



ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN SISTEMAS

**“COMPARACIÓN DE HERRAMIENTAS BENCHMARKING PARA
PRUEBAS UNITARIAS WEB DE APLICACIONES JSF. CASO PRÁCTICO:
SISTEMA DE GESTIÓN DE VIÁTICOS DE LA ESPOCH”**

**“TESIS DE GRADO PREVIA OBTENCIÓN DEL TÍTULO DE
INGENIERO EN SISTEMAS INFORMÁTICOS”**

ANA LUCIA LLALAO CUSHPA

JOSÉ RODOLFO LUIS TAXI CEVALLOS

RIOBAMBA – ECUADOR

-2013-

Agradezco infinitamente a Dios por sus grandes bendiciones en cada momento de mi vida, permitiéndome cumplir mis objetivos, a mis padres por su apoyo incondicional, especialmente a mi madre por su amor y abnegación por inculcarme siempre buenos valores y darme fortaleza para estar siempre de pie ante las adversidades y luchar para alcanzar cada propósito planteado, a toda mi familia y amigos que estuvieron junto a mí con sus palabras de aliento para no dejarme vencer y un sincero agradecimiento a nuestra directora de tesis Ing. Gloria Arcos que con sus conocimientos fue una guía en el desarrollo de nuestra tesis, a la Ing. Lorena Aguirre por su colaboración en el desarrollo de la misma.

Anita Llalao

Agradezco a Dios por el regalo más grande que tenemos en la vida que es el amor de la familia, pues gracias al apoyo de mis padres y hermanos he luchado constantemente para cumplir mis metas, a mis amigos por estar ahí cuando los necesite, a todos mis maestros pues gracias a sus conocimientos impartidos en clases es posible llegar a cumplir este sueño.

José Rodolfo Luisataxi Cevallos

Dedico este trabajo a Dios, a mis padres Carlos y Clementina, a mi hermano Hugo quienes con amor y comprensión estuvieron junto a mí durante mi formación académica permitiendo así llegar a la meta.

A mi prima Luisita, a mis amig@s Elvira, Lorena, José por su valiosa amistad y apoyo incondicional.

Anita Llalao.

Este logro no es solo mío, si lo he conseguido es por todo el apoyo incondicional de mis padres José Vicente y María del Carmen a quienes dedico este trabajo con mucho amor por ser ellos mi fortaleza día a día, a mis hermanos Patricia y Juan Carlos que siempre confiaron en mí.

A todos amigos con quienes compartí mi estadía en mi amada ESPOCH.

José Rodolfo Luisataxi Cevallos

FIRMAS RESPONSABLES Y NOTAS

NOMBRES	FIRMA	FECHA
ING. IVÁN MENES CAMEJO DECANO DE LA FACULTAD DE INFORMÁTICA Y ELECTRÓNICA.	_____	_____
ING. RAÚL ROSERO M. DIRECTOR DE LA ESCUELA DE INGENIERÍA EN SISTEMAS.	_____	_____
ING. GLORIA ARCOS M. DIRECTOR DE TESIS.	_____	_____
ING. LORENA AGUIRRE MIEMBRO DE TESIS.	_____	_____
TLG. CARLOS RODRÍGUEZ DIRECTOR DEL CENTRO DE DOCUMENTACIÓN	_____	_____

NOTA: _____

RESPONSABILIDAD DEL AUTOR

Nosotros, Ana Lucia Llalao Cushpa y José Rodolfo Luisataxi Cevallos, somos los responsables de las ideas, doctrinas y resultados expuestos en esta Tesis y el patrimonio intelectual de la misma pertenecen a la Escuela Superior Politécnica de Chimborazo.

Ana Lucia Llalao Cushpa

José Rodolfo Luisataxi Cevallos

ÍNDICE DE ABREVIATURAS

API	Interfaz de Programación de Aplicaciones.
DOM	Modelo de Objetos del Documento.
ESPOCH	Escuela Superior Politécnica de Chimborazo.
GUI	Interfaz Gráfica de Usuario.
HTML	Lenguaje de Marcado de Hipertexto.
HTTP	Protocolo de Transferencia de Hipertexto.
JSF	Java Server Faces.
JSP	Java Server Pages.
MVC	Modelo Vista Controlador.
XML	Lenguaje de Marcas Extensible.

ÍNDICE GENERAL

Contenido

ÍNDICE DE ABREVIATURAS

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

CAPÍTULO I

MARCO REFERENCIAL

1.1. Antecedentes	20
1.2. Justificación.....	23
1.2.1. Justificación Teórica	23
1.2.2. Justificación Práctica	24
1.3. Objetivos	24
1.3.1. Objetivo General.....	24
1.3.2. Objetivos Específicos	24
1.4. Hipótesis.....	25

CAPÍTULO II

MARCO TEÓRICO

2.1. Benchmarking	27
2.1.1. Tipos de Benchmarking	28
2.1.1.1. Benchmarking Interno.....	28
2.1.1.2. Benchmarking Competitivo	28
2.1.1.3. Benchmarking Cooperativo o Multifuncional	29
2.1.2. Características del Benchmarking	29
2.1.2.1. Calidad	29
2.1.2.2. Productividad	30
2.1.2.3. Tiempo	30
2.1.3. Etapas de Benchmarking	31
1. Determinar a qué se le va a hacer benchmarking.....	31
3. Identificar a los socios del Benchmarking	31

4.	Recopilar y analizar la información de Benchmarking	32
5.	Actuar	32
2.1.4.	Ventajas del Benchmarking	32
2.2.	Pruebas Unitarias	33
2.2.1.	Características de una buena prueba unitaria.....	33
2.2.2.	Beneficios de las pruebas unitarias	35
2.3.	Java Server Faces	38
2.3.1.	Arquitectura	39
2.3.1.1.	Modelo	39
2.3.1.2.	Vista	40
2.3.1.3.	Controlador	40
2.3.2.	Ciclo de Vida	40
2.3.2.1.	Restore View	41
2.3.2.2.	Apply Request Values.....	41
2.3.2.3.	Process Validations	41
2.3.2.4.	Update Model Values.....	42
2.3.2.5.	Invoke Application.....	42
2.3.2.6.	Render Response	42
2.3.3.	Estructura de una aplicación JSF	42
2.3.3.1.	Archivos XHTML.....	43
2.3.3.2.	Archivos XML	43
2.3.3.3.	Archivos Java.....	43
2.3.3.4.	Archivos de paquetes de mensajes (<i>message bundles</i>).	43
2.3.4.	Cómo funciona JSF.....	44

CAPÍTULO III

COMPARACIÓN DE HERRAMIENTAS DE BENCHMARKING PARA PRUEBAS UNITARIAS WEB DE APLICACIONES JSF

3.1.	Determinación de las herramientas a comparar	47
3.1.1.	Httpunit	47
3.1.2.	Htmlunit	48
3.1.3.	Selenium	48
3.1.4.	Jwebunit.....	49

3.1.5.	Shale.....	50
3.1.6.	csUnit.....	50
3.1.7.	Cactus.....	51
3.1.8.	JSFUnit	51
3.2.	Análisis de las herramientas seleccionadas.....	52
3.3.	Descripción de herramientas seleccionadas	56
3.3.1.	Selenium	56
3.3.2.	Cactus.....	57
3.3.3.	JSFUnit	59
3.4.	Determinación de los indicadores de comparación.....	60
3.5.	Creación del prototipo para las pruebas.....	61
3.6.	Realización de las pruebas	62
3.6.1.	Selenium.....	62
3.6.1.1.	Procedimiento	63
3.6.2.	Cactus	70
3.6.2.1.	Procedimiento	71
3.6.3.	JSFUnit.....	77
3.6.3.1.	Procedimiento	77
3.8.	Valoración de herramientas según indicadores definidos.....	85
3.9.	Comprobación de hipótesis	89

CAPITULO IV

DESARROLLO DEL SISTEMA DE GESTIÓN DE VIÁTICOS DE LA ESPOCH

4.1.	Gestión del Proyecto	94
4.2.	Desarrollo de la Metodología XP.....	95
4.2.1.	Fase I: Planificación.....	95
4.2.1.1.	Descripción del Sistema.....	96
4.2.1.2.	Definición del flujo del proceso de Gestión de Viáticos.....	96
4.2.1.3.	Especificación de Requerimientos	99
4.2.2.	Fase II: Diseño de Software.....	139
4.2.2.1.	Diseño de Base de Datos.....	139
4.2.2.2.	Diseño de Interfaces	140
4.2.2.3.	Diccionario de datos.....	144

4.2.3.	Fase III: Codificación	153
4.2.4.	Fase IV: Pruebas de Funcionamiento	157

CONCLUSIONES

RECOMENDACIONES

RESUMEN

SUMARY

GLOSARIO

ANEXOS

BIBLIOGRAFÍA

ÍNDICE DE FIGURAS

Figura II. 1 Arquitectura Java Server Faces	39
Figura II. 2 Ciclo de vida de Java Server Faces.....	41
Figura II. 3 Estructura de directorio de una aplicación JSF.....	43
Figura III. 4 Cuadro Estadístico de Ponderaciones.....	56
Figura III. 5 Arquitectura de Prototipos.....	62
Figura III. 6 Arquitectura SeleniumTest.....	63
Figura III. 7 Selenium: Librerías	64
Figura III. 8 Selenium: CargarCiudadTest.java.....	64
Figura III. 9 Selenium: InsertarCiudadTest.java	65
Figura III. 10 Selenium: EditarCiudadTest.java.....	66
Figura III. 11 Selenium: BuscarCiudadPorCodigo.java	67
Figura III. 12 Selenium: SeleniumTestSuite.java.....	68
Figura III. 13 Inicialización del servidor Selenium	69
Figura III. 14 Ejecución de la clase SeleniumTestSuite	69
Figura III. 15 Selenium: Resultados	70
Figura III. 16 Arquitectura Proyecto TestCactus.....	71
Figura III. 17 Cactus: Librerías.....	72
Figura III. 18 Cactus: Archivo web.xml.....	73
Figura III. 19 Cactus: TestCiudad.java.....	74
Figura III. 20 Cactus: CiudadC.java	75
Figura III. 21 Cactus: Resultados	76
Figura III. 22 Arquitectura Proyecto JSFUnitTest.....	77
Figura III. 23 JSFUnit: Librerías	79
Figura III. 24 JSFUnit: ID en los componentes	79
Figura III. 25 JSFUnit: Archivo web.xml.....	80
Figura III. 26 JSFUnit: Agregar cactus-report.xml	80
Figura III. 27 JSFUnit: JSFUnitTest.java	81
Figura III. 28 JSFUnit: Método testInsertarCiudad	82
Figura III. 29 JSFUnit: Método testCargarCiudades	83
Figura III. 30 JSFUnit: Método testEditarCiudad	83
Figura III. 31 JSFUnit: Método testBuscarCiudadPorCodigo.....	84
Figura III. 32 JSFUnit: Resultados	85
Figura III. 33 Resultado Tiempo de Ejecución.....	86
Figura III. 34 Resultado de líneas de código escritas	87
Figura III. 35 Resultado flexibilidad de cambios	88
Figura III. 36 Resultado navegabilidad por la aplicación.....	89
Figura III. 37 Resultado Análisis Comparativo	92
Figura IV. 38 Fases de la Metodología XP	95
Figura IV. 39 Flujo de proceso (Secretaría de Dependencia).....	97

Figura IV. 40 Flujo de proceso (Jefe de Dependencia/Secretaria del Vicerrectorado de Investigación y Desarrollo).....	98
Figura IV. 41 Flujo de proceso (Vicerrector de Investigación y Desarrollo/Jefe de Movilización).....	98
Figura IV. 42 Usuarios del Sistema de Gestión de Viáticos.....	104
Figura IV. 43 Rol del Usuario Financiero	105
Figura IV. 44 Roles de los Usuarios Secretaria y Jefe de Dependencia.....	105
Figura IV. 45 Roles de los Usuarios Secretaria y Vicerrector de Investigación y Desarrollo.....	105
Figura IV. 46 Rol del Usuario Jefe de Movilización.....	105
Figura IV. 47 Diseño de Base de Datos.....	139
Figura IV. 48 Ingreso Nivel.....	140
Figura IV. 49 Ingreso Zona.....	140
Figura IV. 50 Ingreso Ciudad	141
Figura IV. 51 Ingreso Nivel Zona.....	141
Figura IV. 52 Ingreso Zona Ciudad	141
Figura IV. 53 Ingreso Datos de la Comisión	142
Figura IV. 54 Ingreso de las Actividades	142
Figura IV. 55 Ingreso de los Datos de los Comisionados.....	143
Figura IV. 56 Asignar Vehículo	143
Figura IV. 57 Diagrama de Bloques y Despliegue del Sistema de Gestión de Viáticos	154
Figura IV. 58 Pruebas Unitarias Nivel.....	155
Figura IV. 59 Pruebas Unitarias Zona	155
Figura IV. 60 Pruebas Unitarias Ciudad	155
Figura IV. 61 Pruebas Unitarias Solicitud	156
Figura IV. 62 Pruebas Unitarias Asignación Vehículo.....	156
Figura IV. 63 Pruebas Unitarias Carga Solicitudes y Salvoconductos.....	156

ÍNDICE DE TABLAS

Tabla III. I Leyenda de la Tabla III.II.....	53
Tabla III. II Resumen de comparación de herramientas	54
Tabla III. III Ponderaciones de las herramientas según los parámetros	55
Tabla III. IV Métodos Assert.....	58
Tabla III. V Descripción de Prototipos	61
Tabla III. VI Resultados parámetro Tiempo de ejecución.....	85
Tabla III. VII Resultados parámetro líneas de código	86
Tabla III. VIII Resultados parámetro flexibilidad a cambios	87
Tabla III. IX Resultados parámetro navegabilidad por la arquitectura.....	88
Tabla III. X Pesos de los indicadores	90
Tabla III. XI Resumen de resultados	90
Tabla III. XII Escala de Equivalencias	91
Tabla III. XIII Valoración de indicadores	91
Tabla III. XIV Resultado del análisis comparativo	91
Tabla IV. XV Historia de Usuario Ingreso y modificación de zona.....	106
Tabla IV. XVI Historia de Usuario Ingreso y modificación de nivel.....	106
Tabla IV. XVII Historia de Usuario Ingreso y modificación de ciudad.....	107
Tabla IV. XVIII Historia de Usuario Relacionar nivel zona	107
Tabla IV. XIX Historia de Usuario Asignar una ciudad a una zona	108
Tabla IV. XX Historia de Usuario Reporte Nivel Zona Ciudad.....	108
Tabla IV. XXI Historia de Usuario Crear solicitud de movilización externa.....	109
Tabla IV. XXII Historia de Usuario Crear solicitud de movilización interna.....	109
Tabla IV. XXIII Historia de Usuario Rechazar o anular una solicitud de movilización interna	110
Tabla IV. XXIV Historia de Usuario Rechazar o anular una solicitud de movilización externa.....	111
Tabla IV. XXV Historia de Usuario Aprobar, rechazar o anular una solicitud de movilización interna	111
Tabla IV. XXVI Historia de Usuario Aprobar, rechazar o anular una solicitud de movilización externa.....	112
Tabla IV. XXVII Historia de Usuario Listar solicitudes de movilización internas aprobadas	112
Tabla IV. XXVIII Historia de Usuario Listar solicitudes de movilización externas aprobadas	113
Tabla IV. XXIX Historia de Usuario Editar una solicitud de movilización internas rechazadas.....	113
Tabla IV. XXX Historia de Usuario Editar una solicitud de movilización externas rechazadas.....	114
Tabla IV. XXXI Historia de Usuario Listar solicitudes internas anuladas.....	114

Tabla IV. XXXII Historia de Usuario Listar solicitudes externas anuladas.....	115
Tabla IV. XXXIII Historia de Usuario Editar y aprobar o anular una solicitud de movilización interna	115
Tabla IV. XXXIV Historia de Usuario Editar y aprobar o anular una solicitud de movilización externa.....	116
Tabla IV. XXXV Historia de Usuario Asignar vehículo a solicitudes de movilización internas.....	116
Tabla IV. XXXVI Historia de Usuario Aprobar o anular una solicitud de movilización interna	117
Tabla IV. XXXVII Historia de Usuario Aprobar o anular una solicitud de movilización externa.....	117
Tabla IV. XXXVIII Historia de Usuario Listar solicitudes de movilización internas aprobadas	118
Tabla IV. XXXIX Historia de Usuario Listar solicitudes de movilización externas aprobadas	118
Tabla IV. XL Historia de Usuario Generar Salvoconducto.....	119
Tabla IV. XLI Historia de Usuario Listar solicitudes de movilización internas anuladas	119
Tabla IV. XLII Historia de Usuario Listar solicitudes de movilización externas anuladas	120
Tabla IV. XLIII Historia de Usuario Listar solicitudes de movilización internas aprobadas por fechas.....	120
Tabla IV. XLIV Historia de Usuario Listar solicitudes de movilización externas aprobadas por fechas.....	121
Tabla IV. XLV Historia de Usuario Visualizar número de solicitudes	121
Tabla IV. XLVI Historia de Usuario Reporte de lugares más visitados de un comisionado	122
Tabla IV. XLVII Historia de Usuario Reporte lista comisionados por fechas y dependencia	122
Tabla IV. XLVIII Historia de Usuario Gráfica del comisionado que más ha salido por fechas	123
Tabla IV. XLIX Historia de Usuario Reporte lista de comisionados por fechas.....	123
Tabla IV. L Historia de Usuario Reporte listado de movimiento de vehículos en un rango de fechas	124
Tabla IV. LI Historia de Usuario Grafica de lugares más visitados por los comisionados	124
Tabla IV. LII Historia de Usuario Gráfica de lugares más visitado por los comisionados por fechas.....	125
Tabla IV. LIII Historia de Usuario Listado de salvoconductos aprobados dado fechas	125
Tabla IV. LIV Historia de Usuario Visualizar valor estimado del viático	126
Tabla IV. LV Iteración 1.....	127

Tabla IV. LVI Iteración 2	129
Tabla IV. LVII Iteración 3	131
Tabla IV. LVIII Iteración 4.....	135
Tabla IV. LIX Descripción de la tabla ciudad	144
Tabla IV. LX Descripción de la tabla zona	144
Tabla IV. LXI Descripción de la tabla zona_ciudad	145
Tabla IV. LXII Descripción de la tabla nivel	145
Tabla IV. LXIII Descripción de la tabla nivel_zona	146
Tabla IV. LXIV Estados de la tabla nivel_zona	146
Tabla IV. LXV Descripción de la tabla comisión	146
Tabla IV. LXVI Estados de la tabla comisión	147
Tabla IV. LXVII Descripción de la tabla documento_comision	148
Tabla IV. LXVIII Descripción de la tabla vehiculo	148
Tabla IV. LXIX Descripción de la tabla vehiculo_comision	149
Tabla IV. LXX Descripción de la tabla comisionado.....	150
Tabla IV. LXXI Descripción de la tabla viatico	150
Tabla IV. LXXII Descripción de la tabla detalle_salvoconducto.....	151
Tabla IV. LXXIII Estados de la tabla detalle_salvoconducto	151
Tabla IV. LXXIV Descripción de la tabla salvoconducto.....	152
Tabla IV. LXXV Estados de la tabla salvoconducto	153
Tabla IV. LXXVI Prueba de funcionamiento ingreso y modificación de Zona.....	157
Tabla IV. LXXVII Prueba de funcionamiento ingreso y modificación de Nivel.....	158
Tabla IV. LXXVIII Prueba de funcionamiento ingreso y modificación de Ciudad.....	158
Tabla IV. LXXIX Prueba de funcionamiento relacionar Nivel-Zona	159
Tabla IV. LXXX Prueba de funcionamiento asignar una Ciudad a una Zona	160
Tabla IV. LXXXI Prueba de funcionamiento reporte Nivel Zona Ciudad.....	161
Tabla IV. LXXXII Prueba de funcionamiento creación de Solicitud de Movilización Externa	161
Tabla IV. LXXXIII Prueba de funcionamiento creación de Solicitud de Movilización Interna	162
Tabla VI. LXXXIV Prueba de funcionamiento rechazar o anular una Solicitud de Movilización Interna.....	163
Tabla VI. LXXXV Prueba de funcionamiento rechazar o anular una Solicitud de Movilización Externa.....	164
Tabla VI. LXXXVI Prueba de funcionamiento aprobar, rechazar o anular una Solicitud de Movilización Interna	165
Tabla VI. LXXXVII Prueba de funcionamiento aprobar, rechazar o anular una Solicitud de Movilización Externa.....	166
Tabla VI. LXXXVIII Prueba de funcionamiento listar Solicitudes de Movilización Internas aprobadas	167
Tabla VI. LXXXIX Prueba de funcionamiento listar Solicitudes de Movilización Externas aprobadas	167

Tabla VI. XC Prueba de funcionamiento editar Solicitud de Movilización Interna Rechazada	168
Tabla VI. XCI Prueba de funcionamiento editar Solicitudes de Movilización Externas rechazadas	169
Tabla VI. XCII Prueba de funcionamiento listar Solicitudes de Movilización Internas anuladas	170
Tabla VI. XCIII Prueba de funcionamiento listar Solicitudes de Movilización Externas anuladas	170
Tabla VI. XCIV Prueba de funcionamiento editar, aprobar o anular una Solicitud de Movilización Interna.....	171
Tabla VI. XCV Prueba de funcionamiento editar, aprobar o anular una Solicitud de Movilización Externa.....	172
Tabla VI. XCVI Prueba de funcionamiento asignar vehículo a Solicitudes de Movilización Internas	173
Tabla VI. XCVII Prueba de funcionamiento aprobar o anular una Solicitud de Movilización Interna.....	174
Tabla VI. XCVIII Prueba de funcionamiento aprobar o anular una Solicitudes de Movilización Externa.....	174
Tabla VI. XCIX Prueba de funcionamiento listar Solicitudes e Movilización Internas aprobadas	175
Tabla VI. C Prueba de funcionamiento listar Solicitudes de Movilización Externas aprobadas	176
Tabla VI. CI Prueba de funcionamiento generar Salvoconducto	177
Tabla VI. CII Prueba de funcionamiento listar Solicitudes de movilización Internas anuladas	177
Tabla VI. CIII Prueba de funcionamiento listar Solicitudes de Movilización Internas anuladas	178
Tabla VI. CIV Prueba de funcionamiento listar Solicitudes de Movilización Internas Aprobadas dado fechas	179
Tabla VI. CV Prueba de funcionamiento listar Solicitudes de Movilización Externas aprobadas dado fechas	180
Tabla VI. CVI Prueba de funcionamiento visualizar número de Solicitudes Externas/Internas aprobadas/anuladas	180
Tabla VI. CVII Prueba de funcionamiento reporte de lugares visitados por un Comisionado	181
Tabla VI. CVIII Prueba de funcionamiento reporte listar comisionados por fechas y dependencia	182
Tabla VI. CIX Prueba de funcionamiento grafica estadística de los comisionados con más salidas en un rango de fechas	183
Tabla VI. CX Prueba de funcionamiento reporte lista de comisionados dado un rango de fechas	183

Tabla VI. CXI Prueba de funcionamiento reporte listado de movimiento de vehículos por fechas	184
Tabla VI. CXII Prueba de funcionamiento gráfica de lugares más visitados por los comisionados	185
Tabla VI. CXIII Prueba de funcionamiento gráfica de lugares más visitados por los comisionados por fechas	185
Tabla VI. CXIV Prueba de funcionamiento listado de salvoconductos aprobados por fechas	186
Tabla VI. CXV Prueba de funcionamiento visualizar valor estimado de viáticos	187

INTRODUCCIÓN

Una prueba unitaria es una prueba que comprueba el funcionamiento de un proceso incluido en la aplicación, tanto en las condiciones favorables como en las no favorables (por ejemplo, entradas correctas e incorrectas del usuario para un determinado valor). La realización de pruebas unitarias permite mejorar el desarrollo de aplicaciones web, ya que se reducen los tiempos de depuración y corrección de incidencias.

La presente investigación se encamina a estudiar y determinar cuál es la herramienta de mejor rendimiento para realizar pruebas unitarias en aplicaciones web JSF, estableciendo parámetros de comparación los mismos que serán analizados y probados en diferentes prototipos lo que permitirá el cumplimiento de los objetivos planteados.

En el Capítulo I Marco referencial, se detalla los antecedentes, la justificación de la investigación los objetivos a cumplirse y la hipótesis planteada, la misma que será comprobada al final de la investigación.

El Capítulo II Marco Teórico, comprende el estudio y definiciones de las herramientas de benchmarking, de pruebas unitarias y de java server faces lo que servirá para el desarrollo de esta investigación.

En el Capítulo III Comparación de Herramientas de Benchmarking, presenta la información acerca de la determinación de las herramientas a comparar el análisis y la descripción de las mismas, así también los indicadores de rendimiento, la creación de prototipos, la realización de pruebas, valoración de las herramientas y la comprobación de hipótesis.

En el Capítulo IV se detalla el desarrollo del proceso de Gestión de Viáticos de la ESPOCH en el cual se implementara las pruebas unitarias con la herramienta de mejor rendimiento.

CAPÍTULO I

MARCO REFERENCIAL

1.1. Antecedentes

En la actualidad se ve la necesidad del uso de herramientas Benchmarking para conocer el rendimiento efectivo de los procesos del negocio con el propósito de mejorarlos en base a los resultados por estas herramientas.

“Benchmarking es una herramienta que nos ayuda a comparar y medir el desempeño del negocio contra uno de la clase mundial” [The James Group].

A pesar de la existencia de muchas herramientas tales como: Httpunit, Htmlunit, Selenium, JwebUnit, Shale, Cactus; en el mercado, aún no existe un estudio que permita determinar cuál es la mejor herramienta para cumplir con este objetivo.

En la búsqueda de mejores prácticas en el desarrollo de aplicaciones web, con el propósito de mejorar los resultados en base al rendimiento se ve en la necesidad de realizar pruebas unitarias para determinar posibles incoherencias en los procesos, permitiendo realizar cambios y mejoras en la aplicación de una forma continua, sabiendo que una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código, que sirve para asegurar que cada uno de estos funcione correctamente por separado. Luego, con las Pruebas de Integración, se podrá asegurar el correcto funcionamiento del sistema o subsistema en cuestión.

JSF (JavaServer Faces) es un framework de interfaces para aplicaciones web basadas en tecnología Java y en el patrón MVC(Modelo Vista Controlador), el mismo que ofrece ventajas como: flexibilidad al permitir crear nuestros propios componentes y renderizadores personalizados en la forma que más nos convenga, separación entre el comportamiento y la presentación permitiendo a cada miembro del equipo de desarrollo centrarse en su parte asignada, además es que pueden encontrarse implementaciones de distintos fabricantes esto permite no vincularse exclusivamente con un proveedor concreto, y poder seleccionar el que más interese en cada caso según: el número de componentes que suministra, el rendimiento de estos, soporte proporcionado, precio, política de evolución, etc.

El proceso de Gestión de Viáticos es el acto administrativo, por medio del cual, la autoridad competente de la Escuela Superior Politécnica de Chimborzao, dispone y autoriza la movilización fuera de su lugar habitual de trabajo de las autoridades, empleados, funcionarios, servidores y representantes de gremios declarados en comisión

de servicio en la institución, para cumplir con actividades inherentes a los fines y objetivos de la ESPOCH.

La Comisión de Servicios se determina en una “Solicitud de Movilización”, donde se especificará los nombres, funciones que cumple el Comisionado en la ESPOCH, además, lugar, motivo de la comisión, fecha de salida y retorno, y tipo de transporte a utilizarse, adjuntando para el efecto el correspondiente plan de trabajo aprobado por el Ordenador del Gasto.

Una vez cumplida la comisión de servicios se deberá presentar el formulario de solicitud de liquidación y el Informe de Comisión especificando las tareas cumplidas, aprobado por el jefe inmediato, ante la Dirección Financiera adjuntando, las facturas de gasolina, transporte y hotel.

Este proceso se lo lleva manualmente, en un inicio se lo registraba en un Inventario de Viáticos, debido a la gran demanda de movilizaciones y para agilizar de manera previa la búsqueda de personas que realizaban la gestión, se optó por crear un archivo en Excel en el que se almacenan los datos de cada gestión realizada, este mecanismo no permite obtener información detallada y exacta, no permite filtrar datos de acuerdo a las necesidades de la institución, a más de tener una interfaz muy poco amigable.

Otra de las grandes falencias de este proceso, es al momento de revisar la Solicitud de Movilización, ya que al no contar con un formato estándar para la presentación de estas vienen en diferentes formatos y no cumplen con todos los requisitos necesarios para la creación de la Orden de Movilización, por lo que la secretaria tiene que realizar varias

llamadas telefónicas para poder completar los datos solicitados y verificar la disponibilidad de los vehículos institucionales.

De igual manera al momento de presentar el formulario de solicitud de liquidación y el Informe de Comisión no existe un estándar que facilite la verificación de las gestiones realizadas.

Por lo que la ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO se ve en la necesidad de desarrollar e implementar un Sistema para la Gestión de Viáticos.

En vista de todos los beneficios que nos brinda el uso de JSF y ayudados de las pruebas unitarias, el Sistema a desarrollar cubrirá todas las necesidades presentes actualmente de manera eficiente garantizando así que el proceso de Gestión de Viáticos se lo realice de la mejor manera.

1.2. Justificación

1.2.1. Justificación Teórica

En definitivas cuentas, la tecnología Java Server Faces proporciona una rica arquitectura para manejar el estado de los componentes, procesar los datos, validar la entrada del usuario, y manejar eventos.

El uso de herramientas Benchmarking ayudará en la obtención de un sistema web de calidad ya que en el transcurso del desarrollo del sistema nos permitirá mejorar cada uno de los procesos en base a los resultados obtenidos al realizar las pruebas unitarias.

1.2.2. Justificación Práctica

En vista de la necesidad del Vicerrectorado de Investigación y Desarrollo y el Director Financiero, de mantener en forma actualizada, un registro de los comisionados, objeto y valores entregados, se cree conveniente desarrollar un Sistema de Gestión de Viáticos el mismo que agilizará este proceso, ya que se establecerá un estándar para la creación de Solicitud de Movilización, lo que implica cumplir con todos los requisitos necesarios para generar la Orden de Movilización de manera automática debido a que los datos serán extraídos directamente de la Base de Datos de Recursos Humanos y Movilización, retornando al Sistema Financiero el monto a liquidar a cada comisionado.

1.3. Objetivos

1.3.1. Objetivo General

- Comparar las herramientas de BENCHMARKING para realizar pruebas unitarias web de aplicaciones JSF aplicando al Sistema de Gestión de Viáticos de la ESPOCH.

1.3.2. Objetivos Específicos

- Estudiar las características de las herramientas BENCHMARKING más utilizadas: Httpunit, Htmlunit, Selenium, Jwebunit, Shale, csUnit, Cactus.
- Definir los parámetros para el análisis de las herramientas a utilizar.
- Elaborar prototipos para la realización de pruebas unitarias.
- Determinar cuáles son las herramientas más adecuadas para la realización de pruebas unitarias en aplicaciones web con tecnología JSF.
- Desarrollar un Sistema para la Gestión de Viáticos de la ESPOCH.

1.4. Hipótesis

La comparación de herramientas de benchmarking para pruebas unitarias web de aplicaciones JSF permitirá escoger la herramienta de mejor rendimiento para realizar pruebas unitarias en el Sistema de Gestión de Viáticos de la ESPOCH.

1.5. Métodos y Técnicas

1.5.1. Métodos

El método utilizado como guía para la presente investigación es el método Científico, el cual contempla los siguientes puntos:

- El planteamiento del problema que en este caso es el análisis de herramientas de Benchmarking para Pruebas Unitarias Web de aplicaciones JSF
- El apoyo del proceso previo a la formulación de la Hipótesis.
- Levantamiento de información necesaria.
- Análisis e interpretación de Resultados.
- Proceso de Comprobación de la Hipótesis.
- Para el desarrollo del módulo aplicaremos la Metodología de Desarrollo XP (Programación Extrema).

1.5.2. TÉCNICAS

Para la recopilación de la información necesaria que sustente el presente trabajo de investigación, se ha establecido como técnicas las siguientes:

- Revisión de Artículos Científicos acerca de Benchmarking
- Observación

- Técnicas de Comprobación de hipótesis.
- Pruebas.

CAPÍTULO II

MARCO TEÓRICO

2.1. Benchmarking

El Benchmarking es un proceso de investigación que proporciona información valiosa para el mejoramiento de los procesos con desempeños sub-óptimos y no simplemente da respuestas sencillas sobre el porqué del problema en el proceso. El cambio de las prácticas comerciales por las de otras de clase mundial no es el objetivo del benchmarking.

El benchmarking es medición y recopilación, un proceso que consiste en aprender de otros y obtener los mayores beneficios de lo aprendido y no es simplemente copiar o imitar lo que otros hacen, esta confusión se debe a que muchas personas dentro de la organización no poseen la información adecuada sobre el proceso que se realiza o no conocen la forma en que se implementa.

2.1.1. Tipos de Benchmarking

Existen tres tipos de Benchmarking definidos, todos ellos con el mismo proceso dependiendo del enfoque que sea utilizado para implementar este proceso, estos tipos se diferencian básicamente en el enfoque desde el punto de vista de con quien se hace Benchmarking, a continuación se describe cada uno de los tipos de Benchmarking de fuentes consultadas.

2.1.1.1. Benchmarking Interno

Este tipo de Benchmarking se acostumbra a hacer en compañías multinacionales en las cuales una empresa de determinada región adopta determinados procesos de otra sucursal de la misma compañía, adaptando dicho proceso a la cultura organizacional de la empresa, ya que existen diferencias entre los distintos procesos de trabajo de una organización derivados de las diferencias de aspectos tales como, la geografía, la historia local de la organización, el ambiente laboral, el potencial del mercado adquisitivo de la región y muchos otros. Es un tipo de Benchmarking que las empresas grandes utilizan para identificar al “mejor de la familia” y extender el conocimiento sobre estas prácticas a lo largo de la organización y es realizado como primer paso de lo que será más adelante un estudio enfocado al exterior. El objetivo del Benchmarking interno es identificar los estándares de desarrollo de la organización.

2.1.1.2. Benchmarking Competitivo

Es la comparación de los estándares de una organización, con los de otras empresas (competidoras). Este suele ser el más conocido por las empresas. Podremos observar, por lo tanto, cómo han funcionado nuevas tecnologías o métodos de trabajo en otras

organizaciones .En general consiste en efectuar pruebas de comparación así como investigaciones que nos permitan conocer todas las ventajas y desventajas de nuestros competidores más directos, este trata de evaluar los productos, servicios y procesos de la organización con actividades similares que ha identificado como las más exitosas de la competencia. Se realiza entre competidores pertenecientes a un mismo sector o actividad.

Su objetivo es medir funciones, procesos, productos o servicios de la organización y compararlos con los de la competencia con el fin de lograr que sean los mejores en su clase.

2.1.1.3. Benchmarking Cooperativo o Multifuncional

Consiste en comparar los procesos de la organización con empresas y/o funciones que son competidores directos o indirectos, o simplemente con organizaciones que no compiten entre sí pero que son reconocidas como las mejores en su clase a nivel mundial. Este tipo es el más usado actualmente, existiendo consorcios de grandes empresas que comparten información variada sobre sus procesos. [12]

2.1.2. Características del Benchmarking

2.1.2.1. Calidad

Entre los aspectos tenemos a la calidad, que se refiere al nivel de valor creado de los productos para el cliente sobre el costo de producirlos. Dentro de este aspecto el benchmarking puede ser muy importante para saber la forma en que las otras empresas forman y manejan sus sistemas de calidad, aparte de poder ser usado desde un punto de vista de calidad conforme a la calidad percibida por los clientes, la cual es determinada por la relación con el cliente, la satisfacción del mismo y por último las comparaciones

con la competencia. También se puede ver el aspecto de la calidad conforme a lo que se llama calidad relativa a normas, la cual se refiere a diseñar sistemas de calidad que aseguren que la calidad resultante de los mismos se apegará o cumplirá con especificaciones y estándares predeterminados, lo cual se puede hacer a través de revisar el proceso de desarrollo y diseño, los procesos de producción y distribución y los procesos de apoyo como contabilidad, finanzas, etc. Por último dentro del aspecto de calidad se puede ver lo referente al desarrollo organizacional en base a que tanto nos enfocamos en lo que hacemos, en el desarrollo del recurso humano, en el compromiso e involucramiento del mismo, así como en el entrenamiento.

2.1.2.2. Productividad

El benchmarking de productividad es la búsqueda de la excelencia en las áreas que controlan los recursos de entrada, y la productividad puede ser expresada por el volumen de producción y el consumo de recursos los cuales pueden ser costos o capital.

2.1.2.3. Tiempo

El estudio del tiempo, al igual que de la calidad, simboliza la dirección del desarrollo industrial en los años recientes. Flujos más rápidos en ventas, administración, producción y distribución han recibido una mayor atención como un factor potencial de mejora de la productividad y la competencia. El desarrollo de programas enfocados en el tiempo ha demostrado una habilidad espectacular para recortar los tiempos de entrega. [1]

2.1.3. Etapas de Benchmarking

El proceso de benchmarking se puede describir como un proceso estructurado [“Michael J. Spendolini”].

El modelo del proceso de Benchmarking tiene cinco etapas:

1. Determinar a qué se le va a hacer benchmarking.

La primera etapa del proceso es identificar a los clientes para la información del benchmarking y sus necesidades y definir los asuntos específicos a los cuales se les va a hacer benchmarking. Definido esto, se procede a identificar y a asegurar los recursos necesarios (por ejemplo tiempo, presupuesto, personal) para llevar a cabo una exitosa investigación.

2. Formar un equipo de Benchmarking

Aunque el benchmarking puede ser conducido por individuos, la mayor parte de los esfuerzos de benchmarking son actividades de equipo. El proceso de escoger, orientar y dirigir un equipo es la segunda etapa importante del proceso de benchmarking. Los papeles y las responsabilidades específicas se les asignan a los miembros del equipo. Se introducen herramientas de manejo de proyectos para garantizar que las tareas de benchmarking sean claras para todas las personas involucradas y para que las etapas clave del proyecto sean identificadas.

3. Identificar a los socios del Benchmarking

La etapa del proceso es identificar fuentes de información que se utilizarán para recopilar la información de benchmarking. Estas fuentes son empleados de organizaciones en que

se practica el benchmarking, asesores, analistas, fuentes gubernamentales, literatura de administración y marketing, informes de otras organizaciones y bases de datos computarizadas por nombrar algunas. También se incluye en esta etapa el proceso de identificación de las mejores prácticas industriales y organizacionales.

4. Recopilar y analizar la información de Benchmarking

Durante esta etapa del proceso, se seleccionan los métodos específicos de recopilación de la información. Es importante que los responsables de esta actividad sean expertos en estos métodos. Se contactan los socios del Benchmarking, y se recopila la información de acuerdo con el protocolo establecido, y luego se resume para hacer el análisis. La información se analiza de acuerdo con las necesidades del cliente original, y se producen recomendaciones para la acción.

5. Actuar

Esta etapa del proceso está influenciada por las necesidades del cliente original y por los usos de la información de benchmarking. La acción que se realiza puede oscilar entre producir un solo informe o producir un conjunto de recomendaciones para la implementación real del cambio, basado, al menos en parte, en la información recopilada durante la investigación de benchmarking. Se incluyen cualesquiera pasos siguientes o actividades apropiadas de seguimiento, incluyendo la continuación del proceso de benchmarking. [5]

2.1.4. Ventajas del Benchmarking

Benchmarking permite:

- Identificar oportunidades de innovación a través del descubrimiento de nuevas tecnologías, ya aplicadas en su propio sector u otros diferentes.
- Identificar aquellos procesos en los que existan diferencias significativas respecto al “mejor del sector”, utilizándolo como estímulo para el cambio y como instrumento de seguimiento de las mejoras producidas.
- Conocer la posición relativa frente a empresas del propio sector o de otros, evitando el estancamiento y ofreciendo diferentes alternativas.
- Conocer con suficiente anterioridad nuevas tendencias y direcciones estratégicas y, en función de éstas, gestionar adecuadamente el cambio.
- Detectar cambios y tendencias en los mercados
- Seguimiento a relaciones y desarrollo de planes de colaboración. [20]

2.2. Pruebas Unitarias

Una prueba unitaria es una prueba que comprueba el correcto funcionamiento de una funcionalidad incluida en la aplicación, tanto en las condiciones favorables como en las no favorables (por ejemplo, entradas correctas e incorrectas del usuario para un determinado valor). Las pruebas unitarias normalmente se desarrollan como funciones que utilizan los componentes ya desarrollados, y se lanzan como un conjunto de pruebas que sólo tienen dos posibles resultados: correcta o incorrecta.

2.2.1. Características de una buena prueba unitaria

Las pruebas unitarias se tienen que poder ejecutar sin necesidad de intervención manual. Esta característica posibilita que podamos automatizar su ejecución.

Las pruebas unitarias tienen que poder repetirse tantas veces como uno quiera. Por este motivo, la rapidez de las pruebas tiene un factor clave. Si pasar las pruebas es un proceso lento no se pasarán de forma habitual, por lo que se perderán los beneficios que éstas nos ofrecen.

Las pruebas unitarias deben poder cubrir casi la totalidad del código de nuestra aplicación. Una prueba unitaria será tan buena como su cobertura de código. La cobertura de código marca la cantidad de código de la aplicación que está sometido a una prueba. Por tanto, si la cobertura es baja, significará que gran parte de nuestro código está sin probar.

Las pruebas unitarias tienen que poder ejecutarse independientemente del estado del entorno. Las pruebas tienen que pasar en cualquier ordenador del equipo de desarrollo.

La ejecución de una prueba no puede afectar la ejecución de otra. Después de la ejecución de una prueba el entorno debería quedar igual que estaba antes de realizar la prueba.

Las diferentes relaciones que puedan existir entre módulos deben ser simuladas para evitar dependencias entre módulos.

Es importante conocer claramente cuál es el objetivo del test. Cualquier desarrollador debería poder conocer claramente cuál es el objetivo de la prueba y su funcionamiento. Esto sólo se consigue si se trata el código de pruebas como el código de la aplicación. Es importante tener en cuenta que aunque estas son las características de una buena prueba, no siempre será posible ni necesario cumplir con todas estas reglas y será la experiencia la que nos guiará en la realización de las mismas.

2.2.2. Beneficios de las pruebas unitarias

Con las pruebas unitarias todos ganan. La vida de desarrollador será mucho más fácil, ya que la calidad de su código mejorará, se reducirán los tiempos de depuración y la corrección de incidencias y por tanto el cliente estará mucho más contento porque la aplicación hace lo que él quiere que haga, por lo que ha pagado.

Las pruebas fomentan el cambio y la refactorización. Si consideremos que nuestro código es mejorable podemos cambiarlo sin ningún problema. Si el cambio no estuviera realizado correctamente las pruebas nos avisarán de ello. Seguramente la frase “si funciona no lo toques” a más de uno familiar les resultará familiar. Si hubiera pruebas unitarias, no sería necesario pronunciarla.

Se reducen drásticamente los problemas y tiempos dedicados a la integración. En las pruebas se simulan las dependencias lo que nos permite que podemos probar nuestro código sin disponer del resto de módulos. Por experiencia puede decir que los procesos de integración son más de una vez traumáticos, dejándolos habitualmente para el final del proyecto. La frase “sólo queda integrar” haciendo referencia a que el proyecto está cerca de terminar suele ser engañosa, ya que el periodo de integración suele estar lleno de curvas.

Las pruebas nos ayudan a entender mejor el código, ya que sirven de documentación. A través de las pruebas podemos comprender mejor qué hace un módulo y que se espera de él.

Nos permite poder probar o depurar un módulo sin necesidad de disponer del sistema completo. Aunque seamos los propietarios de toda la aplicación, en algunas situaciones

montar un entorno para poder probar una incidencia es más costoso que corregir la incidencia propiamente dicha. Si partimos de la prueba unitaria podemos centrarnos en corregir el error de una forma más rápida y lógicamente, asegurándonos posteriormente que todo funciona según lo esperado.

Para el mundo Java hay un estándar open source para el desarrollo de pruebas unitarias para los componentes desarrollados en lenguaje Java: JUnit. El framework JUnit se extiende a menudo con otra serie de frameworks ¹que realizan pruebas unitarias más específicas, insertándose a menudo como plugins ²en los entornos de desarrollo.

Toda aplicación Web se ejecuta dentro de un servidor Web o de un servidor de aplicaciones. Estos elementos hacen de contenedor para la aplicación, conteniendo los recursos, datos y objetos que la forman.

Hay varias formas de realizar pruebas unitarias sobre una aplicación Web:

- Pruebas desde el interior del contenedor de la aplicación. En este caso se instala un módulo en el contenedor de la aplicación (normalmente un módulo en el servidor Web o de aplicaciones), que le dota de esta capacidad.
- Pruebas desde el exterior del contenedor de la aplicación. En este caso, mediante aplicaciones externas a la aplicación web, se ejecutan pruebas contra la aplicación, que normalmente son peticiones de páginas o servicios, simulando la interacción de los usuarios o a
- Pruebas sin utilizar un contenedor de la aplicación.

¹ Framework son módulos de software reusables.

² Plugin es un módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.

Para realizar las pruebas unitarias se utiliza normalmente un framework (marco de trabajo) que nos permitirá realizar dichas pruebas en nuestra aplicación.

En el primer caso, el framework se integra dentro del contenedor, bien utilizando la API del contenedor o extendiendo su funcionalidad, por ejemplo, añadiendo un módulo que intercepte el diálogo entre el cliente Web y el servidor. Es el método más complejo, pues supone controlar desde el propio contenedor la aplicación, obligándonos a usar las siempre complejas APIs ³de bajo nivel de contenedor. Este método está especialmente indicado para probar los componentes críticos, el corazón de nuestras aplicaciones, y es el método que normalmente se utiliza para probar los propios frameworks utilizados para el desarrollo de aplicaciones.

En el segundo caso, las pruebas se realizan desde el exterior del contenedor, desde el lado del cliente. Una técnica habitual para ello consiste en implementar el protocolo http en un cliente de pruebas unitarias que simula la interacción entre un usuario y el servidor Web. Este método está especialmente indicado para comprobar la correcta navegación dentro de la aplicación, así como para detectar problemas en la interfaz visual de la misma, ya que normalmente los framework de pruebas pueden inspeccionar la vista devuelta por la aplicación en cada momento.

El tercer método consiste en implementar dentro del framework parte de la funcionalidad del propio contenedor de la aplicación. Este método es especialmente útil para aplicaciones basadas en componentes, ya que los componentes suelen encerrar una funcionalidad muy definida de la aplicación Web, lo que facilita el desarrollo de las

³ API es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

pruebas unitarias. Una ventaja adicional de este método es que es muy sencillo realizar las pruebas unitarias dentro del proceso de compilación, en la misma máquina virtual donde se está compilando la aplicación, sin necesidad de realizar el despliegue de la misma a los servidores web o de aplicación. [15]

2.3. Java Server Faces

JavaServer Faces (JSF) es el framework para aplicaciones Web en Java de Sun Microsystems, liberado apenas en Marzo del 2004, que busca tomar su lugar como estándar entre los muchos de su clase.

JSF es un framework orientado a la interfaz gráfica de usuario (GUI), facilitando el desarrollo de éstas, y que sin embargo, realiza una separación entre comportamiento y presentación, además de proporcionar su propio servlet como controlador, implementando así los principios del patrón de diseño Model-View-Controller (MVC), lo que da como resultado un desarrollo más simple y una aplicación mejor estructurada. El enfoque mencionado anteriormente no es nada nuevo. Lo que hace a JSF tan atractivo, entre muchas otras cosas más, es que brinda un modelo basado en componentes y dirigido por eventos para el desarrollo de aplicaciones Web, que es similar al modelo usado en aplicaciones GUI standalone ⁴durante años [Bergsten, 2004], como es el caso de Swing, el framework estándar para interfaces gráficas de Java.

Otra característica muy importante de JavaServer Faces es que, a pesar de que HTML es su lenguaje de marcado por default⁵, no está limitado a éste ni a ningún otro, pues tiene la capacidad de utilizar diferentes renderers ⁶para los componentes GUI y

⁴ Standalone es Una entidad independiente es algo que no tiene ninguna dependencia.

⁵ Default configuración de un programa por defecto.

⁶ Renderers es el proceso de generar una imagen desde un modelo.

obtener así diferentes salidas para mandar al cliente [Bergsten, 2004]. Así mismo, JSF es suficientemente flexible para soportar diversas tecnologías de presentación [Bergsten, 2004], destacando entre estas JSP, ya que es una tecnología soportada, requerida y especificada para toda implementación de JavaServer Faces.

2.3.1. Arquitectura

El framework JSF implementa el patrón de diseño MVC (Modelo Vista Controlador) como se muestra en la **Figura 1**, Permitiendo una separación clara entre el código de interfaz y el de lógica de negocio.

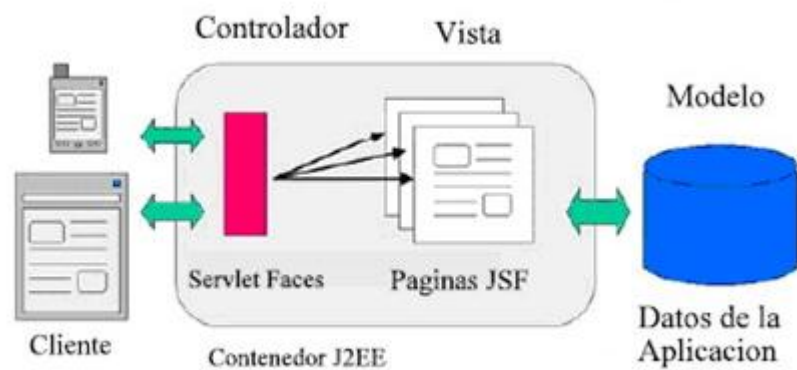


Figura II. 1 Arquitectura Java Server Faces

2.3.1.1. Modelo

Objeto que representa y trabaja directamente gestionando los datos y controlando todas sus transformaciones. El modelo no tiene ninguna referencia de las vistas. JSF se comunica con el modelo de la aplicación a través del fichero **faces-config.xml** donde se detalla un objeto perteneciente a la clase del dominio Usuario, la clase a la que hace referencia y el ámbito de sesión.

2.3.1.2. Vista

Objeto que maneja la presentación visual de los datos gestionados por el Modelo. Genera una representación visual del modelo y muestra los datos al usuario. En nuestro proyecto vendrá determinado por cada una de las páginas JSF. La manera de conectar el modelo con la vista a través de componentes JSF es haciendo referencia a objetos del modelo existentes en el controlador.

2.3.1.3. Controlador

Objeto que actúa sobre los datos del modelo y que entra en acción cuando se realiza una petición por parte de la vista o una actualización por parte del modelo hacia la vista. En JSF opera como un gestor que reacciona ante los eventos provocados por el usuario, procesa sus acciones y los valores de estos eventos, ejecutando código para actualizar el modelo o la vista. La regla de navegación estará definida en el archivo faces-config.xml.

2.3.2. Ciclo de Vida

Otro aspecto muy importante dentro de JavaServer Faces es su ciclo de vida, el cual es similar al de una página JSP: el cliente hace una petición HTTP y el servidor responde con la página en HTML. Sin embargo, debido a las características que ofrece JSF, el ciclo de vida incluye algunos pasos más [Ball, 2003].

El proceso de una petición estándar incluye seis fases, como se muestra en la **Figura II.2**, representada por los rectángulos blancos:

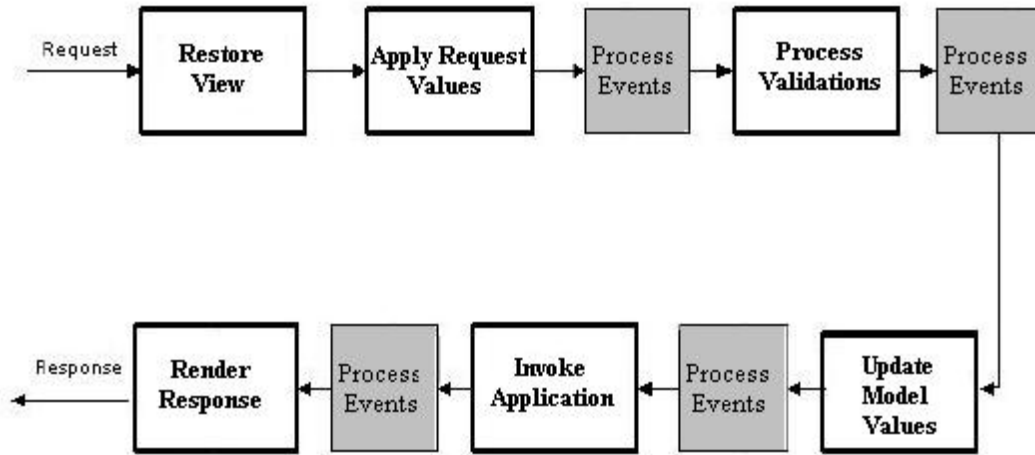


Figura II. 2 Ciclo de vida de Java Server Faces

Los rectángulos grises etiquetados con la leyenda “*Process Events*” representan la ejecución de cualquier evento producido durante el ciclo de vida.

El funcionamiento de cada etapa se describe brevemente a continuación.

2.3.2.1. Restore View

También llamada *Reconstitute Component Tree*, es la primera etapa que se lleva a cabo, e inicia cuando se hace una petición. Su objetivo es la creación de un árbol con todos los componentes de la página en cuestión.

2.3.2.2. Apply Request Values

Cada uno de los componentes del árbol creado en la fase anterior obtiene el valor que le corresponde de la petición realizada y lo almacena.

2.3.2.3. Process Validations

Después de almacenar los valores de cada componente, estos son validados según las reglas que se hayan declarado.

2.3.2.4. Update Model Values

Durante esta fase los valores locales de los componentes son utilizados para actualizar los *beans* que están ligados a dichos componentes [Geary & Geary, 2004]. Esta fase se alcanzará únicamente si todas las validaciones de la etapa anterior fueron exitosas.

2.3.2.5. Invoke Application

Se ejecuta la acción u operación correspondiente al evento inicial que dio comienzo a todo el proceso.

2.3.2.6. Render Response

La respuesta se renderiza y se regresa al cliente. Dependiendo del éxito o fracaso de las tareas en cada una de las fases del ciclo de vida, el flujo normal descrito puede cambiar hacia caminos alternos según sea el caso.

2.3.3. Estructura de una aplicación JSF

Se necesitan dos cosas para correr aplicaciones con JavaServer Faces: un contenedor Web para Java y una implementación de la especificación de JSF [Bergsten, 2004]. Debido a esto, la estructura del directorio de una aplicación JSF podría verse de la manera como lo muestra la **Figura II. 3**.

Los elementos propios de JavaServer Faces que conforman una aplicación son típicamente los siguientes:

2.3.3.1. Archivos XHTML

Constituyen la interfaz gráfica de la aplicación y que contienen las diversas funcionalidades de JSF, como los *tags* que representan los componentes GUI.

2.3.3.2. Archivos XML

Específicamente el archivo *faces-config.xml* que almacena las diferentes configuraciones y elementos a utilizar en la aplicación.

2.3.3.3. Archivos Java

Típicamente desempeñando el rol de *beans*.

2.3.3.4. Archivos de paquetes de mensajes (*message bundles*).

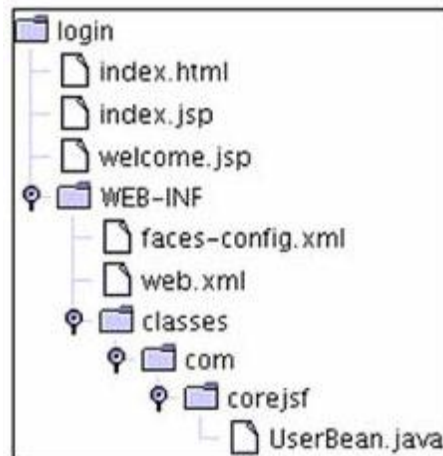


Figura II. 3 Estructura de directorio de una aplicación JSF

2.3.4. Cómo funciona JSF

Normalmente las aplicaciones web se construyen como un conjunto de pantallas con las que va interactuando el usuario. Estas pantallas contienen textos, botones, imágenes, tablas y elementos de selección que el usuario modifica.

Todos estos elementos estarán agrupados en formularios HTML, que es la manera en que las páginas web envían la información introducida por el usuario al servidor.

La principal función del controlador JSF es asociar a las pantallas, clases java que recogen la información introducida y que disponen de métodos que responden a las acciones del usuario. JSF nos resuelve de manera muy sencilla y automática muchas tareas:

- Mostrar datos al usuario en cajas de texto y tablas.
- Recoger los datos introducidos por el usuario en los campos del formulario.
- Controlar el estado de los controles del formulario según el estado de la aplicación, activando, ocultando o añadiendo y eliminando controles y demás elementos
- Realizando validaciones y conversiones de los datos introducidos por el usuario
- Rellenando campos, listas, combos y otros elementos a medida que el usuario va interactuando con la pantalla
- Controlando los eventos que ocurren en los controles (pulsaciones de teclas, botones y movimientos del ratón).

Las aplicaciones JSF están formadas por los siguientes elementos principales:

- Páginas JSP que incluyen los formularios JSF. Estas páginas generarán las vistas de la aplicación
- Beans java que se conectan con los formularios JSF
- Clases java para la lógica de negocio y utilidades.
- Ficheros de configuración, componentes a medida y otros elementos del framework.
- Resto de recursos de la aplicación web: recursos estáticos, javascript y otros elementos. [9]

CAPÍTULO III

COMPARACIÓN DE HERRAMIENTAS DE BENCHMARKING PARA PRUEBAS UNITARIAS WEB DE APLICACIONES JSF.

El Benchmarking es una herramienta que permite contar con un método en donde se pueden evaluar cada uno de los procesos del desarrollo del software, los cuales pueden mejorar significativamente el producto final, ya que permite utilizar la experiencia de otros proyectos similares a través de la comparación de características comunes.

También se puede observar que el Benchmarking es una poderosa herramienta capaz de darnos información valiosa sobre los procesos que se implementan en el desarrollo de software, tales como maximizar el rendimiento, así como también minimizar lo relacionado con el costo de mantenimiento.

En cuanto a las pruebas unitarias, se están convirtiendo poco a poco en una métrica de calidad del software desarrollado actualmente.

Una prueba unitaria es una prueba que comprueba el correcto funcionamiento de una funcionalidad incluida en la aplicación, tanto en las condiciones favorables como en las no favorables (por ejemplo, entradas correctas e incorrectas del usuario para un determinado valor).

3.1. Determinación de las herramientas a comparar

Para la realización de pruebas unitarias en el mundo Java en aplicaciones web se cuenta con varias herramientas en el dominio open source⁷. En este caso se ha escogido las siguientes herramientas.

3.1.1. Httpunit

Es una extensión de Junit⁸ para la realización de pruebas unitarias sobre el protocolo http.

Características

- Emula las partes pertinentes al comportamiento del navegador, incluyendo el envío de formularios, JavaScripts y la redirección automática de páginas.
- No indican el grado de soporte de funciones javascript.
- Su última versión lanzada fue la HttpUnit 1.7, el 20 de Mayo del 2008.
- No es adecuado para realizar pruebas en componentes basados en Modelo Vista Controlador [7].

⁷ **Open Source:** Código abierto es el término con el que se conoce al software distribuido y desarrollado libremente.

⁸ **Junit:** Es un conjunto de bibliotecas que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java.

3.1.2. Htmlunit

Es una extensión de Junit para la realización de pruebas unitarias web. Su funcionamiento se basa en obtener los diferentes elementos que componen la página web devuelta por el servidor, e interactuar sobre ellos.

Características

- No cuenta con una interfaz de usuario.
- Validación para todas las etiquetas html: title, table, etc.
- Sintaxis bastante compleja. Para acceder a los elementos de la página hay que ir instanciando los diferentes objetos que la soportan.
- Gestión parcial de los eventos JavaScript⁹ de los elementos html de la página, todavía no está completo el soporte, aunque sí dispara el onclick().
- El proyecto está bastante activo, y se ha liberado en Marzo 2013 la versión 2.12
- Se utiliza como el browser subyacente mediante diferentes herramientas de código abierto [8].

3.1.3. Selenium

Es un framework en desarrollo al estilo de htmlunit, pero su principal diferencia es que embebe un cliente web real. Se instala como un plugin sobre Firefox o Internet Explorer, y permite trabajar con ellos como cliente web de pruebas, controlando sus funciones, es una herramienta eficaz para la realización de pruebas unitarias.

⁹ JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Características

- Es ideal para probar las interfaces de usuario web.
- Realizar pruebas de compatibilidad con diferentes navegadores.
- Realizar pruebas funcionales de aplicaciones al estilo de htmlunit.
- Su última versión estable es 2.0.0 y fue liberada el 26 Abril del 2013.
- Bibliotecas de cliente para el lenguaje de programación favorito.

3.1.4. Jwebunit

Esta herramienta es una fachada de otra herramienta similar, llamada HttpUnit.

Características

- JWebUnit funciona de manera similar a un navegador, ofreciendo un conjunto de métodos en Java para simular las acciones de un navegador (cargar enlace, rellenar formularios, hacer click) y para verificar los elementos recibidos (el título de la página, las imágenes, las tablas, etc.).
- Esta herramienta ofrece un conjunto muy completo de métodos para interactuar con las páginas HTML.
- No soporta expresiones regulares. Sin embargo, extender JWebUnit para verificar expresiones regulares es una tarea trivial y no realizada por falta de tiempo, lo que se convierte en una limitante para su uso.
- Su última versión estable es la 2.5, lanzada en Octubre 24 del 2010.

3.1.5. Shale

Realmente es un framework para escribir aplicaciones JSF, pero incluye un test-framework que puede ser utilizado para realizar aplicaciones unitarias en aplicaciones JSF sin usar un contenedor, vía objetos mock¹⁰ que emulan al contenedor JSF.

Características

- Manejo de objetos mock que emulan el contexto JSF y el contenedor del Servlet¹¹ de JSF.
- Dispone de elementos para realizar pruebas unitarias sobre servlets y páginas JSP.
- Su última versión estable fue la 1.0.4 lanzada en Marzo 20 del 2008, actualmente esta herramienta esta desuso.

3.1.6. csUnit

Es una herramienta gratuita para realizar pruebas unitarias, y está diseñado para trabajar especialmente con tecnología .NET compatible.

Características

- Facilidad de uso, posee una interfaz amigable para el usuario.
- Viene integrada desde Visual Studio 2005 en adelante.
- Funciona con cualquier lenguaje. NET (C #, VB.NET, Managed C + +, etc).

¹⁰ **Mock** son objetos simulados.

¹¹ Servlet son objetos que corren dentro y fuera del contexto de un contenedor de servlets (ej: Tomcat) y extienden su funcionalidad.

3.1.7. Cactus

Es un framework para la realización de pruebas unitarias de aplicaciones web, y dispone de elementos para realizar pruebas unitarias sobre servlets y páginas jsp.

Características

- Diseñado para probar aplicaciones que siguen el patrón de arquitectura MVC.
- Permite verificar el correcto funcionamiento de los métodos anidados a los beans.
- La documentación de esta herramienta es detallada por lo que su uso es recomendado.
- Su última versión estable es 1.8.1 y fue lanzada en Enero 21 del 2011.

3.1.8. JSFUnit

JSFUnit es un framework de pruebas para aplicaciones JSF. Está diseñada para permitir una completa integración de pruebas y pruebas de unidad de las aplicaciones JSF utilizando un API simplificado.

Características

- JSFUnit corre dentro de un contenedor, el cual provee al desarrollador completo acceso al control de los Beans, el FacesContext, expresiones, etc.
- Pruebas de navegación validando la entrada de usuario sobre el formulario y el acceso a la vista propietaria.
- Verifica que las acciones se ejecutan como se esperan y las condiciones de error se manejan como se espera.
- Su última versión estable es la jboss-JSFUnit-core-2.0.0, lanzada en Septiembre 8 del 2011.

3.2. Análisis de las herramientas seleccionadas

Existe una variedad de cuanto a herramientas de testeo de aplicaciones, en este caso la investigación se enfoca básicamente en el testeo de aplicaciones web JSF, por lo cual se realiza un análisis que permitirá seleccionar las herramientas adecuadas para el cumplimiento de los objetivos planteados, este análisis también implica el descarte de algunas de estas ya que no cumple con los estándares requeridos.

En primer lugar las herramientas deben cumplir con el tipo de pruebas que son de interés para este caso, como son los Test Funcionales que tienen por objetivo probar que los sistemas desarrollados, cumplan con las funciones específicas para los cuales han sido creados.

Las condiciones que deben cumplir las herramientas son las siguientes:

- Emular las acciones de un usuario final.
- Interfaz de grabación.
- Verificar que las acciones se ejecuten correctamente y las condiciones de error puedan ser validadas.
- Debe estar en constante desarrollo.
- Soporta aplicaciones JSF

Las herramientas con las que se va a trabajar deben ofrecer la posibilidad de reproducir las pruebas como si de un usuario final se tratara: ejecutando acciones del navegador automáticamente (clicar un botón, clicar un link, rellenar campos de texto,...).

En segundo lugar se requiere una herramienta que disponga de una interfaz de grabación que sea capaz de generar scripts¹² con la navegación por la página que se desea testear y también sea capaz de reproducirlos.



El tercer punto permite detectar si existe algún error al momento de ejecutar alguna acción.

El cuarto punto es necesario porque se necesita escoger una herramienta que este en constante evolución ya que cada evolución implica una mejora en las bondades que presta la herramienta.

El quinto punto es muy importante en vista de que muchas de las herramientas son para pruebas en páginas estáticas y lo que se busca probar son páginas JSF.

A continuación se procede a evaluar el análisis realizado acerca de las herramientas seleccionadas bajo los parámetros que fueron mencionados anteriormente.

Tabla III. I Leyenda de la Tabla III.II

Dispone	No dispone
	
100%	0%

En la **Tabla III. II**, se muestra un resumen del resultado del análisis de las diferentes herramientas facilitando la interpretación de los mismos.

¹² **Scripts** es un fragmento de código.

Tabla III. II Resumen de comparación de herramientas

Herramienta	Emulador de usuario final	Interfaz de Grabación	Verificación de Acciones	Actualizado	Soporta Aplicaciones Web JSF
Httpunit	✓	✗	✓	✗	✗
Htmlunit	✓	✗	✓	✓	✗
Selenium	✓	✓	✓	✓	✓
Jwebunit	✓	✗	✓	✓	✗
Shale	✓	✗	✓	✗	✓
csUnit	✓	✗	✓	✓	✗
Cactus	✓	✗	✓	✓	✓
JSFUnit	✓	✗	✓	✓	✓

A continuación se da pesos a cada uno de los parámetros los mismos que facilitarían la evaluación de las herramientas.

1. **Emulador de usuario final:** tiene una ponderación del 15% ya que una herramienta de prueba unitaria pretende que la aplicación sea probada antes de ser usada por el cliente real.
2. **Interfaz de grabación:** tiene la menor ponderación que es el 10% ya que si se puede realizar una prueba unitaria sin contar con una interfaz de grabación.
3. **Verificación de acciones:** tiene una ponderación de 30% ya que representa el objetivo mismo de una prueba unitaria.

4. **Actualizado**: su ponderación es de 15% ya que toda actualización implica una mejora de la herramienta, incrementando así las bondades de la misma.
5. **Soporta Aplicaciones Web JSF**: con una ponderación del 30% ya que en esta investigación se requiere específicamente realizar pruebas unitarias en aplicaciones web JSF.

En la **Tabla III.III** se da a conocer cada de las ponderaciones antes expuestas de acuerdo a la herramienta con su respectivo parámetro.

Tabla III. III Ponderaciones de las herramientas según los parámetros

Herramienta	Emulador de usuario final	Interfaz de Grabación	Verificación de Acciones	Actualizado	Soporta Aplicaciones Web JSF	TOTAL
Httpunit	15%	0%	30%	0%	0%	45%
Htmunit	15%	0%	30%	15%	0%	60%
Selenium	15%	10%	30%	15%	30%	100%
Jwebunit	15%	0%	30%	15%	0%	60%
Shale	15%	0%	30%	0%	30%	75%
csUnit	15%	0%	30%	15%	0%	60%
Cactus	15%	0%	30%	15%	30%	90%
JSFUnit	15%	0%	30%	15%	30%	90%

En la **Figura 4**, se observa claramente las herramientas que obtuvieron mayor ponderación una vez realizado el análisis lo que facilitara determinar cuáles son las más adecuadas para realizar pruebas en aplicaciones web JSF.

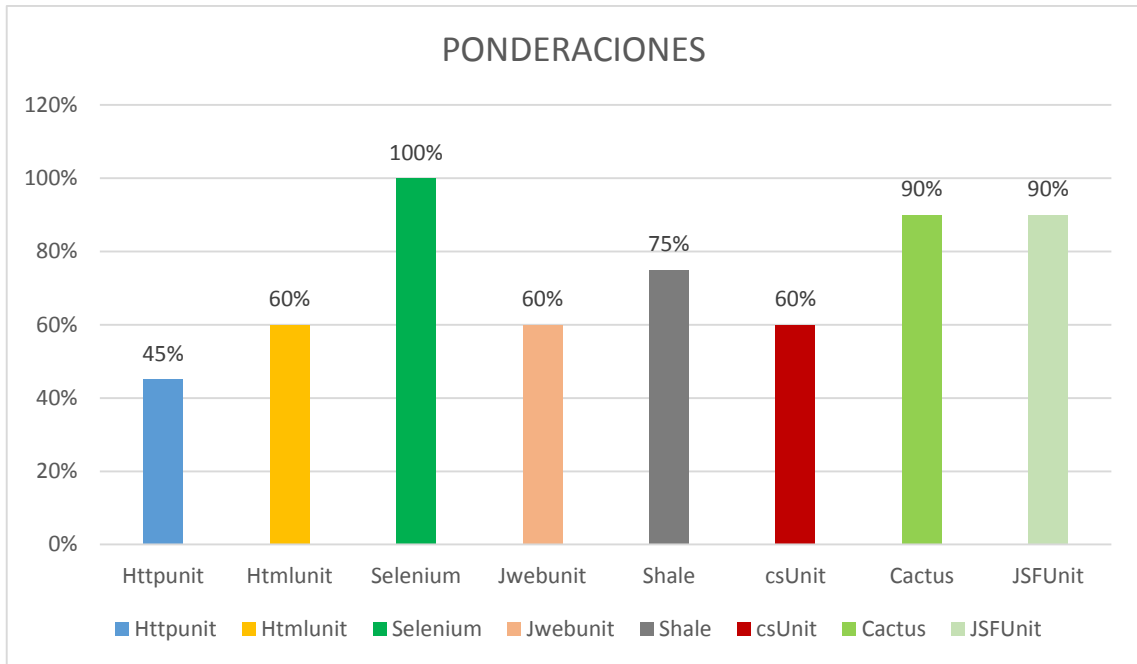


Figura III. 4 Cuadro Estadístico de Ponderaciones

En base a las características expuestas con respecto a cada una de las herramientas y analizando los resultados obtenidos en la **Tabla III. III**, se ha considerado seleccionar las siguientes: Selenium, Cactus y JSFUnit pues son las que más bondades proporcionan al momento de realizar pruebas unitarias en aplicaciones web JSF.

3.3. Descripción de herramientas seleccionadas

3.3.1. Selenium

Selenium es un conjunto de herramientas para automatizar las pruebas sobre aplicaciones web a través de diferentes plataformas.

Características

- Funciona en “diferentes navegadores” y con “diferentes sistemas operativos”.
- Puede ser controlado por muchos lenguajes de programación y frameworks de prueba.

- Permite crear pruebas de regresión.
- Proyecto Open Source (Comunidad OpenQA)

Arquitectura

Está compuesto por:

- ***Selenium IDE***: Creación y mantenimiento de pruebas automatizadas.
- ***Selenium Remote Control (RC)***: Creación de pruebas escritas en lenguajes de programación como Java o C#.
- ***Selenium GRID***: Ejecución de pruebas escritas en los anteriores lenguajes de forma paralela.
- ***Selenium on Rails***: Realización de pruebas sobre aplicaciones Rails con Selenium Core.
- ***Selenium on Ruby***: Proporciona el hub para relacionar Ruby con los proyectos Selenium.
- ***CubicTest***: Plugin gráfico para Eclipse que permite escribir los test de Selenium.

3.3.2. Cactus

Cactus es un marco simple de pruebas unitarias de código Java del lado del servidor (Servlets, EJB, bibliotecas de etiquetas, filtros...).

La intención de Cactus es bajar el costo de escribir las pruebas para el código del lado del servidor. Utiliza JUnit y la amplía.

Cactus implementa una estrategia de in-container, lo que significa que las pruebas se ejecutan en el interior del contenedor.

Estructura

Se implementa el método **setUp()** que se encarga de inicializar variables y objetos usados durante la ejecución de la clase de pruebas unitarias. En la **Figura 4** se muestra el uso del método **Setup()**.

Cabe destacar que para que un método de prueba sea exitoso o no, son usados los métodos “**assert**”, tanto para comparar valores esperados contra valores reales, o directamente hacer que el método de prueba falle. En la **Tabla III. V** se describen los métodos implementados de Cactus.

Tabla III. IV Métodos Assert

Método assert	Comprobación a realizar
assertTrue(expresión)	Verifica que la expresión se evalúe en “true”, en caso contrario marca un fallo en el método de prueba.
assertFalse(expresión)	Verifica que la expresión se evalúe en “false”, en caso contrario el método de prueba falla.
assertEquals(valor esperado, valor real)	Verifica que el valor esperado sea igual al valor real, de lo contrario el método de prueba falla.
assertNotNull(objeto)	Verifica que el objeto no sea nulo, en caso de serlo el método de prueba falla.

assertSame(objeto esperado, objeto real)	Verifica que el objeto esperado sea igual al objeto real, de lo contrario el método de prueba falla.
assertNotSame(objeto esperado, objeto real)	Verifica que el objeto esperado sea diferente al objeto real, en caso contrario el método de prueba falla.
fail()	Forza que el método de prueba finalice con un fallo.

Reglas básicas

Algunas de las reglas básicas para implementar Cactus para la realización de pruebas unitarias son las siguientes:

- Toda clase de prueba desde extender de la clase “ServletTestCase”:

```
public class CiudadC extends ServletTestCase
```

- Todo método de prueba dentro de la clase de prueba debe comenzar con la palabra test:

```
public void testInsertarCiudad() throws Exception
```

3.3.3. JSFUnit

JSFUnit es un framework de pruebas para aplicaciones JSF. Está diseñada para permitir una completa integración de pruebas y pruebas de unidad de las aplicaciones JSF utilizando un API simplificado. JSFUnit corre dentro de un contenedor, el cual provee al

desarrollador completo acceso al control de los Beans, el FacesContext, las expresiones y al componente interno JSF Tree. Al mismo tiempo tiene acceso al HTML de salida enviado para cada uno de los request del cliente.

Características

- Habilidad para correr casos de prueba aislados, fuera del contenedor de aplicaciones o desde los diferentes IDE's, eclipse o netbeans.
- Habilidad para correr pruebas completas al ciclo de vida del HTTP Request-Response.
- Integración con herramientas de construcción tales como Ant y Maven
- Usa la api JSF para llamar los métodos de prueba en vez de revisar los HTML de salida o los objetos de cliente
- Pruebas de navegación validando la entrada de usuario sobre el formulario y el acceso a la vista propietaria
- Verifica que las acciones se ejecutan como se esperan y las condiciones de error se manejan como se espera
- Verifica que los archivos de configuración XML (facesconfig.xml) se carguen correctamente.

3.4. Determinación de los indicadores de comparación

Para la comparación de las herramientas Benchmarking seleccionadas se considera una serie de aspectos sobre el rendimiento de estas, para así determinar que herramienta es la más adecuada para la realización de pruebas unitarias, los indicadores a comparar son los siguientes:

- **Tiempo de ejecución de pruebas:** Permite conocer el tiempo que la herramienta tarda en llevar a cabo una prueba unitaria de un proceso determinado.
- **Líneas de código escritas para las pruebas:** Permite medir la complejidad en la estructura de creación de métodos para ejecutar una prueba unitaria.
- **Flexibilidad para realizar cambios:** Permite valorar la capacidad de realizar cambios en un tiempo moderado.
- **Navegabilidad por la estructura de la aplicación:** Permite valorar la capacidad que tiene la herramienta, para realizar los testing por toda la arquitectura de la aplicación (en este caso MVC)

3.5. Creación del prototipo para las pruebas.

Para la realización de las pruebas unitarias con las diferentes herramientas seleccionadas, se ha escogido cuatro procesos que se efectúan en el Sistema de Gestión de Viáticos estos se describen en la **Tabla III. V.**

Tabla III. V Descripción de Prototipos

No.	Prototipo	Descripción
1	Ingreso de datos de una ciudad.	Permite el ingreso de los datos de una ciudad como son: nombre, y descripción,
2	Actualización de datos de una ciudad.	Permite modificar los datos de una ciudad seleccionada.
3	Carga de datos de las ciudades existentes.	Permite visualizar un listado de las ciudades insertadas.
4	Búsqueda de una ciudad por código.	Permite buscar una ciudad dado su código.

A continuación se detalla el diseño de la arquitectura del proyecto que contiene a estos prototipos.

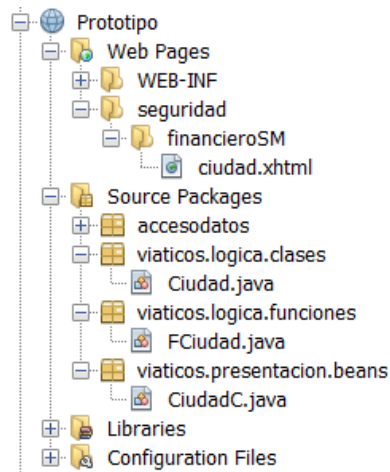


Figura III. 5 Arquitectura de Prototipos

- La carpeta **Web Pages** contiene la vista **ciudad.xhtml** la misma que permite interactuar con el usuario.
- La clase **Ciudad.java** contiene los atributos de la ciudad como son código, nombre y descripción.
- La clase **FCiudad.java** es en la que se desarrollan los métodos necesarios para ingresar u obtener información requerida hacia o desde la base de datos.
- La clase **CiudadC.java** es el controlador que permite la comunicación entre las vistas y las funciones.
- En la carpeta **Libraries** es el contenedor donde se agregan las librerías necesarias para la ejecución de la aplicación.

3.6. Realización de las pruebas

3.6.1. Selenium

Se realizara las pruebas unitarias con el framework selenium empleando la arquitectura que se muestra en la **Figura 6**.

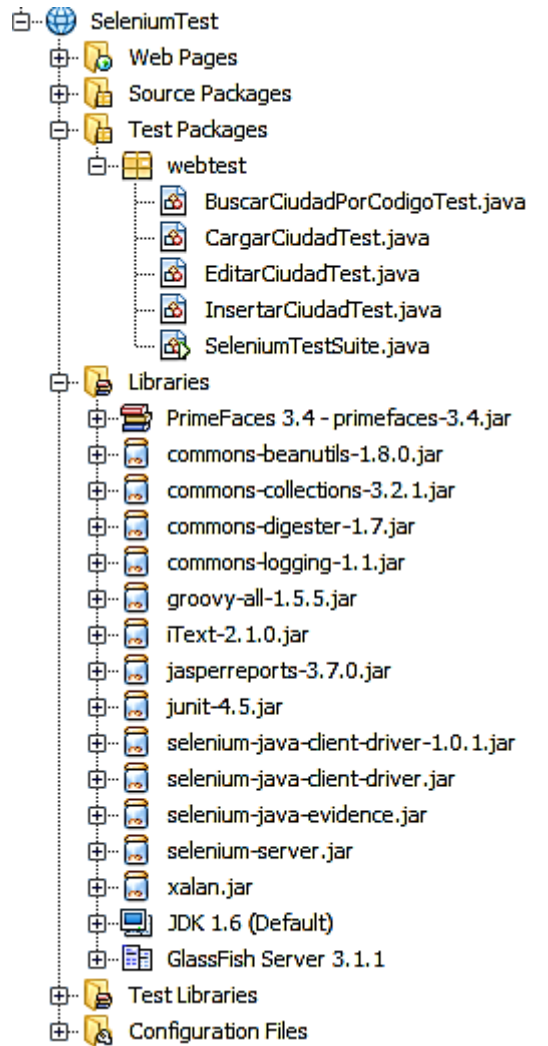


Figura III. 6 Arquitectura SeleniumTest

3.6.1.1. Procedimiento

A continuación se detallan paso a paso el procedimiento para el desarrollo de la prueba unitaria con la herramienta selenium.

- *Crear el proyecto SeleniumTest*
- *Agregamos las librerías:* Para la ejecución de las pruebas de una aplicación son necesarias algunas librerías, las mismas que se muestran en la **Figura 7**.

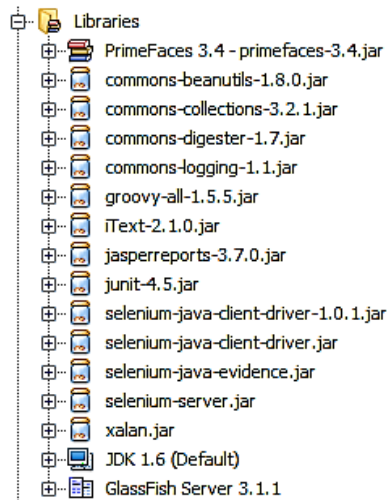


Figura III. 7 Selenium: Librerías

- **Crear las clases test para cada uno de los prototipos definidos:** Cada una de estas clases contiene las instrucciones necesarias para la realización de pruebas unitarias en la aplicación.

Ahora se muestran las clases correspondientes a cada uno de los prototipos.

```
package webtest;

import com.thoughtworks.selenium.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;
import java.util.regex.Pattern;
import junit.framework.TestCase;

public class CargarCiudadTest extends TestCase {

    private Selenium selenium;

    @Before
    public void setUp() throws Exception {
        /**
         * localhost: servidor donde esta montada la aplicacion a testear
         * 444:Puerto del Servidor Selenium *firefox: browser sobre el cual se
         * realizaran las pruebas
         */
        //emula el browser y lo inicializa
        selenium = new DefaultSelenium("localhost", 4444, "**firefox", "http://localhost:8080/");
        selenium.start();
    }

    @Test
    public void testTestSelenium() throws Exception {
        // ejecuta la pagina ciudad.xhtml
        selenium.open("/SeleniumTest/faces/seguridad/financieroSM/ciudad.xhtml");
    }

    @After
    public void tearDown() throws Exception {
        // detiene el browser una vez concluidas las pruebas
        selenium.stop();
    }
}
```

Figura III. 8 Selenium: CargarCiudadTest.java


```
package webtest;

import com.thoughtworks.selenium.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;
import java.util.regex.Pattern;
import junit.framework.TestCase;

public class InsertarCiudadTest extends TestCase {
    private Selenium selenium;
    @Before
    public void setUp() throws Exception {
        /**
         * localhost: servidor donde esta montada la aplicacion a testear
         * 444:Puerto del Servidor Selenium *firefox: browser sobre el cual se
         * realizaran las pruebas
         */
        //emula el browser y lo inicializa
        selenium = new DefaultSelenium("localhost", 4444, "**firefox", "http://localhost:8080/");
        selenium.start();
    }
    @Test
    public void testInsertarCiudad() throws Exception {
        // ejecuta la pagina ciudad.xhtml
        selenium.open("/SeleniumTest/faces/seguridad/financieroSM/ciudad.xhtml");

        // emula la ejecucion de un clic sobre el boton con id=btnNuevaCiudad
        selenium.click("id=frmCiudad:tblCiudad:btnNuevaCiudad");

        // setea los valores para los campos nombreCiudad y observacionCiudad respectivamente
        selenium.type("id=frmNuevaCiudad:nombreCiudad", "NuevaCiudad");
        selenium.type("id=frmNuevaCiudad:observacionCiudad", "NuevaCiudad");

        // emula el evento clic sobre el boton guardarCiudad una vez q se ingreso los datos
        // de la nueva ciudad
        selenium.click("id=frmNuevaCiudad:guardarCiudad");
    }
    @After
    public void tearDown() throws Exception {
        // detiene el browser una vez concluidas las pruebas
        selenium.stop();
    }
}
```

Figura III. 9 Selenium: InsertarciudadTest.java

```
package webtest;

import com.thoughtworks.selenium.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;
import java.util.regex.Pattern;
import junit.framework.TestCase;

public class EditarCiudadTest extends TestCase {
    private Selenium selenium;
    @Before
    public void setUp() throws Exception {
        /**
         * localhost: servidor donde esta montada la aplicacion a testear
         * 444:Puerto del Servidor Selenium *firefox: browser sobre el cual se
         * realizaran las pruebas
         */
        //emula el browser y lo inicializa
        selenium = new DefaultSelenium("localhost", 4444, "*firefox", "http://localhost:8080/");
        selenium.start();
    }
    @Test
    public void testEditarCiudad() throws Exception {
        // ejecuta la pagina ciudad.xhtml
        selenium.open("/SeleniumTest/faces/seguridad/financieroSM/ciudad.xhtml");
        // emula la ejecucion de un clic sobre el boton con id=btnEditarCiudad
        selenium.click("id=frmCiudad:tblCiudad:btnEditarCiudad");

        // setea los valores para los campos nombreCiudadModificar y observacionCiudadModificar
        // respectivamente
        selenium.type("id=frmActualizarCiudad:nombreCiudadModificar", "Nombre ciudad");
        selenium.type("id=frmActualizarCiudad:observacionCiudadModificar", "Descripcion Latacunga");

        // emula el evento clic sobre el boton modificarCiudad una vez q se ingreso los nuevos datos
        // de la ciudad seleccionada
        selenium.click("id=frmActualizarCiudad:modificarCiudad");
    }
    @After
    public void tearDown() throws Exception {
        // detiene el browser una vez concluidas las pruebas
        selenium.stop();
    }
}
```

Figura III. 10 Selenium: EditarCiudadTest.java

```
package webtest;

import com.thoughtworks.selenium.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;
import java.util.regex.Pattern;
import junit.framework.TestCase;

public class BuscarCiudadPorCodigoTest extends TestCase {

    private Selenium selenium;
    @Before
    public void setUp() throws Exception {
        /**
         * localhost: servidor donde esta montada la aplicacion a testear
         * 444:Puerto del Servidor Selenium *firefox: browser sobre el cual se
         * realizaran las pruebas
         */
        //emula el browser y lo inicializa
        selenium = new DefaultSelenium("localhost", 4444, "*firefox", "http://localhost:8080/");
        selenium.start();
    }

    @Test
    public void testBuscarCiudadPorCodigo() throws Exception {
        // ejecuta la pagina busquedaCiudad.xhtml
        selenium.open("/SeleniumTest/faces/seguridad/financieroSM/busquedaCiudad.xhtml");

        // emula la ejecucion de un clic sobre el boton con id=codigoCiudad
        selenium.click("id=frmCiudad:codigoCiudad");

        // emula el ingreso del codigo de la ciudad a buscar
        selenium.type("id=frmCiudad:codigoCiudad", "2");

        // emula el evento clic sobre el boton buscarCiudad una vez q se ingreso
        // el codigo de la ciudad a buscar
        selenium.click("id=frmCiudad:buscarCiudad");
    }

    @After
    public void tearDown() throws Exception {
        selenium.stop();
    }
}
```

Figura III. 11 Selenium: BuscarCiudadPorCodigo.java

- **Agregar la clase SeleniumTestSuite.java:** Esta clase va a permitir ejecutar las clases creadas anteriormente, como se muestra en la **Figura 12**.

```
package webtest;

import junit.framework.TestSuite;
import junit.textui.TestRunner;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import webtest.CargarCiudadTest;
import webtest.InsertarCiudadTest;
import webtest.EditarCiudadTest;
import webtest.BuscarCiudadPorCodigoTest;

@RunWith(Suite.class)
@Suite.SuiteClasses({webtest.CargarCiudadTest.class, webtest.InsertarCiudadTest.class,
    webtest.EditarCiudadTest.class, webtest.BUScarCiudadPorCodigoTest.class})
public class SeleniumTestSuite extends TestSuite{
    @BeforeClass
    public static void setUpClass() throws Exception {
    }
    @AfterClass
    public static void tearDownClass() throws Exception {
    }
    @Before
    public void setUp() throws Exception {
    }
    @After
    public void tearDown() throws Exception {
    }
    public SeleniumTestSuite(String name) {
        super(name);
    }
    public static void main(String[] args) {
        TestRunner.run((junit.framework.Test) suite());
    }
    public static TestSuite suite() {
        TestSuite suite = new SeleniumTestSuite("Test Selenium JSF");
        //Establecemos los casos de prueba a ejecutar
        suite.addTestSuite(CargarCiudadTest.class);
        suite.addTestSuite(InsertarCiudadTest.class);
        suite.addTestSuite(EditarCiudadTest.class);
        suite.addTestSuite(BuscarCiudadPorCodigoTest.class);
        return suite;
    }
}
```

Figura III. 12 Selenium: SeleniumTestSuite.java

- **Ejecutar las pruebas:** Una vez realizadas las clases para cada uno de los prototipos el último paso es ejecutar las pruebas.

Para esto se debe iniciar el servidor Slenium, como se muestra en la **Figura 13**

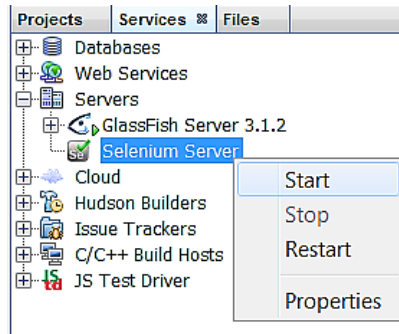


Figura III. 13 Inicialización del servidor Selenium

Una vez que este iniciado el servidor ejecutamos el SeleniumTestSuite.java para verificar el funcionamiento de cada uno de los prototipos creados.

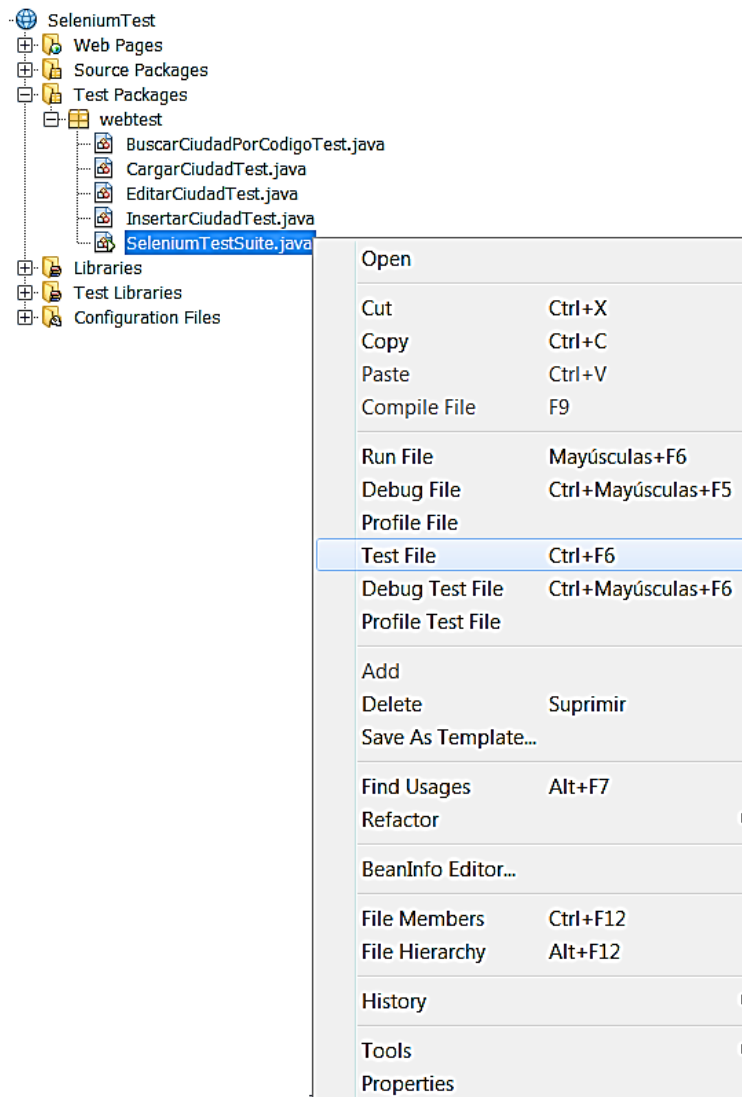


Figura III. 14 Ejecución de la clase SeleniumTestSuite

En la **Figura 15** se observan los resultados obtenidos de la prueba unitaria con la herramienta Selenium.

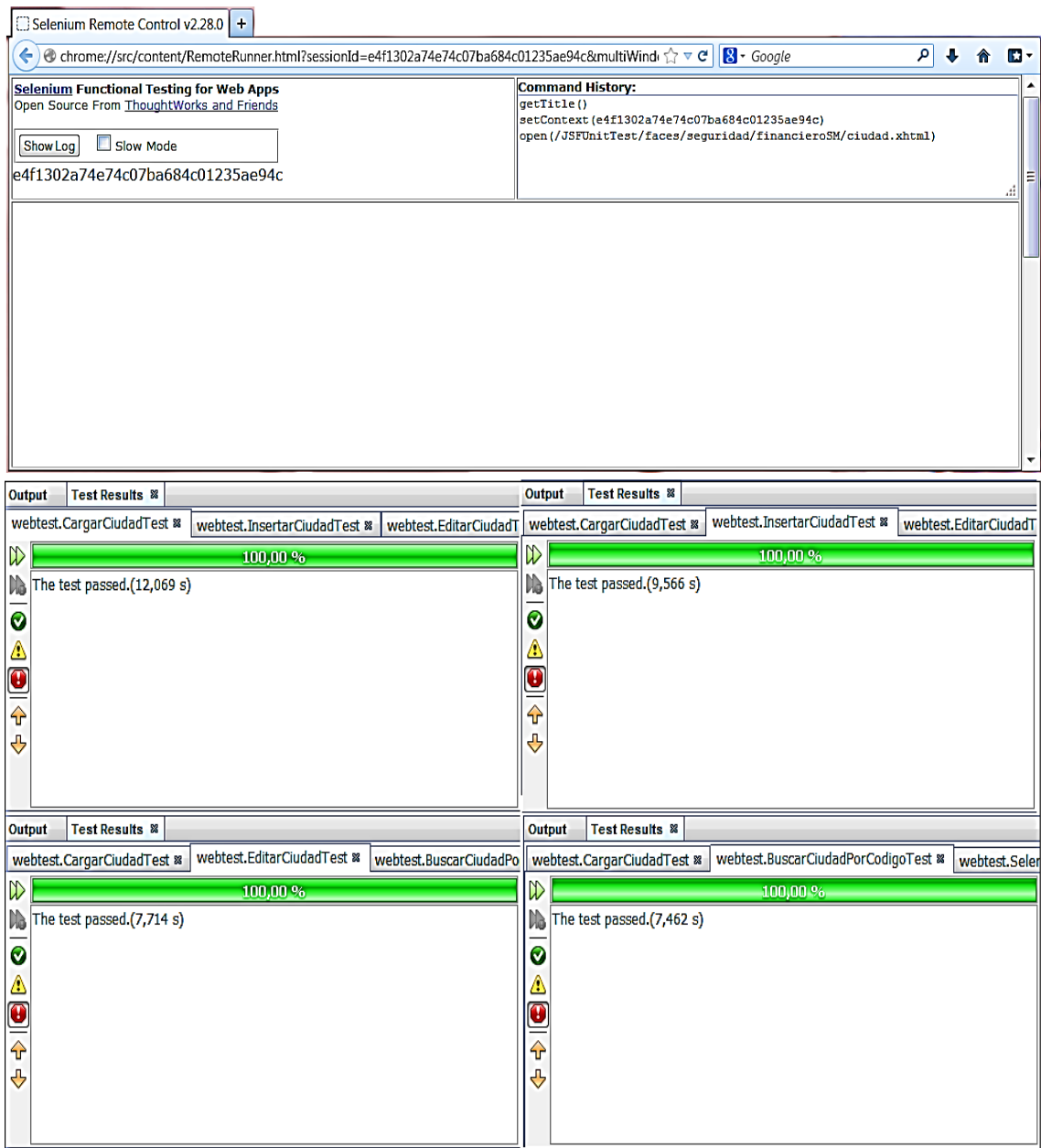


Figura III. 15 Selenium: Resultados

3.6.2. Cactus

Para la realización de pruebas unitarias con el framework cactus se emplea la arquitectura que se muestra en la **Figura 16**.

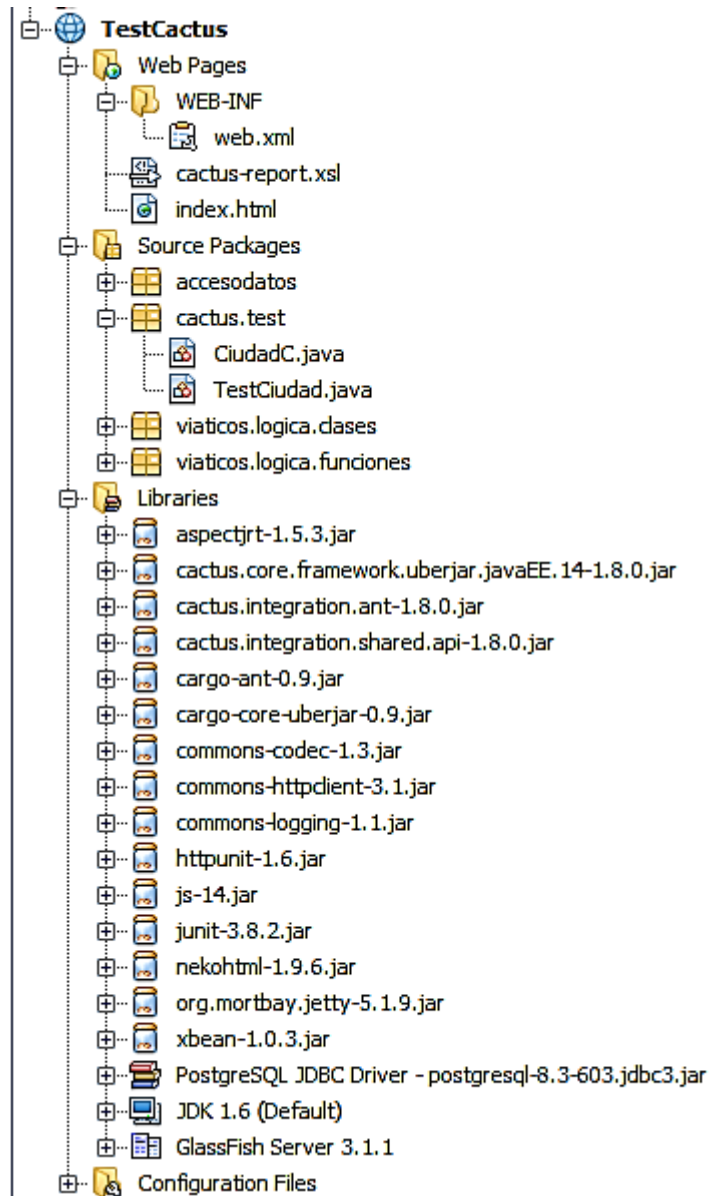


Figura III. 16 Arquitectura Proyecto TestCactus

3.6.2.1. Procedimiento

A continuación se describe paso a paso el procedimiento para el desarrollo de la prueba unitaria.

- *Crear el proyecto TestCactus.*

- **Agregamos las librerías** es necesario para la ejecución de la aplicación y la realización de pruebas unitarias, agregar las librerías que en la **Figura 17** de detallan.

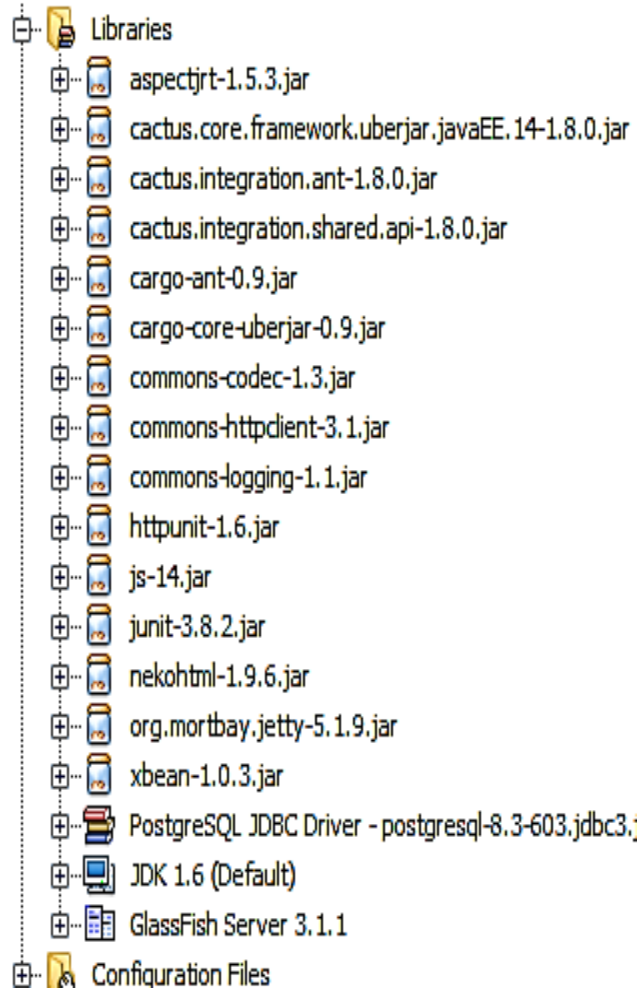


Figura III. 17 Cactus: Librerías

- **Modificar el archivo web.xml** este archivo permite heredar las clases necesarias para el desarrollo de las pruebas unitarias y el mapeo de los métodos que son motivos del test como se muestra en la **Figura 18**.


```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0"
  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-app_3_0.xsd">
  <servlet>
    <servlet-name>ServletTestRunner</servlet-name>
    <servlet-class>
      org.apache.cactus.server.runner.ServletTestRunner
    </servlet-class>
  </servlet>
  <servlet>
    <servlet-name>ServletRedirector</servlet-name>
    <servlet-class>
      org.apache.cactus.server.ServletTestRedirector
    </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>ServletTestRunner</servlet-name>
    <url-pattern>/ServletTestRunner</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>ServletRedirector</servlet-name>
    <url-pattern>/ServletRedirector</url-pattern>
  </servlet-mapping>
</web-app>
```

Figura III. 18 Cactus: Archivo web.xml

- **Agregar la clase *TestCiudad.java*** esta clase permite crear un conjunto de pruebas a la que se puede agregar las clases que son motivo de prueba en este caso se llama a la clase CiudadC que es el controlador en el que se encuentran los prototipos para la prueba, para ello se importan la librerías de ServletTestCase y ServletTestSuite pertenecientes a cactus más la librería Test del framework junit, esta clase es llamada mediante el ServletTestRunner configurado anteriormente en *web.xml*, el código se muestra en la **Figura 19**.

```
package cactus.test;

import junit.framework.Test;
import org.apache.cactus.ServletTestCase;
import org.apache.cactus.ServletTestSuite;

public class TestCiudad extends ServletTestCase {

    public static Test suite() {
        ServletTestSuite ourSuite = new ServletTestSuite("Pruebas Ciudad");
        ourSuite.addTestSuite(cactus.test.CiudadC.class);
        return ourSuite;
    }
}
```

Figura III. 19 Cactus: TestCiudad.java

- **Modificar la clase CiudadC.java** en esta clase se definen y desarrollan los métodos que serán objeto de testeo los mismos que ya se mencionó en la definición de prototipos **Tabla III. VI**, al ser llamados por la clase **TestCiudad** arrojaran los resultados de satisfactorio o error de las pruebas ejecutadas en la **Figura 20** se visualiza el código necesario.

```
package cactus.test;

import java.util.ArrayList;
import org.apache.cactus.ServletTestCase;
import viaticos.logica.clases.Ciudad;
import viaticos.logica.funciones.FCiudad;

public class CiudadC extends ServletTestCase {

    Ciudad ciudad = new Ciudad();
    Ciudad objCiudadSel = new Ciudad();
    private ArrayList<Ciudad> listaCiudades;
    private int codigo;

    @Override
    public void setUp() throws Exception {
        //atributos insertar
        ciudad.setNombre("Quito");
        ciudad.setObservacion("Capital Ecuador");
        //atributos actualizar
        objCiudadSel.setCodigo(3);
        objCiudadSel.setNombre("Riobamba");
        objCiudadSel.setObservacion("Capital Chimborazo");
        //atributos obtener ciudad dado codigo
        codigo = 2;
    }

    public void testInsertarCiudad_Ok() throws Exception {
        boolean words = FCiudad.insertar(ciudad);
        assertEquals(true, words);
    }

    public void testCargarCiudad_Ok() throws Exception {
        listaCiudades = FCiudad.ObtenerCiudades();
        assertNotNull(listaCiudades);
    }

    public void testEditarCiudad() throws Exception {
        assertEquals("Actualizacion Correcta",FCiudad.actualizar(objCiudadSel));
    }

    public void testObtenerCiudadDadoCodigo() throws Exception {
        Ciudad ciudad1 = FCiudad.ObtenerCiudadDadoCodigo(codigo);
        assertNotNull(ciudad1);
    }
}
```

Figura III. 20 Cactus: CiudadC.java

A continuación se describen los métodos definidos en la clase *CiudadC*.

- **Método setUp():** contiene la inicialización de variables y objetos usados durante la ejecución de la clase de pruebas unitarias.

- **Método *testInsertarCiudad()***: permite probar si una los datos de una ciudad se ingresaron correctamente, devolviendo como **true** si el ingreso fue correcto.
- **Método *testCargarCiudad()***: permite probar si los datos de una ciudad se cargaron correctamente, devolviendo una lista de ciudades, en caso de devolver **null** la prueba será errónea.
- **Método *testEditarCiudad()***: permite probar si la modificación de los datos se realizó correctamente devolviendo la cadena “Actualización Correcta” si la modificación se realizó sin inconvenientes.
- **Método *testObtenerCiudadDadoCodigo:*** permite verificar que la búsqueda de una ciudad dado su código sea correcta.
- **Agregar *cactus-report.xml***, este archivo permite dar un formato los resultados de las pruebas unitarias para poder apreciarlos de mejor manera.
- **Ejecutar las pruebas** para esto se ejecuta el proyecto y se visualiza el link “Cactus Test” el mismo que ayudara a dirigirse a la siguiente dirección <http://localhost:8080/TestCactus/ServletTestRunner?suite=cactus.test.TestCiudad&xsl=cactus-report.xml> permitiendo la ejecución del prototipo de pruebas obteniendo los resultados que se muestra en la **Figura 21**

Unit Test Results

Summary

Tests	Failures	Errors	Success rate	Time
4	0	0	100.00%	2.394

Note: *failures* are anticipated and checked for with assertions while *errors* are unanticipated.

TestCase cactus.test.TestCiudad

Name	Status	Type	Time(s)
testInsertarCiudad_Ok	Success		1.820
testCargarCiudad_Ok	Success		0.177
testEditarCiudad	Success		0.176
testObtenerCiudadDadoCodigo	Success		0.075

Figura III. 21 Cactus: Resultados

3.6.3. JSFUnit

Ahora se va a demostrar el uso de este framework utilizando los prototipos creados y explicados anteriormente.

3.6.3.1. Procedimiento

4. Como primer paso se crea el proyecto *JSFUnitTest* con la misma estructura que se definió al crear los prototipos, a más de esto se crea un paquete *TestCiudad* el cual contendrá los test, para entender de mejor manera se muestra como queda la arquitectura del proyecto.

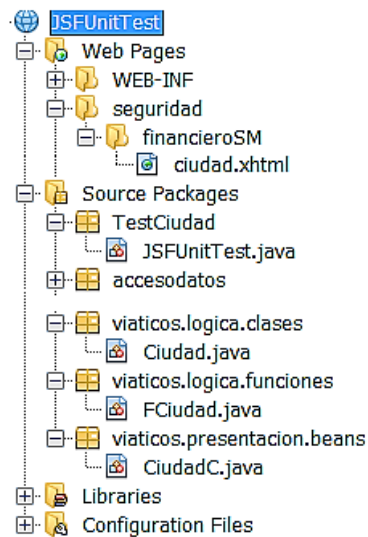


Figura III. 22 Arquitectura Proyecto JSFUnitTest

- En segundo lugar se necesita los jar de JSFUnit. Como JSFUnit utiliza JUnit, Cactus, HtmlUnit y HttpUnit para gran parte de su infraestructura debemos agregarlos, así como también otras librerías que se listan a continuación:
 - Jboss-jsfunit-core-1.3.0.Final.jar
 - aspectjrt-1.2.1.jar
 - cactus-13-1.7.1.jar

- junit-3.8.1.jar
- nekohtml-1.9.14.jar
- htmlunit-2.8.jar
- htmlunit-core-js-2.8.jar
- cssparser-0.9.5.jar
- httpcore-4.0.1.jar
- httpclient-4.0.1.jar
- httpmime-4.0.1.jar
- apache-mime4j-0.6.jar
- commons-io-1.4.jar.jar
- sac-1.3.jar
- commons-lang-2.4.jar
- commons-codec-1.4.jar
- commons-logging-1.1.1.jar
- commons-collections-3.2.1.jar
- commons-httpclient-2.0.2.jar
- cactus-ant-13-1.7.1.jar
- cargo-0.5.jar
- ant-1.5.4.jar
- xercesImpl-2.8.1.jar
- xalan-2.7.0.jar

Una vez que se tiene todas estas librerías se procede a agregarlas al proyecto *JSFUnitTest* que se creó anteriormente, este queda como se muestra en la **Figura 23**.

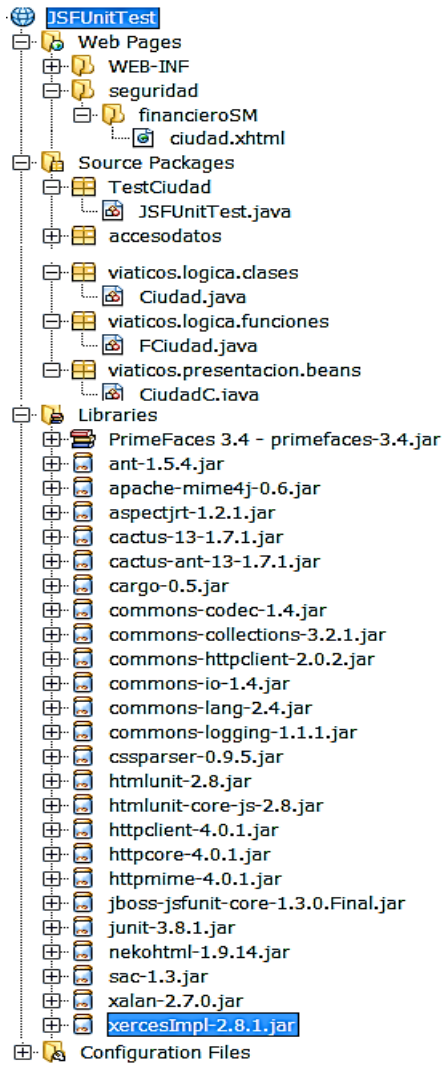


Figura III. 23 JSFUnit: Librerías

- El siguiente paso y muy importante es establecer un **ID** a cada uno de los componentes como a se muestra en la **Figura 24**.

```
<p:dataTable id="tblCiudad" value="#{ciudadC.listaCiudades}" var="ciudad" paginator="true" rows="10" rowKey="#{ciudad.codigo}" selectionMode="single" selection="#{ciudadC.objCiudadSel}">
```

Figura III. 24 JSFUnit: ID en los componentes

Para todo componente JSF sobre una página, se tiene la opción de proveer un ID al componente. Si no se suministra el ID, JSF creará uno para cada componente. Sin

embargo si se permite a JSF crear el ID, se tendrá dificultades para referenciar al componentes en la pruebas.

- El siguiente paso es la configuración del *web.xml* su código se muestra en la **Figura 25**.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" 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-app_3_0.xsd">

<filter>
<filter-name>JSFUnitFilter</filter-name>
<filter-class>org.jboss.jsfunit.framework.JSFUnitFilter</filter-class>
</filter>
<filter-mapping>
<filter-name>JSFUnitFilter</filter-name>
<servlet-name>ServletTestRunner</servlet-name>
</filter-mapping>
<filter-mapping>
<filter-name>JSFUnitFilter</filter-name>
<servlet-name>ServletRedirector</servlet-name>
</filter-mapping>
<servlet>
<servlet-name>ServletRedirector</servlet-name>
<servlet-class>org.jboss.jsfunit.framework.JSFUnitServletRedirector</servlet-class>
</servlet>
<servlet>
<servlet-name>ServletTestRunner</servlet-name>
<servlet-class>org.apache.cactus.server.runner.ServletTestRunner</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>ServletRedirector</servlet-name>
<url-pattern>/ServletRedirector</url-pattern>
</servlet-mapping>
<servlet-mapping>
<servlet-name>ServletTestRunner</servlet-name>
<url-pattern>/ServletTestRunner</url-pattern>
</servlet-mapping>
</web-app>
```

Figura III. 25 JSFUnit: Archivo web.xml

Para poder visualizar los resultados de las pruebas unitarias se debe agregar el archivo *cactus-report.xml*, el cual permite dar formato al informe con los resultados una vez que se ejecuten los test, este archivo se lo debe colocar en la raíz del proyecto como se muestra en la **Figura 26**.

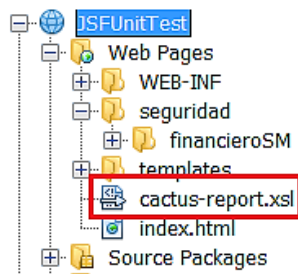


Figura III. 26 JSFUnit: Agregar cactus-report.xml

- Finalmente viene la creación de las pruebas, las cuales son creadas dentro del paquete *TestCiudad*, que como ya se vio tiene como nombre *JSFUnitTest.java*, que no es más que una prueba JUnit creada utilizando la API JSFUnit.

La estructura de esta clase la definimos en la **Figura 27**.

```
package TestCiudad;

import java.io.IOException;
import javax.faces.component.UIComponent;
import static junit.framework.Assert.assertEquals;
import static junit.framework.Assert.assertTrue;
import junit.framework.Test;
import junit.framework.TestSuite;
import org.jboss.jsfunit.jsfession.JSFClientSession;
import org.jboss.jsfunit.jsfession.JSFServerSession;
import org.jboss.jsfunit.jsfession.JSFSession;

public class JSFUnitTest extends org.apache.cactus.ServletTestCase {

    public static Test suite() {
        return new TestSuite(JSFUnitTest.class);
    }

    public void testInsertarCiudad() throws IOException, Exception {...}

    public void testCargarCiudades() throws IOException, Exception {...}

    public void testEditarCiudad() throws IOException, Exception {...}

    public void testBuscarCiudadPorCodigo() throws IOException, Exception {...}

    public JSFUnitTest(String testName) {
        super(testName);
    }

    @Override
    protected void setUp() throws Exception {
        super.setUp();
    }

    @Override
    protected void tearDown() throws Exception {
        super.tearDown();
    }
}
```

Figura III. 27 JSFUnit: JSFUnitTest.java

Como se observa, en esta clase se definió los métodos test para realizar las pruebas sobre los prototipos definidos, a continuación se muestra el contenido de cada uno de estos, con su respectiva explicación.

- *Método testInsertarCiudad()*

Este método permite probar si una ciudad se ingresa correctamente su código se muestra en la **Figura 28**.

```
public void testInsertarCiudad() throws IOException, Exception {  
  
    // Enviar una solicitud HTTP a la página inicial  
    JSFSession jsfSession = new JSFSession("/faces/seguridad/financieroSM/ciudad.xhtml");  
  
    // Un JSFClientSession emula el browser y permite probar el HTML  
    JSFClientSession client = jsfSession.getJSFClientSession();  
  
    // Un JSFServerSession nos da acceso al estado JSF  
    JSFServerSession server = jsfSession.getJSFServerSession();  
  
    // Prueba la navegacion para la pagina inicial  
    assertEquals("/seguridad/financieroSM/ciudad.xhtml", server.getCurrentViewID());  
  
    // Emula un click sobre el boton con id = btnNuevaCiudad  
    client.click("btnNuevaCiudad");  
  
    // Permite asignar valores sobre los componentes con id=nombreCiudad y observacionciudad  
    // respectivamente, los mismos que estan dentro del formulario con id = frmNuevaCiudad  
    client.setValue("frmNuevaCiudad:nombreCiudad", "Ciudad");  
    client.setValue("frmNuevaCiudad:observacionCiudad", "Observacion");  
    client.click("guardarCiudad");  
  
    // Confirma que el componente del sistema con id mensaje1 está en el árbol de  
    // componentes y se ejecuto correctamente  
    UIComponent prompt = server.findComponent("mensaje1");  
    assertTrue(prompt.isRendered());  
  
    // Prueba del managed bean(controlador)  
    // verifica que la ciudad se inserto correctamente  
    assertEquals("Datos Insertados", server.getManagedBeanValue("#{ciudadC.mensajeAlerta}"));  
}
```

permite instanciar el archivo ciudad.xhtml

Figura III. 28 JSFUnit: Método testInsertarCiudad

- *Método testCargarCiudades()*

Este método permite conocer si se están cargando las ciudades que se encuentran ingresadas en la base de datos su código se muestra en la **Figura 29**.

```
public void testCargarCiudades() throws IOException, Exception {  
  
    // Enviar una solicitud HTTP a la página inicial  
    JSFSession jsfSession = new JSFSession("/faces/seguridad/financieroSM/ciudad.xhtml");  
  
    // Un JSFClientSession emula el browser y permite probar el HTML  
    JSFClientSession client = jsfSession.getJSFClientSession();  
  
    // Un JSFServerSession nos da acceso al estado JSF  
    JSFServerSession server = jsfSession.getJSFServerSession();  
  
    // Prueba la navegacion para la pagina inicial  
    assertEquals("/seguridad/financieroSM/ciudad.xhtml", server.getCurrentViewID());  
  
    // Confirma que el componente del sistema con id tblCiudad está en el árbol de  
    // componentes y se ejecuto correctamente  
    UIComponent prompt = server.findComponent("tblCiudad");  
    assertTrue(prompt.isRendered());  
  
    // Prueba del managed bean(controlador)  
    // verifica que las ciudades se cargaron correctamente  
    assertEquals("Ciudades Cargadas", server.getManagedBeanValue("#{ciudadC.mensajeAlerta}"));  
}
```

permite instanciar el archivo ciudad.xhtml

Figura III. 29 JSFUnit: Método testCargarCiudades

- *Método testEditarCiudad()*

Permite verificar si la edición de una ciudad seleccionada se realiza correctamente su código se muestra en la **Figura 30**.

```
public void testEditarCiudad() throws IOException, Exception {  
  
    // Enviar una solicitud HTTP a la página inicial  
    JSFSession jsfSession = new JSFSession("/faces/seguridad/financieroSM/ciudad.xhtml");  
  
    // Un JSFClientSession emula el browser y permite probar el HTML  
    JSFClientSession client = jsfSession.getJSFClientSession();  
  
    // Un JSFServerSession nos da acceso al estado JSF  
    JSFServerSession server = jsfSession.getJSFServerSession();  
  
    // Prueba la navegacion para la pagina inicial  
    assertEquals("/seguridad/financieroSM/ciudad.xhtml", server.getCurrentViewID());  
  
    // Emula un click sobre el boton con id = btnEditarCiudad  
    client.click("btnEditarCiudad");  
  
    // valores de nueva ciudad  
    // permite obtener los datos de la ciudad seleccionada y asignarlos a las variables  
    // nombreCiudadSeleccionada y observacionCiudadSeleccionada respectivamente  
    String nombreCiudadSeleccionada = (String) server.getManagedBeanValue("#{ciudadC.objCiudadSel.nombre}");  
    String observacionCiudadSeleccionada = (String) server.getManagedBeanValue("#{ciudadC.objCiudadSel.observacion}");  
  
    // Permite asignar valores sobre los componentes con id=nombreCiudadModificar y observacionCiudadModificar  
    // respectivamente, los mismos que estan dentro del formulario con id = frmNuevaCiudad  
    client.setValue("frmActualizarCiudad:nombreCiudadModificar", nombreCiudadSeleccionada);  
    client.setValue("frmActualizarCiudad:observacionCiudadModificar", observacionCiudadSeleccionada);  
  
    // Emula un click sobre el boton con id = modificarCiudad una vez que se modifican  
    // los datos de una ciudad seleccionada  
    client.click("modificarCiudad");  
  
    // Confirma que el componente del sistema con id mensaje1 está en el árbol de  
    // componentes y se ejecuto correctamente  
    UIComponent prompt = server.findComponent("mensaje1");  
    assertTrue(prompt.isRendered());  
  
    // Prueba del managed bean(controlador)  
    // verifica que la ciudad seleccionada se modico correctamente  
    assertEquals("Datos Modificados", server.getManagedBeanValue("#{ciudadC.mensajeAlerta}"));  
}
```

permite instanciar el archivo ciudad.xhtml

Figura III. 30 JSFUnit: Método testEditarCiudad

- **Método *testBuscarCiudadPorCodigo()***

Permite evaluar si la búsqueda de una ciudad por el código funciona correctamente su código se muestra en la **Figura 31**.

```
public void testBuscarCiudadPorCodigo() throws IOException, Exception {  
  
    // Enviar una solicitud HTTP a la página inicial  
    JSFSession jsfSession = new JSFSession("/faces/seguridad/financieroSM/busquedaCiudad.xhtml");  
  
    // Un JSFClientSession emula el browser y permite probar el HTML  
    JSFClientSession client = jsfSession.getJSFClientSession();  
  
    // Un JSFServerSession nos da acceso al estado JSF  
    JSFServerSession server = jsfSession.getJSFServerSession();  
  
    // Prueba la navegacion para la pagina inicial  
    assertEquals("/seguridad/financieroSM/busquedaCiudad.xhtml", server.getCurrentViewID());  
  
    // Permite asignar el valor del codigo de la ciudad a buscar  
    // el mismo que se asignan al componente con id=codigoCiudad  
    // que esta dentro del formulario con id = frmCiudad  
    client.setValue("frmCiudad:codigoCiudad", "5");  
  
    // Emula un click sobre el boton con id = buscarCiudad una vez que se  
    // ingreso el codigo de la ciudad a buscar  
    client.click("buscarCiudad");  
  
    // Assert that the prompt component is in the component tree and rendered  
    UIComponent prompt = server.findComponent("mensaje1");  
    assertTrue(prompt.isRendered());  
  
    // Test a managed bean  
    assertEquals("Ciudad encontrada", server.getManagedBeanValue("#{ciudadC.mensajeAlerta}"));  
}
```

Permite instanciar el archivo
busquedaCiudad.xhtml

Figura III. 31 JSFUnit: Método *testBuscarCiudadPorCodigo*

Ahora lo que queda por hacer es probar si todo lo descrito funciona correctamente para esto vamos a escribir en la barra de dirección del browser lo siguiente: <http://localhost:8080/JSFUnitTest/ServletTestRunner?suite=TestCiudad.JSFUnitTest&xsl=cactus-report.xsl>, en donde *localhost* es el nombre del servidor, *JSFUnitTest* es el nombre del proyecto, *ServletTestRunner* nos permite ejecutar los test, *TestCiudad.JSFUnitTest* es el directorio donde se encuentra la clase test, y finalmente *cactus-report.xls* que es quien dará formato al resultado de las pruebas los resultados obtenidos se muestran en la **Figura 32**.

Unit Test Results

Summary

Tests	Failures	Errors	Success rate	Time
4	0	0	100.00%	19.889

Note: *failures* are anticipated and checked for with assertions while *errors* are unanticipated.

TestCase TestCiudad.JSJUnitTest

Name	Status	Type	Time(s)
testInsertarCiudad	Success		6.656
testCargarCiudades	Success		4.699
testEditarCiudad	Success		5.497
testBuscarCiudadPorCodigo	Success		3.036

Figura III. 32 JSJUnit: Resultados

3.8. Valoración de herramientas según indicadores definidos

Una vez realizadas las pruebas unitarias con las herramientas seleccionadas y los prototipos creados para medir el rendimiento de estas herramientas se obtuvo los resultados expuestos a continuación.

- **Indicador 1: *Tiempo de ejecución (en segundos)***

Tabla III. VI Resultados parámetro Tiempo de ejecución

	Selenium	Cactus	JsfUnit
Prototipo 1	12,069	1,820	6,656
Prototipo 2	9,566	0,177	4,699
Prototipo 3	7,714	0,176	5,497
Prototipo 4	7,462	0,075	3,036
Promedio	9.203	0,562	4.972

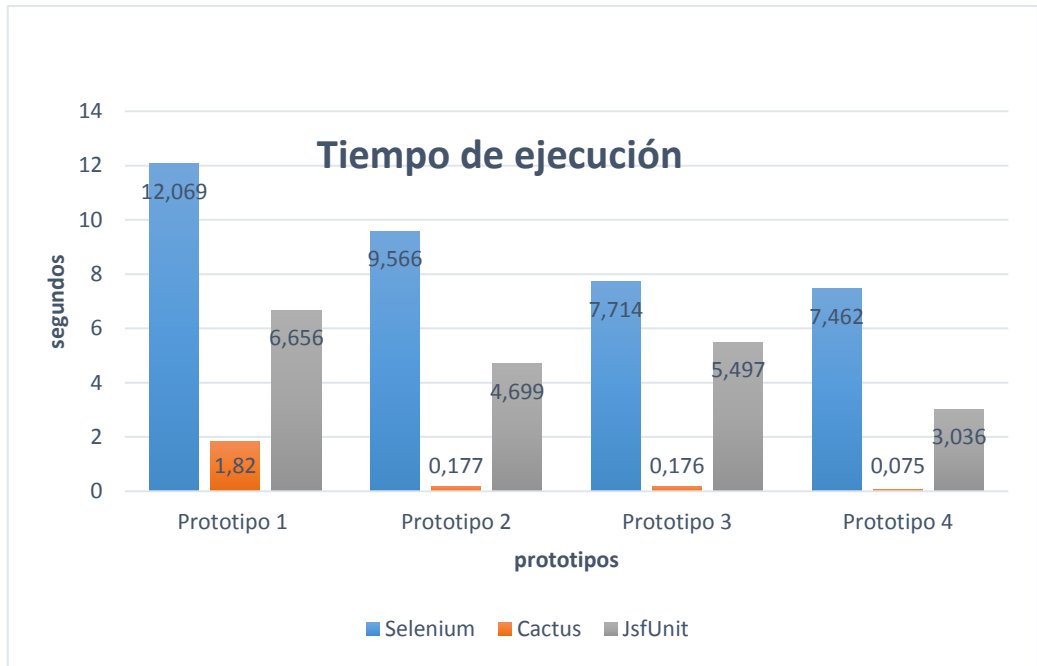


Figura III. 33 Resultado Tiempo de Ejecución

En este indicador el que se destaca es la herramienta cactus ya que es la que menos tiempo tarda en la realización de una prueba unitaria, mientras que JsfUnit y selenium toman mucho más tiempo para la realización de las pruebas.

- **Indicador 2: Líneas de código escritas para las pruebas**

Tabla III. VII Resultados parámetro líneas de código

	Selenium	Cactus	JsfUnit
Prototipo 1	16	6	13
Prototipo 2	20	4	9
Prototipo 3	19	7	16
Prototipo 4	20	5	11
Promedio	18.75	5.5	12.25

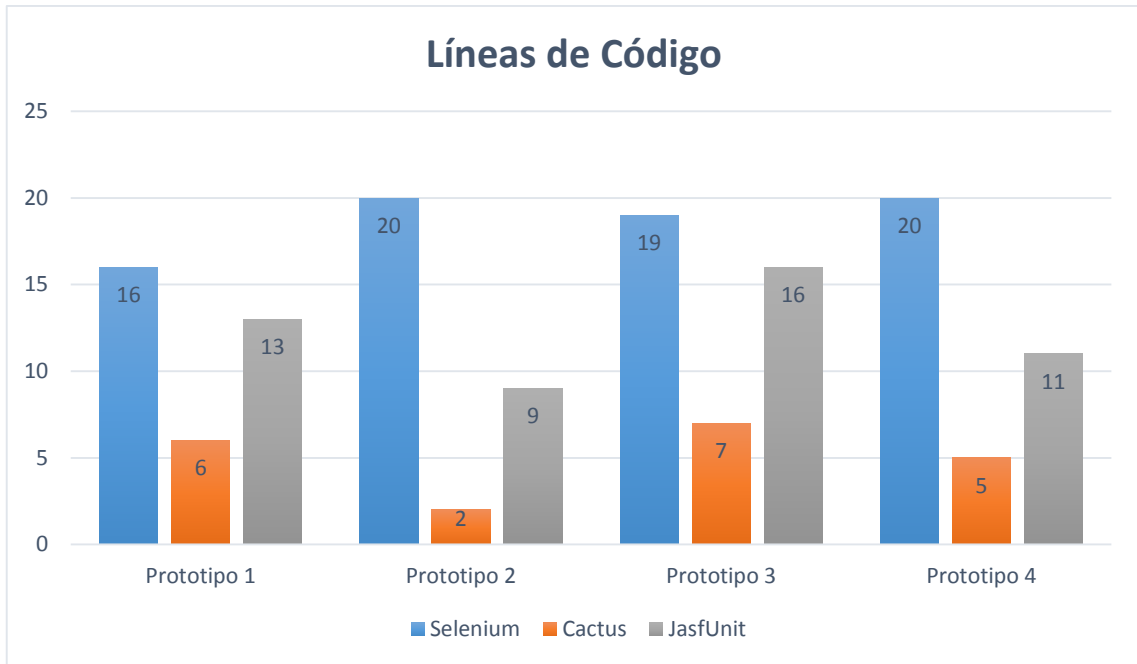


Figura III. 34 Resultado de líneas de código escritas

En la **Figura 34** se puede observar que nuevamente la herramienta cactus es la que menos número de líneas escritas posee, rescatando como puntos positivos al momento de realizar una prueba unitaria.

- **Indicador 3: Flexibilidad para realizar cambios (rango del 1 - 3)**

Tabla III. VIII Resultados parámetro flexibilidad a cambios

	Selenium	Cactus	JsfUnit
Prototipo 1	3	3	3
Prototipo 2	3	3	3
Prototipo 3	3	3	3
Prototipo 4	3	3	3
Promedio	3	3	3

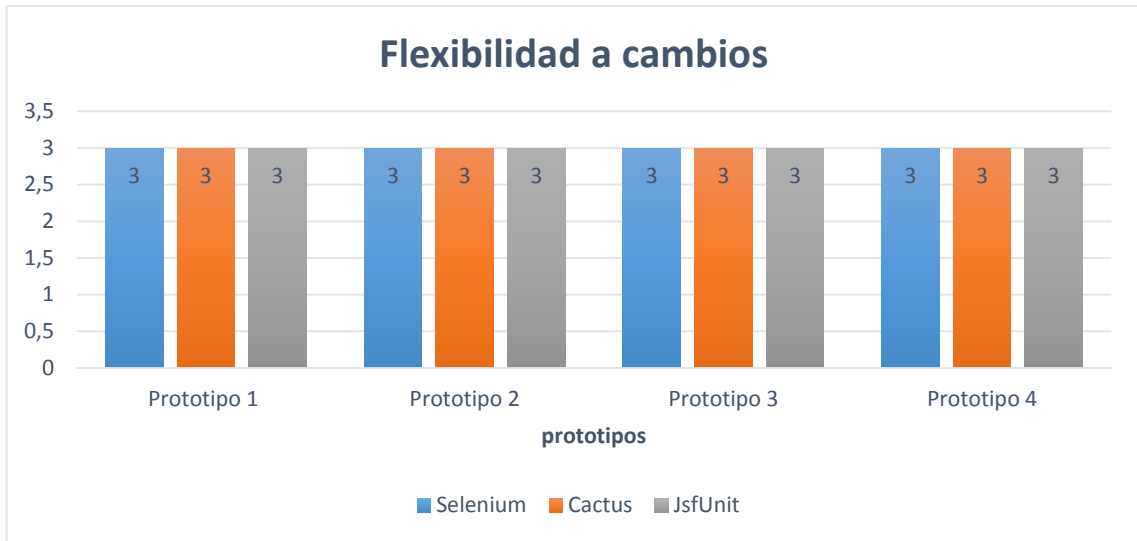


Figura III. 35 Resultado flexibilidad de cambios

En la **Figura 35** se observa un equilibrio ya que las tres herramientas tienen flexibilidad al momento de realizar cambios en la estructura de su código.

- **Indicador 4: Navegabilidad por la arquitectura de la aplicación (rango del 1 - 3)**

La aplicación tiene estructura MVC por lo tanto la navegabilidad tendrá la valoración equitativa por cada capa evaluada, es decir Modelo = 1, Vista =1, Controlador = 1,

: Vista=1; Modelo Vista=2; Modelo Vista Controlador=3.

Tabla III. IX Resultados parámetro navegabilidad por la arquitectura

	Selenium	Cactus	JsfUnit
Prototipo 1	1	2	3
Prototipo 2	1	2	3
Prototipo 3	1	2	3
Prototipo 4	1	2	3
Promedio	1	2	3

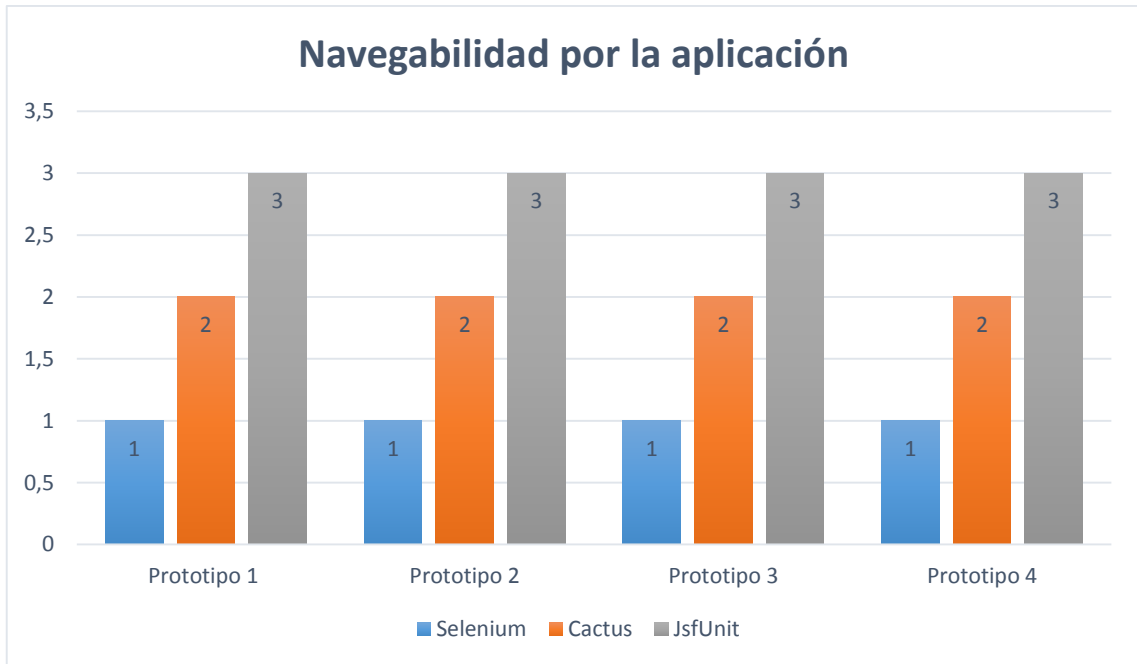


Figura III. 36 Resultado navegabilidad por la aplicación

En cuanto a navegabilidad se trata, se analiza de manera diferente ya que la herramienta que mayor navegabilidad tiene es la que mejores ventajas nos proporciona al realizar una prueba unitaria, en este caso se observa claramente que la herramienta JSFUnit cumple totalmente con lo requerido, es decir permite evaluar toda la arquitectura de la aplicación (MVC).

3.9. Comprobación de hipótesis

Para realizar el análisis comparativo de las herramientas utilizadas para la realización de pruebas unitarias, en la **Tabla III. XI** se establecen los respectivos pesos a los indicadores tomando en cuenta que el indicador uno posee un peso del 20% sobre el rendimiento total, puesto que el tiempo estará en función del proceso a probar, el indicador dos posee un peso del 10% sobre el rendimiento total, pues lo que se busca el resultado de los testing mas no la complejidad en su creación, el tercer indicador posee un peso del 20% pues es importante que se puedan realizar cambios en los casos de pruebas, finalmente el cuarto

indicador es el que mayor peso tiene, pues lo que se busca es realizar una prueba unitaria sobre toda la arquitectura de la aplicación JSF (MVC), por este motivo el peso de este indicado es de 50%, dando la suma de estos pesos un total del 100% lo que permitirá determinar y seleccionar la herramienta de mejor rendimiento.

Tabla III. X Pesos de los indicadores

Indicadores	Pesos
I1: Tiempo de ejecución de pruebas	20%
I2: Líneas de código escritas para las pruebas	10%
I3: Flexibilidad para realizar cambios	20%
I4: Navegabilidad por la arquitectura de la aplicación	50%
Rendimiento	100%

Una vez ejecutadas las pruebas unitarias con las herramientas seleccionadas se obtuvo los resultados expuestos en la **Tabla III. XII**

Tabla III. XI Resumen de resultados

Indicador	Selenium	Cactus	JsfUnit
I1	9,203	0,562	4,972
I2	18,75	5,5	12,25
I3	6	8	9
I4	3,33	6,66	10

Para evaluar las herramientas de benchmarking seleccionadas los resultados obtenidos en la **Tabla III. XII**, se los calificará en una escala del 1 al 3 para estandarizar los valores de resultados, ya que los resultados obtenidos están en diferentes unidades y así facilitar la comparación, en la **Tabla III. XIII** se muestran las escalas que tendrán los resultados.

Tabla III. XII Escala de Equivalencias

Cualitativa	Regular	Bueno	Excelente
Cuantitativa	1	2	3

En la **Tabla III. XV** se muestra la estandarización de los resultados de cada uno de los indicadores.

Tabla III. XIII Valoración de indicadores

Indicador	Selenium	Cactus	JsfUnit
I1	1	3	2
I2	1	3	2
I3	3	3	3
I4	1	2	3
TOTAL	6	11	10

Ahora se procederá a multiplicar la valoración por los pesos de los indicadores respectivamente, en la **Tabla III. XIV** se muestra la valoración de manera porcentual.

Tabla III. XIV Resultado del análisis comparativo

Indicador	Selenium	Cactus	JsfUnit
I1	6.67%	20%	13.33%
I2	3.33%	10%	6.67%
I3	20%	20%	20%
I4	16.67%	33.33%	50%
TOTAL	46.67%	83.33%	90%

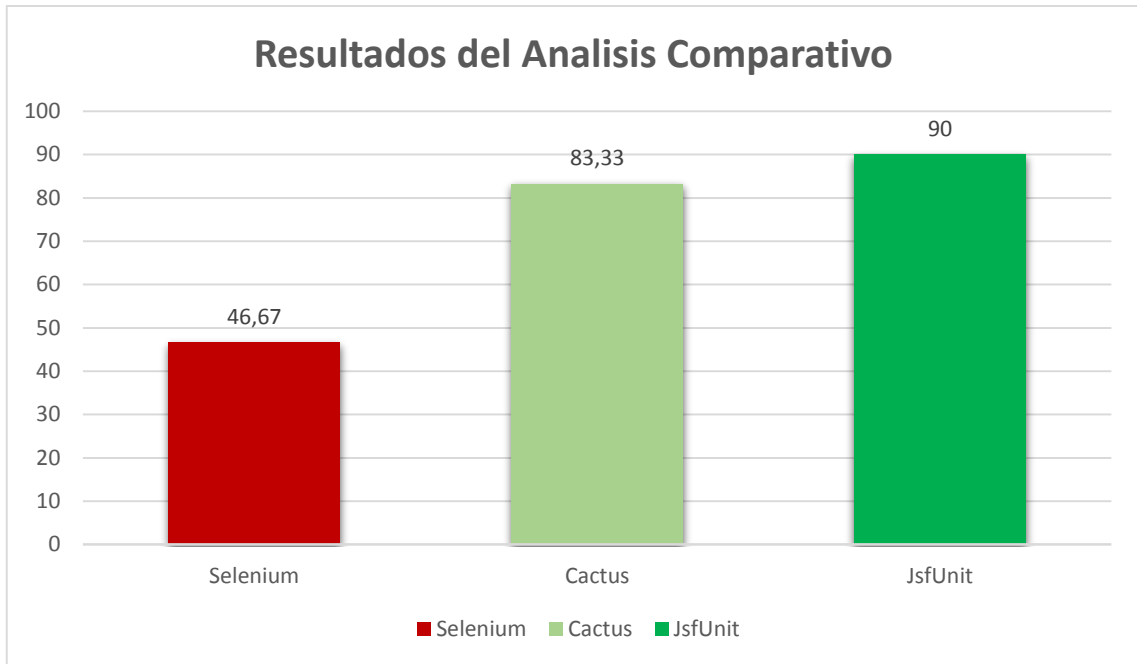


Figura III. 37 Resultado Análisis Comparativo

Una vez realizado el estudio comparativo y observando los resultados obtenidos, se puede indicar que la herramienta:

Selenium: obtiene un rendimiento del 46,67% es la que mayor tiempo tarda en la ejecución de una prueba unitaria, así también tiene el mayor número de líneas de código y realiza los test únicamente en la capa de las vistas aportando todo esto negativamente al momento de realizar una prueba unitaria.

Cactus: obtiene un rendimiento del 83,33% gracias a su ejecución temprana de una prueba y su reducido número de líneas de código escritas lo que hace una herramienta efectiva, sin embargo se ve en desventaja ya que las pruebas unitarias se ejecutan solo sobre dos capas, es decir realiza los test en las capas modelo y controlador.

JSFUnit: obtiene un rendimiento del 90% realiza un test completo, pues su tiempo de ejecución de las pruebas y líneas de código escritas hacen referencia a que lo realiza en toda la arquitectura de la aplicación.

Por lo tanto se demuestra la hipótesis propuesta indicando que la comparación de herramientas de benchmarking para pruebas unitarias web de aplicaciones JSf permitió escoger la herramienta JsfUnit que ofrece mejor rendimiento para realizar pruebas unitarias en el sistema de Gestión de Viáticos.

CAPITULO IV

DESARROLLO DEL SISTEMA DE GESTIÓN DE VIÁTICOS DE LA ESPOCH

4.1. Gestión del Proyecto

Es necesario aplicar una metodología para el correcto desarrollo de un software, la misma que debe adaptarse a las necesidades del sistema a implementarse.

En este capítulo se describe el desarrollo de la metodología XP (*Programación Extrema*), la misma que se aplicó de acuerdo a los requerimientos necesarios para la implementación del sistema de Gestión de Viáticos de la ESPOCH, esta metodología se divide en cuatro fases, cada una de ellas con sus respectivas actividades que se detallan en el transcurso de este capítulo.

La metodología XP comprende cuatro fases que se detallan a continuación:

- Fase I: Planificación
- Fase II: Diseño
- Fase III: Codificación

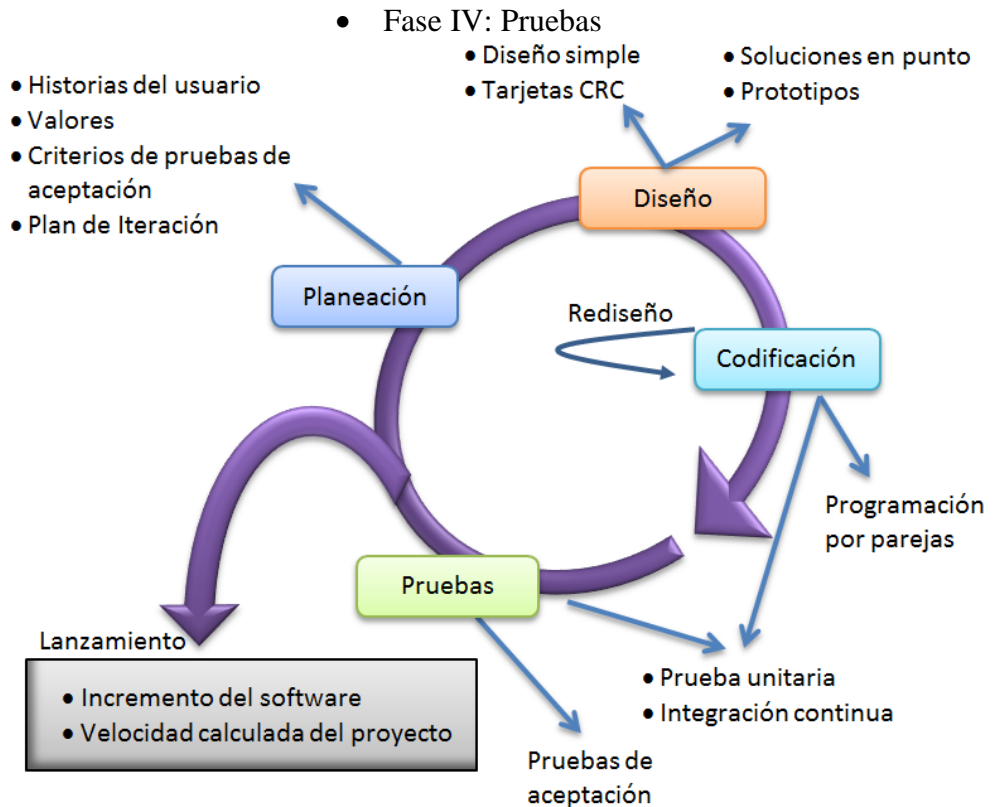


Figura IV. 38 Fases de la Metodología XP

4.2. Desarrollo de la Metodología XP

4.2.1. Fase I: Planificación

En esta fase se desarrollará todas las actividades que permitan conocer, comprender y entender el proceso de Gestión de Viáticos de la ESPOCH, así como también los requerimientos e historias de usuario del mismo, la planificación inicial e iteraciones necesarias para automatizar este proceso y cumplir con los objetivos planteados.

4.2.1.1. Descripción del Sistema

El sistema de Gestión de Viáticos permite la creación de solicitudes de movilización, ya sea con o sin vehículo de la institución, es decir solicitud interna y/o solicitud externa respectivamente, también permite mantener en forma actualizada, un registro de los comisionados, comisiones y vehículos que forman parte de una solicitud, para finalmente generar automáticamente el salvoconducto de una solicitud interna siempre y cuando esta haya sido aprobada.

El sistema se desarrolló en el lenguaje de programación Java con la ayuda del framework JSF, el mismo que se publicó en el servidor de la ESPOCH, permitiendo su acceso desde cualquier navegador web (Mozilla Firefox, Google Chrome, Internet Explorer, Opera, etc.), de la misma manera para el almacenamiento de la información se utilizó el gestor de base de datos PostgreSQL, por su rapidez y bajo consumo de recursos, además tanto Java como PostgreSQL no incluyen costos adicionales de licencia por ser software libre.

4.2.1.2. Definición del flujo del proceso de Gestión de Viáticos.

Para conocer más a fondo el proceso que se desea automatizar es necesario conocer cómo es que se realiza dicho proceso, para esto es necesario definirlo con nuestro cliente, de esta manera evitamos dejar cabos sueltos en cuanto a requerimientos se refiere, en este caso se lo realizó conjuntamente con el Vicerrectorado de Investigación y Desarrollo pues son ellos los expertos en este proceso.

A continuación se presenta el flujo del proceso desde cuando se crea una solicitud de movilización hasta cuando es aprobada por el Vicerrector de Investigación y Desarrollo y generado el respectivo salvoconducto de ser el caso.

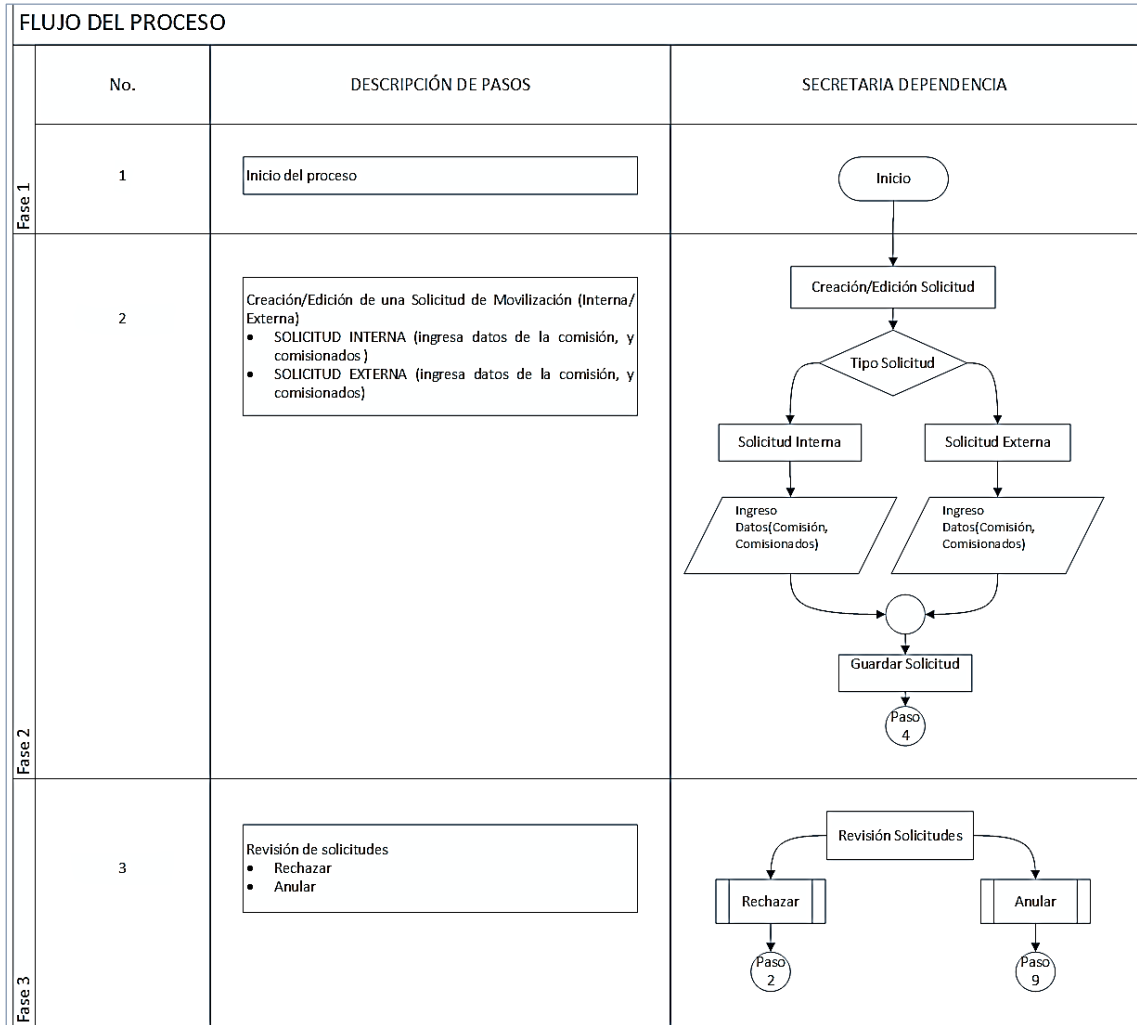


Figura IV. 39 Flujo de proceso (Secretaria de Dependencia)

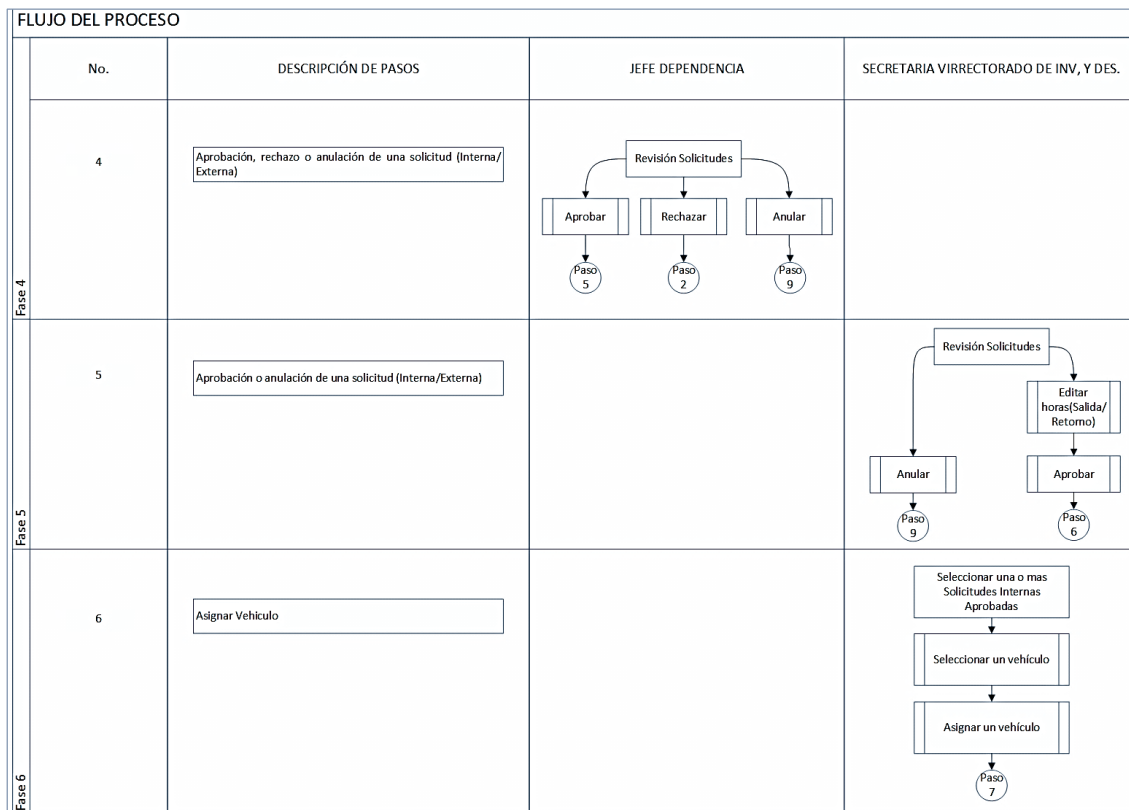


Figura IV. 40 Flujo de proceso (Jefe de Dependencia/Secretaria del Vicerrectorado de Investigación y Desarrollo)

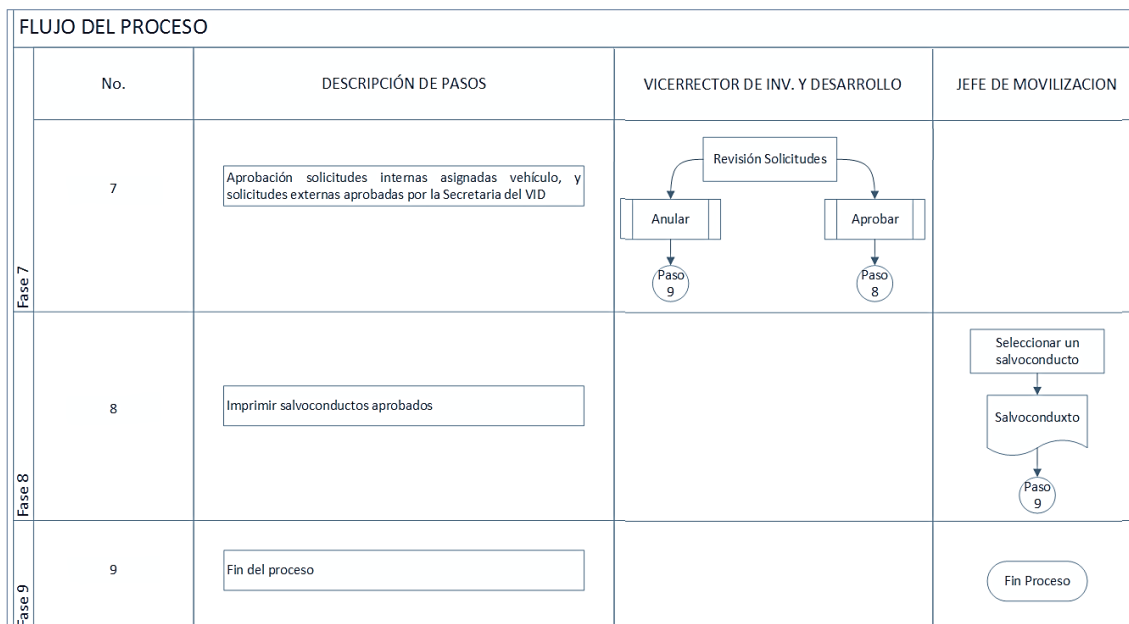


Figura IV. 41 Flujo de proceso (Vicerrector de Investigación y Desarrollo/Jefe de Movilización)

4.2.1.3. Especificación de Requerimientos

Comprende la descripción completa del comportamiento del sistema de Gestión de Viáticos, como la perspectiva del producto, requerimientos no funcionales, incluye un conjunto de historias de usuarios que describen cada uno de los requerimientos funcionales, también un plan de iteraciones a seguir para el diseño e implementación del sistema.

Perspectiva del producto

El sistema de Gestión de Viáticos es parte del Sistema de Movilización que interaccionará con el Sistema de Recursos Humanos mediante el uso de servicios web, pues este último nos proveerá información relevante para cumplir con el proceso de creación de solicitudes.

Requerimientos funcionales

1. El sistema permitirá al usuario financiero ingresar y modificar los datos de una zona.
2. El sistema permitirá al usuario financiero ingresar y modificar los datos de un nivel.
3. El sistema permitirá al usuario financiero ingresar y modificar los datos de una ciudad.
4. El sistema permitirá al usuario financiero relacionar un nivel con una zona y asignar el valor correspondiente a estos.
5. El sistema permitirá al usuario financiero asignar una ciudad a una zona.
6. El sistema permitirá al usuario financiero visualizar un reporte del Nivel Zona Ciudad.

7. El sistema permitirá a la secretaria de una dependencia ingresar los datos de una solicitud de movilización externa.
8. El sistema permitirá a la secretaria de una dependencia ingresar los datos de una solicitud de movilización interna.
9. El sistema permitirá a la secretaria de una dependencia rechazar o anular una solicitud de movilización interna de su dependencia.
10. El sistema permitirá a la secretaria de una dependencia rechazar o anular una solicitud de movilización externa de su dependencia.
11. El sistema permitirá al jefe de dependencia aprobar, rechazar o anular una solicitud de movilización interna de su dependencia.
12. El sistema permitirá al jefe de dependencia aprobar, rechazar o anular una solicitud de movilización externa de su dependencia.
13. El sistema permitirá al jefe y secretaria de dependencia visualizar la lista de solicitudes de movilización internas aprobadas.
14. El sistema permitirá al jefe y secretaria de dependencia visualizar la lista de solicitudes de movilización externas aprobadas.
15. El sistema permitirá a la secretaria de dependencia editar una solicitud de movilización interna rechazada.
16. El sistema permitirá a la y secretaria de dependencia editar una solicitud de movilización externa rechazada.
17. El sistema permitirá al jefe y secretaria de dependencia visualizar la lista de solicitudes de movilización internas anuladas.
18. El sistema permitirá al jefe y secretaria de dependencia visualizar la lista de solicitudes de movilización externas anuladas.

19. El sistema permitirá a la secretaria de Investigación y Desarrollo editar y aprobar o anular una solicitud de movilización interna pendiente enviada por las dependencias.
20. El sistema permitirá a la secretaria de Investigación y Desarrollo editar y aprobar o anular una solicitud de movilización externa pendiente enviada por las dependencias.
21. El sistema permitirá a la secretaria de Investigación y Desarrollo asignar un vehículo a una o varias solicitudes de movilización internas enviadas por las dependencias.
22. El sistema permitirá al vicerrector de Investigación y Desarrollo aprobar o anular las solicitudes de movilización interna.
23. El sistema permitirá al vicerrector de Investigación y Desarrollo aprobar o anular una solicitud de movilización externa.
24. El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar la lista de solicitudes de movilización internas aprobadas.
25. El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar la lista de solicitudes de movilización externas aprobadas.
26. El sistema permitirá al jefe de movilización, vicerrector y secretaria de Investigación y Desarrollo generar un salvoconducto.
27. El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar la lista de solicitudes de movilización internas anuladas.
28. El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar la lista de solicitudes de movilización externas anuladas.

29. El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar la lista de solicitudes de movilización internas aprobadas en un rango de fechas.
30. El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar la lista de solicitudes de movilización externas aprobadas en un rango de fechas.
31. El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar el número de solicitudes según el estado seleccionado en un rango de fechas.
32. El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar un listado de los lugares a los que ha viajado un comisionado y visualizar el respectivo detalle de la comisión, dado la cédula del comisionado.
33. El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar un listado de comisionados y el detalle de la comisión dado un rango de fechas y una dependencia específica.
34. El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar de manera gráfica el comisionado que más ha salido en un rango de fechas.
35. El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar un listado de comisionados y el detalle de la comisión dado un rango de fechas.
36. El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar el listado del movimiento de vehículos y el detalle del mismo en un rango de fechas.

37. El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar de manera gráfica el lugar más visitado por los comisionados.
38. El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar de manera gráfica el lugar más visitado por los comisionados dado un rango de fechas.
39. El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo un listado de salvoconductos aprobados dado un rango de fechas.
40. El sistema permitirá a cada comisionado visualizar el valor estimado del viático dado el número de su cédula.

Requerimientos No funcionales

- ***Amigabilidad***

El sistema proporcionara una interfaz gráfica amigable, fácil de utilizar e intuitiva para el usuario.

- ***Disponibilidad***

Es sistema estará disponible las 24 horas ya sea para crear una solicitud, para cambiar el estado de las mismas, o para la generación de reportes con la finalidad de agilizar el proceso.

- ***Fiabilidad***

El sistema es confiable ya que es sometido a continuas pruebas y validaciones para medir su grado de eficiencia.

- ***Mantenibilidad***

Mantenimiento de la base de datos.

- **Seguridad**

El sistema tendrá un formulario de autenticación.

Uso de sesiones de usuario.

Definición de usuarios

Tomando en cuenta cada uno de los requerimientos, se definen los usuarios que interactuarán directamente con el Sistema de Gestión de Viáticos, así como también los roles de cada uno de estos.

A continuación se muestran dichos usuarios, en el orden en que se lleva el proceso de creación, aprobación y generación de salvoconductos.

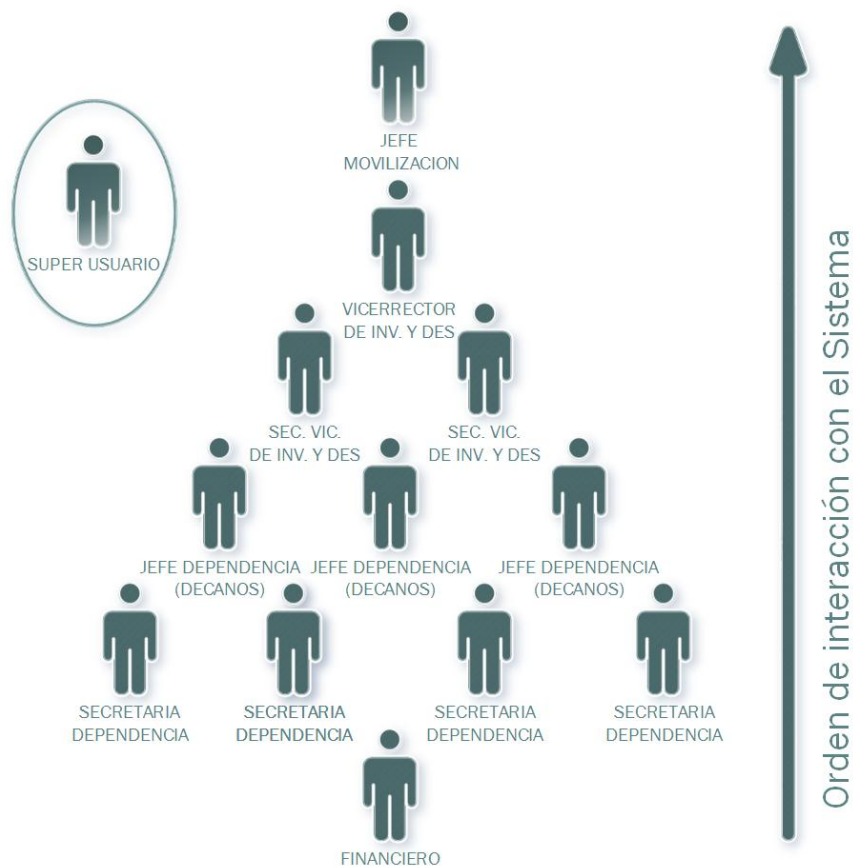


Figura IV. 42 Usuarios del Sistema de Gestión de Viáticos

Definición de roles

Teniendo en cuenta las necesidades de cada uno de los usuarios antes mencionados se definió y asignó los roles como se muestra a continuación.



Figura IV. 43 Rol del Usuario Financiero

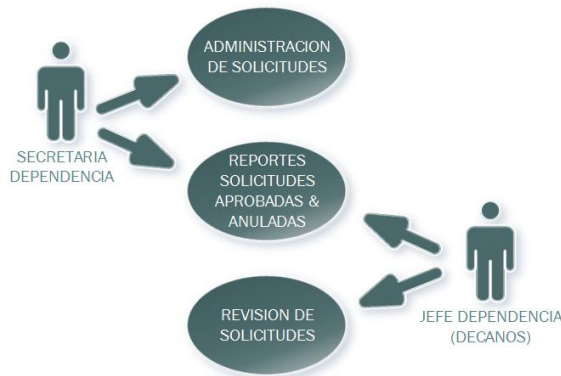


Figura IV. 44 Roles de los Usuarios Secretaria y Jefe de Dependencia



Figura IV. 45 Roles de los Usuarios Secretaria y Vicerrector de Investigación y Desarrollo



Figura IV. 46 Rol del Usuario Jefe de Movilización

Historias de Usuario

Una historia de usuario es una representación de un requerimiento de software escrito en una o dos frases utilizando el lenguaje común del usuario, a continuación se detalla cada una de estas con la finalidad de evitar la elaboración de una gran cantidad de documentos formales, sin requerir mucho tiempo para administrarles, así también se muestra el grupo de usuarios que utilizaran el sistema con sus respectivos roles y permisos.

Tabla IV. XV Historia de Usuario Ingreso y modificación de zona

HISTORIA DE USUARIO	
Numero: 1	Usuario: Financiero
Nombre Historia: Ingreso y modificación de zona.	Iteración Asignada: 1
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: Ana Llalao	
Descripción: El usuario ingresará o modificará los datos de una zona (nombre, observación)	
Observaciones:	

Tabla IV. XVI Historia de Usuario Ingreso y modificación de nivel

HISTORIA DE USUARIO	
Numero: 2	Usuario: Financiero
Nombre Historia: Ingreso y modificación de nivel.	Iteración Asignada: 1
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: José Luisataxi	
Descripción: El usuario ingresará o modificará los datos de un nivel (nombre, observación)	

Tabla IV. XVII Historia de Usuario Ingreso y modificación de ciudad

HISTORIA DE USUARIO	
Numero: 3	Usuario: Financiero
Nombre Historia: Ingreso y modificación de ciudad	Iteración Asignada: 1
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: José Luisataxi	
Descripción: El usuario ingresará o modificará los datos de una ciudad (nombre, observación)	
Observaciones:	

Tabla IV. XVIII Historia de Usuario Relacionar nivel zona

HISTORIA DE USUARIO	
Numero: 4	Usuario: Financiero
Nombre Historia: Relacionar nivel zona	Iteración Asignada: 1
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: Ana Llalao	
Descripción: El usuario relacionarán una zona con un nivel y se asignará un valor, las zonas y niveles se seleccionarán desde una lista desplegable, la misma que contendrá el listado de estas, posteriormente presionará un botón aceptar para finalizar la actividad.	
Observaciones: El valor mencionado se cargará automáticamente dicho valor esta ya establecido en el Reglamento de Gestión de Viáticos de la ESPOCH	

Tabla IV. XIX Historia de Usuario Asignar una ciudad a una zona

HISTORIA DE USUARIO	
Numero: 5	Usuario: Financiero
Nombre Historia: Asignar una ciudad a una zona	Iteración Asignada: 1
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: Ana Llalao	
Descripción: El usuario asignará las ciudades a una zona determinada, las zonas y ciudades se seleccionarán desde una lista desplegable, la misma que contendrá un listado tanto de ciudades como de zonas, posteriormente presionará un botón aceptar para finalizar la actividad.	
Observaciones:	

Tabla IV. XX Historia de Usuario Reporte Nivel Zona Ciudad

HISTORIA DE USUARIO	
Numero: 6	Usuario: Financiero
Nombre Historia: Reporte nivel zona ciudad	Iteración Asignada: 1
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: José Luisataxi	
Descripción: El usuario visualizara un reporte en el que se detalla las ciudades que pertenecen a un zona y el valor del viatico de acuerdo a la zona y nivel que corresponda.	
Observaciones:	

Tabla IV. XXI Historia de Usuario Crear solicitud de movilización externa

HISTORIA DE USUARIO	
Numero: 7	Usuario: Secretaria de Dependencia
Nombre Historia: Crear solicitud de movilización externa	Iteración Asignada: 1
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: Ana Llalao – José Luisataxi	
<p>Descripción: El usuario podrá ingresar los datos de una solicitud de movilización interna, la misma que contiene los datos de la comisión y de los comisionados. Entre los datos de la comisión tenemos la fecha de emisión, la misma que se carga automáticamente, la ciudad de destino que se seleccionará de una lista desplegable, la fecha salida y retorno que será seleccionada de un calendario respectivamente, la hora inicio y fin de la actividad, hora de la salida y retorno, empresa y motivo de la comisión, además tendrá una opción que permite cargar un archivo pdf el mismo que contendrá las actividades a realizarse en esta comisión, siendo este último de carácter opcional ya que posee un campo que le permitirá digitar las actividades a realizarse.</p> <p>Los datos de los comisionados se obtienen ingresando el número de cédula, presionando el botón buscar y se cargarán los datos del comisionado en una tabla en la cual debe asignarse un nivel, este nivel será seleccionado desde una lista desplegable, finalmente se guarda la solicitud.</p>	
Observaciones: Los datos de los comisionados se obtendrán con un servicio web que proveerá el departamento de Recursos Humanos.	

Tabla IV. XXII Historia de Usuario Crear solicitud de movilización interna

HISTORIA DE USUARIO	
Numero: 8	Usuario: Secretaria de Dependencia
Nombre Historia: Crear solicitud de movilización interna	Iteración Asignada: 2
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto

Programador Responsable: Ana Llalao – José Luisataxi
<p>Descripción: El usuario podrá ingresar los datos de una solicitud de movilización interna, la misma que contiene los datos de la comisión y de los comisionados. Entre los datos de la comisión tenemos la fecha de emisión, la misma que se carga automáticamente, la ciudad de destino que se seleccionará de una lista desplegable, la fecha salida y retorno que será seleccionada de un calendario respectivamente, la hora inicio y fin de la actividad, hora de la salida y retorno, empresa y motivo de la comisión, además tendrá una opción que permite cargar un archivo pdf el mismo que contendrá las actividades a realizarse en esta comisión, siendo este último de carácter opcional ya que posee un campo que le permitirá digitar las actividades a realizarse.</p> <p>Los datos de los comisionados se obtienen ingresando el número de cédula, presionando el botón buscar y se cargarán los datos del comisionado en una tabla en la cual debe asignarse un nivel, este nivel será seleccionado desde una lista desplegable, finalmente se guarda la solicitud.</p>
<p>Observaciones: Los datos de los comisionados se obtendrán con un servicio web que proveerá el departamento de Recursos Humanos.</p>

Tabla IV. XXIII Historia de Usuario Rechazar o anular una solicitud de movilización interna

HISTORIA DE USUARIO	
Numero: 9	Usuario: Secretaria de Dependencia
Nombre Historia: Rechazar o anular una solicitud de movilización interna	Iteración Asignada: 2
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: Ana Llalao	
<p>Descripción: El usuario visualiza un listado de solicitudes de movilización internas ingresadas con estado pendiente, permitiendo seleccionar una de estas, ver su detalle (datos solicitud de movilización) de la misma y cambiar el estado de pendiente ha anulado o rechazado.</p>	
Observaciones:	

Tabla IV. XXIV Historia de Usuario Rechazar o anular una solicitud de movilización externa

HISTORIA DE USUARIO	
Numero: 10	Usuario: Secretaria de Dependencia
Nombre Historia: Rechazar o anular una solicitud de movilización externa	Iteración Asignada: 2
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: José Luisataxi	
Descripción: El usuario visualiza un listado de solicitudes de movilización externas ingresadas con estado pendiente, permitiendo seleccionar una de estas, ver su detalle (datos solicitud de movilización) de la misma y cambiar el estado de pendiente ha anulado o rechazado.	
Observaciones:	

Tabla IV. XXV Historia de Usuario Aprobar, rechazar o anular una solicitud de movilización interna

HISTORIA DE USUARIO	
Numero: 11	Usuario: Jefe de Dependencia
Nombre Historia: Aprobar, rechazar o anular una solicitud de movilización interna	Iteración Asignada: 2
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: Ana Llalao	
Descripción: El usuario visualiza un listado de solicitudes de movilización internas pendientes, permitiendo seleccionar una de estas, ver su detalle (datos solicitud de movilización) de la misma y cambiar el estado de pendiente ha aprobado, anulado o rechazado.	
Observaciones:	

Tabla IV. XXVI Historia de Usuario Aprobar, rechazar o anular una solicitud de movilización externa

HISTORIA DE USUARIO	
Numero: 12	Usuario: Jefe de Dependencia
Nombre Historia: Aprobar, rechazar o anular una solicitud de movilización externa	Iteración Asignada: 2
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: Ana Llalao	
Descripción: El usuario visualiza un listado de solicitudes de movilización externas pendientes, permitiendo seleccionar una de estas, ver su detalle (datos solicitud de movilización) de la misma y cambiar el estado de pendiente ha aprobado, anulado o rechazado.	
Observaciones:	

Tabla IV. XXVII Historia de Usuario Listar solicitudes de movilización internas aprobadas

HISTORIA DE USUARIO	
Numero: 13	Usuario: Secretaria / Jefe de Dependencia
Nombre Historia: Listar solicitudes de movilización internas aprobadas	Iteración Asignada: 2
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: José Luisataxi	
Descripción: Los usuarios pueden visualizar la lista de solicitudes de movilización internas aprobadas y el detalle de cada una de estas.	
Observaciones:	

Tabla IV. XXVIII Historia de Usuario Listar solicitudes de movilización externas aprobadas

HISTORIA DE USUARIO	
Numero: 14	Usuario: Secretaria / Jefe de Dependencia
Nombre Historia: Listar solicitudes de movilización externas aprobadas	Iteración Asignada: 2
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: José Luisataxi	
Descripción: Los usuarios pueden visualizar la lista de solicitudes de movilización externas aprobadas y el detalle de cada una de estas.	
Observaciones:	

Tabla IV. XXIX Historia de Usuario Editar una solicitud de movilización internas rechazadas

HISTORIA DE USUARIO	
Numero: 15	Usuario: Secretaria de Dependencia
Nombre Historia: Editar una solicitud de movilización internas rechazadas	Iteración Asignada: 3
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: Ana Llalao – José Luisataxi	
Descripción: El usuario visualiza la lista de solicitudes de movilización internas rechazadas, seleccionan unas de ellas y editan los datos del vehículo y datos de la comisión (Fechas, Empresa, Motivo, Actividades) y retorna la solicitud modificada a la lista de solicitudes pendientes.	
Observaciones:	

Tabla IV. XXX Historia de Usuario Editar una solicitud de movilización externas rechazadas

HISTORIA DE USUARIO	
Numero: 16	Usuario: Secretaria de Dependencia
Nombre Historia: Editar una solicitud de movilización externas rechazadas	Iteración Asignada: 3
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: Ana Llalao – José Luisataxi	
Descripción: El usuario visualiza la lista de solicitudes de movilización externas rechazadas, seleccionan unas de ellas y editan los datos de la comisión (Fechas, Empresa, Motivo, Actividades) y retorna la solicitud modificada a la lista de solicitudes pendientes.	
Observaciones:	

Tabla IV. XXXI Historia de Usuario Listar solicitudes internas anuladas

HISTORIA DE USUARIO	
Numero: 17	Usuario: Secretaria / Jefe de Dependencia
Nombre Historia: Listar solicitudes internas anuladas	Iteración Asignada: 3
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: José Luisataxi	
Descripción: Los usuarios visualizan la lista de solicitudes de movilización internas anuladas por la secretaria y jefe de dependencia y el detalle de cada una de estas.	
Observaciones:	

Tabla IV. XXXII Historia de Usuario Listar solicitudes externas anuladas

HISTORIA DE USUARIO	
Numero: 18	Usuario: Secretaria / Jefe de Dependencia
Nombre Historia: Listar solicitudes externas anuladas	Iteración Asignada: 3
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: José Luisataxi	
Descripción: Los usuarios visualizan la lista de solicitudes de movilización externas anuladas por la secretaria y jefe de dependencia y el detalle de cada una de estas.	
Observaciones:	

Tabla IV. XXXIII Historia de Usuario Editar y aprobar o anular una solicitud de movilización interna

HISTORIA DE USUARIO	
Numero: 19	Usuario: Secretaria de Inv. y Desarrollo
Nombre Historia: Editar y aprobar o anular una solicitud de movilización interna	Iteración Asignada: 3
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: Ana Llalao	
Descripción: El usuario visualiza un listado de solicitudes de movilización internas rechazadas, permitiendo seleccionar una de estas, ver su detalle (datos solicitud de movilización), editar la hora de salida y retorno de la comisión y aprobarlo o anularlo según su criterio.	
Observaciones:	

Tabla IV. XXXIV Historia de Usuario Editar y aprobar o anular una solicitud de movilización externa

HISTORIA DE USUARIO	
Numero: 20	Usuario: Secretaria de Inv. y Desarrollo
Nombre Historia: Editar y aprobar o anular una solicitud de movilización externa	Iteración Asignada: 3
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: José Luisataxi	
Descripción: El usuario visualiza un listado de solicitudes de movilización externas rechazadas, permitiendo seleccionar una de estas, ver su detalle (datos solicitud de movilización), editar la hora de salida y retorno de la comisión y aprobarlo o anularlo según su criterio.	
Observaciones:	

Tabla IV. XXXV Historia de Usuario Asignar vehículo a solicitudes de movilización internas

HISTORIA DE USUARIO	
Numero: 21	Usuario: Secretaria de Inv. y Desarrollo
Nombre Historia: Asignar vehículo a solicitudes de movilización internas	Iteración Asignada: 3
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: Ana Llalao – José Luisataxi	
Descripción: El usuario visualiza un listado de solicitudes de movilización internas pendientes, permitiendo seleccionar una de estas, y agregar a una nueva lista para poder asignar un vehículo, estas comisiones tendrán el mismo lugar de destino y la misma fecha y hora de salida y retorno, para la asignación de un vehículo se presenta un listado de vehículos disponibles, siendo la placa de estos el filtro de búsqueda, una vez seleccionado el vehiculó se mostraran los siguientes datos: nombre, apellido y cedula del conductor así también la marca, número, fecha matrícula, tipo, placa, motor y color del vehículo.	

Tabla IV. XXXVI Historia de Usuario Aprobar o anular una solicitud de movilización interna

HISTORIA DE USUARIO	
Numero: 22	Usuario: Vicerrector de Inv. y Desarrollo
Nombre Historia: Aprobar o anular una solicitud de movilización interna	Iteración Asignada: 3
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: Ana Llalao	
Descripción: El usuario visualiza un listado de solicitudes de movilización internas (Salvoconductos) pendientes, permitiendo seleccionar una o varias de estas, y cambiar el estado de pendiente ha aprobado o anulado.	
Observaciones:	

Tabla IV. XXXVII Historia de Usuario Aprobar o anular una solicitud de movilización externa

HISTORIA DE USUARIO	
Numero: 23	Usuario: Vicerrector de Inv. y Desarrollo
Nombre Historia: Aprobar o anular una solicitud de movilización externa	Iteración Asignada: 3
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: Ana Llalao	
Descripción: El usuario visualiza un listado de solicitudes de movilización externas pendientes, permitiendo seleccionar una de estas y cambiar el estado de pendiente ha aprobado o anulado.	
Observaciones:	

Tabla IV. XXXVIII Historia de Usuario Listar solicitudes de movilización internas aprobadas

HISTORIA DE USUARIO	
Numero: 24	Usuario: Secretaria / Vicerrector de Inv. y Desarrollo
Nombre Historia: Listar solicitudes de movilización internas aprobadas	Iteración Asignada: 3
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: José Luisataxi	
Descripción: Los usuarios pueden visualizar la lista de solicitudes de movilización internas aprobadas y el detalle de cada una de estas.	
Observaciones:	

Tabla IV. XXXIX Historia de Usuario Listar solicitudes de movilización externas aprobadas

HISTORIA DE USUARIO	
Numero: 25	Usuario: Secretaria / Vicerrector de Inv. y Desarrollo
Nombre Historia: Listar solicitudes de movilización externas aprobadas	Iteración Asignada: 3
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: José Luisataxi	
Descripción: Los usuarios pueden visualizar la lista de solicitudes de movilización externas aprobadas y el detalle de cada una de estas.	
Observaciones:	

Tabla IV. XL Historia de Usuario Generar Salvoconducto

HISTORIA DE USUARIO	
Numero: 26	Usuario: Secretaria / Vicerrector de Inv. y Desarrollo / Jefe de Movilización
Nombre Historia: Generar Salvoconducto	Iteración Asignada: 3
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: Ana Llalao – José Luisataxi	
Descripción: Los usuarios visualizan un listado de solicitudes de movilización internas aprobadas, selecciona una de ellas y genera el salvoconducto el mismo que puede ser impreso inmediatamente.	
Observaciones:	

Tabla IV. XLI Historia de Usuario Listar solicitudes de movilización internas anuladas

HISTORIA DE USUARIO	
Numero: 27	Usuario: Secretaria / Vicerrector de Inv. y Desarrollo
Nombre Historia: Listar solicitudes de movilización internas anuladas	Iteración Asignada: 4
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: José Luisataxi	
Descripción: Los usuarios pueden visualizar la lista de solicitudes de movilización internas anuladas y el detalle de cada una de estas.	
Observaciones:	

Tabla IV. XLII Historia de Usuario Listar solicitudes de movilización externas anuladas

HISTORIA DE USUARIO	
Numero: 28	Usuario: Secretaria / Vicerrector de Inv. y Desarrollo
Nombre Historia: Listar solicitudes de movilización externas anuladas	Iteración Asignada: 4
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: José Luisataxi	
Descripción: Los usuarios pueden visualizar la lista de solicitudes de movilización externas anuladas y el detalle de cada una de estas.	
Observaciones:	

Tabla IV. XLIII Historia de Usuario Listar solicitudes de movilización internas aprobadas por fechas

HISTORIA DE USUARIO	
Numero: 29	Usuario: Secretaria / Vicerrector de Inv. y Desarrollo
Nombre Historia: Listar solicitudes de movilización internas aprobadas por fechas	Iteración Asignada: 4
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: Ana Llalao	
Descripción: Los usuarios pueden visualizar la lista de solicitudes de movilización internas aprobadas y el detalle de cada una de estas en un rango de fechas seleccionado desde un calendario.	
Observaciones:	

Tabla IV. XLIV Historia de Usuario Listar solicitudes de movilización externas aprobadas por fechas

HISTORIA DE USUARIO	
Numero: 30	Usuario: Secretaria / Vicerrector de Inv. y Desarrollo
Nombre Historia: Listar solicitudes de movilización externas aprobadas por fechas	Iteración Asignada: 4
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: Ana Llalao	
Descripción: Los usuarios pueden visualizar la lista de solicitudes de movilización externas aprobadas y el detalle de cada una de estas en un rango de fechas seleccionado desde un calendario.	
Observaciones:	

Tabla IV. XLV Historia de Usuario Visualizar número de solicitudes

HISTORIA DE USUARIO	
Numero: 31	Usuario: Secretaria / Vicerrector de Inv. y Desarrollo
Nombre Historia: Visualizar número de solicitudes	Iteración Asignada: 4
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: José Luisataxi	
Descripción: Los usuarios pueden visualizar el número de solicitudes internas y externas aprobadas, anuladas respectivamente en un rango de fechas.	
Observaciones:	

Tabla IV. XLVI Historia de Usuario Reporte de lugares más visitados de un comisionado

HISTORIA DE USUARIO	
Numero: 32	Usuario: Secretaria / Vicerrector de Inv. y Desarrollo
Nombre Historia: Reporte de lugares más visitados de un comisionado	Iteración Asignada: 4
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: José Luisataxi	
Descripción: Los usuarios pueden visualizar un listado de los lugares a los que ha viajado un comisionado y el respectivo detalle de la comisión, dado la cédula del comisionado.	
Observaciones:	

Tabla IV. XLVII Historia de Usuario Reporte lista comisionados por fechas y dependencia

HISTORIA DE USUARIO	
Numero: 33	Usuario: Secretaria / Vicerrector de Inv. y Desarrollo
Nombre Historia: Reporte lista comisionados por fechas y dependencia	Iteración Asignada: 4
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: Ana Llalao	
Descripción: Los usuarios pueden visualizar el listado de comisionados y el detalle de información de la comisión en la que participo dado un rango de fechas y una dependencia específica.	
Observaciones:	

Tabla IV. XLVIII Historia de Usuario Gráfica del comisionado que más ha salido por fechas

HISTORIA DE USUARIO	
Numero: 34	Usuario: Secretaria / Vicerrector de Inv. y Desarrollo
Nombre Historia: Gráfica del comisionado que más ha salido por fechas	Iteración Asignada: 4
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: Ana Llalao – José Luisataxi	
Descripción: Los usuarios pueden visualizar de manera gráfica el comisionado que más ha salido en comisiones en un rango de fechas seleccionado desde un calendario.	
Observaciones:	

Tabla IV. XLIX Historia de Usuario Reporte lista de comisionados por fechas

HISTORIA DE USUARIO	
Numero: 35	Usuario: Secretaria / Vicerrector de Inv. y Desarrollo
Nombre Historia: Reporte lista de comisionados por fechas	Iteración Asignada: 4
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: Ana Llalao	
Descripción: Los usuarios pueden visualizar la lista de comisionados y el detalle de información de la comisión en la que participo dado un rango de fechas.	
Observaciones:	

Tabla IV. L Historia de Usuario Reporte listado de movimiento de vehículos en un rango de fechas

HISTORIA DE USUARIO	
Numero: 36	Usuario: Secretaria / Vicerrector de Inv. y Desarrollo
Nombre Historia: Reporte listado de movimiento de vehículos en un rango de fechas	Iteración Asignada: 4
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: José Luisataxi	
Descripción: Los usuarios pueden visualizar un listado del movimiento de vehículos dado un rango de fechas.	
Observaciones:	

Tabla IV. LI Historia de Usuario Grafica de lugares más visitados por los comisionados

HISTORIA DE USUARIO	
Numero: 37	Usuario: Secretaria / Vicerrector de Inv. y Desarrollo
Nombre Historia: Grafica de lugares más visitados por los comisionados	Iteración Asignada: 4
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: Ana Llalao – José Luisataxi	
Descripción: Los usuarios de Nivel 3 pueden visualizar de manera gráfica los lugares más visitados por los comisionados.	
Observaciones:	

Tabla IV. LII Historia de Usuario Gráfica de lugares más visitado por los comisionados por fechas

HISTORIA DE USUARIO	
Numero: 38	Usuario: Secretaria / Vicerrector de Inv. y Desarrollo
Nombre Historia: Gráfica de lugares más visitados por los comisionados por fechas	Iteración Asignada: 4
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: Ana Llalao – José Luisataxi	
Descripción: Los usuarios de pueden visualizar de manera gráfica los lugares más visitados por los comisionados dado un rango de fechas.	
Observaciones:	

Tabla IV. LIII Historia de Usuario Listado de salvoconductos aprobados dado fechas

HISTORIA DE USUARIO	
Numero: 39	Usuario: Secretaria / Vicerrector de Inv. y Desarrollo
Nombre Historia: Listado de salvoconductos aprobados dado fechas	Iteración Asignada: 4
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: José Luisataxi	
Descripción: Los usuarios pueden visualizar el listado de salvoconductos aprobados seleccionando el rango de fechas desde un calendario.	
Observaciones:	

Tabla IV. LIV Historia de Usuario Visualizar valor estimado del viático

HISTORIA DE USUARIO	
Numero: 40	Usuario: Servidor Publico
Nombre Historia: Visualizar valor estimado del viático	Iteración Asignada: 4
Prioridad en Negocio: Alto	Riesgo en Desarrollo: Alto
Programador Responsable: Ana Llalao	
Descripción: Los usuarios pueden visualizar un el valor estimado de cada viatico y su respectivo detalle dado su número de cedula.	
Observaciones:	

Planificación Inicial

Se desarrolló un plan de actividades previo al desarrollo de la aplicación, en el mismo se detallan las actividades, el responsable y el tiempo en que se desarrolla cada una de estas, como se indica en el **Anexo 1**.

Plan de iteraciones

El desarrollo se divide en tres iteraciones, en cada una de ellas se eligen las historias de usuarios a desarrollar y las tareas de desarrollo las mismas que se describen en las siguientes tablas.

a) Iteración 1

En esta iteracion se presentara el ingreso y modificacion de una zona, ciudad y nivel las relaciones exixtentes entre si y ademas el ingreso de una solicitud de movilizacion externa en la **Tabla IV. LV**, se describe cada uno de los requerimientos con sus actividades.

Tabla IV. LV Iteración 1

ID	Título	Asignado a
1	El sistema permitirá al usuario financiero ingresar y modificar los datos de una zona.	Ana Llalao
	- Crear clases	
	- Crear funciones en PostgreSQL	
	- Crear funciones en Java	
	- Crear controladores	
	- Crear Vistas	
2	El sistema permitirá al usuario financiero ingresar y modificar los datos de un nivel.	José Luisataxi
	- Crear clases	
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Crear controladores	
	- Crear Vistas	
3	El sistema permitirá al usuario financiero ingresar y modificar los datos de una ciudad.	José Luisataxi
	- Crear clases	
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Crear controladores	
	- Crear Vistas	
4	El sistema permitirá al usuario financiero relacionar un nivel con una zona y asignar el valor correspondiente a estos.	Ana Llalao

	- Crear clases	
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Crear controladores	
	- Crear Vistas	
5	El sistema permitirá al usuario financiero asignar una ciudad a una zona.	Ana Llalao
	- Crear clases	
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Crear controladores	
	- Crear Vistas	
6	El sistema permitirá al usuario financiero visualizar un reporte del Nivel Zona Ciudad.	José Luisataxi
	- Crear clases	
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Crear controladores	
	- Crear Vistas	
7	El sistema permitirá a la secretaria de una dependencia ingresar los datos de una solicitud de movilización externa.	José Luisataxi Ana Llalao
	- Crear clases	
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	

	- Crear controladores	
	- Crear Vistas	

b) Iteración 2

En esta iteración se desarrollara el ingreso de una solicitud de movilización interna, así como también la aprobación, rechazo o anulación de las solicitudes de movilización, a continuación en la **Tabla IV. LVI**, se detalla cada una de las actividades a realizarse.

Tabla IV. LVI Iteración 2

ID	Título	Asignado a
1	El sistema permitirá a la secretaria de una dependencia ingresar los datos de una solicitud de movilización interna.	José Luisataxi Ana Llalao
	- Crear clases	
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Crear controladores	
	- Crear Vistas	
2	El sistema permitirá a la secretaria de una dependencia rechazar o anular una solicitud de movilización interna de su dependencia.	Ana Llalao
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	

3	El sistema permitirá a la secretaria de una dependencia rechazar o anular una solicitud de movilización externa de su dependencia.	José Luisataxi
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
4	El sistema permitirá al jefe de dependencia aprobar, rechazar o anular una solicitud de movilización interna de su dependencia.	Ana LLalao
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
5	El sistema permitirá al jefe de dependencia aprobar, rechazar o anular una solicitud de movilización externa de su dependencia.	Ana Llalao
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
6	El sistema permitirá al jefe y secretaria de dependencia visualizar la lista de solicitudes de movilización internas aprobadas.	José Luisataxi
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	

7	El sistema permitirá al jefe y secretaria de dependencia visualizar la lista de solicitudes de movilización externas aprobadas.	José Luisataxi
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	

c) Iteración 3

Se realizará la edición de las solicitudes de movilización rechazadas, se listará las solicitudes de movilización anuladas, también se permitirá la edición y aprobación o anulación de las solicitudes de movilización, la asignación de un vehículo a las comisiones y la aprobación de las mismas.

Tabla IV. LVII Iteración 3

ID	Título	Asignado a
1	El sistema permitirá a la secretaria de dependencia editar una solicitud de movilización interna rechazada.	José Luisataxi Ana Llalao
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
2	El sistema permitirá a la secretaria de dependencia editar una solicitud de movilización externa rechazada.	José Luisataxi Ana Llalao
	- Crear funciones en Java	

	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
3	El sistema permitirá al jefe y secretaria de dependencia visualizar la lista de solicitudes de movilización internas anuladas.	José Luisataxi
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
4	El sistema permitirá al jefe y secretaria de dependencia visualizar la lista de solicitudes de movilización externas anuladas.	José Luisataxi
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
5	El sistema permitirá a la secretaria de Investigación y Desarrollo editar y aprobar o anular una solicitud de movilización interna pendiente enviada por las dependencias.	Ana Llalao
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
6	El sistema permitirá a la secretaria de Investigación y Desarrollo editar y aprobar o anular una solicitud de movilización externa pendiente enviada por las dependencias.	José Luisataxi

	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
7	El sistema permitirá a la secretaria de Investigación y Desarrollo asignar un vehículo a una o varias solicitudes de movilización internas enviadas por las dependencias.	José Luisataxi Ana Llalao
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
8	El sistema permitirá al vicerrector de Investigación y Desarrollo aprobar o anular las solicitudes de movilización interna.	Ana Llalao
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
9	El sistema permitirá al vicerrector de Investigación y Desarrollo aprobar o anular una solicitud de movilización externa.	Ana Llalao
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	

10	El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar la lista de solicitudes de movilización internas aprobadas.	José Luisataxi
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
11	El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar la lista de solicitudes de movilización externas aprobadas.	José Luisataxi
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
12	El sistema permitirá al jefe de movilización, vicerrector y secretaria de Investigación y Desarrollo generar un salvoconducto.	José Luisataxi Ana Llalao
	- Crear clases	
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	

d) Iteración 4

En esta iteración se detalla cada uno de los reportes solicitados en la **Tabla IV.**

LVIII, se detalla cada una de las actividades a desarrollarse.

Tabla IV. LVIII Iteración 4

ID	Título	Asignado a
1	El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar la lista de solicitudes de movilización internas anuladas.	José Luisataxi
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
2	El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar la lista de solicitudes de movilización externas anuladas.	José Luisataxi
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
3	El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar la lista de solicitudes de movilización internas aprobadas en un rango de fechas.	Ana Llalao
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
4	El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar la lista de solicitudes de movilización externas aprobadas en un rango de fechas.	Ana Llalao
	- Crear funciones en Java	

	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
5	El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar el número de solicitudes según el estado seleccionado en un rango de fechas.	José Luisataxi
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
6	El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar un listado de los lugares a los que ha viajado un comisionado y visualizar el respectivo detalle de la comisión, dado la cédula del comisionado.	José Luisataxi
	- Crear clases	
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
7	El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar un listado de comisionados y el detalle de la comisión dado un rango de fechas y una dependencia específica.	Ana Llalao
	- Crear clases	
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	

	- Crear Vistas	
8	El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar de manera gráfica el comisionado que más ha salido en un rango de fechas.	José Luisataxi Ana Llalao
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
9	El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar un listado de comisionados y el detalle de la comisión dado un rango de fechas.	Ana Llalao
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
10	El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar el listado del movimiento de vehículos y el detalle del mismo en un rango de fechas.	José Luisataxi
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
11	El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar de manera gráfica el lugar más visitado por los comisionados.	José Luisataxi Ana Llalao
	- Crear funciones en Java	

	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
12	El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo visualizar de manera gráfica el lugar más visitado por los comisionados dado un rango de fechas.	José Luisataxi Ana Llalao
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
13	El sistema permitirá al vicerrector y secretaria de Investigación y Desarrollo un listado de salvoconductos aprobados dado un rango de fechas.	José Luisataxi
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	
14	El sistema permitirá a cada comisionado visualizar el valor estimado del viático dado el número de su cédula.	Ana Llalao
	- Crear funciones en Java	
	- Crear funciones en PostgreSQL	
	- Modificar controladores	
	- Crear Vistas	

4.2.2. Fase II: Diseño de Software

4.2.2.1. Diseño de Base de Datos

A continuación se visualiza el diseño de la base de datos para el desarrollo del sistema de Gestión de Viáticos de la ESPOCH.

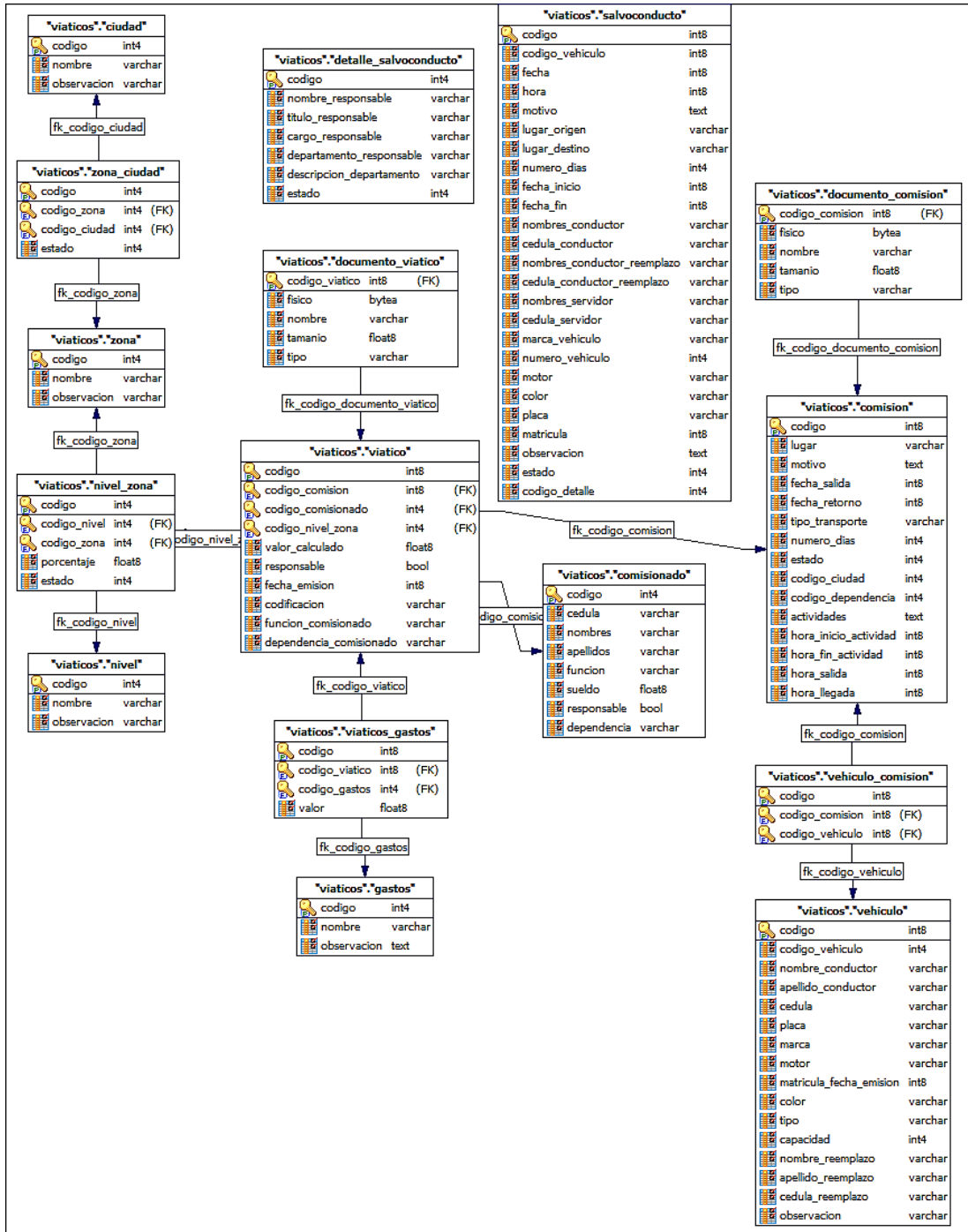


Figura IV. 47 Diseño de Base de Datos

4.2.2.2. Diseño de Interfaces

La metodología XP sugiere que hay que conseguir diseños simples y sencillos, hay que procurar hacerlo todo lo menos complicado posible para conseguir un diseño fácilmente entendible que costara menos tiempo y esfuerzo en desarrollar, para lo cual se propone los siguientes prototipos.

Este prototipo de formulario, titulado 'Nuevo Nivel', está diseñado con un encabezado gris que contiene el título. El cuerpo del formulario está dividido en tres secciones horizontales: la primera contiene el campo de texto 'Nombre:'; la segunda contiene el campo de texto 'Observación:'. En la parte inferior, se encuentran dos botones de acción: 'Aceptar' y 'Cancelar', ambos con un efecto de sombra y un borde sutil.

Figura IV. 48 Ingreso Nivel

Este prototipo de formulario, titulado 'Nueva Zona', sigue el mismo patrón de diseño que el anterior. Tiene un encabezado gris con el título 'Nueva Zona'. El formulario está dividido en tres secciones: 'Nombre:', 'Observación:' y dos botones 'Aceptar' y 'Cancelar' en la base.

Figura IV. 49 Ingreso Zona

Nueva Ciudad

Nombre:

Observación:

Figura IV. 50 Ingreso Ciudad

Nuevo Nivel Zona

Zona	Nivel	Valor
Zona A <input type="button" value="v"/>	Nivel 2 <input type="button" value="v"/>	70.00 <input type="text"/>

Figura IV. 51 Ingreso Nivel Zona

Nueva Zona Ciudad

Zona	Ciudad
Zona A <input type="button" value="v"/>	Riobamba <input type="button" value="v"/>

Figura IV. 52 Ingreso Zona Ciudad

Agregar Solicitud

Datos Comisión **Registre las Actividades** Datos Comisionados

Fecha Emisión:	<input type="text" value="01/10/2012"/>	Ciudad:	<input type="text" value="Quito"/>	
Fecha Salida:	<input type="text" value="03/10/2012"/>	Fecha Retorno:	<input type="text" value="03/10/2012"/>	
Hora Inicio Actividad:	<input type="text" value="09:00"/>	Hora Fin Actividad:	<input type="text" value="12:00"/>	
Hora Salida:	<input type="text" value="05:00"/>	Hora Retorno:	<input type="text" value="16:00"/>	
Empresa:	<input type="text" value="ESPE"/>	Número Días:	<input type="text" value="1"/>	
Motivo:	<input type="text" value="Reunión para tratar asuntos varios"/>			

Fisico:

Nombre:

Figura IV. 53 Ingreso Datos de la Comisión

Agregar Solicitud

Datos Comisión **Registre las Actividades** Datos Comisionados

Figura IV. 54 Ingreso de las Actividades

Agregar Solicitud

Datos Comisión Registre las Actividades Datos Comisionados

DATOS DEL RESPONSABLE

Ingrese el número de cédula:

Cédula	Apellidos	Nombres	Función	Nivel
0604799756	Duran	Fernando	Decano	Nivel1

DATOS COMISIONADOS

Ingrese el número de cédula:

Cédula	Apellidos	Nombres	Función	Nivel
--------	-----------	---------	---------	-------

Figura IV. 55 Ingreso de los Datos de los Comisionados

SOLICITUDES INTERNAS PENDIENTES

Cédula	Apellidos	Nombres	Función	Ciudad	Lugar	F. Emisión	F. Salida	F. Retorno
06089	Herrera	Anibal	Director	Quito	ESPOL	03-10-2012	03-10-2012	03-13-2012

Cédula	Apellidos	Nombres	Función	Ciudad
0604799756	Duran	Fernando	Decano	Quito

Seleccione

Conductor: Cédula:

Capacidad: Tipo Vehículo:

Marca: Placa:

Número: Motor:

Matrícula: Color:

Reemplazo:

Cédula:

Observación:

Figura IV. 56 Asignar Vehículo

4.2.2.3. Diccionario de datos

Una buena práctica de ingeniería es generar un diccionario de datos, pues es fundamental a la hora de conocer su estructura por técnicos ajenos a un proyecto.

A continuación se describe cada una de las tablas creadas, con sus respectivos atributos.

- **Tabla ciudad**

Esta tabla permite almacenar las distintas ciudades a las que pueden dirigirse las comisiones.

Tabla IV. LIX Descripción de la tabla ciudad

Nombre de la columna	Tipo de dato	Primary Key	Not Null	AutoInc
Código	int4	SI	SI	SI
Nombre	Varchar	NO	NO	NO
observacion	Varchar	NO	NO	NO

- **Tabla zona**

Esta tabla almacenará las zonas en las que está dividido el país, las mismas que contendrán ciudades.

Tabla IV. LX Descripción de la tabla zona

Nombre de la columna	Tipo de dato	Primary Key	Not Null	AutoInc
Código	int4	SI	SI	SI
Nombre	Varchar	NO	NO	NO
observación	Varchar	NO	NO	NO

- **Tabla zona_ciudad**

En esta tabla se almacenará la relación entre la ciudad y la zona.

Tabla IV. LXI Descripción de la tabla zona_ciudad

Nombre de la columna	Tipo de dato	Primary Key	Not Null	AutoInc
código	int4	SI	SI	SI
codigo_zona	int4	NO	NO	NO
codigo_ciudad	int4	NO	NO	NO
estado	int4	NO	NO	NO

- **Tabla nivel**

Esta tabla contendrá los niveles en los que están divididos quienes trabajan en la ESPOCH.

Tabla IV. LXII Descripción de la tabla nivel

Nombre de la columna	Tipo de dato	Primary Key	Not Null	AutoInc
código	int4	SI	SI	SI
codigo_zona	int4	NO	NO	NO
codigo_ciudad	int4	NO	NO	NO
estado	int4	NO	NO	NO

- **Tabla nivel_zona**

Aquí almacenaremos la relación de un nivel con una zona, y un monto en dependencia de esta.

Tabla IV. LXIII Descripción de la tabla nivel_zona

Nombre de la columna	Tipo de dato	Primary Key	Not Null	AutoInc
código	int4	SI	SI	SI
codigo_zona	int4	NO	NO	NO
codigo_ciudad	int4	NO	NO	NO
estado	int4	NO	NO	NO

El atributo *estado* permitirá saber si esta activa o inactiva la relación entre un nivel y una zona, a continuación se muestra los estados y sus descripciones.

Tabla IV. LXIV Estados de la tabla nivel_zona

Estado	Descripción
0	Relación nivel_zona Inactiva.
1	Relación nivel_zona Activa.

- **Tabla comisión**

En esta tabla se almacenará los datos generales de la comisión como a continuación de describe.

Tabla IV. LXV Descripción de la tabla comisión

Nombre de la columna	Tipo de dato	Primary Key	Not Null	AutoInc
Código	int8	SI	SI	SI
Lugar	varchar	NO	NO	NO
Motivo	Text	NO	NO	NO
fecha_salida	int8	NO	NO	NO
fecha_retorno	int8	NO	NO	NO
tipo_transporte	varchar	NO	NO	NO

numero_dias	int4	NO	NO	NO
Estado	int4	NO	NO	NO
codigo_ciudad	int4	NO	NO	NO
codigo_dependencia	int4	NO	NO	NO
actividades	text	NO	NO	NO
hora_inicio_actividad	int8	NO	NO	NO
hora_fin_actividad	int8	NO	NO	NO
hora_salida	int8	NO	NO	NO
hora_llegada	int8	NO	NO	NO

El atributo *estado* permite conocer el estado en el que se encuentra una comisión, a continuación se detalla los posibles estados en los que se puede encontrar:

Tabla IV. LXVI Estados de la tabla comisión

Estado	Descripción
0	Comisión aprobada por el Jefe de Dependencia/pendiente para la Secretaria del Vicerrectorado de Investigación y Desarrollo.
1	Comisión aprobada por el Vicerrector de Investigación y Desarrollo.
2	Comisión pendiente para el Jefe de Dependencia.
3	Comisión rechazada por el Jefe de Dependencia o Secretaria
4	Comisión anulada por el Jefe de Dependencia o Secretaria
5	Comisión anulada por el Vicerrectorado de Investigación y Desarrollo.
6	Comisión aprobada por la Secretaria del Vicerrectorado de Investigación y Desarrollo/pendiente para la misma para asignación de vehículo.
7	Comisión asignada vehículo por la Secretaria del Vicerrectorado de Investigación y Desarrollo/pendiente para el Vicerrector de Investigación y Desarrollo.

- **Tabla documento_comision**

En esta tabla se almacenará un archivo PDF con la descripción de las actividades a realizar en la comisión, cabe recalcar que este archivo es opcional.

Tabla IV. LXVII Descripción de la tabla documento_comision

Nombre de la columna	Tipo de dato	Primary Key	Not Null	AutoInc
codigo_comision	int8	SI	SI	SI
físico	bytea	NO	NO	NO
nombre	varchar	NO	NO	NO
tamano	float8	NO	NO	NO
Tipo	varchar	NO	NO	NO

- **Tabla vehículo**

Esta tabla permitirá llevar datos históricos de los vehículos y conductores asignados a cada comisión generada y aprobada.

Tabla IV. LXVIII Descripción de la tabla vehiculo

Nombre de la columna	Tipo de dato	Primary Key	Not Null	AutoInc
código	int8	SI	SI	SI
codigo_vehiculo	int4	NO	NO	NO
nombre_conductor	Varchar	NO	NO	NO
apellido_conductor	Varchar	NO	NO	NO
cedula	Varchar	NO	NO	NO
Placa	Varchar	NO	NO	NO
Marca	Varchar	NO	NO	NO

Motor	Varchar	NO	NO	NO
matricula_fecha_emision	int8	NO	NO	NO
Color	Varchar	NO	NO	NO
Tipo	Varchar	NO	NO	NO
Capacidad	int4	NO	NO	NO
nombre_reemplazo	Varchar	NO	NO	NO
apellido_reemplazo	Varchar	NO	NO	NO
cedula_reemplazo	Varchar	NO	NO	NO
Observación	Varchar	NO	NO	NO

- **Tabla vehiculo_comision**

Esta tabla nos permitirá almacenar la relación entre una comisión y el vehículo que se le asigne.

Tabla IV. LXIX Descripción de la tabla vehiculo_comision

Nombre de la columna	Tipo de dato	Primary Key	Not Null	AutoInc
codigo	int8	SI	SI	SI
codigo_comision	int8	NO	NO	NO
codigo_vehiculo	int8	NO	NO	NO

- **Tabla comisionado**

Esta tabla contendrá los datos de cada uno de los funcionarios de la ESPOCH que salgan en una comisión.

Tabla IV. LXX Descripción de la tabla comisionado

Nombre de la columna	Tipo de dato	Primary Key	Not Null	AutoInc
código	int4	SI	SI	SI
cedula	varchar	NO	NO	NO
nombres	varchar	NO	NO	NO
apellidos	varchar	NO	NO	NO
función	varchar	NO	NO	NO
Sueldo	float8	NO	NO	NO
responsable	Bool	NO	NO	NO
dependencia	Varchar	NO	NO	NO

- **Tabla viatico**

Aquí se almacena las relaciones entre la comisión y los comisionados asignados a ella, así como también la relación con *nivel_zona* en dependencia a la ciudad de destino de la comisión.

Tabla IV. LXXI Descripción de la tabla viatico

Nombre de la columna	Tipo de dato	Primary Key	Not Null	AutoInc
código	int8	SI	SI	SI
codigo_comision	int8	NO	NO	NO
codigo_comisionado	int4	NO	NO	NO
codigo_nivel_zona	int4	NO	NO	NO
valor_calculado	float8	NO	NO	NO

responsable	Bool	NO	NO	NO
fecha_emision	int8	NO	NO	NO
codificación	varchar	NO	NO	NO
funcion_comisionado	varchar	NO	NO	NO
dependencia_comisionado	varchar	NO	NO	NO

• **Tabla detalle salvoconducto**

Esta tabla permitirá almacenar datos relevantes del salvoconducto como se describe a continuación.

Tabla IV. LXXII Descripción de la tabla detalle_salvoconducto

Nombre de la columna	Tipo de dato	Primary Key	Not Null	AutoInc
código	int4	SI	SI	SI
nombre_responsable	varchar	NO	NO	NO
titulo_responsable	varchar	NO	NO	NO
cargo_responsable	varchar	NO	NO	NO
departamento_responsable	varchar	NO	NO	NO
descripcion_departamento	varchar	NO	NO	NO
estado	int4	NO	NO	NO

Al igual que en otras tabla el atributo *estado* permitirá saber si está activo o inactivo el detalle del salvoconducto

Tabla IV. LXXIII Estados de la tabla detalle_salvoconducto

Estado	Descripción
0	Detalle salvoconducto Inactivo.
1	Detalle salvoconducto Activo.

- **Tabla salvoconducto**

Esta tabla permitirá almacenar los salvoconductos aprobados.

Tabla IV. LXXIV Descripción de la tabla salvoconducto

Nombre de la columna	Tipo de dato	Primary Key	Not Null	AutoInc
código	int8	SI	SI	SI
codigo_vehiculo	int8	NO	NO	NO
fecha	int8	NO	NO	NO
hora	int8	NO	NO	NO
motivo	Text	NO	NO	NO
lugar_origen	varchar	NO	NO	NO
lugar_destino	varchar	NO	NO	NO
numero_dias	int4	NO	NO	NO
fecha_inicio	int8	NO	NO	NO
fecha_fin	int8	NO	NO	NO
nombres_conductor	varchar	NO	NO	NO
cedula_conductor	varchar	NO	NO	NO
nombres_conductor_reemplazo	varchar	NO	NO	NO
cedula_conductor_reemplazo	varchar	NO	NO	NO
nombres_servidor	varchar	NO	NO	NO
cedula_servidor	varchar	NO	NO	NO
marca_vehiculo	varchar	NO	NO	NO
numero_vehiculo	int4	NO	NO	NO
motor	varchar	NO	NO	NO

color	varchar	NO	NO	NO
placa	varchar	NO	NO	NO
matricula	int8	NO	NO	NO
observación	Text	NO	NO	NO
estado	int4	NO	NO	NO
codigo_detalle	int4	NO	NO	NO

El atributo *estado* permitirá saber si el salvoconducto fue generado, aprobado o impreso, a continuación mostramos los distintos estados en los que se puede presentar el salvoconducto.

Tabla IV. LXXV Estados de la tabla salvoconducto

Estado	Descripción
1	Salvoconducto pendiente para el Vicerrector de Investigación y Desarrollo.
2	Salvoconducto aprobado por el Vicerrector de Investigación y Desarrollo.
3	Salvoconducto impreso por el Jefe de Movilización.

4.2.3. Fase III: Codificación

Como punto de partida para empezar con el desarrollo de la aplicación, es necesario tener una idea clara de la estructura y como interactuaran los servidores tanto de base de datos, como el web con los clientes, por lo cual se ha diseñado el siguiente diagrama, permitiendo de esta manera enfocarse en la mejor solución.

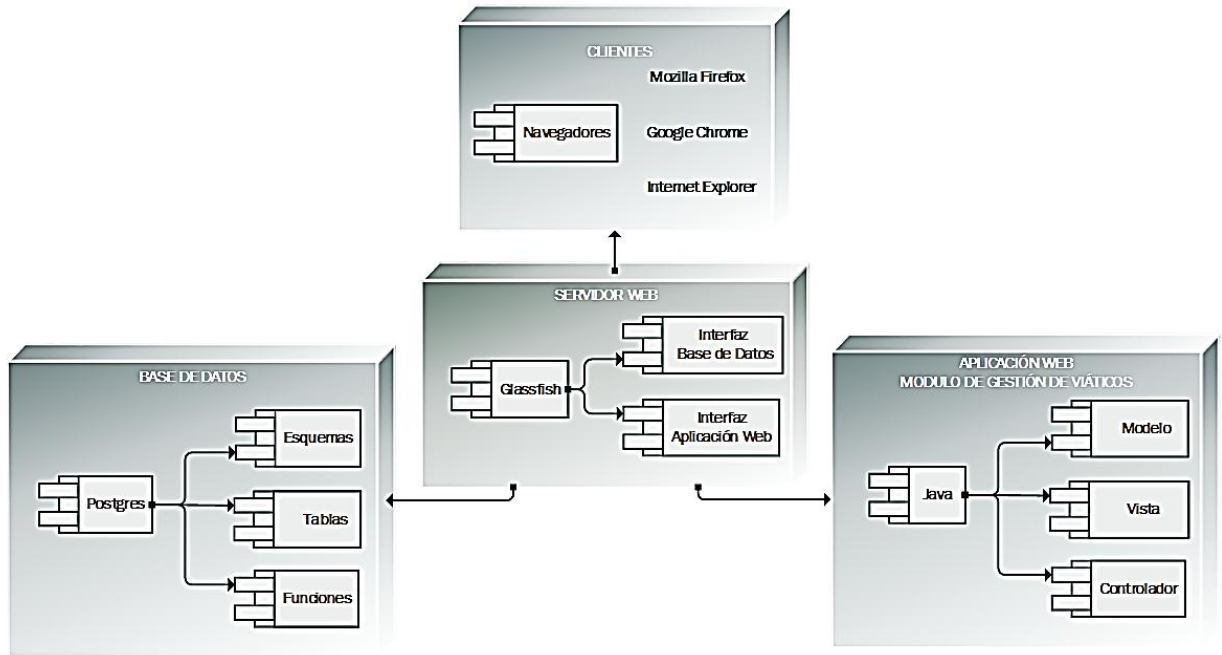


Figura IV. 57 Diagrama de Bloques y Despliegue del Sistema de Gestión de Viáticos

Lenguaje

El sistema de Gestión de Viáticos se desarrolla en lenguaje Java, pues provee de muchas ventajas para cumplir con los objetivos planteados, una de las principales es el uso del framework JSF(Java Server Faces), que es un entorno que permite enriquecer nuestras aplicaciones web, pues cuenta con una gran gama de componentes que hacen que la experiencia del usuario al utilizar una aplicación web sea agradable, otra de las ventajas es que se basa en patrón MVC(Modelo Vista Controlador), que es una arquitectura flexible y que permite independencia entre la presentación y el comportamiento.

Pruebas Unitarias con JSFUnit

Previa la entrega del sistema de Gestión de Viáticos se realizó las pruebas unitarias con la herramienta JSFUnit, la misma que permitió verificar el correcto funcionamiento del sistema, y lo más importante que cumpla con los requerimientos establecidos al inicio del desarrollo de la aplicación obteniendo los siguientes resultados.

En la **Figura 58**, se muestra la ejecución de pruebas unitarias del ingreso, carga y edición de los niveles.

Summary

Tests	Failures	Errors	Success rate	Time
3	0	0	100.00%	27.280

Note: *failures* are anticipated and checked for with assertions while *errors* are unanticipated.

TestCase Test.nivelTest

Name	Status	Type	Time(s)
testInsertarNivel	Success		17.942
testCargarNiveles	Success		4.169
testEditarNivel	Success		5.078

Figura IV. 58 Pruebas Unitarias Nivel

En la **Figura 59**, se muestra la ejecución de pruebas unitarias del ingreso, carga y edición de las zonas.

Summary

Tests	Failures	Errors	Success rate	Time
3	0	0	100.00%	26.753

Note: *failures* are anticipated and checked for with assertions while *errors* are unanticipated.

TestCase Test.zonaTest

Name	Status	Type	Time(s)
testInsertarZona	Success		17.477
testCargarZonas	Success		3.840
testEditarZona	Success		5.344

Figura IV. 59 Pruebas Unitarias Zona

En la **Figura 60**, se muestra la ejecución de pruebas unitarias del ingreso, carga, edición y búsqueda de las ciudades.

Summary

Tests	Failures	Errors	Success rate	Time
4	0	0	100.00%	28.265

Note: *failures* are anticipated and checked for with assertions while *errors* are unanticipated.

TestCase Test.ciudadTest

Name	Status	Type	Time(s)
testInsertarCiudad	Success		17.262
testCargarCiudades	Success		3.757
testEditarCiudad	Success		4.495
testBuscarCiudadPorCodigo	Success		2.659

Figura IV. 60 Pruebas Unitarias Ciudad

En la **Figura 61**, se muestra la ejecución de pruebas unitarias del ingreso, carga, edición de las solicitudes.

Summary

Tests	Failures	Errors	Success rate	Time
3	0	0	100.00%	32.016

Note: *failures* are anticipated and checked for with assertions while *errors* are unanticipated.

TestCase Test.solicitudTest

Name	Status	Type	Time(s)
testInsertarSolicitud	Success		20.248
testCargarSolicitudes	Success		5.795
testEditarSolicitud	Success		5.872

Figura IV. 61 Pruebas Unitarias Solicitud

En la **Figura 62**, se muestra la ejecución de pruebas unitarias de la asignación del vehículo a una o más solicitudes, así como también la aprobación de las mismas y la generación de salvoconductos.

Summary

Tests	Failures	Errors	Success rate	Time
3	0	0	100.00%	29.729

Note: *failures* are anticipated and checked for with assertions while *errors* are unanticipated.

TestCase Test.asignacionVehiculo_y_AprobacionSolicitudTest

Name	Status	Type	Time(s)
testAsignacionVehiculo	Success		18.252
testAprobacionSolicitud	Success		5.097
testGenerarSalvoconducto	Success		6.308

Figura IV. 62 Pruebas Unitarias Asignación Vehículo

En la **Figura 63**, se muestra la ejecución de pruebas unitarias de la carga de solicitudes y salvoconductos.

Summary

Tests	Failures	Errors	Success rate	Time
2	0	0	100.00%	24.098

Note: *failures* are anticipated and checked for with assertions while *errors* are unanticipated.

TestCase Test.cargaSolicitudesYSalvoconductosAprobadosTest

Name	Status	Type	Time(s)
testCargaSolcitiidesAprobadas	Success		18.401
testCargaSalvoconductos	Success		5.605

Figura IV. 63 Pruebas Unitarias Carga Solicitudes y Salvoconductos

4.2.4. Fase IV: Pruebas de Funcionamiento

En esta fase se verifica el correcto funcionamiento del sistema, ingresando los datos necesarios para su utilización.

A continuación se muestran las pruebas realizadas por cada requerimiento.

Tabla IV. LXXVI Prueba de funcionamiento ingreso y modificación de Zona

HISTORIA 1	
Ingreso y Modificación de Zona	
Descripción	El usuario financiero una vez que ingrese al sistema correctamente podrá ingresar y modificar los datos de una zona (nombre, observación).
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none">• El usuario ingresa su clave y contraseña.• En el menú Administración Ciudad, Nivel, Zona selecciona la opción Administración Zona.• Selecciona el botón nuevo o modificar.• Ingresa o modifica los datos de la zona como nombre y observación.• Presiona el botón aceptar.• El proceso de Ingreso o modificación de Zona finaliza.
Resultado Esperado	Luego de ingresar los datos de la zona mostrar un mensaje de notificación del éxito del ingreso o modificación de los mismos.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla IV. LXXVII Prueba de funcionamiento ingreso y modificación de Nivel

HISTORIA 2	
Ingreso y Modificación de Nivel	
Descripción	El usuario financiero una vez que ingrese al sistema correctamente podrá ingresar y modificar los datos de un Nivel (nombre, observación).
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Administración Ciudad, Nivel, Zona selecciona la opción Administración Nivel. • Selecciona el botón nuevo o modificar. • Ingresa o modifica los datos del nivel como nombre y observación. • Presiona el botón aceptar. • El proceso de Ingreso o modificación de Nivel finaliza.
Resultado Esperado	Luego de ingresar los datos del nivel mostrar un mensaje de notificación del éxito del ingreso o modificación del mismo.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla IV. LXXVIII Prueba de funcionamiento ingreso y modificación de Ciudad

HISTORIA 3	
Ingreso y Modificación de Ciudad	
Descripción	El director financiero una vez que ingrese al sistema correctamente podrá ingresar y modificar los datos de una ciudad (nombre, observación).
Condiciones de ejecución	El usuario debe estar registrado en el sistema

Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Administración Ciudad, Nivel, Zona selecciona la opción Administración Ciudad. • Selecciona el botón nuevo o modificar. • Ingresa o modifica los datos de una ciudad como nombre y observación. • Presiona el botón aceptar. • El proceso de Ingreso y modificación de Ciudad finaliza.
Resultado Esperado	Luego de ingresar o modificar los datos de una ciudad mostrar un mensaje de notificación del éxito del ingreso o modificación de la misma.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla IV. LXXIX Prueba de funcionamiento relacionar Nivel-Zona

HISTORIA 4	
Relacionar nivel-zona	
Descripción	El usuario financiero una vez que ingrese al sistema correctamente podrá relacionar un nivel con una zona, los niveles y las zonas se seleccionarán desde una lista desplegable, la misma que contendrá el listado de estas, posteriormente presionará un botón aceptar para finalizar la actividad.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Administración Ciudad, Nivel, Zona selecciona la opción Relacionar Nivel-Zona. • Selecciona el botón nuevo. • Selecciona un nivel y una zona. • Presiona el botón aceptar. • El proceso de relacionar un nivel con una zona finaliza.

Resultado Esperado	Luego de seleccionar un nivel y una zona y presionar el botón aceptar mostrar un mensaje de notificación del éxito del ingreso de los datos.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla IV. LXXX Prueba de funcionamiento asignar una Ciudad a una Zona

HISTORIA 5	
Asignar una Ciudad a una Zona	
Descripción	El director financiero una vez que ingrese al sistema correctamente podrá asignar las ciudades con una zona determinada, las zonas y ciudades se seleccionarán desde una lista desplegable, la misma que contendrá el listado de las mismas, posteriormente presionará un botón aceptar para finalizar la actividad.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Administración Ciudad, Nivel, Zona selecciona la opción Asignar Ciudad a Zona. • Selecciona el botón nuevo. • Selecciona una ciudad y una zona. • Presiona el botón aceptar. • El proceso de asignación de ciudades a una zona finaliza.
Resultado Esperado	Luego de seleccionar la ciudad y zona y presionar el botón aceptar mostrar un mensaje de notificación del éxito del ingreso de los datos.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla IV. LXXXI Prueba de funcionamiento reporte Nivel Zona Ciudad

HISTORIA 6	
Reporte Nivel Zona Ciudad	
Descripción	El director financiero una vez que ingresa al sistema correctamente visualizara un reporte en el que se detalla las ciudades que pertenecen a un zona y el valor del viatico de acuerdo a la zona.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú selecciona la opción Administración Ciudad, Nivel, Zona selecciona la opción Reporte Nivel Zona Ciudad. • El proceso de reporte Nivel Zona Ciudad finaliza.
Resultado Esperado	Luego de seleccionar la opción Reporte Nivel Zona Ciudad visualizar un reporte en el que se detalla las ciudades que pertenecen a un zona y el valor del viatico de acuerdo a la zona.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla IV. LXXXII Prueba de funcionamiento creación de Solicitud de Movilización Externa

HISTORIA 7	
Crear Solicitud de Movilización Externa	
Descripción	La secretaria de dependencia podrá ingresar los datos de una solicitud de movilización interna, la misma que contiene los datos de la comisión y de los comisionados. Entre los datos de la comisión tenemos la fecha de emisión, la misma que se carga automáticamente, la ciudad de destino que se seleccionará de una lista desplegable, la fecha salida y retorno que será seleccionada de un calendario respectivamente, la hora inicio y fin de la actividad, hora de la salida y retorno, empresa y motivo de la comisión, además tendrá una opción que permite cargar un archivo pdf el mismo que contendrá las actividades a realizarse en esta comisión,

	<p>siendo este último de carácter opcional ya que posee un campo que le permitirá digitar las actividades a realizarse.</p> <p>Los datos de los comisionados se obtienen ingresando el número de cédula, presionando el botón buscar y se cargarán los datos del comisionado en una tabla en la cual debe asignarse un nivel, este nivel será seleccionado desde una lista desplegable, finalmente se guarda la solicitud.</p>
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Crear Solicitudes selecciona la opción Crear Solicitud Externa. • Selecciona el botón nuevo. • Ingresa las actividades, datos de la comisión, datos del comisionado. • Presiona el botón Guardar. • El proceso de Crear Solicitud de Movilización Externa finaliza.
Resultado Esperado	Luego de ingresar los datos solicitados y presionar el botón Guardar mostrar un mensaje de notificación del éxito del ingreso de los datos.
Evaluación de la prueba	<p><input type="checkbox"/> Mala</p> <p><input type="checkbox"/> Aceptable</p> <p><input checked="" type="checkbox"/> Satisfactoria</p>

Tabla IV. LXXXIII Prueba de funcionamiento creación de Solicitud de Movilización Interna

HISTORIA 8	
Crear Solicitud de Movilización Interna	
Descripción	La secretaria de dependencia podrá ingresar los datos de una solicitud de movilización interna, la misma que contiene los datos de la comisión y de los comisionados. Entre los datos de la comisión tenemos la fecha de emisión, la misma que se carga automáticamente, la ciudad de destino que se seleccionará de una lista desplegable, la fecha salida y retorno que será seleccionada de un calendario respectivamente, la hora inicio y fin de la actividad, hora de la salida y retorno, empresa y motivo de la comisión, además

	<p>tendrá una opción que permite cargar un archivo pdf el mismo que contendrá las actividades a realizarse en esta comisión, siendo este último de carácter opcional ya que posee un campo que le permitirá digitar las actividades a realizarse.</p> <p>Los datos de los comisionados se obtienen ingresando el número de cédula, presionando el botón buscar y se cargarán los datos del comisionado en una tabla en la cual debe asignarse un nivel, este nivel será seleccionado desde una lista desplegable, finalmente se guarda la solicitud.</p>
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Crear Solicitudes selecciona la opción Crear Solicitud Interna. • Selecciona el botón nuevo. • Ingresa las actividades, datos de la comisión, datos del comisionado. • Presiona el botón Guardar. • El proceso de Crear Solicitud de Movilización Interna finaliza.
Resultado Esperado	Luego de ingresar los datos solicitados y presionar el botón Guardar mostrar un mensaje de notificación del éxito del ingreso de los datos.
Evaluación de la prueba	<p><input type="checkbox"/> Mala</p> <p><input type="checkbox"/> Aceptable</p> <p><input checked="" type="checkbox"/> Satisfactoria</p>

Tabla VI. LXXXIV Prueba de funcionamiento rechazar o anular una Solicitud de Movilización Interna

HISTORIA 9	
Rechazar o Anular una Solicitud de Movilización Interna	
Descripción	La secretaria de dependencia visualiza un listado de solicitudes internas ingresadas con estado pendiente, permitiendo seleccionar una de estas, ver su detalle (datos solicitud de movilización) de la misma y cambiar el estado de pendiente ha anulado o rechazado.

Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Crear Solicitudes selecciona la opción Crear Solicitud Interna. • Selecciona una solicitud. • Clic en el botón Revisar. • Ir a la pestaña Visto bueno. • Seleccionar la opción Rechazado / Anulado. • Presiona el botón guardar. • El proceso de Rechazar o Anular una Solicitud de Movilización Interna.
Resultado Esperado	Luego de cambiar el estado de la solicitud y presionar el botón Guardar mostrar un mensaje de notificación del éxito de la modificación del estado de la solicitud.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. LXXXV Prueba de funcionamiento rechazar o anular una Solicitud de Movilización Externa

HISTORIA 10	
Rechazar o Anular una Solicitud de Movilización Externa	
Descripción	La secretaria de dependencia visualiza un listado de solicitudes externas ingresadas con estado pendiente, permitiendo seleccionar una de estas, ver su detalle (datos solicitud de movilización) de la misma y cambiar el estado de pendiente ha anulado o rechazado.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Crear Solicitudes selecciona la opción Crear Solicitud Externa. • Selecciona una solicitud. • Clic en el botón Revisar. • Ir a la pestaña Visto bueno.

	<ul style="list-style-type: none"> • Seleccionar la opción Rechazado / Anulado. • Presiona el botón guardar. • El proceso de Rechazar o Anular una Solicitud de Movilización Externa finaliza.
Resultado Esperado	Luego de cambiar el estado de la solicitud y presionar el botón Guardar mostrar un mensaje de notificación del éxito de la modificación del estado de la solicitud.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. LXXXVI Prueba de funcionamiento aprobar, rechazar o anular una Solicitud de Movilización Interna

HISTORIA 11	
Aprobar, Rechazar o Anular una Solicitud de Movilización Interna	
Descripción	El jefe de dependencia visualiza un listado de solicitudes de movilización internas pendientes, permitiendo seleccionar una de estas, ver su detalle (datos solicitud de movilización) de la misma y cambiar el estado de pendiente ha aprobado, anulado o rechazado.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Revisar Solicitudes selecciona la opción Revisar Solicitud Interna Pendiente. • Selecciona una solicitud. • Clic en el botón Revisar. • Ir a la pestaña Visto Bueno. • Seleccionar la opción Aprobado/ Rechazado / Anulado. • Presiona el botón Guardar. • El proceso de Aprobar, Rechazar o Anular una Solicitud de Movilización Interna finaliza.
Resultado Esperado	Luego de cambiar el estado de la solicitud y presionar el botón Guardar mostrar un mensaje de notificación del éxito de la modificación del estado de la solicitud.




Evaluación de la prueba	 Mala  Aceptable  Satisfactoria
-------------------------	--

Tabla VI. LXXXVII Prueba de funcionamiento aprobar, rechazar o anular una Solicitud de Movilización Externa




HISTORIA 12 Aprobar, Rechazar o Anular una Solicitud de Movilización Externa	
Descripción	El jefe de dependencia visualiza un listado de solicitudes de movilización externas pendientes, permitiendo seleccionar una de estas, ver su detalle (datos solicitud de movilización) de la misma y cambiar el estado de pendiente ha aprobado, anulado o rechazado.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Revisar Solicitudes selecciona la opción Revisar Solicitud Externa Pendiente. • Selecciona una solicitud. • Clic en el botón Revisar. • Ir a la pestaña Visto Bueno. • Seleccionar la opción Aprobado/ Rechazado / Anulado. • Presiona el botón Guardar. • El proceso de Aprobar, Rechazar o Anular una Solicitud de Movilización Externa finaliza.
Resultado Esperado	Luego de cambiar el estado de la solicitud y presionar el botón Guardar mostrar un mensaje de notificación del éxito de la modificación del estado de la solicitud.
Evaluación de la prueba	 Mala  Aceptable  Satisfactoria

Tabla VI. LXXXVIII Prueba de funcionamiento listar Solicitudes de Movilización Internas aprobadas

HISTORIA 13	
Listar Solicitudes de Movilización Internas Aprobadas	
Descripción	La secretaria y jefe de dependencia pueden visualizar la lista de solicitudes de movilización internas aprobadas por el Jefe de dependencia y el detalle de cada una de estas.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Solicitudes Aprobadas Jefe Dependencia selecciona la opción Solicitud Interna Aprobada. • Selecciona una solicitud. • Clic en el botón Revisar. • Presiona el botón Aceptar. • El proceso de Listar Solicitudes de Movilización Internas Aprobadas finaliza.
Resultado Esperado	Luego presionar el botón Revisar, visualizar el detalle de información de la Solicitud de Movilización seleccionada.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. LXXXIX Prueba de funcionamiento listar Solicitudes de Movilización Externas aprobadas

HISTORIA 14	
Listar Solicitudes de Movilización Externas Aprobadas	
Descripción	La secretaria y jefe de dependencia pueden visualizar la lista de solicitudes de movilización externas aprobadas por el Jefe de dependencia y el detalle de cada una de estas.
Condiciones de ejecución	El usuario debe estar registrado en el sistema

Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Solicitudes Aprobadas Jefe Dependencia selecciona la opción Solicitud Externa Aprobada. • Selecciona una solicitud. • Clic en el botón Revisar. • Presiona el botón Aceptar. • El proceso de Listar Solicitudes de Movilización Externas Aprobadas finaliza.
Resultado Esperado	Luego presionar el botón Revisar, visualizar el detalle de información de la Solicitud de Movilización seleccionada.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. XC Prueba de funcionamiento editar Solicitud de Movilización Interna Rechazada

HISTORIA 15	
Editar Solicitudes de Movilización Internas Rechazadas	
Descripción	La secretaria de dependencia puede visualizar la lista de solicitudes de movilización internas rechazadas por la secretaria o jefe de dependencia, el detalle de cada una de estas permitiendo editar sus datos.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Modificar Solicitudes selecciona la opción Solicitud Interna Rechazada. • Selecciona una solicitud. • Clic en el botón Editar. • Modificar datos de la solicitud. • Presiona el botón Modificar. • El proceso de Editar Solicitud de Movilización Internas Rechazadas finaliza.

Resultado Esperado	Luego presionar el botón Modificar, mostrar un mensaje de notificación del éxito de la modificación de los datos de la solicitud.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. XCI Prueba de funcionamiento editar Solicitudes de Movilización Externas rechazadas

HISTORIA 16	
Editar Solicitudes de Movilización Externas Rechazadas	
Descripción	La secretaria de dependencia puede visualizar la lista de solicitudes de movilización externas rechazadas por la secretaria o jefe de dependencia, el detalle de cada una de estas permitiendo editar sus datos.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Modificar Solicitudes selecciona la opción Solicitud Externa Rechazada. • Selecciona una solicitud. • Clic en el botón Editar. • Modificar datos de la solicitud. • Presiona el botón Modificar. • El proceso de Editar Solicitud de Movilización Externas Rechazadas finaliza.
Resultado Esperado	Luego presionar el botón Modificar, mostrar un mensaje de notificación del éxito de la modificación de los datos de la solicitud.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. XCII Prueba de funcionamiento listar Solicitudes de Movilización Internas anuladas

HISTORIA 17	
Listar Solicitudes de Movilización Internas Anuladas	
Descripción	La secretaria y jefe de dependencia pueden visualizar la lista de solicitudes de movilización internas anuladas por la secretaria o jefe de dependencia y el detalle de cada una de estas.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Solicitudes Anuladas Jefe Dependencia selecciona la opción Solicitud Interna Anulada. • Selecciona una solicitud. • Clic en el botón Revisar. • Presiona el botón Aceptar. • El proceso de Listar Solicitudes de Movilización Internas Anuladas finaliza.
Resultado Esperado	Luego presionar el botón Revisar, visualizar el detalle de información de la Solicitud de Movilización seleccionada.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. XCIII Prueba de funcionamiento listar Solicitudes de Movilización Externas anuladas

HISTORIA 18	
Listar Solicitudes de Movilización Externas Anuladas	
Descripción	La secretaria y jefe de dependencia pueden visualizar la lista de solicitudes de movilización externas anuladas por la secretaria o Jefe de dependencia y el detalle de cada una de estas.
Condiciones de ejecución	El usuario debe estar registrado en el sistema

Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Solicitudes Anuladas Jefe Dependencia selecciona la opción Solicitud Externa Anulada. • Selecciona una solicitud. • Clic en el botón Revisar. • Presiona el botón Aceptar. • El proceso de Listar Solicitudes de Movilización Externas Anuladas finaliza.
Resultado Esperado	Luego presionar el botón Revisar, visualizar el detalle de información de la Solicitud de Movilización seleccionada.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. XCIV Prueba de funcionamiento editar, aprobar o anular una Solicitud de Movilización Interna

HISTORIA 19	
Editar, Aprobar o Anular una Solicitud de Movilización Interna.	
Descripción	La secretaria de Investigación y Desarrollo visualiza un listado de solicitudes de movilización internas pendientes, permitiendo seleccionar una de estas, ver su detalle (datos solicitud de movilización), editar la hora de salida y retorno de la comisión y aprobarlo o anularlo según su criterio.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Revisar Solicitudes Pendientes selecciona la opción Solicitud Interna Pendiente. • Selecciona una solicitud. • Clic en el botón Revisar. • Modificar la hora de salida y retorno de la comisión. • Presiona el botón Aprobar / Anular. • El proceso de Editar, Aprobar o Anular una Solicitud de Movilización Interna finaliza.

Resultado Esperado	Luego presionar el botón Aprobar, mostrar un mensaje de notificación del éxito de la modificación de los datos de la solicitud y aprobación de la misma.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. XCV Prueba de funcionamiento editar, aprobar o anular una Solicitud de Movilización Externa

HISTORIA 20	
Editar, Aprobar o Anular una Solicitud de Movilización Externa.	
Descripción	La secretaria de Investigación y Desarrollo visualiza un listado de solicitudes de movilización internas pendientes, permitiendo seleccionar una de estas, ver su detalle, editar la hora de salida y retorno de la comisión y aprobarlo o anularlo según su criterio.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Revisar Solicitudes Pendientes selecciona la opción Solicitud Externa Pendiente. • Selecciona una solicitud. • Clic en el botón Revisar. • Modificar la hora de salida y retorno de la comisión. • Presiona el botón Aprobar / Anular. • El proceso de Editar, Aprobar o Anular una Solicitud de Movilización Externa finaliza.
Resultado Esperado	Luego presionar el botón Aprobar, mostrar un mensaje de notificación del éxito de la modificación de los datos de la solicitud y aprobación de la misma.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. XCVI Prueba de funcionamiento asignar vehículo a Solicitudes de Movilización Internas

HISTORIA 21	
Asignar Vehículo a Solicitudes de Movilización Internas.	
Descripción	La secretaria de Investigación y Desarrollo visualiza un listado de solicitudes de movilización internas pendientes, permitiendo seleccionar una de estas, y agregar a una nueva lista para poder asignar un vehículo, estas comisiones tendrán el mismo lugar de destino y la misma fecha y hora de salida y retorno, para la asignación de un vehículo se presenta un listado de vehículos disponibles, siendo la placa de estos el filtro de búsqueda, una vez seleccionado el vehiculó se mostraran los siguientes datos: nombre, apellido y cedula del conductor así también la marca, número, fecha matrícula, tipo, placa, motor y color del vehículo.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Asignar Vehículo Solicitudes Internas selecciona la opción Asignar Vehículo. • Selecciona una solicitud. • Clic en el botón Agregar Comisión. • Seleccionar un vehículo. • Ingresar datos del conductor reemplazo en caso de existir. • Presiona el botón Guardar. • El proceso de Asignar Vehículo a Solicitudes de Movilización Internas finaliza.
Resultado Esperado	Luego presionar el botón Guardar, mostrar un mensaje de notificación del éxito de la asignación del vehículo a las comisiones.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. XCVII Prueba de funcionamiento aprobar o anular una Solicitud de Movilización Interna

HISTORIA 22	
Aprobar o Anular una Solicitud de Movilización Interna	
Descripción	El vicerrector de Investigación y Desarrollo visualiza un listado de solicitudes de movilización internas (Salvoconductos) pendientes, permitiendo seleccionar una o varias de estas, y cambiar el estado de pendiente ha aprobado o anulado.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Aprobar Solicitudes selecciona la opción Aprobación Solicitud Interna. • Selecciona una o varias solicitudes. • Presiona el botón Aprobar/Anular. • El proceso de Aprobar o Anular una Solicitud de Movilización Interna finaliza.
Resultado Esperado	Luego de cambiar presionar el botón Aprobar o Anular mostrar un mensaje de notificación del éxito del cambio de estado de la solicitud.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. XCVIII Prueba de funcionamiento aprobar o anular una Solicitudes de Movilización Externa

HISTORIA 23	
Aprobar o Anular una Solicitud de Movilización Externa	
Descripción	El vicerrector de Investigación y Desarrollo visualiza un listado de solicitudes de movilización externas pendientes, permitiendo seleccionar una o varias de estas, y cambiar el estado de pendiente ha aprobado o anulado.

Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Aprobar Solicitudes selecciona la opción Aprobación Solicitud Externa. • Selecciona una o varias solicitudes. • Presiona el botón Aprobar/Anular. • El proceso de Aprobar o Anular una Solicitud de Movilización Externa finaliza.
Resultado Esperado	Luego de cambiar presionar el botón Aprobar o Anular mostrar un mensaje de notificación del éxito del cambio de estado de la solicitud.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. XCIX Prueba de funcionamiento listar Solicitudes e Movilización Internas aprobadas

HISTORIA 24	
Listar Solicitudes de Movilización Internas Aprobadas	
Descripción	La secretaria y vicerrector de Investigación y Desarrollo pueden visualizar la lista de solicitudes de movilización internas aprobadas por el vicerrector de investigación y desarrollo y el detalle de cada una de estas.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Solicitudes Aprobadas VID selecciona la opción Internas Aprobadas. • Selecciona una solicitud. • Clic en el botón Revisar. • Presiona el botón Aceptar. • El proceso de Listar Solicitudes de Movilización Internas Aprobadas finaliza.

Resultado Esperado	Visualizar la lista de Solicitudes de Movilización Internas.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. C Prueba de funcionamiento listar Solicitudes de Movilización Externas aprobadas

HISTORIA 25	
Listar Solicitudes de Movilización Externas Aprobadas	
Descripción	La secretaria y vicerrector de Investigación y Desarrollo pueden visualizar la lista de solicitudes de movilización externas aprobadas por el vicerrector de investigación y desarrollo y el detalle de cada una de estas.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Solicitudes Aprobadas VID selecciona la opción Externas Aprobadas. • Selecciona una solicitud. • Clic en el botón Revisar. • Presiona el botón Aceptar. • El proceso de Listar Solicitudes de Movilización Externas Aprobadas finaliza.
Resultado Esperado	Visualizar la lista de Solicitudes de Movilización Externas.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. CI Prueba de funcionamiento generar Salvoconducto

HISTORIA 26	
Generar Salvoconducto	
Descripción	El jefe de movilización, la secretaria y vicerrector de Investigación y Desarrollo visualizan un listado de solicitudes de movilización internas aprobadas, selecciona una de ellas y genera el salvoconducto el mismo que puede ser impreso inmediatamente.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none">• El usuario ingresa su clave y contraseña.• En el menú Solicitudes Aprobadas VID selecciona la opción Internas Aprobadas.• Selecciona una solicitud.• Clic en el botón Generar Salvoconducto.• El proceso de Generar Salvoconducto finaliza.
Resultado Esperado	Visualizar el Salvoconducto listo para su impresión.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. CII Prueba de funcionamiento listar Solicitudes de movilización Internas anuladas

HISTORIA 27	
Listar Solicitudes de Movilización Internas Anuladas	
Descripción	La secretaria y vicerrector de Investigación y Desarrollo pueden visualizar la lista de solicitudes de movilización internas anuladas por la secretaria o vicerrector de investigación y desarrollo y el detalle de cada una de estas.
Condiciones de ejecución	El usuario debe estar registrado en el sistema

Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Solicitudes Anuladas VID selecciona la opción Internas Anuladas. • Selecciona una solicitud. • Clic en el botón Revisar. • Presiona el botón Aceptar. • El proceso de Listar Solicitudes de Movilización Internas Anuladas finaliza.
Resultado Esperado	Visualizar la lista de Solicitudes de Movilización Internas Anuladas.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. CIII Prueba de funcionamiento listar Solicitudes de Movilización Internas anuladas

HISTORIA 28	
Listar Solicitudes de Movilización Internas Anuladas	
Descripción	La secretaria y vicerrector de Investigación y Desarrollo pueden visualizar la lista de solicitudes de movilización externas anuladas por la secretaria o vicerrector de investigación y desarrollo y el detalle de cada una de estas.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Solicitudes Anuladas VID selecciona la opción Externas Anuladas. • Selecciona una solicitud. • Clic en el botón Revisar. • Presiona el botón Aceptar. • El proceso de Listar Solicitudes de Movilización Externas Anuladas finaliza.
Resultado Esperado	Visualizar la lista de Solicitudes de Movilización Externas Anuladas.

Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria
--------------------------------	--

Tabla VI. CIV Prueba de funcionamiento listar Solicitudes de Movilización Internas Aprobadas dado fechas

HISTORIA 29 Listar Solicitudes de Movilización Internas Aprobadas dado fechas	
Descripción	La secretaria y vicerrector de Investigación y Desarrollo pueden visualizar la lista de solicitudes de movilización internas aprobadas por el vicerrector de investigación y desarrollo y el detalle de cada una de estas en un rango de fechas seleccionado desde un calendario.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Solicitudes Aprobadas por Dependencia selecciona la opción Internas Aprobadas Fechas. • Selecciona la fecha inicio. • Selecciona la fecha fin. • Selecciona una dependencia. • Presionar el botón Buscar. • El proceso de Listar Solicitudes de Movilización Internas Aprobadas Dado Fechas finaliza.
Resultado Esperado	Visualizar la lista de Solicitudes de Movilización Internas Aprobadas en el rango de fechas especificado.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. CV Prueba de funcionamiento listar Solicitudes de Movilización Externas aprobadas dado fechas

HISTORIA 30	
Listar Solicitudes de Movilización Externas Aprobadas dado fechas	
Descripción	La secretaria y vicerrector de Investigación y Desarrollo pueden visualizar la lista de solicitudes de movilización externas aprobadas por el vicerrector de investigación y desarrollo y el detalle de cada una de estas en un rango de fechas seleccionado desde un calendario.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Solicitudes Aprobadas por Dependencia selecciona la opción Externas Aprobadas Fechas. • Selecciona la fecha inicio. • Selecciona la fecha fin. • Selecciona una dependencia. • Presionar el botón Buscar. • El proceso de Listar Solicitudes de Movilización Externas Aprobadas Dado Fechas finaliza.
Resultado Esperado	Visualizar la lista de Solicitudes de Movilización Externas Aprobadas en el rango de fechas especificado.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. CVI Prueba de funcionamiento visualizar número de Solicitudes Externas/Internas aprobadas/anuladas

HISTORIA 31	
Visualizar Número de Solicitudes Externas/Internas aprobadas/anuladas	
Descripción	La secretaria y vicerrector de Investigación y Desarrollo pueden visualizar el número de solicitudes aprobadas, anuladas en un rango de fechas.

Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Cantidad Solicitudes Aprobadas-Anuladas selecciona la opción Cantidad Solicitudes. • Selecciona la fecha inicio. • Selecciona la fecha fin. • Presionar el botón Buscar. • El proceso Visualizar Número de Solicitudes finaliza.
Resultado Esperado	Visualizar en número de solicitudes internas y externas aprobadas y anuladas respectivamente en el rango de fechas especificado.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. CVII Prueba de funcionamiento reporte de lugares visitados por un Comisionado

HISTORIA 32	
Reporte de Lugares Visitados por un Comisionado.	
Descripción	La secretaria y vicerrector de Investigación y Desarrollo visualizan un listado de los lugares a los que ha viajado un comisionado y el respectivo detalle de la comisión, dado la cédula del comisionado.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Comisionados selecciona la opción Comisiones por Comisionado. • Ingresar el número de cedula del comisionado. • Presionar el botón Buscar. • El proceso Reporte de Lugares Visitados por un Comisionado finaliza.
Resultado Esperado	Visualizar el listado de los lugares a los que ha viajado un comisionado y ver el detalle de cada uno de estas.

Evaluación de la prueba	<input type="checkbox"/> Mala
	<input type="checkbox"/> Aceptable
	<input checked="" type="checkbox"/> Satisfactoria

Tabla VI. CVIII Prueba de funcionamiento reporte listar comisionados por fechas y dependencia

HISTORIA 33	
Reporte Listar Comisionados por fechas y Dependencia	
Descripción	La secretaria y vicerrector de Investigación y Desarrollo pueden visualizar el listado de comisionados y el detalle de información de la comisión en la que participo dado un rango de fechas y una dependencia específica.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Comisionados selecciona la opción Comisionados por Fecha y Dependencia. • Selecciona la fecha inicio. • Selecciona la fecha fin. • Selecciona una dependencia. • Presionar el botón Buscar. • El proceso Reporte Listar Comisionados por fechas y Dependencia finaliza.
Resultado Esperado	Visualizar el listado de los comisionados y ver el detalle de cada uno de estos.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. CIX Prueba de funcionamiento grafica estadística de los comisionados con más salidas en un rango de fechas

HISTORIA 34	
Gráfica estadística de Comisionados que más han salido en un rango de fechas.	
Descripción	La secretaria y vicerrector de Investigación y Desarrollo pueden visualizar de manera gráfica el comisionado que más ha salido en comisiones en un rango de fechas seleccionado desde un calendario.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Comisionados selecciona la opción Comisionado con Mayor salidas. • Selecciona la fecha inicio. • Selecciona la fecha fin. • Presionar el botón Buscar. • El proceso Gráfica del Comisionado que más ha Salido en un Rango de Fechas finaliza.
Resultado Esperado	Visualizar el listado de los comisionados que tienen mayores salidas.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. CX Prueba de funcionamiento reporte lista de comisionados dado un rango de fechas

HISTORIA 35	
Reporte Lista de Comisionados dado un rango de fechas	
Descripción	La secretaria y vicerrector de Investigación y Desarrollo pueden visualizar de manera gráfica el comisionado que más ha salido en comisiones en un rango de fechas seleccionado desde un calendario.
Condiciones de ejecución	El usuario debe estar registrado en el sistema

Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Comisionados selecciona la opción Comisionados por Fechas. • Selecciona la fecha inicio. • Selecciona la fecha fin. • Presionar el botón Buscar. • El proceso Reporte Lista de Comisionados dado un Rango de Fechas finaliza.
Resultado Esperado	Visualizar el listado de los comisionados.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. CXI Prueba de funcionamiento reporte listado de movimiento de vehículos por fechas

HISTORIA 36	
Reporte Listado de Movimiento de vehículos por fechas.	
Descripción	La secretaria y vicerrector de Investigación y Desarrollo pueden visualizar un listado del movimiento de vehículos dado un rango de fechas.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Movimiento Vehículos selecciona la opción Movimiento Vehículos. • Selecciona la fecha inicio. • Selecciona la fecha fin. • Presionar el botón Buscar. • El proceso Reporte Listado de Movimiento de Vehículos por Fechas finaliza.
Resultado Esperado	Visualizar el listado de los vehículos y ver el detalle de cada uno de estos.

Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria
--------------------------------	--

Tabla VI. CXII Prueba de funcionamiento gráfica de lugares más visitados por los comisionados

HISTORIA 37	
Grafica de lugares más visitados por los comisionados.	
Descripción	La secretaria y vicerrector de Investigación y Desarrollo pueden visualizar de manera gráfica los lugares más visitados por los comisionados.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Lugares Visitados selecciona la opción Lugar Más Visitado. • El proceso Gráfica de lugares más visitados por los comisionados finaliza.
Resultado Esperado	Visualizar gráficamente cual es el lugar más visitado.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. CXIII Prueba de funcionamiento gráfica de lugares más visitados por los comisionados por fechas

HISTORIA 38	
Grafica de lugares más visitados por los comisionados por fechas.	
Descripción	La secretaria y vicerrector de Investigación y Desarrollo pueden visualizar de manera gráfica los lugares más visitados por los comisionados dado un rango de fechas.

Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Lugares Visitados selecciona la opción Lugar Más Visitado Fechas. • Selecciona la fecha inicio. • Selecciona la fecha fin. • Presionar el botón Buscar. • El proceso Grafica de lugares más visitados por los comisionados por fechas finaliza.
Resultado Esperado	Visualizar un listado y gráficamente cual es el lugar más visitado.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. CXIV Prueba de funcionamiento listado de salvoconductos aprobados por fechas

HISTORIA 39	
Listado de salvoconductos aprobados por fechas.	
Descripción	Los usuarios pueden visualizar el listado de salvoconductos aprobados seleccionando el rango de fechas desde un calendario.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su clave y contraseña. • En el menú Salvoconductos selecciona la opción Salvoconductos Aprobados dado Fechas. • Selecciona la fecha inicio. • Selecciona la fecha fin. • Presionar el botón Buscar. • El proceso Listado de salvoconductos aprobados por fechas finaliza.

Resultado Esperado	Visualizar un listado de salvoconductos aprobados.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

Tabla VI. CXV Prueba de funcionamiento visualizar valor estimado de viáticos

HISTORIA 40	
Visualizar valor estimado del viático.	
Descripción	Los usuarios pueden visualizar un el valor estimado de cada viatico y su respectivo detalle dado su número de cedula.
Condiciones de ejecución	El usuario debe estar registrado en el sistema
Entrada	<ul style="list-style-type: none"> • El usuario ingresa su número de cedula. • Presionar el botón Buscar. • El proceso Visualizar valor estimado del viatico as finaliza.
Resultado Esperado	Visualizar el valor estimado de sus viáticos.
Evaluación de la prueba	<input type="checkbox"/> Mala <input type="checkbox"/> Aceptable <input checked="" type="checkbox"/> Satisfactoria

CONCLUSIONES

- Se ha estudiado sobre las diferentes herramientas de BENCHMARKING más utilizadas, pudiendo determinar que Selenium, Cactus y JsfUnit son las que mayores ventajas posee al momento de realizar una prueba unitaria, Selenium se destaca a nivel de interfaz, Cactus y JsfUnit en cambio tiene un gran potencial en una arquitectura MVC.
- Realizando un análisis de las bondades que ofrecen las diferentes herramientas se pudo determinar cómo parámetros para el análisis el tiempo de ejecución de pruebas, la efectividad al realizar las pruebas, las líneas de código escritas para las pruebas, la flexibilidad para realizar cambios, la navegabilidad por la estructura de la aplicación, los mismos que contribuyeron grandemente para determinar cuál es la herramienta de mejor rendimiento para realizar pruebas unitarias en aplicaciones JSF.
- Se desarrolló el sistema de Gestión de Viáticos de la ESPOCH, del cual se escogió un proceso como prototipo para realizar las pruebas unitarias con las herramientas seleccionadas.
- JSFUnit es la más completa para la realización de pruebas unitarias, obteniendo los siguientes resultados: 13,33% en el tiempo de ejecución de pruebas, 6,67% en líneas de código escritas para las pruebas, 20% en la flexibilidad para realizar cambios y 50% en la navegabilidad por la arquitectura de la aplicación, obteniendo un total de 90% de rendimiento frente al 83,33% y 46,67% de las herramientas de software Cactus y Selenium respectivamente.
- Se demostró la hipótesis propuesta indicando que la comparación de herramientas de benchmarking para pruebas unitarias web de aplicaciones JSf permitió escoger la

herramienta JsfUnit que ofrece mejor rendimiento al momento de realizar pruebas unitarias en el sistema de Gestión de Viáticos.

RECOMENDACIONES

- Realizar pruebas unitarias en las aplicaciones web para detectar errores en ejecuciones tempranos y de manera fácil y asegurar la apropiada navegación dentro del sistema, ingreso de datos, procesamiento y recuperación.
- Realizar un caso de prueba por cada requisito para verificar que se cumpla con lo establecido.
- Se recomienda el uso de la herramienta JSFUnit para la realización de pruebas unitarias en aplicaciones web JSF.
- Dedicar más esfuerzo al estudio de herramientas de testing, pues reduce tiempos al momento de probar aplicaciones de gran volumen.

RESUMEN

La finalidad de esta tesis es comparar las herramientas de BENCHMARKING para realizar pruebas unitarias web de aplicaciones Java Server Faces(JSF) aplicadas al Sistema de Gestión de Viáticos de la Escuela Superior Politécnica de Chimborazo desarrollado en el Departamento de Sistemas y Telemática, el mismo que será utilizado por el personal administrativo, docente y de servicio de la institución.

Los métodos utilizados como guía para la presente investigación son el método científico que contempla el planteamiento del problema, levantamiento de información necesaria, análisis e interpretación de resultados para la comprobación de hipótesis, y el método experimental el cual permite desarrollar prototipos para realizar el análisis de rendimiento en las herramientas de software como Selenium, Cactus y JSFUnit.

Con la estadística descriptiva se determinó que la herramienta de software JSFUnit es la más completa para la realización de pruebas unitarias, obteniendo los siguientes resultados: 13,33% en el tiempo de ejecución de pruebas, 6,67% en líneas de código escritas para las pruebas, 20% en la flexibilidad para realizar cambios y 50% en la navegabilidad por la arquitectura de la aplicación, obteniendo un total de 90% de rendimiento frente al 83,33% y 46,67% de las herramientas de software Cactus y Selenium respectivamente.

La realización de pruebas unitarias permite mejorar el desarrollo de aplicaciones web, ya que se reducen los tiempos de depuración y corrección de incidencias.

Se recomienda la utilización de la herramienta JSFUnit para la realización de pruebas unitarias en aplicaciones web Java Server Faces (JSF).

SUMMARY

This paper objective is to compare BENCHMARKING tools in order to make Java Server Faces (JSF) web applications unit testing applied to the travel allowance management system of Escuela Superior Politecnica de Chimborazo developed in the Telematics and System Department which will be used by the administrative, teaching and service staff.

Scientific method was used as a guide for the present research. It considers the problem identifying, information gathering, analysis and interpretation of results, hypothesis testing as experimental method lets develop prototypes to carry out the performance analysis in the software tools such as Selenium, Cactus and JSFUnit respectively.

By descriptive statistics JSFUnit software tool was determined as the whole one to make unit tests, and the following results were gotten: 13, 33% in the test performance period, 6, 67% in code lines written for test, 20% in the flexibility to make changes and 50% in the navigability for the application design. A total of 90% of performance related to 83, 33% and 46, 67% of Cactus and Selenium software tools were gotten.

Unit test carrying out improves the web application development because depuration period and incidences corrections are reduced.

It is recommended to use the JSFUnit tool to carry out Java Server Faces (JSF) web application unit testing.

GLOSARIO

API (Interfaz de programación de aplicaciones).- 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 usados generalmente en las bibliotecas.

APLICACIONES WEB.- aplicación web es aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. En otras palabras, es una aplicación software que se codifica en un lenguaje soportado por los navegadores web en la que se confía la ejecución al navegador.

APPLETS.- es un componente de una *aplicación* que se ejecuta en el contexto de otro programa, por ejemplo un navegador web. El *applet* debe ejecutarse en un *contenedor*, que lo proporciona un programa anfitrión, mediante un *plugin*, o en aplicaciones como teléfonos móviles que soportan el modelo de programación por 'applets'.

DOM (Modelo de Objetos del Documento).- es esencialmente una interfaz de programación de aplicaciones (API) que proporciona un conjunto estándar de objetos para representar documentos HTML y XML.

HISTORIAS DE USUARIO: Una historia de usuario es una representación de un requerimiento de software escrito en una o dos frases utilizando el lenguaje común del usuario.

HTML (Lenguaje de Marcado de Hipertexto).- es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

JAVASCRIPT.- es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

JSF: Es un marco de trabajo de interfaces de usuario del lado del servidor para aplicaciones web.

METODOLOGÍA XP: Es una metodología de desarrollo de la ingeniería de software, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad.

NETBEANS: Es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

PENCIL: Es una herramienta de software libre para crear representaciones de la creación de pantallas.

POSTGRESQL: Es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente.

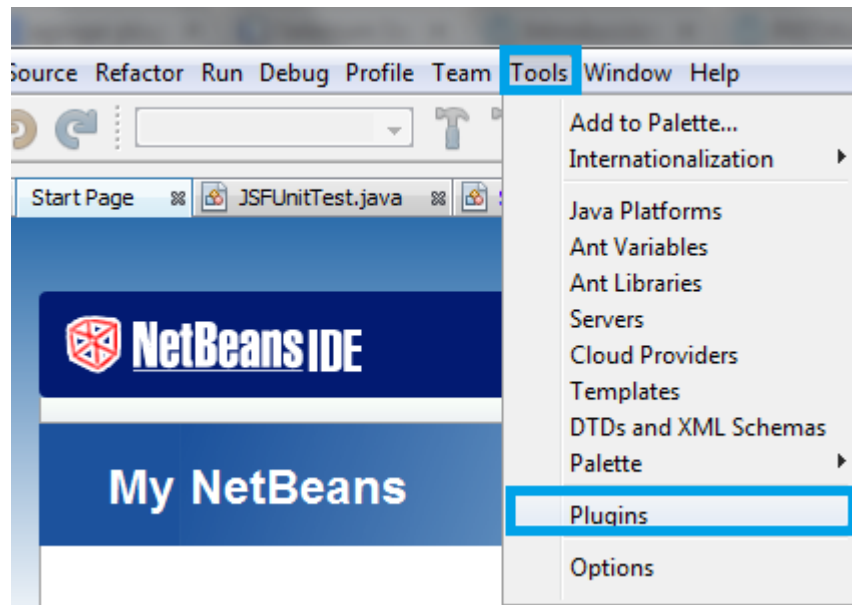
XML (Lenguaje de Marcas Extensible).- es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específico.

ANEXOS

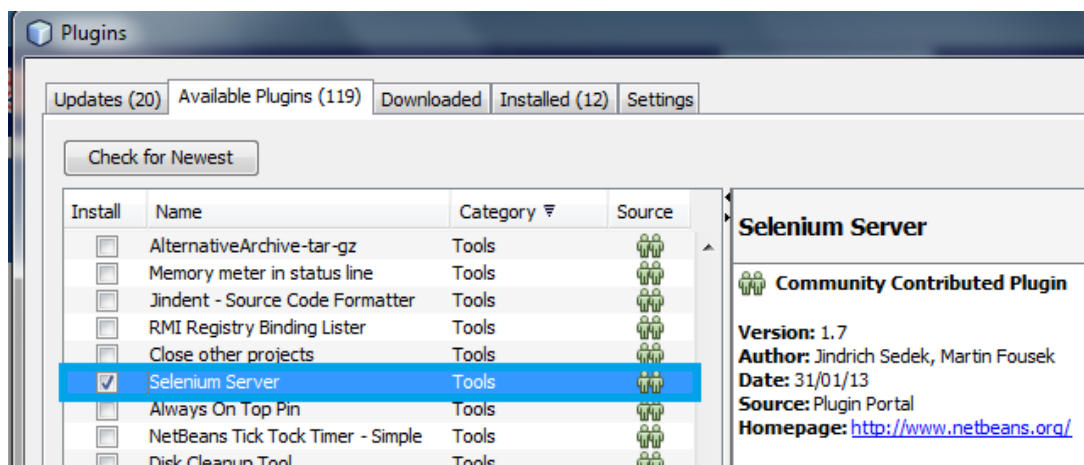
Anexo No 1

Instalación del plugin para Selenium en Netbeans

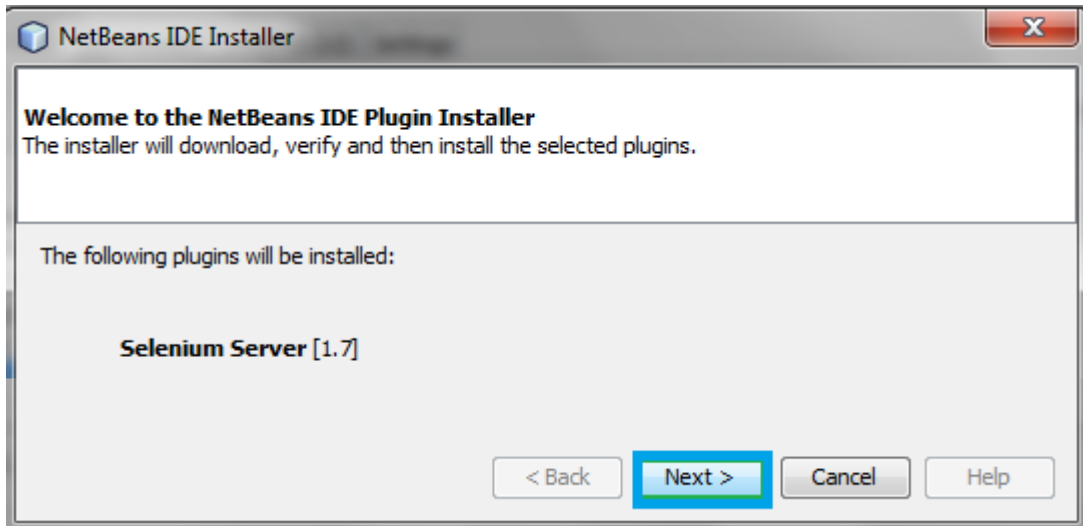
1. Entrar en Netbeans.
2. Luego en el menú “Tools => Plugins”



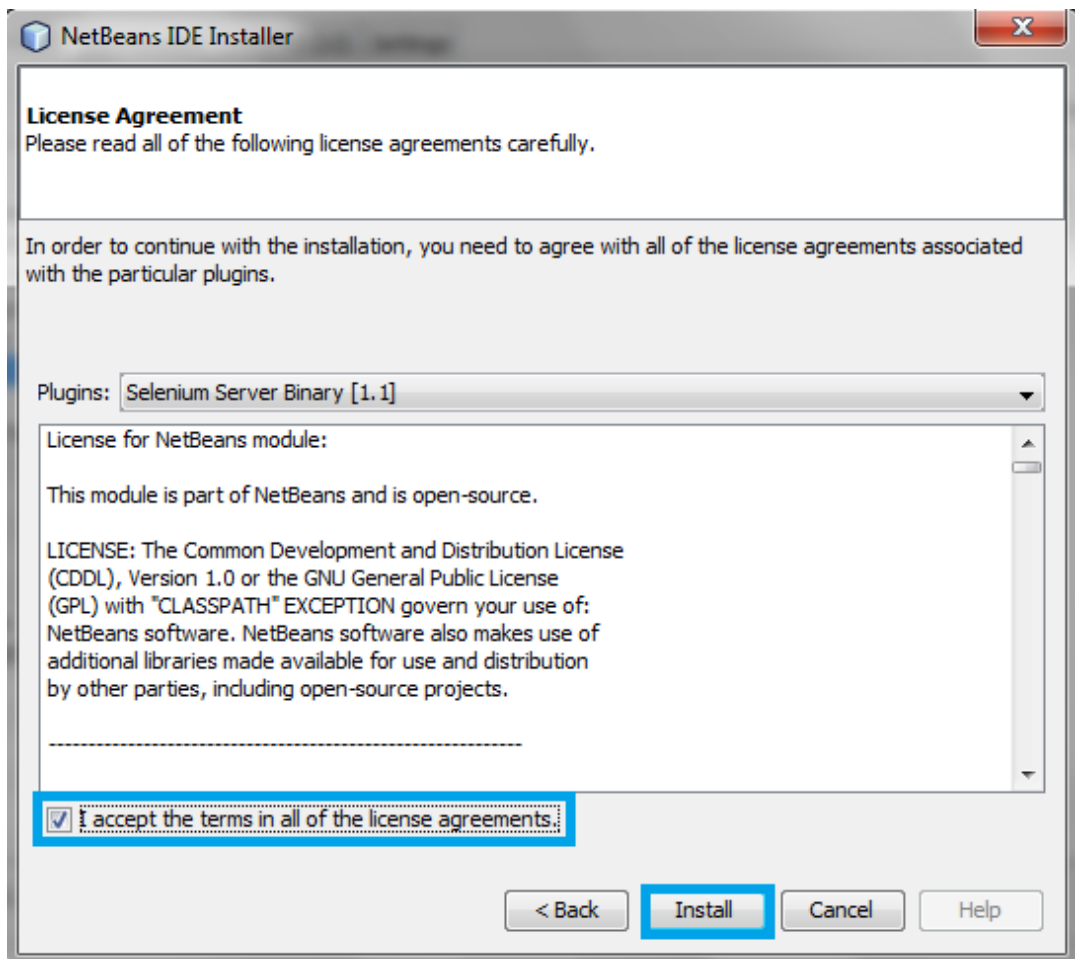
3. Allí buscar la opción *Available Plugins* (Plugins Disponibles), seleccionar Selenium Server y dar clic en *Install*



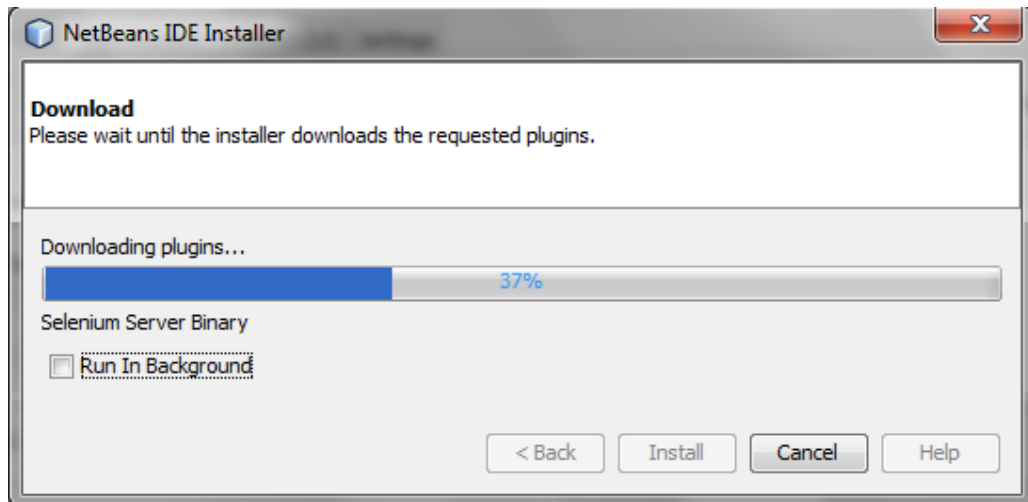
4. Se visualiza la versión del servidor Selenium que se instalara y presione *Next*.



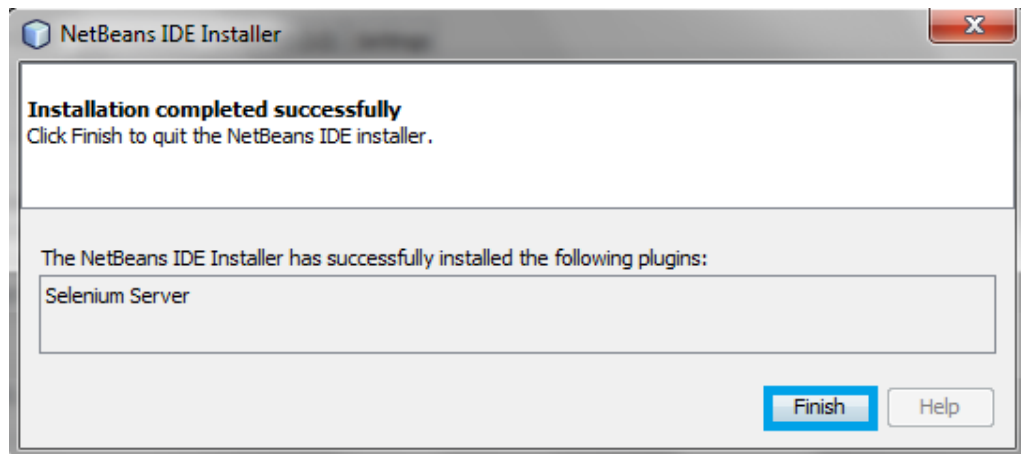
5. Aceptar los acuerdos de licencia y presionar *Install*.



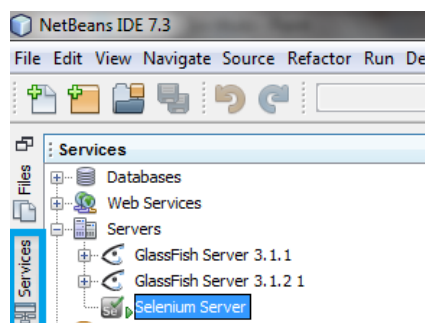
6. Esperar mientras se descarga e instala el plugin Selenium Server.



7. Finalizar la instalación dando clic en el botón *Finish*.



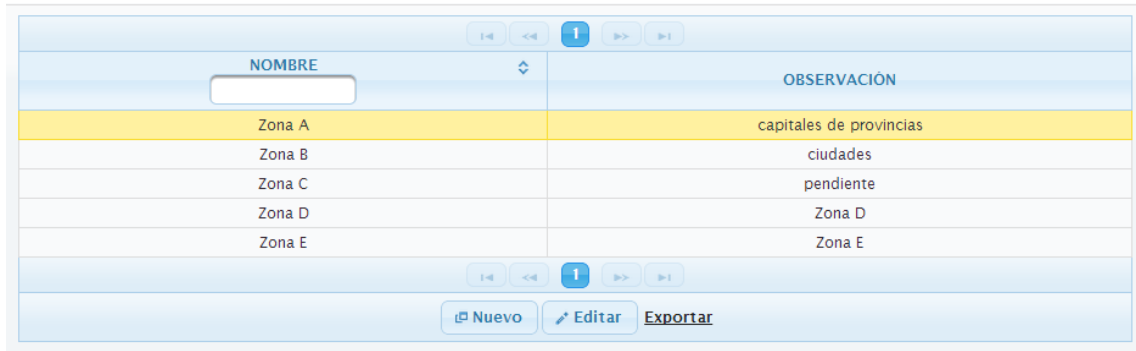
8. Ir a la opción Services => Server y se visualiza el servidor Selenium instalado e inicializado.



Anexo No 2

Realización de pruebas unitarias con la herramienta JSFUnit de los procesos más importantes del Sistema de Gestión de Viáticos de ESPOCH.

1. Módulo de Gestión de Zonas



NOMBRE	OBSERVACIÓN
Zona A	capitales de provincias
Zona B	ciudades
Zona C	pendiente
Zona D	Zona D
Zona E	Zona E

```
import java.io.IOException;
import javax.faces.component.UIComponent;
import static junit.framework.Assert.assertEquals;
import static junit.framework.Assert.assertTrue;
import junit.framework.Test;
import junit.framework.TestSuite;
import org.jboss.jsfunit.jsfession.JSFClientSession;
import org.jboss.jsfunit.jsfession.JSFServerSession;
import org.jboss.jsfunit.jsfession.JSFSession;

public class zonaTest extends org.apache.cactus.ServletTestCase {

    public static Test suite() {
        return new TestSuite(zonaTest.class);
    }

    public void testInsertarZona() throws IOException, Exception {

        JSFSession jsfSession = new JSFSession("/faces/seguridad/financieroSM/zona.xhtml");

        JSFClientSession client = jsfSession.getJSFClientSession();

        JSFServerSession server = jsfSession.getJSFServerSession();

        assertEquals("/seguridad/financieroSM/zona.xhtml", server.getCurrentViewID());

        client.click("btnNuevaZona");

        client.setValue("frmNuevaZona:nombreZona", "Zona");
        client.setValue("frmNuevaZona:observacionZona", "Observacion");
        client.click("guardarZona");

        UIComponent prompt = server.findComponent("mensaje1");
        assertTrue(prompt.isRendered());

        assertEquals("Datos Insertados", server.getManagedBeanValue("#{zonaC.mensajeAlerta}"));
    }
}
```

```

public void testCargarZonas() throws IOException, Exception {

    JSFSession jsfSession = new JSFSession("/faces/seguridad/financieroSM/zona.xhtml");

    JSFClientSession client = jsfSession.getJSFClientSession();

    JSFServerSession server = jsfSession.getJSFServerSession();

    assertEquals("/seguridad/financieroSM/zona.xhtml", server.getCurrentViewID());

    UIComponent prompt = server.findComponent("tblZona");
    assertTrue(prompt.isRendered());

    assertEquals("Zonas Cargadas", server.getManagedBeanValue("#{zonaC.mensajeAlerta}"));

}

public void testEditarZona() throws IOException, Exception {

    JSFSession jsfSession = new JSFSession("/faces/seguridad/financieroSM/zona.xhtml");

    JSFClientSession client = jsfSession.getJSFClientSession();

    JSFServerSession server = jsfSession.getJSFServerSession();

    assertEquals("/seguridad/financieroSM/zona.xhtml", server.getCurrentViewID());

    client.click("btnEditarZona");

    String nombreZonaSeleccionada =
    (String) server.getManagedBeanValue("#{zonaC.objZonaSel.nombre}");

    String observacionZonaSeleccionada =
    (String) server.getManagedBeanValue("#{zonaC.objZonaSel.observacion}");

    client.setValue("frmActualizarZona:nombreZonaModificar", nombreZonaSeleccionada);
    client.setValue("frmActualizarZona:observacionZonaModificar", observacionZonaSeleccionada);

    client.click("modificarZona");

    UIComponent prompt = server.findComponent("mensaje1");
    assertTrue(prompt.isRendered());

    assertEquals("Datos Modificados", server.getManagedBeanValue("#{zonaC.mensajeAlerta}"));

}
}

```

2. Módulo de Gestión de Niveles

Nombre	Observacion
Nivel 1	Nivel 1
Nivel 2	Nivel 2
Nivel 3	Nivel 3
Nivel 4	Nivel 4
Nivel 5	Nivel 5
Nivel 6	Nivel 6
Nivel 7	Nivel 7
Nivel 8	Nivel 8

```

import java.io.IOException;
import javax.faces.component.UIComponent;
import static junit.framework.Assert.assertEquals;
import static junit.framework.Assert.assertTrue;
import junit.framework.Test;
import junit.framework.TestSuite;
import org.jboss.jsfunit.jsfession.JSFClientSession;
import org.jboss.jsfunit.jsfession.JSFServerSession;
import org.jboss.jsfunit.jsfession.JSFSession;

public class nivelTest extends org.apache.cactus.ServletTestCase {

    public static Test suite() {
        return new TestSuite(nivelTest.class);
    }
    public void testInsertarNivel() throws IOException, Exception {

        JSFSession jsfSession = new JSFSession("/faces/seguridad/financieroSM/nivel.xhtml");

        JSFClientSession client = jsfSession.getJSFClientSession();

        JSFServerSession server = jsfSession.getJSFServerSession();

        assertEquals("/seguridad/financieroSM/nivel.xhtml", server.getCurrentViewID());

        // Emula un click sobre el boton con id = btnNuevaCiudad
        client.click("btnNuevaNivel");

        client.setValue("frmNuevoNivel:nombreNivel", "Nivel");
        client.setValue("frmNuevoNivel:observacionNivel", "Observacion");
        client.click("guardarNivel");

        UIComponent prompt = server.findComponent("mensaje1");
        assertTrue(prompt.isRendered());

        assertEquals("Datos Insertados", server.getManagedBeanValue("#{nivelC.mensajeAlerta}"));
    }

    public void testCargarNiveles() throws IOException, Exception {

        JSFSession jsfSession = new JSFSession("/faces/seguridad/financieroSM/nivel.xhtml");

        JSFClientSession client = jsfSession.getJSFClientSession();

        JSFServerSession server = jsfSession.getJSFServerSession();

        assertEquals("/seguridad/financieroSM/nivel.xhtml", server.getCurrentViewID());

        UIComponent prompt = server.findComponent("tblNivel");
        assertTrue(prompt.isRendered());

        assertEquals("Niveles Cargados", server.getManagedBeanValue("#{nivelC.mensajeAlerta}"));
    }
}

```



```

public void testEditarNivel() throws IOException, Exception {

    JSFSession jsfSession = new JSFSession("/faces/seguridad/financieroSM/nivel.xhtml");

    JSFClientSession client = jsfSession.getJSFClientSession();

    JSFServerSession server = jsfSession.getJSFServerSession();

    assertEquals("/seguridad/financieroSM/nivel.xhtml", server.getCurrentViewID());

    client.click("btnEditarNivel");

    String nombreNivelSeleccionada =
        (String) server.getManagedBeanValue("#{nivelC.objNivelSel.nombre}");
    String observacionNivelSeleccionada =
        (String) server.getManagedBeanValue("#{nivelC.objNivelSel.observacion}");

    client.setValue("frmActualizarNivel:nombreNivelModificar",
        nombreNivelSeleccionada);
    client.setValue("frmActualizarNivel:observacionNivelModificar",
        observacionNivelSeleccionada);

    client.click("btnModificarNivel");

    UIComponent prompt = server.findComponent("mensaje1");
    assertTrue(prompt.isRendered());

    assertEquals("Datos Modificados", server.getManagedBeanValue("#{nivelC.mensajeAlerta}"));
}

}

public void testEditarSolicitudInterna() throws IOException, Exception {

    JSFSession jsfSession =
new JSFSession("/faces/seguridad/secretariaDependenciaSM/crearSolicitudInterna.xhtml");

    JSFClientSession client = jsfSession.getJSFClientSession();

    JSFServerSession server = jsfSession.getJSFServerSession();

    assertEquals("/seguridad/secretariaDependenciaSM/crearSolicitudInterna.xhtml",
        server.getCurrentViewID());

    client.click("btnEditarSolicitud");

    client.setValue("frmEditarSolicitud:actividades", "ACTIVIDADES COMISION");

    client.setValue("frmEditarSolicitud:hsalida", "2");
    client.setValue("frmEditarSolicitud:hlllegada", "2");
    client.setValue("frmEditarSolicitud:empresa", "empresa");
    client.setValue("frmEditarSolicitud:motivo", "MOTIVO");

    client.click("btnModificar");

    UIComponent prompt = server.findComponent("mensaje1");
    assertTrue(prompt.isRendered());

    assertEquals("Datos MODificados",
        server.getManagedBeanValue("#{solicitudInternaC.mensajeAlerta}"));
}

}

```

3. Módulo de Gestión de Ciudades

NOMBRE	OBSERVACION
Ambato	Capital de Tungurahua
Cuenca	Capital de Azuay
Santo Domingo	ciudad
Riobamba	Capital de Chimborazo
Quito	Capital de Pichincha
Guayaquil	Capital del Guayas

```
import java.io.IOException;
import javax.faces.component.UIComponent;
import static junit.framework.Assert.assertEquals;
import static junit.framework.Assert.assertTrue;
import junit.framework.Test;
import junit.framework.TestSuite;
import org.jboss.jsfunit.jsfession.JSFClientSession;
import org.jboss.jsfunit.jsfession.JSFServerSession;
import org.jboss.jsfunit.jsfession.JSFSession;
|
public class ciudadZonaTest extends org.apache.cactus.ServletTestCase {

    public static Test suite() {
        return new TestSuite(ciudadZonaTest.class);
    }
    public void testInsertarCiudad() throws IOException, Exception {

        JSFSession jsfSession = new JSFSession("/faces/seguridad/financieroSM/ciudad.xhtml");

        JSFClientSession client = jsfSession.getJSFClientSession();

        JSFServerSession server = jsfSession.getJSFServerSession();

        assertEquals("/seguridad/financieroSM/ciudad.xhtml", server.getCurrentViewID());

        client.click("btnNuevaCiudad");

        client.setValue("frmNuevaCiudad:nombreCiudad", "Ciudad");
        client.setValue("frmNuevaCiudad:observacionCiudad", "Observacion");
        client.click("guardarCiudad");

        UIComponent prompt = server.findComponent("mensaje1");
        assertTrue(prompt.isRendered());

        assertEquals("Datos Insertados", server.getManagedBeanValue("#{ciudadC.mensajeAlerta}"));
    }
}
```

```

public void testCargarCiudades() throws IOException, Exception {

    JSFSession jsfSession = new JSFSession("/faces/seguridad/financieroSM/ciudad.xhtml");

    JSFClientSession client = jsfSession.getJSFClientSession();

    JSFServerSession server = jsfSession.getJSFServerSession();

    assertEquals("/seguridad/financieroSM/ciudad.xhtml", server.getCurrentViewID());

    UIComponent prompt = server.findComponent("tblCiudad");
    assertTrue(prompt.isRendered());

    assertEquals("Ciudades Cargadas", server.getManagedBeanValue("#{ciudadC.mensajeAlerta}"));

}

public void testEditarCiudad() throws IOException, Exception {

    JSFSession jsfSession = new JSFSession("/faces/seguridad/financieroSM/ciudad.xhtml");

    JSFClientSession client = jsfSession.getJSFClientSession();

    JSFServerSession server = jsfSession.getJSFServerSession();

    assertEquals("/seguridad/financieroSM/ciudad.xhtml", server.getCurrentViewID());

    client.click("btnEditarCiudad");

    String nombreCiudadSeleccionada =
    (String) server.getManagedBeanValue("#{ciudadC.objCiudadSel.nombre}");

    String observacionCiudadSeleccionada =
    (String) server.getManagedBeanValue("#{ciudadC.objCiudadSel.observacion}");

    client.setValue("frmActualizarCiudad:nombreCiudadModificar", nombreCiudadSeleccionada);
    client.setValue("frmActualizarCiudad:observacionCiudadModificar", observacionCiudadSeleccionada);

    client.click("modificarCiudad");

    UIComponent prompt = server.findComponent("mensaje1");
    assertTrue(prompt.isRendered());

    assertEquals("Datos Modificados", server.getManagedBeanValue("#{ciudadC.mensajeAlerta}"));

}

public void testBuscarCiudadPorCodigo() throws IOException, Exception {

    JSFSession jsfSession = new JSFSession("/faces/seguridad/financieroSM/busquedaCiudad.xhtml");

    JSFClientSession client = jsfSession.getJSFClientSession();

    JSFServerSession server = jsfSession.getJSFServerSession();

    assertEquals("/seguridad/financieroSM/busquedaCiudad.xhtml", server.getCurrentViewID());

    client.setValue("frmCiudad:codigoCiudad", "5");

    client.click("buscarCiudad");

    UIComponent prompt = server.findComponent("mensaje1");
    assertTrue(prompt.isRendered());

    assertEquals("Ciudad encontrada", server.getManagedBeanValue("#{ciudadC.mensajeAlerta}"));

}
}

```

4. Módulo de Gestión de Solicitudes

LISTADO DE SOLICITUDES INTERNAS CREADAS							
CÉDULA	APELLIDOS	NOMBRES	FUNCIÓN	CIUDAD	LUGAR	FECHA EMISIÓN	Estado
0604261883	Herrera	Anibal	Tec. Informatico	Quito	jjj	2013-04-22	Pendiente
0604261883	Herrera	Anibal	Tec. Informatico	Quito	U	2013-05-14	Pendiente
0603094517	Palacios Campana	Diego Bernardo	Tecnico Informatico	Quito	sd	2013-06-21	Pendiente
0603094517	Palacios Campana	Diego Bernardo	Tecnico Informatico	Riobamba	sss	2013-06-21	Pendiente
0604261883	Herrera	Anibal	Tec. Informatico	Quito	empresa	2013-06-23	Pendiente

```
import java.io.IOException;
import javax.faces.component.UIComponent;
import static junit.framework.Assert.assertEquals;
import static junit.framework.Assert.assertTrue;
import junit.framework.Test;
import junit.framework.TestSuite;
import org.jboss.jsfunit.jsfsession.JSFClientSession;
import org.jboss.jsfunit.jsfsession.JSFServerSession;
import org.jboss.jsfunit.jsfsession.JSFSession;

public class solicitudTest extends org.apache.cactus.ServletTestCase {

    public static Test suite() {
        return new TestSuite(solicitudTest.class);
    }
}
```

```

public void testInsertarSolicitudInterna() throws IOException, Exception {

    JSFSession jsfSession =
new JSFSession("/faces/seguridad/secretariaDependenciaSM/crearSolicitudInterna.xhtml");

    JSFClientSession client = jsfSession.getJSFClientSession();

    JSFServerSession server = jsfSession.getJSFServerSession();

    assertEquals("/seguridad/secretariaDependenciaSM/crearSolicitudInterna.xhtml",
server.getCurrentViewID());

    client.click("btnNuevaSolicitud");

    client.setValue("frmAgregarSolicitud:actividades", "ACTIVIDADES COMISION");

    client.setValue("frmAgregarSolicitud:cmbCiudad", "2");
    client.setValue("frmAgregarSolicitud:fechaSalida", "2");
    client.setValue("frmAgregarSolicitud:fechaRetorno", "2");
    client.setValue("frmAgregarSolicitud:hinicio", "2");
    client.setValue("frmAgregarSolicitud:hfin", "2");
    client.setValue("frmAgregarSolicitud:hsalida", "2");
    client.setValue("frmAgregarSolicitud:hllegada", "2");
    client.setValue("frmAgregarSolicitud:empresa", "empresa");
    client.setValue("frmAgregarSolicitud:motivo", "MOTIVO");

    client.setValue("frmAgregarSolicitud:txtBuscarR", "0604261883");

    client.click("btnBuscar");

    client.setValue("frmAgregarResponsable:cmbNivelR", "1");

    client.click("btnAgregarResponsable");
    assertEquals("OK", server.getManagedBeanValue("#{solicitudInternaC.responsable}"));

    client.click("btnGuardar");

    UIComponent prompt = server.findComponent("mensaje1");
    assertTrue(prompt.isRendered());

    assertEquals("Datos Insertados",
server.getManagedBeanValue("#{solicitudInternaC.mensajeAlerta}"));
}

```

```

public void testCargarSolicitudesInternasCreadas() throws IOException, Exception {

    JSFSession jsfSession =
new JSFSession("/faces/seguridad/secretariaDependenciaSM/crearSolicitudInterna.xhtml");

    JSFClientSession client = jsfSession.getJSFClientSession();

    JSFServerSession server = jsfSession.getJSFServerSession();

    assertEquals("/seguridad/secretariaDependenciaSM/crearSolicitudInterna.xhtml",
server.getCurrentViewID());

    UIComponent prompt = server.findComponent("mensaje1");
    assertTrue(prompt.isRendered());

    prompt = server.findComponent("tblViatico");
    assertTrue(prompt.isRendered());

    assertEquals("solicitudes internas cargadas",
server.getManagedBeanValue("#{solicitudInternaC.mensajeAlerta}"));
}

```

```

public void testEditarSolicitudInterna() throws IOException, Exception {

    JSFSession jsfSession =
new JSFSession("/faces/seguridad/secretariaDependenciaSM/crearSolicitudInterna.xhtml");

    JSFClientSession client = jsfSession.getJSFClientSession();

    JSFServerSession server = jsfSession.getJSFServerSession();

    assertEquals("/seguridad/secretariaDependenciaSM/crearSolicitudInterna.xhtml",
server.getCurrentViewID());

    client.click("btnEditarSolicitud");

    client.setValue("frmEditarSolicitud:actividades", "ACTIVIDADES COMISION");

    client.setValue("frmEditarSolicitud:hsalida", "2");
    client.setValue("frmEditarSolicitud:hllegada", "2");
    client.setValue("frmEditarSolicitud:empresa", "empresa");
    client.setValue("frmEditarSolicitud:motivo", "MOTIVO");

    client.click("btnModificar");

    UIComponent prompt = server.findComponent("mensaje1");
    assertTrue(prompt.isRendered());

    assertEquals("Datos MODificados",
server.getManagedBeanValue("#{solicitudInternaC.mensajeAlerta}"));

}
}

```

5. Revisión de Solicitudes

CÉDULA	APELLIDOS	NOMBRES	FUNCIÓN	CIUDAD	LUGAR	FECHA EMISIÓN	FECHA SALIDA	FECHA RETORNO	Estado
0604261883	Herrera	Anibal	Tec. Informatico	Ambato	df	2013-04-22	2013-04-30	2013-04-30	Pendiente
0604261883	Herrera	Anibal	Tec. Informatico	Ambato	df	2013-04-22	2013-04-30	2013-04-30	Pendiente

CÉDULA	APELLIDOS	NOMBRES	FUNCIÓN	CIUDAD	LUGAR
No records found.					

Seleccione

Cédula:

Reemplazo:

Observación:

Conductor:

Cédula:

Capacidad: Tipo Vehículo:

Marca: Placa:

Número: Motor:

Matrícula: Color:

```

import java.io.IOException;
import javax.faces.component.UIComponent;
import static junit.framework.Assert.assertEquals;
import static junit.framework.Assert.assertTrue;
import junit.framework.Test;
import junit.framework.TestSuite;
import org.jboss.jsfunit.jsfession.JSFClientSession;
import org.jboss.jsfunit.jsfession.JSFServerSession;
import org.jboss.jsfunit.jsfession.JSFSession;

public class asignacionVehiculo_y_AprobacionSolicitudTest extends org.apache.cactus.ServletTestCase {

    public static Test suite() {
        return new TestSuite(asignacionVehiculo_y_AprobacionSolicitudTest.class);
    }

    public void testAprobacionSolicitud() throws IOException, Exception {

        JSFSession jsfSession =
            new JSFSession("/faces/seguridad/secretariaVidSM/revisarSolicitudInternaPendiente.xhtml");

        JSFClientSession client = jsfSession.getJSFClientSession();

        JSFServerSession server = jsfSession.getJSFServerSession();

        assertEquals("/seguridad/secretariaVidSM/revisarSolicitudInternaPendiente.xhtml",
            server.getCurrentViewID());

        client.click("btnRevisar");

        client.setValue("frmEditarSolicitud:horaSalida", "hora salida");
        client.setValue("frmEditarSolicitud:horaRetorno", "hora retono");

        client.click("btnGuardarCambios");

        assertEquals("Cambios Guardados",
            server.getManagedBeanValue("#{SolicitudInternaC.mensajeAlerta}"));
    }

    public void testAsignacionVehiculo() throws IOException, Exception {

        JSFSession jsfSession =
            new JSFSession("/faces/seguridad/secretariaVidSM/asignarVehiculo.xhtml");

        JSFClientSession client = jsfSession.getJSFClientSession();

        JSFServerSession server = jsfSession.getJSFServerSession();

        assertEquals("/seguridad/secretariaVidSM/asignarVehiculo.xhtml",
            server.getCurrentViewID());

        client.click("btnAgregarComision");

        client.setValue("frmAsiganraVehiculo:cmbVehiculos", "1");

        client.click("btnGuardar");

        UIComponent prompt = server.findComponent("mensaje1");
        assertTrue(prompt.isRendered());

        assertEquals("Asignacion exitosa",
            server.getManagedBeanValue("#{solicitudInternaCC.mensajeAlerta}"));
    }
}

```

6. Carga de Solicitudes(Salvoconductos) Aprobadas

CÉDULA	NOMBRES SERVIDOR	CIUDAD	CONDUCTOR	VEHICULO
0604261883	Marco Bonilla	Ambato	WILSON FERNANDO SALAZAR ORTIZ	HEA698 Color BLANCO
0604261883	Marco Bonilla	Ambato	GILBERTO MESIAS GAVIDIA FLORES	HEA692 Color PLOMO

[Generar reporte](#)

```
import java.io.IOException;
import javax.faces.component.UIComponent;
import static junit.framework.Assert.assertEquals;
import static junit.framework.Assert.assertTrue;
import junit.framework.Test;
import junit.framework.TestSuite;
import org.jboss.jsfunit.jsfsession.JSFClientSession;
import org.jboss.jsfunit.jsfsession.JSFServerSession;
import org.jboss.jsfunit.jsfsession.JSFSession;

public class cargaSolicitudesYSalvoconductosAprobadosTest extends org.apache.cactus.ServletTestCase {

    public static Test suite() {
        return new TestSuite(cargaSolicitudesYSalvoconductosAprobadosTest.class);
    }

    public void testCargaSolcitudesAprobadas() throws IOException, Exception {

        JSFSession jsfSession =
            new JSFSession("/faces/seguridad/reportesSM/ListadoSolicitudesInternasAprobadas.xhtml");

        JSFClientSession client = jsfSession.getJSFClientSession();

        JSFServerSession server = jsfSession.getJSFServerSession();

        assertEquals("/seguridad/reportesSM/ListadoSolicitudesInternasAprobadas",
            server.getCurrentViewID());

        UIComponent prompt = server.findComponent("tblSoliitudesAprobadas");
        assertTrue(prompt.isRendered());

        prompt = server.findComponent("mensaje1");
        assertTrue(prompt.isRendered());

        assertEquals("Datos Cargados",
            server.getManagedBeanValue("#{solicitudInternaC.mensajeAlerta}"));
    }
}
```


BIBLIOGRAFÍA

1. BENCHMARKING

<http://www.monografias.com/trabajos3/bench/bench.shtml>

(2012/05/20)

2. CACTUS-REPORT.XSL

<http://jakarta.apache.org/cactus/misc/cactus-report.xsl>

(2012/08/05)

3. CICLO DE VIDA EN JSF

<http://micro-howto.blogspot.com/2010/01/ciclo-de-vida-en-jsf.html>

(2012/05/01)

4. DESCARGAS SELENIUM

<http://docs.seleniumhq.org/download/>

(2013/06/08)

5. EL PROCESO DE BENCHMARKING

http://www.minsa.gob.pe/dgsp/documentos/decs/2006/SegPac/El_Proceso_de_Benchmarking.pdf

(2012/05/20)

6. ESPECIFICACIÓN E IMPLEMENTACIÓN DE CASOS DE PRUEBA

http://www.lsi.us.es/~javierj/investigacion_ficheros/EICP.pdf

(2012/12/09)

7. HTTPUNIT

<http://www.javaworld.com/javaworld/jw-04-2004/jw-0419-httpunit.html>

(2012/04/20)

8. HTMLUNIT

<http://htmlunit.sourceforge.net/>

(2012/04/20)

9. INTRODUCCIÓN A JSF JAVA SERVER FACES

http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/viveros_s_ca/capitulo3.pdf

(2012/04/20)

10. JSFUNIT

http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/247#Contenidos_relacionados

(2013/06/04)

11. JWEBUNIT

<http://jwebunit.sourceforge.net/>

(2013/04/11)

12. NATURALEZA DEL BENCHMARKING Y SU APLICACIÓN

<http://www.tesis.ufm.edu.gt/pdf/2701.pdf>

(2012/06/29)

13. PRUEBAS DE SOFTWARE: HERRAMIENTAS: PRUEBAS UNITARIAS

http://www.kybele.etsii.urjc.es/docencia/IS_LADE/2011-2012/Material/Pruebas%20de%20SoftwareHerramientas.pdf

(2012/06/20)

14. PRUEBAS UNITARIAS WEB PARA APLICACIONES JSF.

<http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=PruebasUnitariasWeb>

(2012/07/02)

15. PRUEBAS UNITARIAS

<http://si.ua.es/es/documentacion/c-sharp/documentos/pruebas/07pruebasunitarias.pdf>

(2012/06/30)

16. SELENIUM REMOTE CONTROL

http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=selenium_rc

(2013/06/08)

17. TUTORIAL CACTUS

<https://today.java.net/pub/a/today/2004/12/09/testejb.html>

(2013/05/24)

18. TUTORIAL JSFUNIT

<http://www.mastertheboss.com/jsf/jsfunit-tutorial>

(2013/06/04)

19. TUTORIAL SELENIUM ID

<http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=seleniumIDE>

(2013/06/04)

20. VENTAJAS DEL BENCHMARKING

http://html.rincondelvago.com/benchmarking_competitividad.html

(2012/05/20)