad”*.

Tabla IV.1: Ponderación de la Severidad

Criterio	Valor	Equivalencia
Visible Información sin riesgo	0	Muy Bajo
Visible Datos comunes	1	Bajo
Visible Datos Importantes	2	Medio
Vulnerabilidad expuesta	3	Alto

Fuente: Propuesta de Técnicas de Aseguramiento de Aplicaciones web desarrolladas en Java

Muy Bajo o Informativo: Se pueden visualizar información como, servidores que pueden estar levantados en el servidor, protocolos, que no generan riesgo o que son muy generales.

Bajo: La aplicación permite ver datos como puerto de escucha del servidor como 80 (http), 5432 (postgres), 3306 (mysql), etc. Datos que deberían estar ocultos.

Medio: Puertos abiertos y aceptando conexiones, información del servidor como versiones, plataformas, etc. Datos que con un análisis más profundo se pueden explotar vulnerabilidades.

Alto: Cuando ya se tiene una vulnerabilidad, y se puede explotar la vulnerabilidad con scripts o herramientas de penetración.

Para valorar el nivel de seguridad se considera la definición dada en ISO 27001: “*la seguridad de la información consiste en la preservación de su confidencialidad, integridad y disponibilidad, así estos tres términos constituyen la base sobre la que se cimienta todo el edificio de la seguridad de la información*”. [34]

En base a la definición de ISO 27001, expuesta en el párrafo anterior, se establece la igualdad de la fórmula 1, donde la **S**eguridad es igual a la **C**onfidencialidad más la **D**isponibilidad más la **I**ntegridad, según se visualiza en la fórmula 1.

$$S = C + D + I \quad (1)$$

En el presente trabajo del 100% del nivel de seguridad a ser medido se ha distribuido porcentualmente de la siguiente manera:

Tabla IV.2: Distribución del Porcentaje del Nivel de Seguridad

	Porcentaje
Confidencialidad	35%
Disponibilidad	30%
Integridad	35%

Fuente: Propuesta de Técnicas de Aseguramiento de Aplicaciones web desarrolladas en Java

Para las fórmulas 2, 3, 4 que se visualizan a continuación se tiene:

D: Disponibilidad

C: Confidencialidad

I: Integridad

gV: Gravedad de la vulnerabilidad

vC, vD, Vi: número de vulnerabilidades encontradas de Confidencialidad, de Disponibilidad y de Integridad respectivamente.

Para obtener la valoración de la Confidencialidad (C) se realizará con la fórmula 2.

$$C = \frac{35}{1 + \sum(vC + gV/10)} \quad (2)$$

Para obtener la valoración de la Disponibilidad (D) se realizará con la fórmula 3.

$$D = \frac{30}{1 + \sum(vD + gV/10)} \quad (3)$$

Para obtener la valoración de la Integridad (I) se realizará con la fórmula 4.

$$I = \frac{35}{1 + \sum(vI + gV/10)} \quad (4)$$

La recolección de los datos se lo realiza con las herramientas de software: Wapiti, Acunetix, SQL Inject Me, XSS Me, CSRF Tester; las mismas que están detalladas en el capítulo III (Instrumentos de Recolección de Datos). A su vez los datos recopilados son sintetizados en el cuadro guía que se expone como tabla IV.3.

Tabla IV.3: Cuadro guía para el escaneo de vulnerabilidades

Herramientas	Vulnerabilidad	PSCP		PCCP	
		nV	gV	nV	gV

Los datos que contiene la tabla IV.3 son:

Herramienta: Software de escaneo de vulnerabilidades.

Vulnerabilidad: Nombre de la vulnerabilidad.

PSCP: Prototipo sin componentes de seguridad personalizados y JSF+ Prime Mobile.

PCCP: Prototipo con componentes de seguridad personalizados y JSF+ Prime Mobile.

nV: Número de vulnerabilidades encontradas al realizar el escaneo.

gV: Gravedad de la vulnerabilidad.

Cabe resaltar que los indicadores de seguridad como Confidencialidad, Disponibilidad e Integridad tomados para este trabajo de investigación se los trata a nivel de interfaz de usuario.

3.10. Variable Independiente

Se ha monitoreado el Framework JSF + PrimeFaces Mobile y los componentes de seguridad con cinco herramientas software para escaneo de vulnerabilidades para aplicaciones web.

3.11. Variable Dependiente

Para desarrollar una aplicación web móvil con un nivel de seguridad, se ha clasificado a las vulnerabilidades seleccionadas para este trabajo de investigación (XSS, SQL Injection, CSRF, Exposición de Datos Sensibles) en base al indicador ya sea de: Confidencialidad Disponibilidad o Integridad.

Indicador 1: Confidencialidad

La confidencialidad se refiere al acceso a la información únicamente por personas que cuenten con la debida autorización.

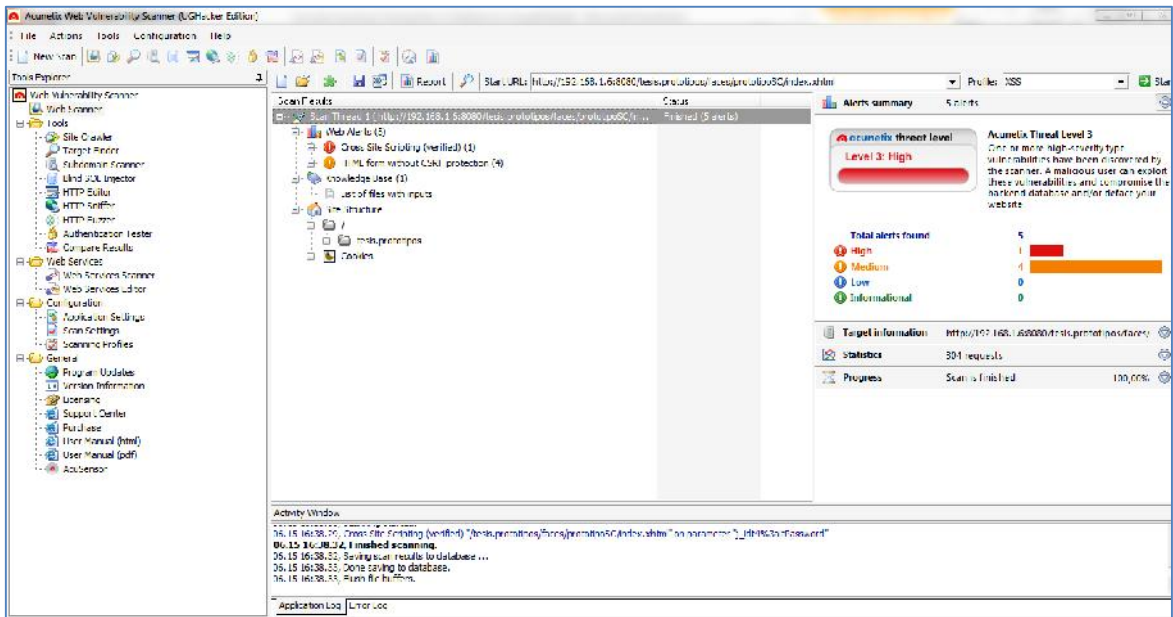


Figura IV.1: Escaneo de vulnerabilidades con software Acunetix

Para realizar el escaneo de las vulnerabilidades se lo realiza en el prototipo de prueba 1 y prototipo de prueba 2, los mismos que están detallados en el capítulo III (Prototipos de Prueba).

Tabla IV.4: Escaneo de vulnerabilidades de Confidencialidad

Herramientas	Vulnerabilidad	PSCP		PCCP	
		nV	gV	nV	gV
Wapiti	XSS	0	3	0	3
	CSRF	0	3	0	3
Acunetix	XSS	5	3	0	3
	CSRF	4	3	4	3
XSS Me	XSS	5	3	3	3
Revisión de Código Fuente	Falta de Confidencial de los Datos	10	3	0	3
CSRFTester	CSRF	0	3	0	3

Fuente: Propuesta de Técnicas de Aseguramiento de Aplicaciones web desarrolladas en Java

En base a los resultados obtenidos mediante el escaneo de las vulnerabilidades con las herramientas, se aplica la fórmula 2.

Tabla IV.5. Aplicación de fórmula 2 en Confidencialidad

Herramienta	Prototipo Sin Componentes personalizados			Prototipo Con Componentes personalizados		
	nV	gV	$\frac{35}{1 + \sum (vC + gV/10)}$	nV	gV	$\frac{35}{1 + \sum (vC + gV/10)}$
Wapiti	0	3	35	0	3	35
	0	3		0	3	
Acunetix	5	3	14.96	0	3	25.46
	4	3		4	3	
XSS Me	5	3	14	3	3	18.42
Revisión de Código Fuente	10	3	8.75	0	3	35
CSRFTester	0	3	35	0	3	35
	Promedio de Confidencialidad		21.54%	Promedio de Confidencialidad		29.78%

Fuente: Propuesta de Técnicas de Aseguramiento de Aplicaciones web desarrolladas en Java

De acuerdo a los resultados obtenidos aplicando las herramientas de escaneo de vulnerabilidades como se visualiza en la tabla IV.4 se ha obtenido que: **la confidencialidad del prototipo utilizando componentes personalizados con la respectiva seguridad es de 29.78% y la confidencialidad del prototipo sin utilizar componentes personalizados es de 21.54%**

La confidencialidad del prototipo incrementa en un 8.24 % utilizando componentes personalizados con la respectiva seguridad con respecto al no utilizar componentes personalizados como se visualiza en la figura IV.1.

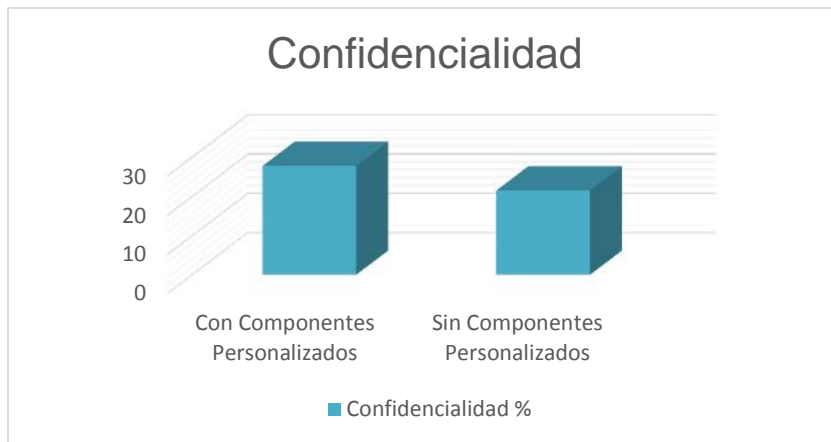


Figura IV.2: Comparación de la Confidencialidad

Indicador 2: Disponibilidad

Se refiere al acceso a la información y a los sistemas por personas autorizadas en el momento que así lo requieran.

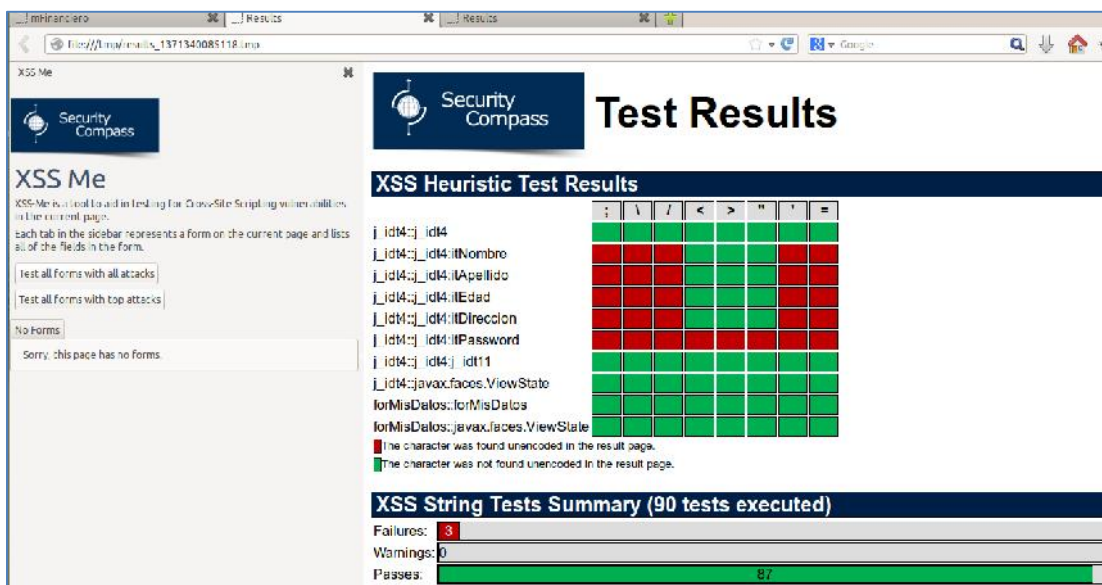


Figura IV.3: Escaneo de vulnerabilidad con XSS Me

Tabla IV.6: Resumen de escaneo de vulnerabilidades de Disponibilidad

Herramientas	Vulnerabilidad	PSCP		PCCP	
		nV	gV	nV	gV
Wapiti	XSS	0	3	0	3
Acunetix	XSS	5	3	0	3
XSS Me	XSS	5	3	3	3

Fuente: Propuesta de Técnicas de Aseguramiento de Aplicaciones web desarrolladas en Java

En base a los resultados obtenidos mediante el escaneo de las vulnerabilidades con las herramientas, se aplica la fórmula 3.

Tabla IV.7: Aplicación de fórmula 3 para Disponibilidad

Herramienta	Prototipo Sin Componentes personalizados			Prototipo Con Componentes personalizados		
	nV	gV	$\frac{30}{1 + \sum(\frac{30}{vI + gV/10})}$	nV	gV	$\frac{30}{1 + \sum(\frac{30}{vI + gV/10})}$
Wapiti	0	3	30	0	3	30
Acunetix	5	3	12	0	3	30
XSS Me	5	3	12	3	3	18.42
	Promedio de Disponibilidad		18%	Promedio de Disponibilidad		26.14%

Fuente: Propuesta de Técnicas de Aseguramiento de Aplicaciones web desarrolladas en java

De acuerdo a los resultados obtenidos aplicando las herramientas de escaneo de vulnerabilidades como se visualiza en la tabla IV.4 se ha obtenido que: **la disponibilidad del prototipo utilizando componentes personalizados con la respectiva seguridad es de 26.14% y la disponibilidad del prototipo sin utilizar componentes personalizados es de 18%.**

La disponibilidad del prototipo incrementa en un 8.14 % utilizando componentes personalizados con la respectiva seguridad con respecto al no utilizar componentes personalizados como se visualiza en la figura IV.2.

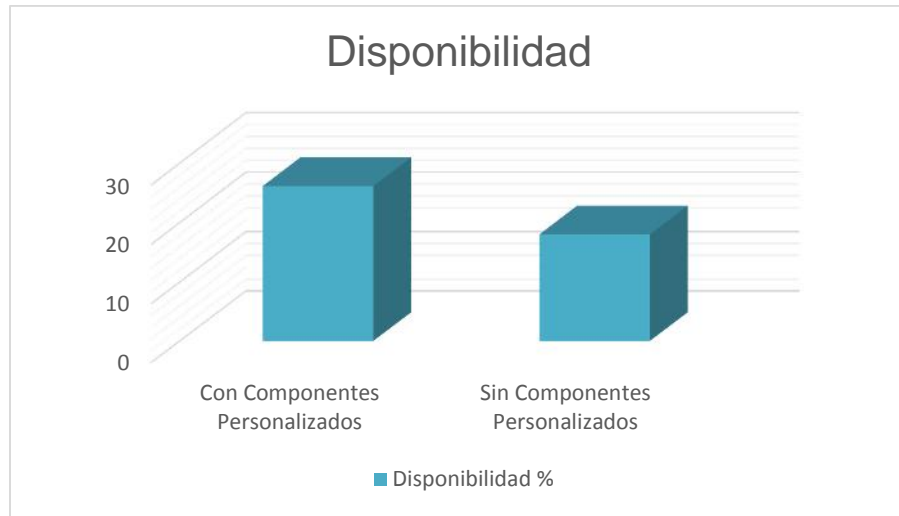


Figura IV.4: Comparación de la Disponibilidad

Indicador 3: Integridad

Se refiere al mantener con exactitud la información tal cual fue generada, sin ser manipulada o alterada por personas o procesos no autorizados.

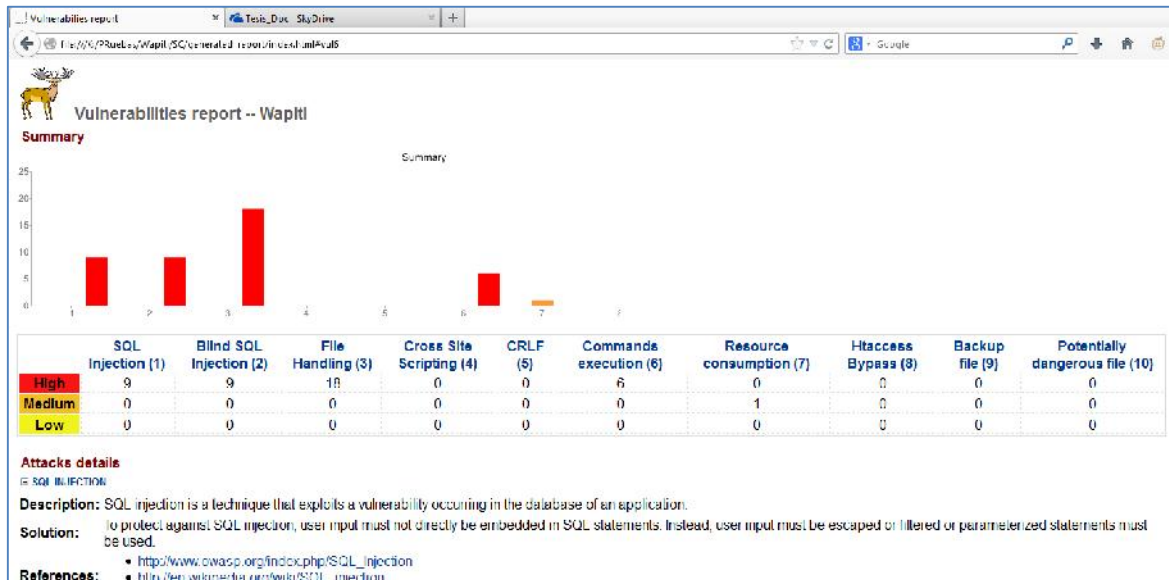


Figura IV.5: Escaneo de vulnerabilidades con Wapiti

Tabla IV.8: Resumen de escaneo de vulnerabilidades de Integridad

Herramientas	Vulnerabilidad	PSCP		PCCP	
		nV	gV	nV	gV
Wapiti	SQL Injection	9	3	2	3
	XSS	0	3	0	3
	CSRF	0	3	0	3
Acunetix	SQL Injection	7	3	2	3
	XSS	5	3	0	3
	CSRF	4	3	4	3
SQL Inject Me	SQL Injection	85	3	0	3
XSS Me	XSS	5	3	3	3
Scripts	Inyección de Código	5	3	0	3
CSRFTester	CSRF	0	3	0	3

Fuente: Propuesta de Técnicas de Aseguramiento de Aplicaciones web desarrolladas en Java

En base a los resultados obtenidos mediante el escaneo de las vulnerabilidades con las herramientas se aplica la fórmula 4.

Tabla IV.9: Aplicación de fórmula 3 para Integridad

Herramienta	Prototipo Sin Componentes personalizados			Prototipo Con Componentes personalizados		
	nV	gV	$\frac{35}{1 + \sum(\frac{35}{vD + gV/10})}$	nV	gV	$\frac{35}{1 + \sum(\frac{35}{vD + gV/10})}$
Wapiti	9	3	26.48	2	3	30.62
	0	3				
	0	3				
Acunetix	7	3	13.74	2	3	24.25
	5	3				
	4	3				
SQL Inject Me	85	3	1.32	0	3	35
XSS Me	5	3	14	3	3	18.42
Scripts	5	3	14	0	3	35
CSRFTester	0	3	35	0	3	35
	Promedio de Integridad		17.42%	Promedio de Integridad		29.72%

Fuente: Propuesta de Técnicas de Aseguramiento de Aplicaciones web desarrolladas en java

De acuerdo a los resultados obtenidos aplicando las herramientas de escaneo de vulnerabilidades como se visualiza en la tabla IV.4 se ha obtenido que: **la integridad del prototipo utilizando componentes personalizados con la respectiva seguridad es de 29.72% y la integridad del prototipo sin utilizar componentes personalizados es de 17.42%.**

La integridad del prototipo incrementa en un 12.3 % utilizando componentes personalizados con la respectiva seguridad con respecto al no utilizar componentes personalizados como se visualiza en la figura IV.3.



Figura IV.6: Comparación de la Integridad

3.12. Resumen General de Equivalencias

En base a los valores obtenidos de cada uno de los indicadores, se los reemplaza en la fórmula 1 la cual es de la seguridad, para el prototipo con componentes personalizados y para el prototipo sin componentes personalizados respectivamente.

Prototipo Con Componentes Personalizados

$$PCCP = C PCCP + D PCCP + I PCCP$$

$$PCCP = 29.78 + 26.14 + 29.72$$

$$PCCP = 85.64\%$$

Prototipo Sin Componentes Personalizados

$$PSCP = C PSCP + D PSCP + I PSCP$$

$$PSCP = 21.54 + 18 + 17.42$$

$$PSCP = 56.96\%$$

En la tabla IV.10 se tiene la síntesis de los indicadores con sus respectivos porcentajes, para el prototipo con componentes personalizados y para el prototipo sin componentes personalizados.

Tabla IV.10: Síntesis general de porcentajes

Indicadores	Prototipo Con Componentes Personalizados	Prototipo Sin Componentes Personalizados
Confidencialidad (%)	29.78 %	21.54 %
Disponibilidad (%)	26.14 %	18 %
Integridad (%)	29.72 %	17.42 %
Total (100%)	85.64%	56.96%

Elaborado por: Gustavo Parco, Jenny Perez

En la figura IV.4 se visualiza la variación de los indicadores con sus respectivos porcentajes, para el prototipo con componentes personalizados y para el prototipo sin componentes personalizados.

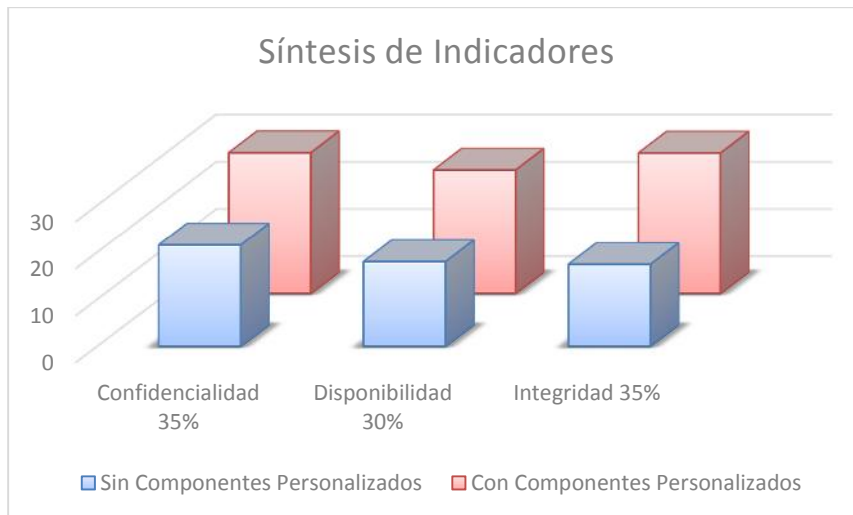


Figura IV.7 Síntesis de Indicadores

3.13. Prueba de Hipótesis

Valoración del Nivel de Seguridad

Para la valoración del nivel de seguridad se lo realiza en base a la tabla IV.11, la cual se ha estimado de la siguiente manera:

Tabla IV.11: Valoración del Nivel de Seguridad

Rango (%)	Equivalencia
0 a < 30	No Adecuado
30 a < 60	Medio Adecuado
60 a 100	Adecuado

Fuente: Metodología de Clasificación de riesgo OWASP

Al realizar la tabulación de los resultados obtenidos se ha obtenido que al utilizar JSF+ PrimeMobile y los componentes personalizados con la seguridad, se obtiene un porcentaje de 85.64% y al situar este valor de acuerdo a la tabla IV.11 anteriormente expuesta, el nivel de seguridad se sitúa en el rango entre 60 a 100 y equivalente a *Adecuado* respecto a utilizar componentes de JSF+ PrimeMobile se obtiene un porcentaje de 56.96% este nivel se sitúa en el rango de 30 a < 60 y es equivalente a *Medio Adecuado*, con lo cual se comprueba la hipótesis planteada.

Prototipo Con Componentes Personalizados = 85.64 %

Prototipo Sin Componentes Personalizados= 56.96%

Los resultados anteriormente mencionados se representan en la figura IV.8



Figura IV.8: Comparación de la Seguridad

Al realizar el análisis en los dos prototipos (prototipo con componentes personalizados y prototipo sin componentes personalizados) se ha verificado que al utilizar JSF+ PrimeMobile y los componentes personalizados con la seguridad el nivel de seguridad se incrementa en un porcentaje de 28.68%.

Aumento de Seguridad=85.64% - 56.96%

Aumento de Seguridad =28.68%

CAPITULO V

IMPLEMENTACIÓN DEL SISTEMA DE CONSULTAS Y MONITOREO DE PROCESOS DE TRÁMITE DE PAGO PARA EL DEPARTAMENTO FINANCIERO (ESPOCH)

4.1. Metodología XP (Xtreme Programming)

El proceso de ingeniería de software se efectúa con la utilización de la metodología XP, esta es una metodología ágil, ya que estas metodologías se promueve la comunicación cara a cara entre los miembros del proyecto y el cliente; en especial XP porque permite pequeñas mejoras, unas tras otras y la simplicidad en el código, lo que lleva a una comunicación más completa, especialmente si se puede reducir el equipo de programadores.

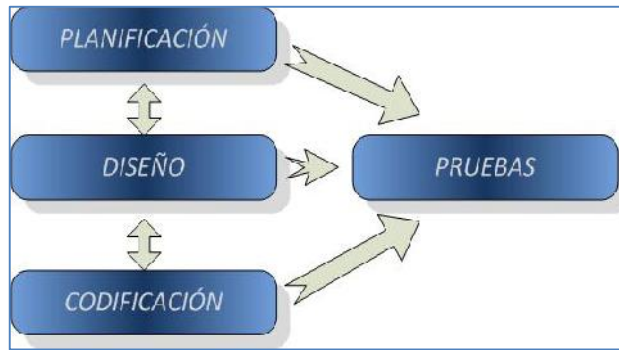


Figura V.1: Fases de la Metodología XP

4.1.1. Fase de Planificación

4.1.1.1. Planificación Inicial

La planificación del proyecto se realizó mediante el estudio del problema y los requerimientos. Mediante la redacción de las historias de usuario se efectuó la planificación inicial la cual fue variando en el transcurso de la misma, cambiando y mejorando las historias en base a concepción del problema. A continuación en la Tabla V.I se visualiza el equipo de trabajo con los integrantes y los roles.

Tabla V.1: Equipo de Trabajo

Miembro	Grupo	Roles XP	Metodología
Gustavo Parco	Tesista	Rastreador, Testeador, Programador	XP
Jenny Perez	Tesista	Rastreador, Testeador, Programador	
Ing. Gloria Arcos Ing. Diego Palacios	Consultor	Entrenador	

Elaborado por: Jenny Perez, Gustavo Parco

Recolección de Información

Para poder recolectar información acerca de cómo se realiza desde la fase inicial en la cual se recogen los documentos hasta la fase final que es la de la aprobación del trámite y ya puede el

usuario cobrar su comprobante de pago, se nos proporciona información a modo de entrevista de la cual se obtiene valiosa información para el desarrollo e implementación del sistema de consulta y monitoreo de trámites de pago.

- ¿Qué requisitos se requieren presentar para realizar los trámites (bienes y servicios)?
- ¿Qué trámites se realizan con más frecuencia en el departamento?
- ¿Cómo es que se logra la aprobación de un trámite?
- ¿Cuáles son las fases por las que debe pasar un comprobante?
- ¿En qué fase se demora más un comprobante?
- ¿Con qué frecuencia los usuarios se acercan a averiguar sobre el avance/proceso de sus comprobantes?
- ¿Qué tiempo estimado se demora en aprobarse los trámites del usuario?
- ¿Para averiguar el avance del trámite en base a que lo hace aun nombre o un número del comprobante, en caso de no acordarse del código del comprobante como se lo ayuda al usuario?
- ¿A parte de la información acerca del avance de su trámite que información más desea obtener las personas que hacen las averiguaciones?
- ¿Qué tipo de información sería importante o le gustaría que fuera expuesta a los usuarios?

4.1.1.2. Historias de Usuario

A continuación se visualiza en la Tabla V.2 las historias de usuario con la prioridad que representa en el desarrollo, así como el riesgo y esfuerzo de cada una de ellas al ser implementadas, también se indica en la iteración que se ha planeado ser implementada.

Tabla V.2: Historias de usuario

Nº	Nombre	Prioridad	Riesgo	Esfuerzo	Iteración
1	Consulta de Datos Personales	Media	Bajo	Bajo	1
2	Consultar Datos de Empresas	Media	Bajo	Bajo	1

Nº	Nombre	Prioridad	Riesgo	Esfuerzo	Iteración
3	Consultar Datos de Instituciones Financieras	Media	Bajo	Bajo	1
4	Comprobantes de pago	Alto	Medio	Medio	2
5	Buscar Comprobantes de pago	Alto	Medio	Medio	2
6	Mostrar Todos los Comprobantes De Pago	Medio	Bajo	Medio	2
7	Buscar Comprobantes por Titular	Alto	Bajo	Medio	2
8	Buscar Comprobantes por Titular Empresa	Alto	Bajo	Medio	2
9	Búsqueda Mostrar Todos	Medio	Bajo	Medio	2
10	Buscar Comprobantes por Fechas	Medio	Bajo	Medio	2
11	Buscar Comprobantes Anulados	Alto	Bajo	Medio	2
12	Buscar Comprobantes por cobrar.	Medio	Bajo	Medio	2
13	Seguimiento al Comprobante de Pago.	Alto	Bajo	Medio	2
14	Fase Actual de Comprobante de Pago	Alto	Bajo	Medio	2
15	Seguimiento Detallado de Comprobante de Pago	Alto	Bajo	Medio	2
16	Requisitos Solicitados del Comprobante de Pago	Alto	Bajo	Medio	2
17	Comprobantes de Venta de Comprobante de Pago	Bajo	Bajo	Medio	2
18	Tiempo Total De Comprobante de Pago	Medio	Medio	Medio	2
19	Fase que más se Demora Comprobante de Pago	Medio	Bajo	Medio	2
20	Notificaciones de Comprobantes por Cobrar	Medio	Bajo	Medio	3
21	Sumatoria de Comprobantes Cobrados	Medio	Bajo	Medio	3
22	Consultas Generales	Bajo	Bajo	Bajo	3

Nº	Nombre	Prioridad	Riesgo	Esfuerzo	Iteración
23	Requisitos según Bien o Servicio	Alto	Bajo	Bajo	3
24	Información de Fases del Comprobante de Pago	Medio	Bajo	Bajo	3
25	Datos para llenar facturas	Bajo	Bajo	Bajo	3
26	Ayuda	Medio	Bajo	Bajo	3
27	Vista Principal del Sistema.	Alto	Bajo	Medio	1
28	Autenticación de usuarios	Alto	Bajo	Medio	1

Elaborado por: Jenny Perez, Gustavo Parco

Para la valoración de la prioridad, riesgo y del esfuerzo se ha tomado los siguientes valores: Alto, Medio y Bajo.

4.1.1.3. Planificación de Publicaciones:

4.1.1.3.1. Iteraciones

Iteración Primera: En esta iteración se pretende implementar las siguientes historias de usuario como se visualiza en la Tabla V.3:

Tabla V.3: Plan de entrega iteración 1

Nº	Historia de Usuario	Duración en semanas
H1	Consulta de Datos Personales	1,5
H2	Consultar Datos de Empresas	1
H3	Consultar Datos de Instituciones Financieras	1
H27	Vista Principal del Sistema.	1,5
H28	Autenticación de usuarios	2

Elaborado por: Jenny Perez, Gustavo Parco

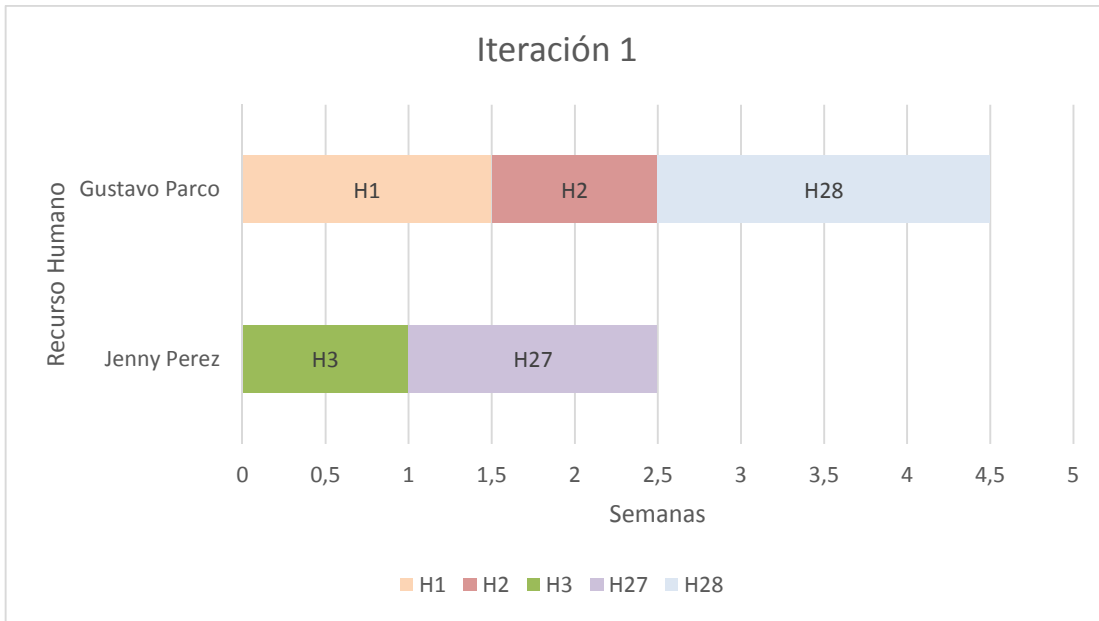


Figura V.2: Plan de entrega Iteración 1

Iteración Segunda: En esta iteración se pretende entregar las siguientes historias de usuarios como se visualiza en la Tabla V.4:

Tabla V.4: Plan de entrega Iteración 2

N°	Historia de Usuario	Duración en semanas
H4	Comprobantes de pago	0,5
H5	Buscar Comprobantes de pago	0,5
H6	Mostrar Todos los Comprobantes De Pago	0,5
H7	Buscar Comprobantes por Titular	0,5
H8	Buscar Comprobantes por Titular Empresa	0,5
H9	Búsqueda Mostrar Todos	0,5
H10	Buscar Comprobantes por Fechas	1
H11	Buscar Comprobantes Anulados	0,5
H12	Buscar Comprobantes por cobrar.	0,5
H13	Seguimiento al Comprobante de Pago.	0,5
H14	Fase Actual de Comprobante de Pago	0,5
H15	Seguimiento Detallado de Comprobante de Pago	1

N°	Historia de Usuario	Duración en semanas
H16	Requisitos Solicitados del Comprobante de Pago	0,5
H17	Comprobantes de Venta de Comprobante de Pago	0,5
H18	Tiempo Total De Comprobante de Pago	1
H19	Fase que más se Demora Comprobante de Pago	0,5

Elaborado por: Jenny Perez, Gustavo Parco

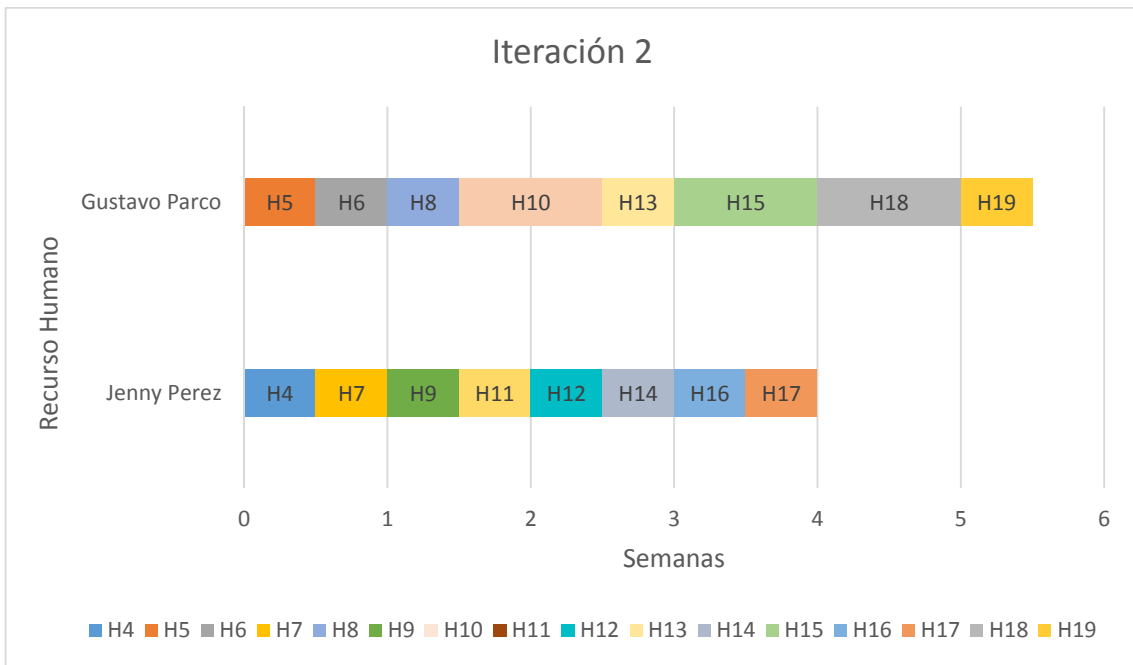


Figura V.3: Plan de entrega Iteración 2

Iteración Tercera: En esta iteración se pretende entregar las siguientes historias de usuarios como se visualiza en la Tabla V.5:

Tabla V.5: Plan de entrega Iteración 3

N°	Historia de Usuario	Duración en semanas
20	Notificaciones de Comprobantes por Cobrar	1
21	Sumatoria de Comprobantes Cobrados	1
22	Consultas Generales	0,5
23	Requisitos según Bien o Servicio	0,5
24	Información de Fases del Comprobante de Pago	0,5
25	Datos para llenar facturas	0,5
26	Ayuda	0,5

Elaborado por: Jenny Perez, Gustavo Parco

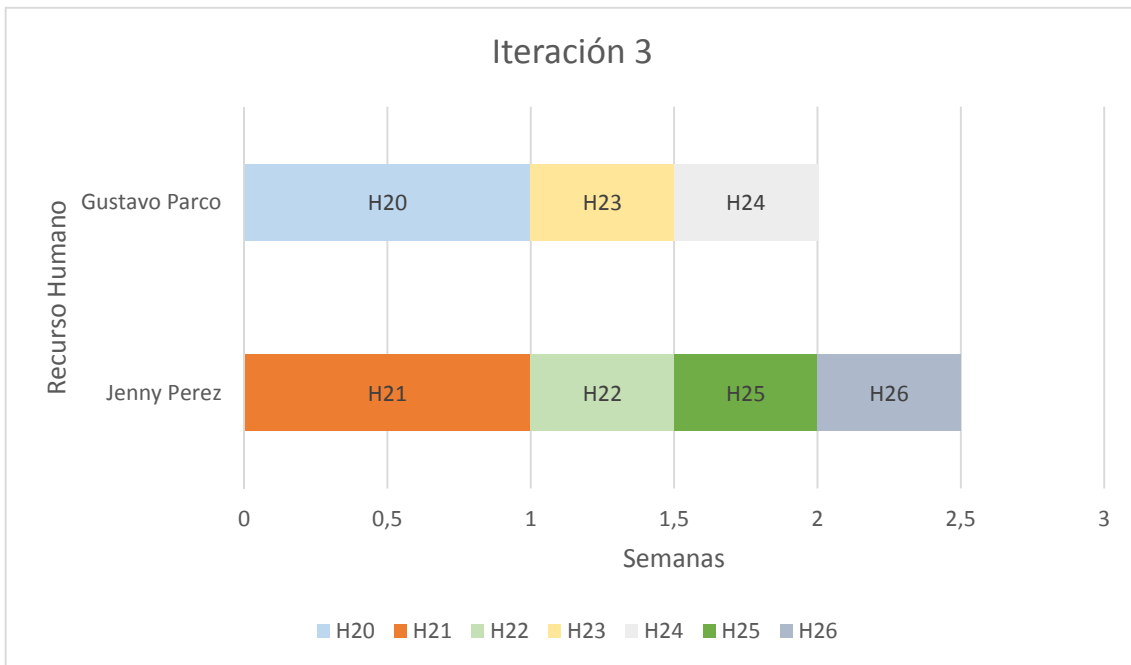


Figura V.4: Plan de entrega Iteración 3

A continuación se tiene las historias de usuario las cuales se han ubicado en base a la planificación inicial de las iteraciones. Se ha tomado una historias de usuario de cada una de las iteraciones si se desea conocer las restantes por favor dirigirse al manual técnico de la aplicación.

Iteración 1: La historia de usuario nº 1 se refiere a la consulta de datos personales del usuario como se visualiza en la Tabla V.6.

Tabla V.6: Historia de usuario Consulta Datos Personales

Historia de Usuario	
Número: 1	Usuario: Titular
Nombre historia: Consulta de Datos Personales	
Prioridad en negocio: Media	Riesgo en desarrollo: Bajo
Esfuerzo: Bajo	Iteración asignada: 1
Programador responsable:	
Descripción: El usuario puede visualizar sus datos personales y verificar si estos son los apropiados.	
Observaciones:	

Elaborado por: Gustavo Parco, Jenny Perez

Iteración 2: La historia de usuario nº4 se refiere a las consultas avanzadas sobre los comprobantes de pago del usuario como se visualiza en la Tabla V.7.

Tabla V.7: Historia de usuario Comprobante de pago

Historia de Usuario	
Número: 4	Usuario: Titular
Nombre historia: Comprobantes de pago	
Prioridad en negocio: Alta	Riesgo en desarrollo: Medio
Esfuerzo: Medio	Iteración asignada: 2
Programador responsable:	
Descripción: El usuario se autentica en el sistema, selecciona la opción del menú referente a Comprobantes de pago y desde esta opción puede realizar las diferentes consultas avanzadas para buscar y determinar en qué estado se encuentra determinado comprobante de pago, visualizar la fase por la cual está atravesando, si están listos para ser cobrados, comprobantes de pago de persona o de empresa anulados.	
Observaciones:	

Elaborado por: Gustavo Parco, Jenny Perez

Iteración 3: La historia de usuario nº20 se refiere a la consulta de notificaciones acerca de comprobantes de pago listos para ser cobrados como se visualiza en la Tabla V.8.

Tabla V.8: Historia de usuario Notificaciones de Comprobantes por Cobrar

Historia de Usuario	
Número: 20	Usuario: Titular
Nombre historia: Notificaciones de Comprobantes por Cobrar	
Prioridad en negocio: Media	Riesgo en desarrollo: Bajo
Esfuerzo: Medio	Iteración asignada: 3
Programador responsable:	
Descripción: En el ingreso a la vista principal se le notifica al usuario que tiene comprobantes de pago listos para ser cobrados.	
Observaciones:	

Elaborado por: Jenny Perez Gustavo Parco

4.1.2. Fase de Diseño

4.1.2.1. Descripción del Diseño de Base de Datos

A continuación se visualiza en la figura V.5 el diseño de la base de datos para el desarrollo del sistema de Consultas de Trámite de Pago Web Móvil, la base de datos cuenta con los esquemas master y gastos con los cuales se trabajó para la implementación del sistema.

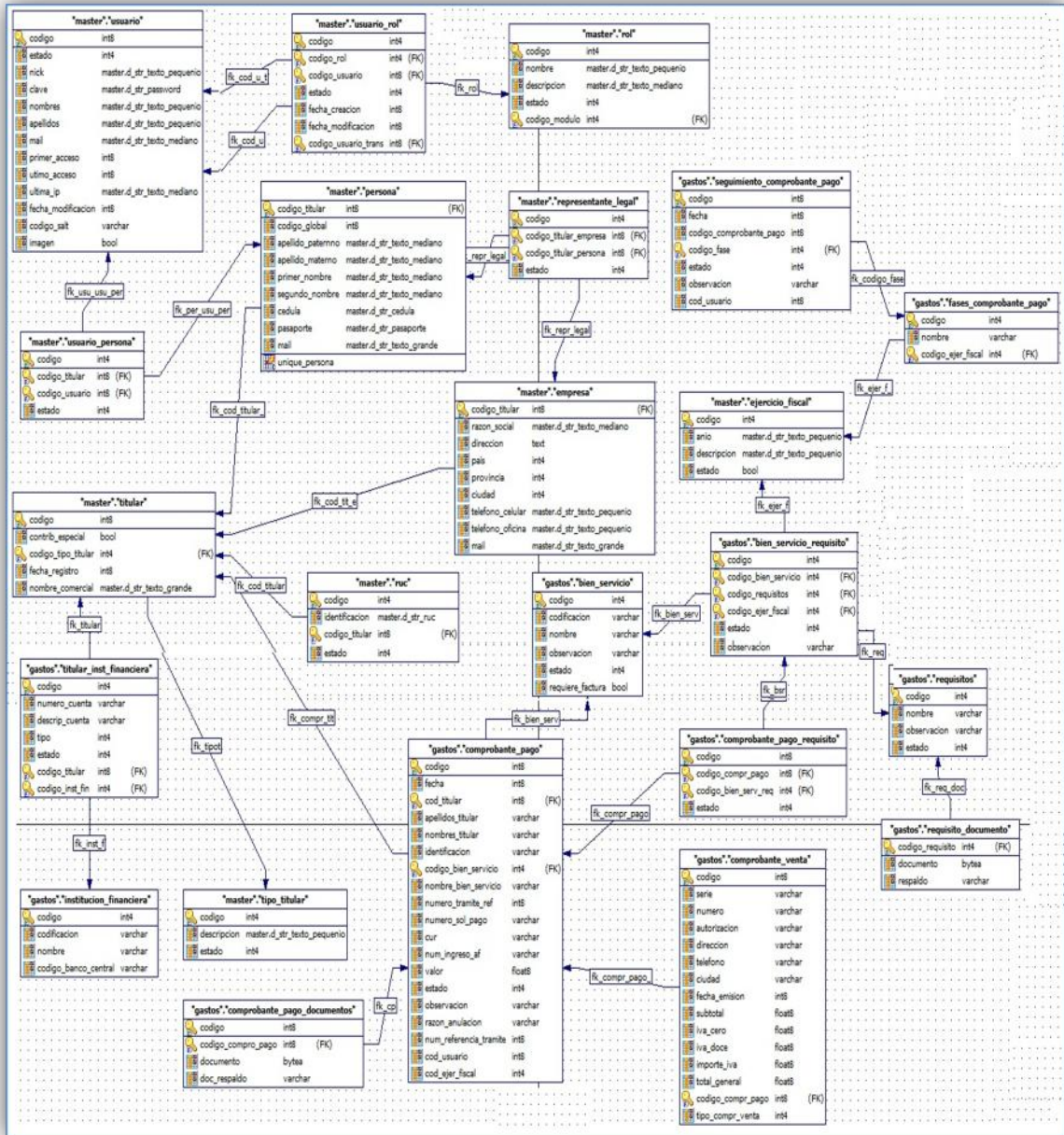


Figura V.5: Diseño de base de Datos "Financiero"

- Es importante preservar la información del usuario y que esta solo sea visible a quien corresponda únicamente, por ello es necesario la autenticación del usuario (tablas involucradas: Usuario, usuario_persona, persona), el modelo de datos para este proceso se visualiza

continuación en la Figura V.6; si se desea conocer a cerca de ello por favor dirigirse al manual técnico de la aplicación:

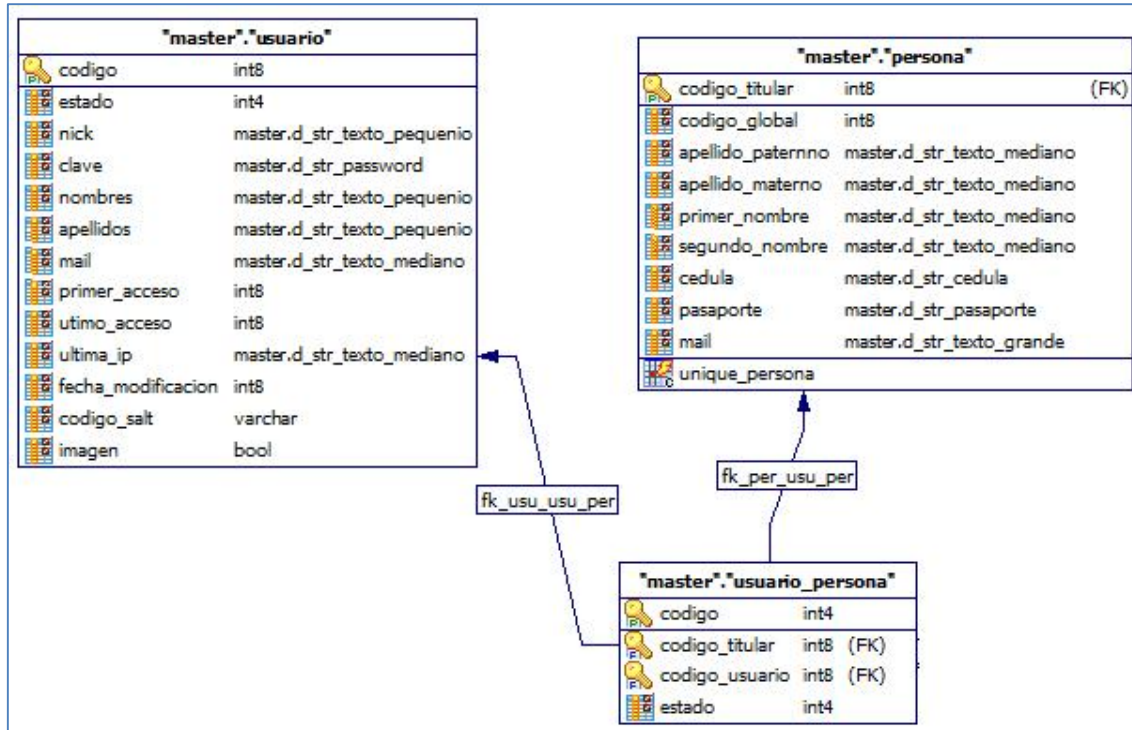


Figura V.6: Modelo de Datos para Autenticación

4.1.2.2. Diseño de Interfaces

El diseño de las vistas para la aplicación web móvil se propone como se visualiza a continuación el cual puede ir cambiando en el transcurso del desarrollo. Se ha tomado una de las imágenes como ejemplo, si se desea conocer las restantes por favor dirijase al manual técnico de la aplicación.

- El punto de partida tanto para usuarios autenticados y aquellos que deseen visitar el sistema de pago web móvil se propone el diseño del interfaz gráfica en la Figura V.7:



Figura V.7: Interfaz Gráfica del Menú principal

4.1.2.3. Diagramas de Caso de Uso

Los casos de uso se desarrollan para poder definir los límites del sistema y las relaciones del sistema y el entorno.

- **Diagrama de Caso de Uso:** A continuación se visualiza en la Figura V.8 el diagrama de caso de uso el cual involucra al rol "Usuario Titular" y la interacción del mismo y el sistema, en este rol se encuentra los usuarios que poseen su respectivo usuario y contraseña.

Si se desea conocer de los restantes casos de uso por favor diríjase al manual técnico de la aplicación.

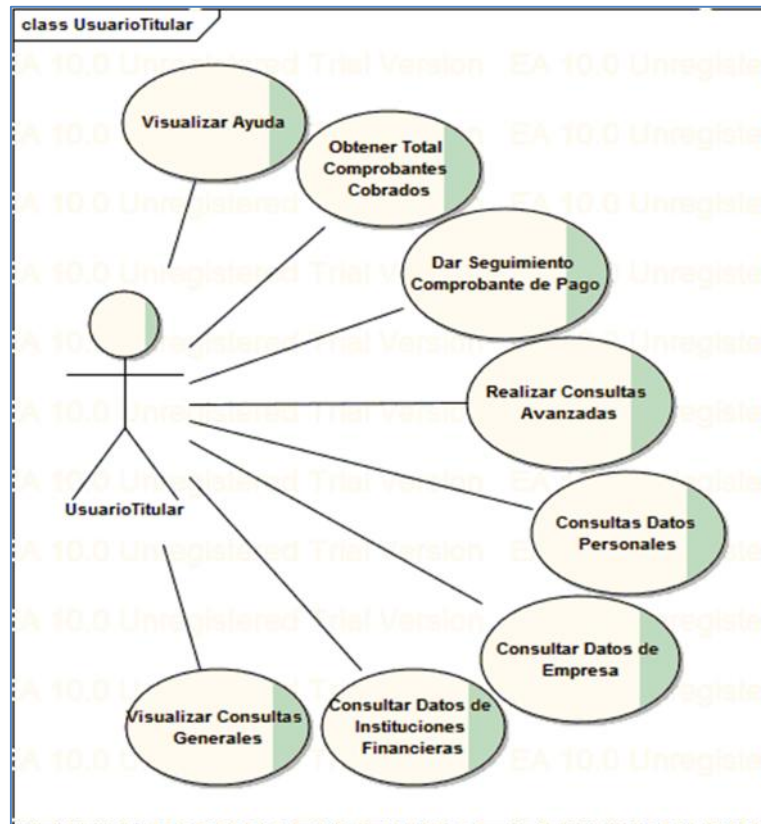


Figura V.8: Diagrama de caso de uso para el rol Usuario Titular

Casos de Uso Detallado: A continuación se visualiza en la Tabla V.9 el comportamiento para la autenticación de usuarios.

Caso De Uso Autenticación_De_Usuarios

Tabla V.9: Caso de uso Detallado Autenticación de Usuarios

Identificador de caso de uso	CU_AUTENTICACION	
Nombre del caso de uso	Autenticación de usuarios	
Actores	Usuario Titular	
Propósito	Permitir el acceso a información crítica únicamente al usuario apropiado	
Visión general	El usuario ingresa a la aplicación, y se autentica ingresando su usuario y contraseña, y realiza sus consultas.	
Tipo	Primario, esencial	
Referencias	Funciones: Historia 28	
Curso típico de eventos		
Acciones del Actor	Respuesta del sistema	
<ul style="list-style-type: none"> ▪ Usuario ingresa al sistema 		
<ul style="list-style-type: none"> ▪ Selecciona Ingresar 	<ul style="list-style-type: none"> ▪ Despliega vista Login 	
	<ul style="list-style-type: none"> ▪ Solicita usuario y contraseña 	
<ul style="list-style-type: none"> ▪ Ingresar usuario y contraseña 	<ul style="list-style-type: none"> ▪ Valida los datos proporcionados. 	
	<ul style="list-style-type: none"> ▪ Despliega vista de consultas. 	
Cursos alternativos		
Línea 6: Mensaje: "Error de validación: se necesita un valor. Usuario:", si no ingresa nombre de usuario.		

Línea 6: Mensaje: "Error de validación: se necesita un valor. Contraseña:", si no ingresa contraseña.

Línea 6: Mensaje: "Datos incorrectos: se necesita un valor. Usuario:", si no ingresa nombre de usuario o contraseña.
--

Elaborado por: Gustavo Parco, Jenny Perez

4.1.2.4. Diagrama de Procesos

En la Figura V.9 se visualiza el proceso inicial para realizar consultas generales, si se desea conocer los diagramas de flujo de procesos restantes por favor dirigirse al manual técnico de la aplicación.

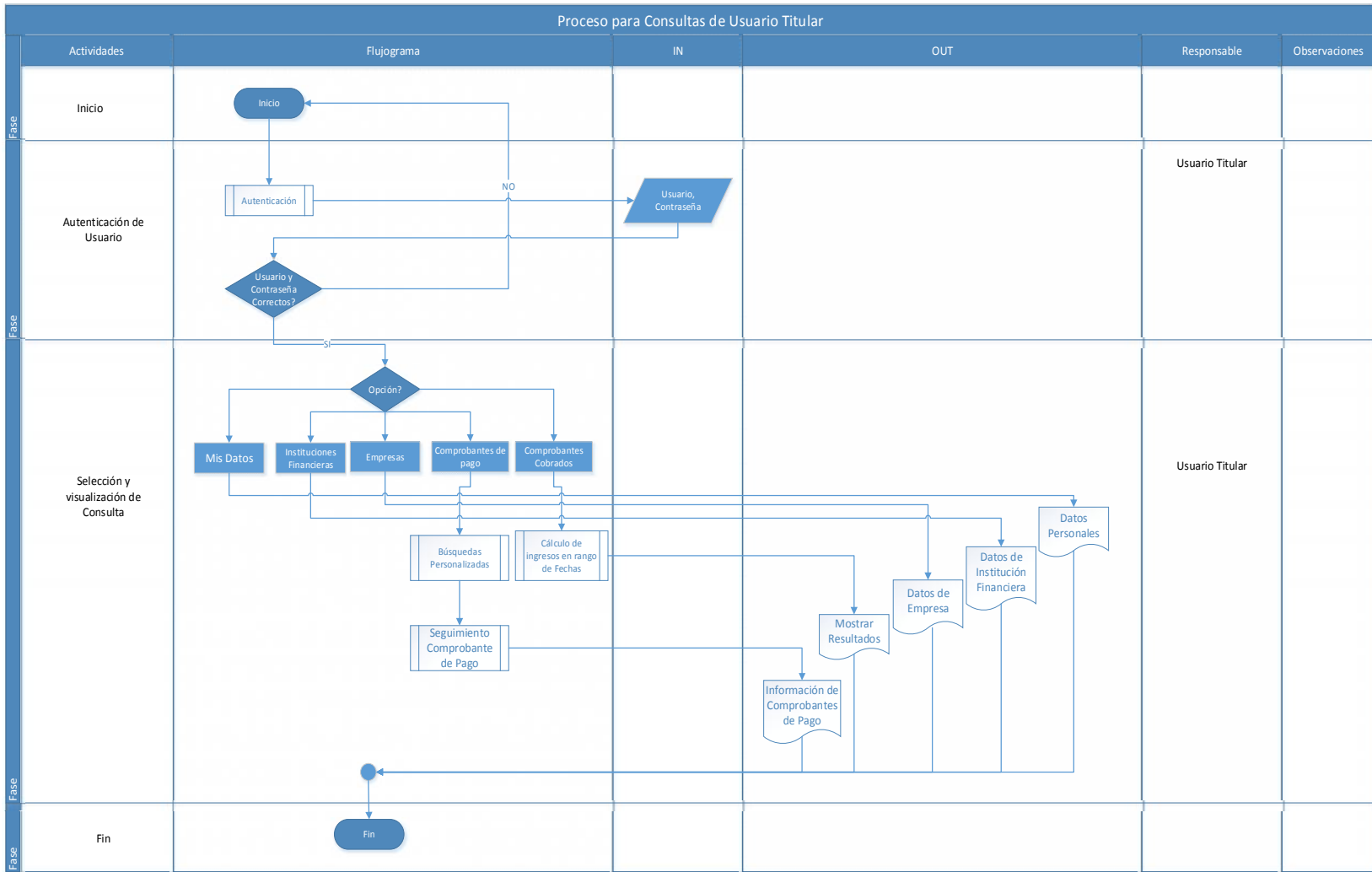


Figura V.9: Flujo de Procesos para realizar consultas con los Comprobantes de Pago

4.1.2.5. Arquitectura

Mediante la Figura V.10 se visualiza la arquitectura de la aplicación web móvil “Consultas Y Monitoreo De Trámites De Pago Web Móvil”:



Figura V.10: Arquitectura de la aplicación web móvil

4.1.2.6. Tareas de las Historias de Usuario

A continuación se visualizan las tareas realizadas para la implementación de las historias de usuario. Se ha tomado como ejemplo una de las tareas, si se desea conocer las restantes por favor diríjase al manual técnico de la aplicación.

La tarea nº 2 se refiere a las tareas para la Implementación de clases, funciones, controladores; estas se visualizan en la Tabla V.10.

Tabla V.10: Tarea de implementación de clases, funciones, controladores

Tarea	
Número de tarea: 2	Número de historia: 23
Nombre de tarea: Implementación de clases, funciones, controladores	
Tipo de tarea: Desarrollo	Esfuerzo: 3
Fecha inicio: 26-09-2012	Fecha fin: 28-09-2012
Programador responsable: Gustavo Parco	
Descripción: Implementación de las clases, funciones, controladores y mapeo de las consultas para lista de requisitos, requisitos por bien servicio.	

Elaborado por: Gustavo Parco, Jenny Perez

4.1.3. Fase de Codificación

En esta fase se realiza la codificación de las historias de usuarios, las cuales fueron expuestas por el cliente.

- En base al estándar sugerido por los Ingenieros del Desitel se ha codificado las clases, el mapeo de clases y funciones se lo ha ordenado en paquetes.

4.1.3.1. Paquetes

- El paquete “accesodatos” contiene la configuración para la conexión a la base de datos.
- EL paquete “gastos.logica.clases” y “master.logica.clases” contiene todas las clases que forman del sistema de Consultas y Monitoreo de Tramites de Pago Móvil. Los nombres de las clases inician con letra mayúscula seguida de letras minúsculas y coinciden con los nombres de las tablas de la base de datos como: “public class Bien Servicio”, los nombres de los atributos de las clases coinciden con los atributos de las tablas de la base de datos y también el tipo de dato, cada uno de los atributos de las clases tienen codificado los respectivos setter y getter.
- El paquete “gastos.logica.funciones” y “master.logica.funciones” contiene todas las funciones de las clases para obtener los datos desde la base de datos. Los nombres de las funciones empiezan con la letra F mayúscula para indicar que es función seguida del nombre de la clase y en letras minúsculas como “public class FBienServicio”. Estas funciones contienen los métodos en base a las consultas implementadas en la base de datos.
- El paquete “gastos.presentacion.beans” contiene los controladores de las clases. Los nombres de los controladores Empiezan con letra mayúscula, seguida de letra minúsculas al final del nombre lleva la letra C mayúscula la cual indica que es controlador así: “public class ConsultaBien ServicioC”.

Si se desea conocer o explorar acerca de los paquetes por favor diríjase al código fuente de la aplicación que se encuentra en el cd.

4.1.3.2. Diagrama de Despliegue

El diagrama de despliegue de la aplicación web móvil “Consultas Y Monitoreo De Trámites De Pago Web Móvil” se visualiza en la Figura IV.10.

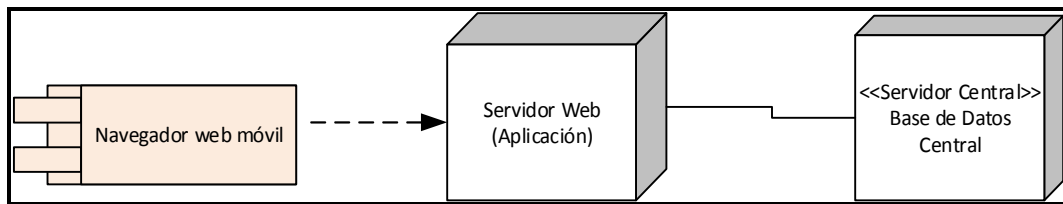


Figura V.11: Diagrama de despliegue

4.1.3.3. Diagrama de Componentes

El diagrama de componentes correspondiente a la aplicación web móvil “Consultas Y Monitoreo De Trámites De Pago Web Móvil” se visualiza en la Figura IV.11.

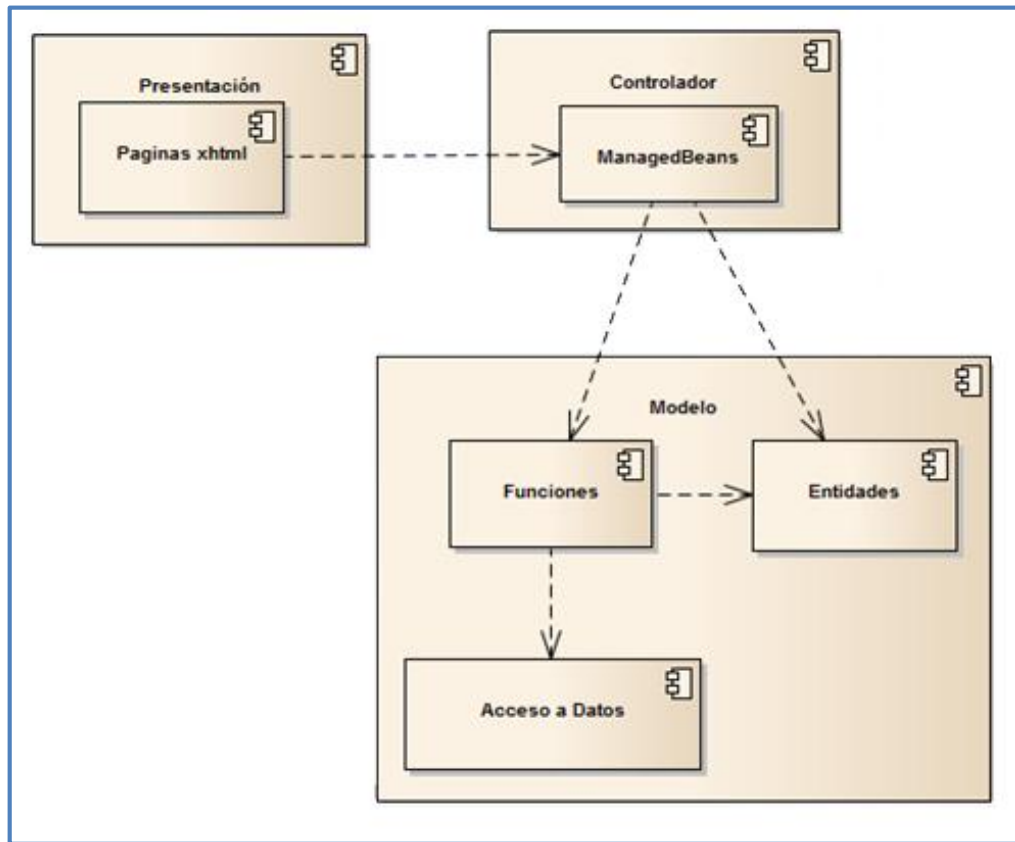


Figura V.12: Diagrama de Componentes

4.1.4. Fase de Pruebas

Para la comprobación del correcto funcionamiento del sistema se realizan pruebas sobre el código implementado. A continuación se describe las pruebas realizadas sobre las historias de usuarios. Se ha tomado la Autenticación correcta y la autenticación incorrecta de usuarios, si se desea conocer más acerca de ello por favor dirigirse al manual técnico de la aplicación en la sección “Fase de Pruebas”.

4.1.4.1. Pruebas Funcionales

A continuación se describe las pruebas funcionales relacionadas a las historias de usuario referentes a los Usuarios autenticados.

- **Historia de usuario: *Autenticación correcta de usuarios.***
 - **Descripción:** El usuario una vez que ha ingresado a la vista principal de la aplicación seleccionará la opción “Ingresar”, se desplegará la vista *Login* donde el usuario ingresará el Usuario y la Contraseña, internamente se verificará si estos datos son los correctos y posteriormente ingresará al sistema.
 - **Condiciones de Ejecución**
 - El usuario deberá estar registrado en el sistema.
 - **Entrada**
 - El usuario seleccionará la opción Ingresar
 - En la vista que se despliega tras la selección ingresará el usuario y la contraseña y presionará el botón Ingresar.
 - Internamente se verificará si el usuario está registrado e ingresará a realizar sus consultas.
 - El proceso de autenticación se considera finalizado.
 - **Resultado Esperado**
 - Tras el ingreso de usuario y contraseña, si el procesado ha sido correcto en la base de datos se registrarán datos de su ingreso y el usuario podrá realizar sus consultas.
 - **Evaluación de la prueba**
 - Prueba satisfactoria

- **Historia de usuario: *Autenticación incorrecta de usuarios.***
 - **Descripción:** El usuario una vez que ha ingresado a la vista principal de la aplicación seleccionará la opción “*Ingresar*” entonces se desplegará la vista Login donde el usuario ingresará el usuario y la contraseña, internamente se procesará los datos proporcionando y se verificará un error en los mismos para lo cual se avisará al usuario mediante un mensaje “Datos Incorrectos”.
 - **Condiciones de ejecución**
 - El usuario proporcionará datos incorrectos.
 - **Entrada**
 - El usuario seleccionará la opción Ingresar.
 - En la vista que se despliega tras la selección ingresará el usuario y la contraseña y presionará el botón Ingresar.
 - Internamente se verificará si el usuario está registrado y en el caso que ocurra un error se le indicará mediante un mensaje indicando que los datos son incorrectos.
 - El proceso de autenticación se considera finalizado.
 - **Resultado esperado**
 - Si el nombre de usuario o contraseña son incorrectos no puede ingresar al sistema.
 - **Evaluación de la prueba**
 - Prueba satisfactoria

4.1.5. Pruebas de Stress y Rendimiento

Mediante la utilización de la herramienta JMeter se realiza las pruebas en la aplicación web móvil, ya que esta herramienta está diseñada para realizar pruebas de rendimiento y pruebas funcionales sobre aplicaciones Web. También se trabaja con la herramienta BadBoy la cual permite grabar navegaciones Web y usarlas en pruebas de estrés sencillas

Para realizar las pruebas en la aplicación web móvil se realiza lo siguiente:

- Utilizando el software badBoy Se captura los parámetros implícitos que se transmiten en cada petición de las páginas xhtml, estas variables implícitas son: viewstate y jsessionID las cuales serán válidas para realizar las peticiones al servidor.
- Se graba una secuencia de navegación capturada como: autenticación del usuario, se visualiza datos del Titular, se realizan búsquedas avanzadas, se da seguimiento a un comprobante de pago; como se visualiza en la Figura V.13:

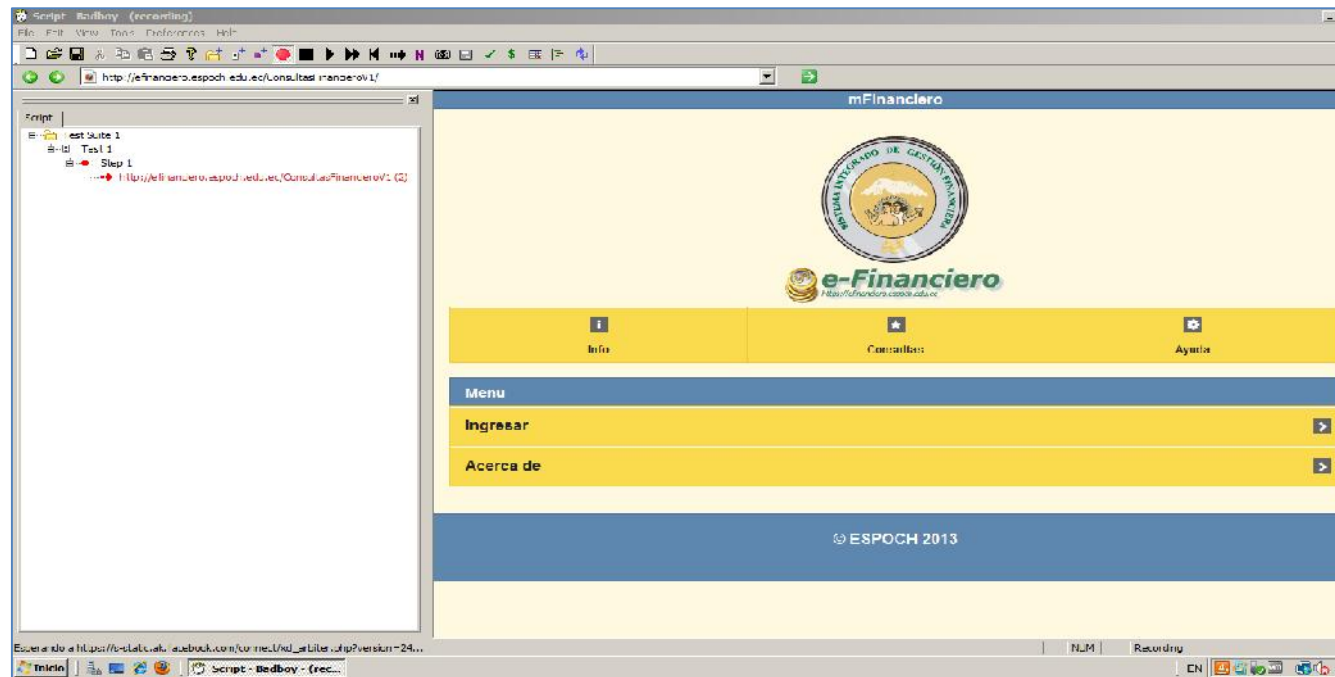


Figura V.13: Captura de variables con la herramienta badBoy

- Una vez que se ha capturado estas variables se exporta el script “*pruebas.jmx*” a JMeter.
- En JMeter se abre el script “*pruebas.jmx*” y mediante este procedemos a realizar las pruebas. Se ingresa en las propiedades: el número de “hilos” es decir el número de usuarios a simular, el período de subida (en segundos) es decir el tiempo que debiera llevarle a JMeter lanzar todos los hilos (por ejemplo si se seleccionan 10 hilos y el periodo de subida es de 100 segundos, entonces cada hilo comenzará 10 segundos después de que el hilo anterior haya sido lanzado) y el contador del bucle que es el número de veces a realizar el test. Para este test se utiliza 400 hilos y el periodo de subida es un segundo; esto se visualiza en la Figura V.14:

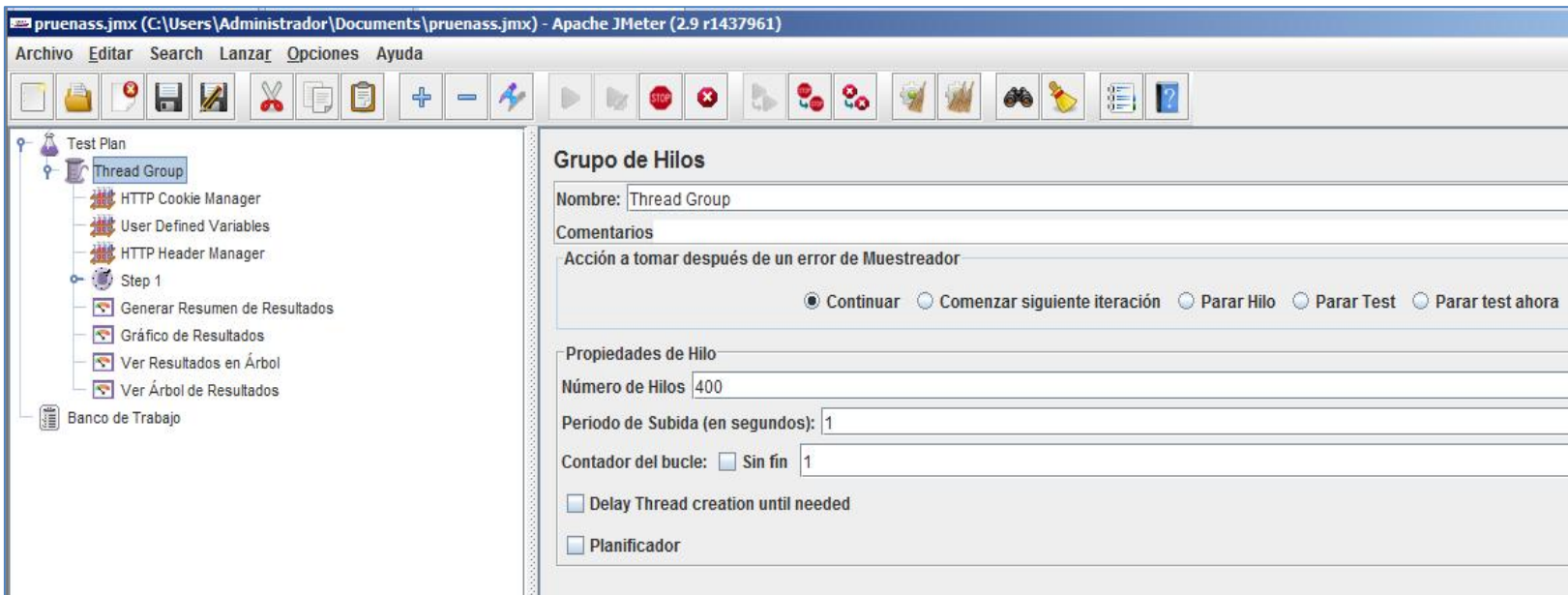


Figura V.14: Propiedades de Hilos en JMeter

Se selecciona “Reporte Resumen” y así ir observando cómo se van desarrollando los resultados, como se visualiza en la Figura V.15; este resumen crea una fila por cada petición y de cada una de estas filas se obtiene la siguiente información:

- **Etiqueta:** El nombre de la muestra (conjunto de muestras).
- **Muestras:** Cantidad total de veces que se realiza un request.
- **Media:** Media aritmética de los tiempos de respuesta (response time) de la aplicación.
- **Mediana:** Mediana aritmética de los tiempos de respuesta (response time) de la aplicación.
- **Línea de 90%:** Tiempo de respuesta en el que se encuentra el 90% de los requests.
- **Min:** Mínimo tiempo de respuesta para el request.
- **%Error:** porcentaje de request con errores
- **Rendimiento:** Es la cantidad de request que el servidor procesa por hora.
- **Kb/Sec:** Cantidad de Kb que el servidor procesa por segundo
- **Max:** Máximo tiempo de respuesta para el request.

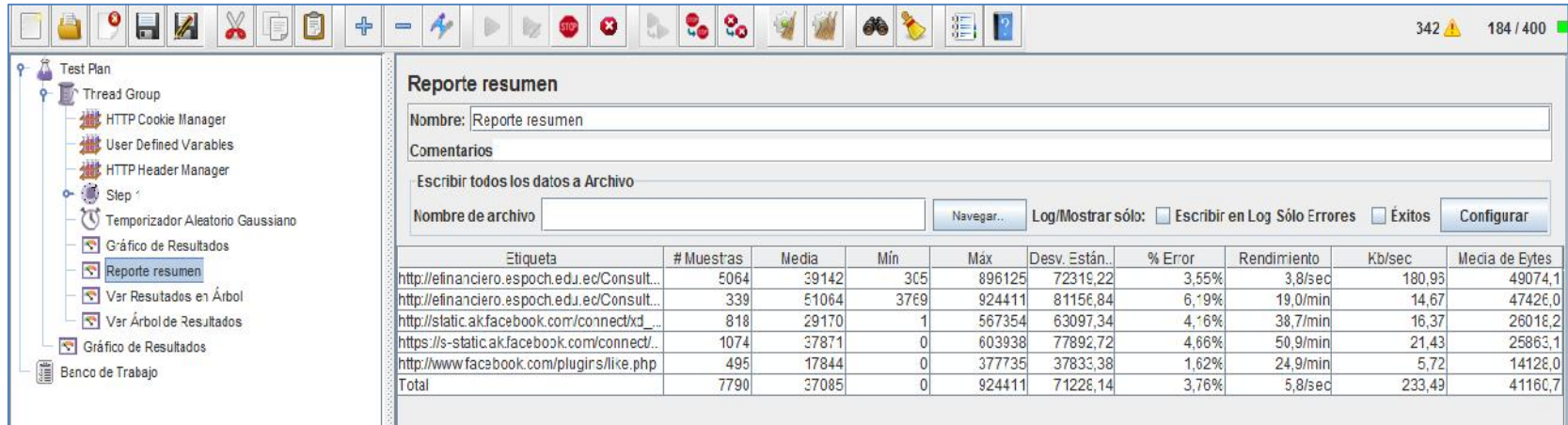


Figura V.15: Resultados obtenidos en JMeter

Podemos observar que las pruebas se ha realizado con cierto porcentaje de error como se observa en la columna “%Error” el cual se encuentra entre 3% y 6.5%. El rendimiento nos muestra que para la simulación con 400 usuarios relacionado en un tiempo de subida de un segundo el servidor es capaz de aceptar 5.8 peticiones por segundo. La latencia para cada conjunto de pruebas no supera el valor de 90950 milisegundos, esta se visualiza en el eje “y” de la figura V.16:

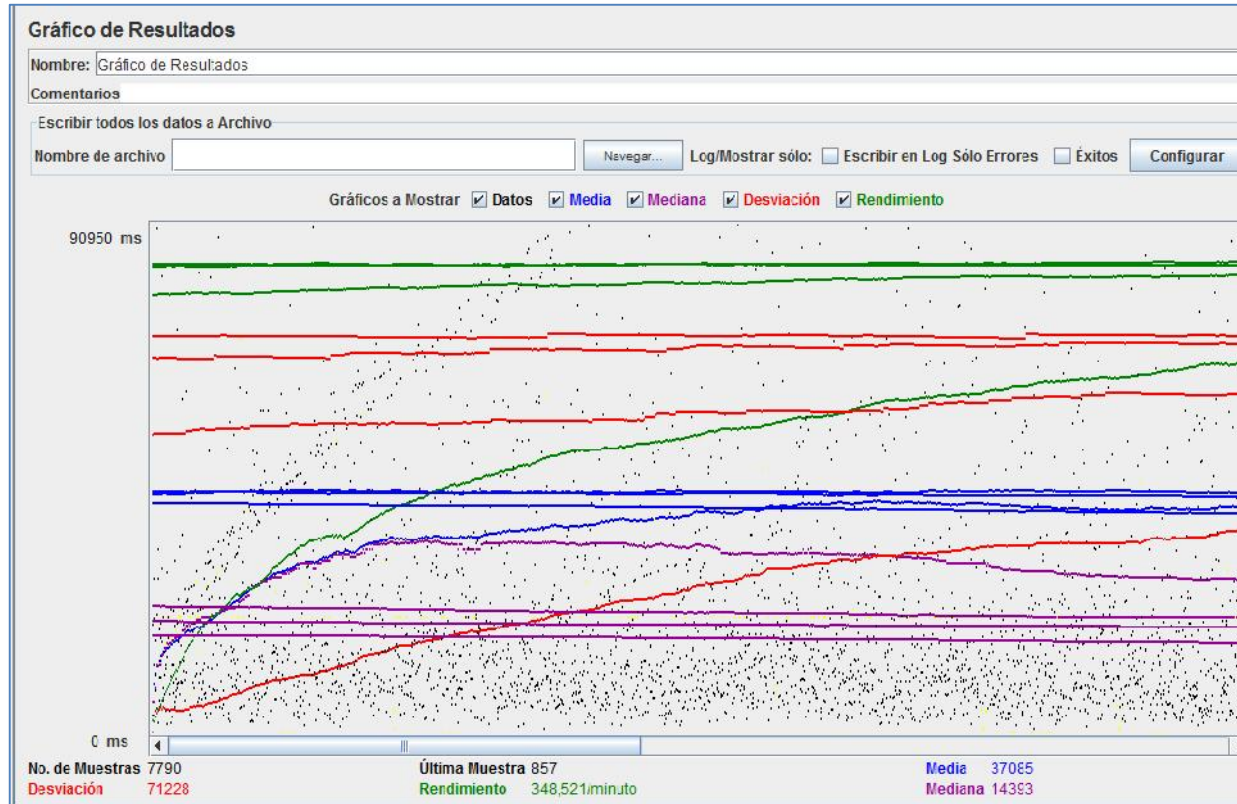


Figura V.16: Grafico de Resultados

Donde:

- **Latencia (eje y):** entendida como el tiempo de espera para la renderización de la página, el tiempo en obtener respuesta del servidor
- **Datos:** muestra los valores actuales de los datos.
- **Media:** representa la Media.

- **Mediana:** dibuja la Mediana.
- **Desviación:** muestra la Desviación Estándar (una medida de la Variación).
- **Rendimiento:** representa el número de muestras por unidad de tiempo.
- **Última muestra:** representa el tiempo transcurrido para la muestra en uso. [35]

Mediante la Figura V.17 se observa la opción “Ver Árbol de Resultados” la cual permite observar el resultado de cada uno de los requests realizados por JMeter, además de la pantalla a la que accedió cada request. En este caso en particular las peticiones que tienen en la columna “Estado” el símbolo verde indica que la petición se realizó con éxito como se visualiza en la Figura V.18 y en el caso de aquellas que tienen el símbolo rojo significa que hubo éxito en la petición como se visualiza en la Figura V.19.

Muestra #	Tiempo de comienzo	Nombre del hilo	Etiqueta	Tiempo de Muestra (...)	Estado	Bytes	Latency
1	18:22:54.573	Thread Group 1-8	http://efinanciero.es...	5815		51794	885
2	18:22:55.101	Thread Group 1-23	http://efinanciero.es...	12254		51794	826
3	18:23:00.477	Thread Group 1-275	http://efinanciero.es...	8274		14136	4458
4	18:22:55.211	Thread Group 1-15	http://efinanciero.es...	16700		51774	1600
5	18:22:55.105	Thread Group 1-65	http://efinanciero.es...	17575		51774	1697
6	18:23:01.072	Thread Group 1-347	http://efinanciero.es...	12004		14136	4467
7	18:22:55.282	Thread Group 1-30	http://efinanciero.es...	18678		51774	2434
8	18:22:55.213	Thread Group 1-22	http://efinanciero.es...	19869		51774	1668
9	18:23:01.318	Thread Group 1-351	http://efinanciero.es...	14193		14136	4225
10	18:22:54.858	Thread Group 1-29	http://efinanciero.es...	20830		51734	668
11	18:23:00.595	Thread Group 1-320	http://efinanciero.es...	15245		14136	4943

Figura V.17: Árbol de Resultados en JMeter

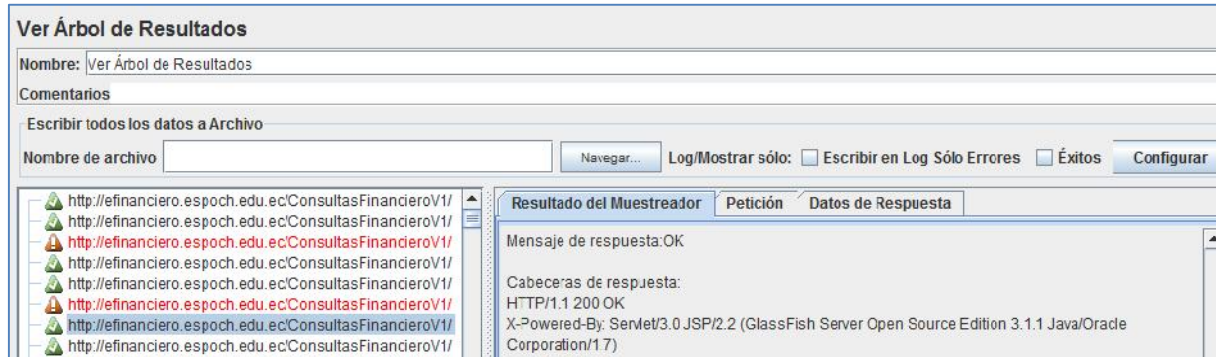


Figura V.18: Respuesta exitosa

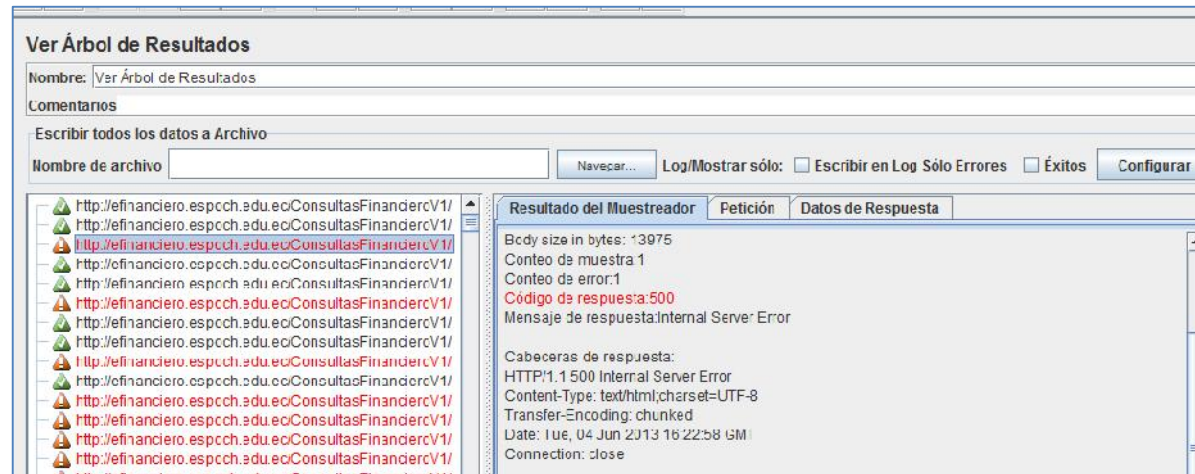


Figura V.19: Respuesta fallida

CONCLUSIONES

- Mediante la utilización de JSF+ PrimeMobile y componentes personalizados construidos para la aplicación web móvil SCMTPM se obtiene un nivel de seguridad de 85.64%, el cual se encuentra en el rango de 60 a 100 que equivale a *Adecuado*
- Al utilizar JSF+ PrimeMobile y los componentes personalizados el nivel de seguridad en la aplicación web móvil SCMTPM se incrementa en un valor de 28.68%. Este valor resulta de la suma de: Confidencialidad (8.24%), Disponibilidad (8.14%) e Integridad (12.3%).
- Toda aplicación es vulnerable, por lo tanto con este trabajo de investigación se pretende mitigar las vulnerabilidades a nivel de interfaz de usuario.
- Al realizar la descripción de los componentes de seguridad que se utilizaron en la aplicación móvil fue necesario conocer las necesidades del usuario, de esta manera se pudo crear los componentes personalizados inputtext seguro, labelseguro y validadores personalizados, ya que la aplicación está orientada a realizar consultas de tramites de pago; debido a ello en lo que más se encontró dificultad para implementar los componentes fue en las configuraciones de archivos xml por lo tanto se requirió una investigación más minuciosa acerca de ello.
- Para poder determinar las potencialidades de los componentes personalizados con la seguridad se lo hizo en base a los parámetros de confidencialidad, disponibilidad e integridad ya que estos hacen parte del concepto de seguridad sugerido por ISO 27001; de esta evaluación se obtuvo que al utilizar los componentes personalizados para exponer la información de los usuarios el nivel de seguridad se incrementa en un 28.68%.
- En la implementación del sistema de consultas y monitoreo de trámites de pago se trabajó con la metodología XP la cual permite realizar programación en pareja lo cual fue de gran ayuda para poder llevar a cabo cada uno de los requerimientos que la aplicación móvil requería.

RECOMENDACIONES

- Los componentes personalizados y JSF+PrimeMobile deben ser considerados para su implementación ya que estos pueden ser muy útiles para la creación de nuevas aplicaciones móviles ya que estos brindan un buen nivel de seguridad.
- El sistema de Consultas y Monitoreo de Trámites de Pago cuenta con un nivel de seguridad por lo tanto es idóneo para realizar consultas a cerca de los trámites que los usuarios realizan en el Departamento Financiero.
- Las vulnerabilidades analizadas en este trabajo de investigación se han escogido para el análisis a nivel de interfaz de usuario, existen otras como: referencia directa insegura a objetos, redirecciones y reenvíos no válidos, falta de control de acceso en el nivel de funciones, entre otras las cuales es importante que también sean analizadas.
- Si se desea construir otro componente personalizado es importante que se comprenda cada uno de los términos necesarios para que este proceso sea más fácil de llevarlo a cabo.
- Para realizar el escaneo de vulnerabilidades con herramientas software de una aplicación se debe elegir en base a la facilidad de adquirirlas ya sean estas de software propietario o libre y también siguiendo la buena aceptación de estas dentro de la comunidad de individuos dedicados a estas tareas de escaneo.
- La encriptación y cifrado de la información sensible es importante, por lo que se debe usar adecuadamente los algoritmos existentes más no crearlos ya que para ello se requiere un estudio profundo.
- Cuando se implementa una aplicación ya sea esta de escritorio o web es importante definir una correcta metodología de lo contrario una inadecuada elección llevaría a provocar retrasos en el desarrollo y las iteraciones de la aplicación.

RESUMEN

Análisis de componentes de seguridad a nivel de interfaz de usuario en Java Server Faces (JSF) + PrimeMobile. Caso práctico Departamento Financiero de la Escuela Superior Politécnica de Chimborazo.

Se aplicó el método científico para avalar el trabajo de investigación y realizar el análisis e interpretación de los resultados que se han obtenido, se ha utilizado la metodología Programación Extrema (eXtreme Programming) para la implementación del Sistema de Consultas y Monitoreo de Trámites de Pago Móvil (SCMTPM), también se empleó componentes personalizados, herramientas de software libre para el desarrollo de la aplicación como el Framework JSF 2.0, PrimeFaces 3.3.1, PrimeMobile 0.9.3, la librería Enterprise Security API (ESAPI), páginas xhtml y JDBC, PostgreSQL, el servidor de aplicaciones web Glassfish 3.1.2 para la publicación de la aplicación web.

En base al análisis que se ha realizado y a los parámetros establecidos para determinar el nivel de seguridad adecuado (Confidencialidad, Disponibilidad e Integridad), se ha obtenido que mediante el uso de JSF+ PrimeMobile y los componentes personalizados con seguridad el nivel de seguridad es de 85.64 %, y mediante el uso de componentes de JSF+ PrimeMobile el nivel de seguridad es de 56.96%, por lo cual utilizando JSF+ PrimeMobile y componentes personalizados con seguridad se obtiene un incremento de 28,68% en el nivel de seguridad.

Concluimos de esta manera que utilizando JSF+ PrimeMobile y componentes personalizados con seguridad se obtiene un nivel adecuado de seguridad el cual se encuentra en los parámetros establecidos en este trabajo de investigación para la aplicación web móvil SCMTPM.

Se recomienda el uso del Sistema de Consulta y Monitoreo de Trámites de Pago Móvil ya que mediante este los usuarios podrán realizar las consultas acerca del avance de sus trámites de pago hasta el momento que ya estén listos los comprobantes para ser cobrados sin necesidad de acercarse al Departamento Financiero, debido a que este sistema solo expone información pero de forma segura y adecuada.

SUMMARY

Analysis of the components of security to an user level interface in Java Server Faces (JSF) + PrimeMobile. Practical Case Financial Department of Higher School Politechnic of Chimborazo.

It was applied the scientific method to guarantee the investigation work and make the analysis and interpretation of the outcomes that have been obtained, it has been used the Extreme Programming Methodology (extreme Programming) for the implementation of the Query and Monitoring System of Procedures of Payment Mobile (SCMTPM), also it was applied personalized component, free software tools for the development of the application such as: Framework JSF 2.0, PrimeFaces 3.3.1, PrimeMobile 0.9.3, the library Enterprise Security API (ESAPI), xhtml pages and .JDBC, PostgreSQL, the application web server Glassfish 3.1.2 for the publication of the web application.

In based on the analysis that has been made and the established parameters to determine the level of adequate security (Confidentiality, Availability, and Integrity), it has been obtained that by using the JSF+ PrimeMobile and the personalized components with security the security level is 86,64% and through the usage of the components of JSF+PrimeMobile the security level is of 56,96% in consequence using JSF+PrimeMobile and the personalized components with security it is gotten an increase of 28,68% in the security level.

It is concluded on this way that using JSF+PrimeMobile and the personalized component with security it is obtained an adequate level of security which is found within the established parameters in this investigation work for the SCMTPM mobile web application.

It is recommended that the usage of the Query and Monitoring System of the Procedures of Mobile Payment since with it the users will be able to make queries about the advancement of their payment procedures till the moment that be ready the vouchers to be charged without need to get to the Financial Department, due to this system only exposes data but in a safe and adequate manner.

GLOSARIO DE TERMINOS

A

API: Es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas.

C

Codificar: La codificación toma la información y la transforma en otra cosa. Esto puede ser tan simple como la codificación de un mensaje escrito en una forma cifrada. Esto hace que el mensaje sea difícil de descifrar para los demás.

D

Decodificar: es tomar la información y devolverla a su forma original.

Deserializar: Proceso de recuperar los datos almacenados y persistidos previamente por una aplicación

E

ESAPI: Es una colección gratis y abierta de todos los métodos de seguridad que un desarrollador necesita para construir una aplicación Web segura publicada por OWASP.

F

Framework: es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado.

Fuzzing: es una técnica de testeado de software, a menudo automatizado o semi automatizado, que consiste en proporcionar datos inválidos, inesperados, o aleatorios a las entradas de un programa de ordenador.

H

Hibernate: es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones.

HTTP 500: Error Interno del Servidor. Cuando se obtiene este error, significa que el "script" que se estaba ejecutando (generalmente es un archivo .cgi, .asp, .php, .cfm, etc.) tuvo un error.

L

Licencia ASL. 2.0 (Apache License o Apache Software License): Licencia permisiva ya que permite realizar todo tipo de operaciones con el código fuente, incluidos forks y desarrollos propietarios.

Lista Blanca: una lista o registro de entidades que, por una razón u otra, pueden obtener algún privilegio particular, servicio, movilidad, acceso o reconocimiento.

Lista Negra (*blacklisting*): es la compilación que identifica a quienes serán denegados, no reconocidos u obstaculizados.

O

OWASP: Open Web Application Security Project, proyecto de código abierto dedicado a determinar y combatir las vulnerabilidades que hacen que el software sea inseguro.

S

Serializar: Es el proceso de almacenar el estado de un objeto o una parte de sus miembros para su uso por otra aplicación o instancia. La serialización binaria es un medio de almacenar los datos en una representación binaria.

ANEXOS

Anexo 1

Código del componente personalizado.

```
import javax.faces.component.FacesComponent;
import javax.faces.component.UIInput;
import javax.faces.context.FacesContext;
import tesis.validadores.Validator;

@FacesComponent(value = "tesis.inputEntry")
public class InputEntryComponent extends UIInput {

    private String label;

    public InputEntryComponent(String label) {
        this.label = label;
    }

    public InputEntryComponent() {
        this.setRendererType("tesis.inputEntry");
    }

    public String getLabel() {
        return label;
    }

    public void setLabel(String label) {
        this.label = label;
    }

    // Overridden methods
    public String getFamily() {
        return "tesis.inputEntry";
    }

    public void restoreState(FacesContext ctxt, Object state) {
        Object val[] = (Object[]) state;
        super.restoreState(ctxt, val[0]);
        label = (String) val[1];
    }

    public Object saveState(FacesContext ctxt) {
        Object val[] = new Object[2];
        val[0] = super.saveState(ctxt);
        val[1] = label;
        return ((Object) val);
    }
}
```


Anexo 2

Código del render personalizado.

```
@FacesRenderer(rendererType = "tesis.componente.renderers.InputEntryRenderer",
componentFamily = "tesis.inputEntry")
public class InputEntryRenderer extends Renderer {

    public InputEntryRenderer() {
    }

    public void decode(FacesContext ctxt, UIComponent cmp) {
        InputEntryComponent ieCmp = (InputEntryComponent) cmp;
        Map requestMap = ctxt.getExternalContext().getRequestParameterMap();
        String clientId = cmp.getClientId(ctxt);
        String val = (String) requestMap.get(clientId);
        ((UIInput) ieCmp).setSubmittedValue(val);
    }

    public void encodeBegin(FacesContext ctxt, UIComponent cmp)
        throws IOException {
        InputEntryComponent ieCmp = (InputEntryComponent) cmp;
        ResponseWriter respWr = ctxt.getResponseWriter();
        encodeLabel(respWr, ieCmp);
        encodeInput(respWr, ieCmp);
        encodeMessage(ctxt, respWr, ieCmp);

        respWr.flush();
    }

    @Override
    public Object getConvertedValue(FacesContext ctxt, UIComponent cmp,
        Object subVal) throws ConverterException {
        Object convVal = null;
        ValueExpression valExpr = cmp.getValueExpression("value");
        if (valExpr != null) {
            Class valType = valExpr.getType(ctxt.getELContext());
            if (valType != null) {
                convVal = subVal;
                if (!valType.equals(Object.class) && !valType.equals(String.class)) {
                    Converter converter = ((UIInput) cmp).getConverter();
                    converter = ctxt.getApplication().createConverter(valType);
                    if (converter != null) {
                        convVal = converter.getAsObject(ctxt, cmp, (String) subVal);
                    }
                }
            }
        }
        return convVal;
    }
}
```

Anexo 3

Código personalizado del archivo tld

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE facelet-taglib PUBLIC
"-//Sun Microsystems, Inc.//DTD Facelet Taglib 1.0//EN"
"http://java.sun.com/dtd/facelet-taglib_1_0.dtd">
<facelet-taglib xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-facelettaglibrary_2_0.xsd"
    version="2.0">
    <namespace>http://tesis/componente/</namespace>
    <tag>
        <tag-name>InputtextSeguro</tag-name>
        <component>
            <component-type>inputEntry</component-type>
            <renderer-type>tesis.componente.renderers.InputEntryRenderer</renderer-type>
        </component>
    </facelet-taglib>
```

Anexo 4

Creación de la creación de la página xhtml y llamado al componente creado.

```
<f:view xmlns="http://www.w3.org/1999/xhtml"
  xmlns:f="http://java.sun.com/jsf/core"
  xmlns:p="http://primefaces.org/ui"
  xmlns:pm="http://primefaces.org/mobile"
  contentType="text/html"
  renderKitId="PRIMEFACES_MOBILE"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:c="http://java.sun.com/jsp/jstl/core"
  xmlns:mc="http://tesis/componente/">

  <pm:page title="Tesis" >
    <pm:view id="Prueba" swatch="e">
      <pm:header title="Prueba" swatch="b">

        </pm:header>
        <pm:content>
          <h:form>

            <mc:InputtextSeguro id="Gustavo" label="Nombre Tesis"
              required = "true"
              value = "#{usuariaC.usuario.nombre}"
              errorStyleClass = "error"
              requiredMessage = "El valor es necesario!" >
            </mc:InputtextSeguro>

            <p:commandButton      actionListener="#{usuariaC.ver}"      value="Probar"
update="ot"/>

          </h:form>
        </pm:content>
      </pm:view>
    </pm:page>
  </f:view>
```

Anexo 5

Código del validator

```
public void validate(FacesContext context, UIComponent c, Object val) throws
ValidatorException {
    String value = (String) val;
    if (!value.isEmpty()) {
        try {
            Validador.validateValue(value);
        } catch (IntrusionException ex) {
            FacesMessage message = new FacesMessage();
            message.setDetail("Caracteres no permitidos");
            message.setSummary("Error");
            message.setSeverity(FacesMessage.SEVERITY_ERROR);
            throw new ValidatorException(message);
        } catch (ValidationException ex) {
            FacesMessage message = new FacesMessage();
            message.setDetail("Caracteres no permitidos");
            message.setSummary("Error");
            message.setSeverity(FacesMessage.SEVERITY_ERROR);
            throw new ValidatorException(message); } } }
}
```

Anexo 6

Código de llamada al validador y al componente creado

```
f:view xmlns="http://www.w3.org/1999/xhtml">
  <pm:page title="mFinanciero" >
    <pm:view id="Prueba" swatch="e">
      <pm:header title="Prueba" swatch="b">

        </pm:header>
        <pm:content>
          <h:form>
            <p:messages id="messages" showDetail="true" autoUpdate="true" />
            <mc:InputtextSeguro id="mcp" label="Nombre Tesis"
              required = "true"
              value = "#{usuariaC.usuario.nombre}"
              errorStyleClass ="error"
              requiredMessage = "El valor es necesario!">
              <f:validator validatorId="tesis.validadores.ValidadorXSS_SQLInjection"
for="mcp"/>
            </mc:InputtextSeguro>
            <p:commandButton actionListener="#{usuariaC.ver}" value="Probar"/>
          </h:form>
        </pm:content>
      </pm:view>
    </pm:page>
  </f:view>
```

Anexo 7

Código de métodos de encriptación:

```
package tesis.validadores;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class StringEncryptor {

    public static String MD2 = "MD2";
    public static String MD5 = "MD5";
    public static String SHA1 = "SHA-1";
    public static String SHA256 = "SHA-256";
    public static String SHA384 = "SHA-384";
    public static String SHA512 = "SHA-512";

    private static String toHexadecimal(byte[] digest) {
        String hash = "";
        for (byte aux : digest) {
            int b = aux & 0xff;
            if (Integer.toHexString(b).length() == 1) {
                hash += "0";
            }
            hash += Integer.toHexString(b);
        }
        return hash;
    }

    public static String getStringMessageDigest(String message, String algorithm) {
        byte[] digest = null;
        byte[] buffer = message.getBytes();
        try {
            MessageDigest messageDigest = MessageDigest.getInstance(algorithm);
            messageDigest.reset();
            messageDigest.update(buffer);
            digest = messageDigest.digest();
        } catch (NoSuchAlgorithmException ex) {
            System.out.println("Error creando Digest");
        }
        return toHexadecimal(digest);
    }
}
```

Anexo 8

Prototipo para verificar la falta de confidencialidad de los datos.



El prototipo muestra una interfaz de usuario con un encabezado azul que contiene un botón 'Atras' con una flecha hacia atrás y el título 'Mis Datos'. El contenido principal es un recuadro amarillo con un encabezado azul que dice 'Mis Datos'. Dentro de este recuadro, se listan los datos personales de un usuario:

Nombre:	Juan
Apellido:	Perez
Password:	123456
Edad:	15
Direccion:	Av. Leopoldo Freire

Anexo 9

Escaneo de vulnerabilidades con la herramienta software Acunetix en el prototipo sin el uso de componentes personalizados.

The screenshot displays the Acunetix Web Vulnerability Scanner interface. The main window shows the scan results for a target URL: `http://192.168.1.6:8080/tesis/usuario/usuarioSC/index.htm`. The scan results are categorized into several groups:

- Web Alerts (5)
- Class Site Suitability (verified) (1)
- 1-ML form without Csrf protection (4)
- Knowledge base (3)
- Use of files with inputs
- Use structure
- misconfigurations
- cookies

The Alerts summary panel on the right indicates a **Level 3: High** threat level. It provides a breakdown of the total alerts found:

Severity	Count
High	1
Medium	4
Low	0
Informational	0

Additional summary information includes:

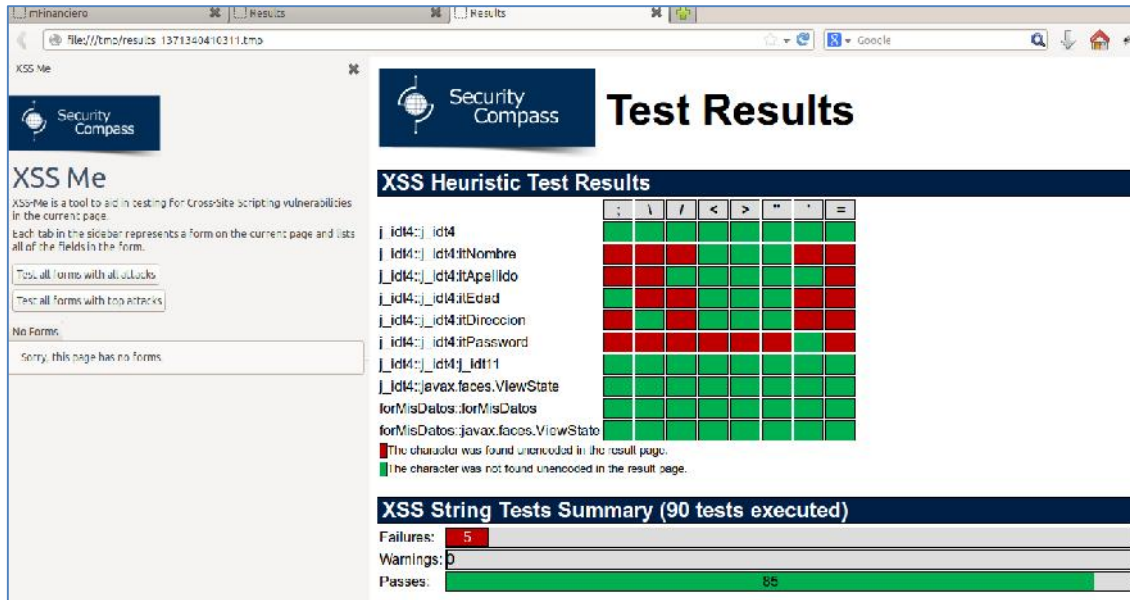
- Target information: `http://192.168.1.6:8080/tesis/usuario/usuarioSC/index.htm`
- Statistics: 301 requests
- Progress: Scan is finished 100.0%

The Activity Window at the bottom shows the following log entries:

```
16:38:19: Done the finishing (method) "test/premios/tesis/premiosSC/index.htm" on response "HTTP/1.1 200 OK"
16:38:20: Finished scanning.
16:38:21: Saving scan results to database ...
16:38:22: Done saving to database.
16:38:23: Scan finished.
```


Anexo 10

Mediante la utilización de la herramienta software XSS Se me realiza el escaneo de la vulnerabilidad XSS en el prototipo sin el uso de componentes personalizados.



The screenshot displays the Security Compass XSS Me tool interface. The main window shows the 'Test Results' section, which includes a table of 'XSS Heuristic Test Results' and a summary of 'XSS String Tests Summary (90 tests executed)'. The table lists various form fields and their test results, with a legend indicating that red cells represent characters found unencoded in the result page, and green cells represent characters not found unencoded.

Field	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10
j_id14: j_id14	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
j_id14: j_id14#Nombre	Red	Red	Red	Red	Red	Red	Red	Red	Red	Red
j_id14: j_id14#Apellido	Red	Red	Red	Red	Red	Red	Red	Red	Red	Red
j_id14: j_id14#Edad	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
j_id14: j_id14#Direccion	Red	Red	Red	Red	Red	Red	Red	Red	Red	Red
j_id14: j_id14#Password	Red	Red	Red	Red	Red	Red	Red	Red	Red	Red
j_id14: j_id14_id111	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
j_id14: javax.faces.ViewState	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
forMisDatos: forMisDatos	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
forMisDatos: javax.faces.ViewState	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green

XSS String Tests Summary (90 tests executed)

Failures:	5
Warnings:	0
Passes:	85

BIBLIOGRAFIA

1. **IULIANO, P.**, "Incorporando seguridad a las componentes de Interfaz de Usuario del framework JSF (Java Server Faces) con soporte para clientes heterogéneos.", Facultad de Informática, Universidad de la Plata, Argentina Buenos Aires, Tesis, 2010, Pp., 26-37, 50-54.

2013/06/14

2. **PAGUAY, Paul.**, "Propuesta De Técnicas De Aseguramiento De Aplicaciones Web Desarrolladas En Java", Escuela De Postgrado Y Educación Continua, Escuela Superior Politécnica de Chimborazo, Riobamba Ecuador, Maestría en Interconectividad de Redes, Tesis, 2013, Pp., 54- 67.

2013/06/14

BIBLIOGRAFIA DE INTERNET

1. **APACHE JMETER**

<http://es.scribd.com/doc/42834314/JMeter-Manual-de-Usuario-v1-1>.

2013/04/30

2. **API ENTERPRISE SECURITY OWASP**

<https://www.owasp.org/index.php/ESAPI>.

2013/06/11

3. CROSS-SITE SCRIPTING (XSS)

[https://www.owasp.org/index.php/Cross-site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Cross-site_Scripting_(XSS))

2013/06/11

4. DETECTANDO VULNERABILIDADES CSRF (CROSS-SITE REQUEST FORGERY)

<http://www.seguridad.unam.mx/noticia/?noti=637>

2013/05/04

5. DESARROLLO DE APLICACIONES WEB SEGURAS CON JAVA

[http://java.ciberaula.com/articulo/aplicaciones_web_seguras/.](http://java.ciberaula.com/articulo/aplicaciones_web_seguras/)

2013/06/01

6. DOM-BASED XSS

[http://www.mediawiki.org/wiki/DOM-based_XSS#.](http://www.mediawiki.org/wiki/DOM-based_XSS#)

2013/06/11

7. FIREBUG

<http://es.wikipedia.org/wiki/Firebug>

2013/06/06

8. INTRODUCCIÓN A LA CRIPTOGRAFÍA

<http://www.dma.fi.upm.es/java/matematicadiscreta/aritmeticamodular/criptografia.htm>

|

2013/06/04

9. INTRODUCCIÓN A LOS SERVICIOS WEB EN JAVA

http://www.programacion.com/articulo/introduccion_a_los_servicios_web_en_java_1_90

2013/03/15

10.JAVA: CALCULAR HASH

<http://www.dbsnippets.com/2012/08/21/java-calcular-hash/>.

2013/06/04

11.JAVA CRYPTOGRAPHY ARCHITECTURE (JCA) REFERENCE GUIDE

<http://docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/CryptoSpec.html#Concepts>

2013/06/05

12.JAVA SERVER FACES

https://www.owasp.org/index.php/Java_Server_Faces

2013/06/01

13.LA INFORMATICA DEL HACKER

<http://root-neztgul.blogspot.com/2011/10/burp-suite-herramienta-de-auditoria-web.html>.

2013/06/06

14.MALWARE EN SMARTPHONES

http://www.bdigital.org/Documents/Malware_Smartphones.pdf.

2012/11/12

15.OWASP TOP 10 - 2013 - RELEASE CANDIDATE

https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project.

2013/03/05

16.OWASP TOP TEN PROJECT

https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project.

2013/03/05

17.PRIMEFACES

<http://es.wikipedia.org/wiki/PrimeFaces>

2013/06/09

18.PRIME FACES MOBILE

<http://primefaces.org/showcase/mobile/index.jsf>

2013/06/09

19.PROYECTO WEBSCARAB OWASP

https://www.owasp.org/index.php/Proyecto_WebScarab_OWASP

2013/05/17

20.¿QUÉ ES LA SEGURIDAD INFORMÁTICA?

http://www.acpdesarrollo.com/seguridad_informatica_murcia.php.

2012/12/02

21. QUÉ ES UN SGSI?

<http://www.iso27000.es/sgsi.html#section2a>

2013/06/15

22. SEGURIDAD EN APLICACIONES MÓVILES

http://www.acis.org.co/fileadmin/Base_de_Conocimiento/VI_JornadaSeguridad/Fabian_Molina_VIJNSI.pdf.

2012/06/06

23. SKIPFISH

<http://code.google.com/p/skipfish/>

2013/06/05

24. SOFTWARE DE SEGURIDAD

<http://www.net-security.org/software.php?id=633>

2013/06/15

25. SQL Inject Me

<https://addons.mozilla.org/es/firefox/addon/sql-inject-me/>

2013/06/06

26.TOP 10 2013-A1-INJECTION

https://www.owasp.org/index.php/Top_10_2013-Injection.

2013/03/05

27.TOP 10 2013-A3-CROSS-SITE SCRIPTING (XSS)

[https://www.owasp.org/index.php/Top_10_2013-Cross-Site_Scripting_\(XSS\)](https://www.owasp.org/index.php/Top_10_2013-Cross-Site_Scripting_(XSS)).

2013/05/31

28.TOP 10 2013-A6-SENSITIVE DATA EXPOSURE

https://www.owasp.org/index.php/Top_10_2013-A6.

2013/06/10

29.WAPITI

<http://www.ict-romulus.eu/web/wapiti/home>.

2013/06/06

30.XSRF ATTACK PREVENTION

<https://java.net/jira/browse/JAVASERVERFACES-812>

2013/01/03

31.XSS Me

<https://addons.mozilla.org/es/firefox/search/?q=XSS+ME&appver=21.0&platform=windows>

2013/06/06