ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN SISTEMAS

"ANÁLISIS DE LA TECNOLOGÍA JAVASERVER FACES, COMO FRAMEWORK DE DESARROLLO DE APLICACIONES WEB, CASO PRÁCTICO: PROCESOS DE COMERCIALIZACIÓN DE COMBUSTIBLES DE EP-PETROECUADOR"

TESIS DE GRADO

PREVIA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS

Miguel Ángel Baldeón Ordóñez

RIOBAMBA – ECUADOR

2011

A todos los que contribuyeron de una u otra forma apoyándonos desinteresadamente durante el desarrollo de esta tesis de grado, especialmente a la Unidad de Tecnologías de Información y Comunicación de EP Petroecuador, quienes colaboraron de manera desinteresada en la realización de esta tesis.

Esta tesis va dedicada de manera especial a mis

padres Rocío y Miguel, por sus, consejos, apoyo y comprensión

que me inspiraron a superarme constantemente, a mis profesores

por su apoyo incondicional, formando siempre en mí la

clase de persona que aspiro ser; y de manera especial muy especial a

Maggy, por todo su amor y apoyo.

F I R M A S D E R E S P O N S A B L E S

NOMBRE	**FIRMA**	**FECHA**
Ing. Iván Menes Camejo **DECANO FACULTAD INFORMÁTICA Y ELECTRÓNICA**	----------	----------
Ing. Raúl Rosero **DIRECTOR DE LA ESCUELA DE INGENIERÍA EN SISTEMAS**	----------	----------
Ing. Ivonne Rodríguez **DIRECTOR DE TESIS**	----------	----------
Ing. Danny Velasco **MIEMBRO DEL TRIBUNAL**	----------	----------
Lcdo. Carlos Rodríguez **DIR. CENTRO DE DOCUMENTACIÓN**	----------	----------

N O T A D E L A T E S I S ----------

"Yo, MIGUEL ÁNGEL BALDEÓN ORDÓÑEZ, soy responsables de las ideas, doctrinas y resultados expuestos en esta tesis; y, el patrimonio intelectual de la Tesis de Grado pertenece a la ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO".

Miguel Ángel Baldeón Ordóñez

Abreviaturas

JDO	Java Data Object
AJAX	Asynchronous JavaScript And XML - JavaScript y XML Asíncrono.
API	Application Programming Interface – Interfaz de Programación de Aplicaciones.
AS/400	Servidor de aplicaciones / base de datos IBM.
AWT	Abstract Window Toolkit - Juego de Herramientas de Ventana Abstracta.
BPEL	Tecnologías SOA.
CSS	Cascading Style Sheets - Hojas de Estilos en Cascada.
DB	Data base – Base de datos.
DB2/400	Integrated Data Base - Base de Datos Integrada
DBMS	Database Management System - Sistema de Gerencia de la Base de Datos
DDS	Data Description Specification - Especificaciones de la Descripción de los Datos
DHTML	Dynamic HTML - HTML Dinámico
DOM	Document Object Model - Modelo de Objetos del Documento
EBML	Extensible Binary Meta Language - Meta Lenguaje Binario Extendible
ECJ	Compilador que viene con Eclipse.
GCj	Integración del compilador de Java
GEF	Graphic Editing Framework - Framework para la Edición Gráfica
GWT	Google Web Toolkit
HTML	HyperText Markup Language – Lenguaje de Etiquetado de Hipertexto
HTTP	Hypertext Markup Language - Protocolo de Transferencia de Hipertexto
IBM	International Bussiness Machines – Máquinas de Negocio Internacional

IDE	Integrated Developement Environment - Entorno de Desarrollo Integrado.
IFS	Integrated File System - Sistema de Ficheros Integrado
IMS	Information Management System - Sistema de Gerencia de Información
J2EE	Java 2 Enterprise Edition
JDBC	Java DataBase Conectivity - Conectividad de Bases de Datos con Java
JDK	Java Development Kit - Kit de Desarrollo de Java.
JDT	Java Development Toolkit - Herramientas de Desarrollo Java
JNI	Java Native Interface - Interfaces Nativas de Java
JRE	Java Runtime Enviroment - Entorno en tiempo de Ejecución de Java.
JS	JavaScript
JSF	Java Server Faces
JSNI	JavaScript Native Interface - Interfaces Nativas de Java Script.
JSO	Java Script Object – Objeto Java Script
JSON	JavaScript Object Notation – Notación de Objetos de JavaScript
JSP	JavaServer Pages.
JVM	Java Virtual Machine - Máquina Virtual de Java
LAMP	Linux + Apache + Mysql + PHP
LIC	Licensed Internal Code - Código Interno Licenciado
MI	Machine Interfaz
NBM	NetBeans Module – Módulo Netbeans
OO	Orientación a objetos
RAD	Rational Application Developer
RIA	Rich internet Application, Aplicaciones Ricas en Internet
RMI	Remote Mechanism Method Invocation - Mecanismo de Invocación de Métodos Remotos.

RPC	Remote Procedure Service - Llamadas a Procedimientos Remotos
SDK	Software Development Kit - Kit de Desarrollo de Software
SOA	Service Oriented Architecture – Arquitectura Orientada a Servicios
SOAP	Simple Object Access Protocol – Protocolo de Acceso a Objetos Simples
SPA	Services Paradigm Adoption - Servicios de Adopción Paradigma
SQL	Structured Query Language - Lenguaje de Interrogación Estructurado
SRS	Especificación de requerimientos de software
SWT	Standard Widget Toolkit
UML	Unified Modeling Languaje - Unificado de Modelado Idioma
URL	Uniform Resource Locator – Localizador Uniforme de Recursos
WSAD	WebSphere Studio Application Developer
XHTML	eXtensible HyperText Markup Language - Lenguaje de Etiquetado de Hipertexto Extensible
XML	Extended Markup Language – Lenguaje de Marcado Extendido
Facelets	Es un sistema de plantillas open source para usarse con JSF, el lenguaje requiere entradas válidas XML para su funcionamiento.
JSP	Java Server Pages

TABLA DE CONTENIDO

Portada

Agradecimiento

Dedicatoria

Abreviaturas

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPÍTULO I

1. MARCO REFERENCIAL 18

1.1. PROBLEMATIZACIÓN 18

1.2. JUSTIFICACIÓN 20

1.3. OBJETIVOS 24

1.3.1. Objetivo General 24

1.3.2. Objetivos Específicos 24

1.4. HIPÓTESIS 25

1.5. MÉTODOS Y TÉCNICAS 25

1.5.1. Métodos 25

1.5.2. Técnicas 27

CAPÍTULO II

2. MARCO TEÓRICO 28

2.1. INTRODUCCIÓN 28

2.2. APLICACIONES WEB 29

2.2.1. Ventajas de las aplicaciones web 30

2.2.2. Desventajas de las aplicaciones web 31

2.2.3. Servidor de aplicaciones 32

2.3. LA TECNOLOGÍA AJAX 34

2.3.1. Modelo AJAX de Aplicaciones Web 35

2.3.2. Beneficios 37

2.3.3. Ventajas 38

2.3.4. Desventajas 39

2.4. PLATAFORMA RIA (Rich Internet Application) 40
2.4.1. Ventajas de la plataforma RIA 40
2.4.2. Desventajas de la plataforma RIA 42
2.5. PATRONES DE DISEÑO 43
2.5.1. Objetivos de los patrones de diseño 43
2.5.2. Categoría de patrones: 44
2.5.3. Patrones Creacionales: 46
2.5.4. Aplicación en ámbitos concretos 52
2.6. FRAMEWORK DE DESARROLLO 53
2.6.1. Dominios de un framework: 54
2.6.2. Arquitectura básica de un framework: 54
2.7. FRAMEWORK DE APLICACIONES WEB 56
2.7.1. Arquitectura 57
2.7.2. Características 58
2.7.3. Comparación de los principales frameworks de aplicaciones Web 62
2.7.4. Comparación entre los principales frameworks de aplicaciones web 64

CAPÍTULO III 66
3. ESTUDIO DE LA TECNOLOGÍA JAVA SERVER FACES 66
3.1. INTRODUCCIÓN 66
3.2. JAVASERVER FACES 67
3.3. CARACTERÍSTICAS PRINCIPALES 68
3.4. BENEFICIOS DE LA TECNOLOGÍA JSF: 71
3.5. UNA APLICACIÓN JAVASERVER FACES 73
3.6. ESTRUCTURA BÁSICA DE UNA APLICACIÓN JSF 76
3.6.1. Estructura de directorios para una aplicación JSF 77
3.6.2. Las páginas jsf 78
3.6.3. Ficheros de configuración de aplicación JSF: faces-context.xml, web.xml 79
3.6.4. Configuración de despliegue de la aplicación 82
3.7. MODELO VISTA CONTROLADOR EN JSF 83
3.7.1. Ventajas Modelo Vista Controlador 84
3.7.2. Modelo 85
3.7.3. Vista 87
3.7.4. Controlador 87

3.8. CICLO DE VIDA DE UNA PÁGINA JAVASERVER FACES 89
3.8.1. Escenarios de Procesamiento del Ciclo de Vida de una Petición 89
3.8.2. Ciclo de vida estándar en una aplicación JSF 92
3.9. MANAGED BEANS 98
3.9.1. Concepto 98
3.9.2. Atributos 99
3.9.3. Expresiones para valores inmediatos y directos 100
3.9.4. Ámbitos de los beans 100
3.10. NAVEGACIÓN 105
3.10.1. Concepto 106
3.10.2. Navegación Estática 106
3.10.3. Navegación dinámica 108
3.10.4. Navegación avanzada 109
3.10.5. Redirección 110
3.10.6. Comodines 111
3.10.7. Etiqueta from – action 112
3.11. ETIQUETAS BÁSICAS JSF 112

CAPÍTULO IV

4. ANÁLISIS COMPARATIVO 120
4.1. INTRODUCCIÓN 120
4.2. Determinación de las herramientas a comparar 121
4.2.1. JavaServer Faces 122
4.2.2. Struts 123
4.3. Análisis de las herramientas 124
4.3.1. JSF 124
4.3.2. Struts 134
4.4. Determinación de los parámetros de comparación 138
4.4.1. Parámetro 1: Comportamiento 138
4.4.2. Parámetro 2: Conectividad 139
4.5. Determinación de las variables para los parámetros de Comparación 139
4.6. Análisis comparativo 140
4.6.1. Parámetro 1: Comportamiento 143
4.6.4.1. Determinación de las variables 143

4.6.4.2. Interpretación de la valorización 147
4.6.2. Parámetro 2: Conectividad 152
4.6.4.3. Determinación de las variables 152
4.6.4.4. Interpretación de la valorización 155
4.7. Análisis de los resultados 157
4.8. Demostración de la Hipótesis 163
4.8.1. Tiempos de respuesta usando Struts 164
4.8.2. Tiempos de respuesta usando JSF 166
4.8.3. Análisis matemático 169

CAPÍTULO V

5. Diseño y desarrollo de la aplicación 174
5.1. Introducción 174
5.2. Metodología para el desarrollo del proyecto 174
5.2.1. Fase de incepción 176
Plan de desarrollo de software 176
Definiciones, siglas y abreviaturas 176
Entregables del proyecto 176
Plan de proyecto 178
Objetivos de las iteraciones 178
Visión 179
Alcance 179
Definición del problema 180
Definición de posicionamiento del producto: 181
Resumen de los usuarios: 181
Ambiente de los usuarios 182
Necesidades clave de los interesados 182
Alternativa y competencias 183
Perspectiva del producto 184
Características del producto 184
Otros requisitos del producto 185
Riesgos 185
Introducción 185
Propósito 186

Alcance 186
Visión general 186
Riesgos 186
Casos de Uso Generales 190
Especificación de casos de uso 191
Aseguramiento de precios 191
Flujo de Eventos 191
Flujos alternativos 193
Requerimientos especiales 193
5.2.2. Fase de elaboración 194
Arquitectura de software 194
Propósito 194
Referencia arquitectónica 194
Vistas de casos de uso 195
Paquetes y capas de subsistema 197
Vista de Despliegue 198
Capas 200
Vista de datos 200
Vista de objetos 202
5.2.3. Fase de Construcción 206
Documento de orden de trabajo de ingeniería de software 206
Evaluación estatus del proyecto 206
Completar el análisis, diseño, desarrollo y pruebas 206
Generación de versiones 206
5.2.4. Fase de Transición 207
CONCLUSIONES
RECOMENDACIONES
RESUMEN
SUMMARY
Glosario
Bibliografía
Anexos

ÍNDICE DE TABLAS

Tabla II. I: Frameworks aplicaciones web que trabajan con Java 62

Tabla II. II: Frameworks PHP más extendidos 63

Tabla II. III: Frameworks .Net más extendidos 63

Tabla II. IV: Comparación entre principales frameworks de aplicaciones web 64

Tabla III. V. Roles típicos en una aplicación JSF 76

Tabla III. VI. Escenarios de procesamiento del ciclo de vida de una petición 90

Tabla III. VII. Escenarios de procesamiento del ciclo de vida de una petición 104

Tabla IV. VIII: Determinación de las variables de los parámetros de comparación 140

Tabla IV. IX. Escala de calificación para parámetros de comparación 141

Tabla IV. X. Escala valoración Cuantitativa 141

Tabla IV. XI. Resumen variables parámetro Comportamiento 147

Tabla IV. XII. Resumen variables parámetro Conectividad 154

Tabla IV. XIII. Tabla general de resultados 158

Tabla V. XIV: Plan de fases 178

Tabla V. XV: Objetivo de las iteraciones 178

Tabla V. XVI: Resumen de los usuarios 182

Tabla V. XVII: Necesidades de los interesados 183

Tabla V. XVIII: Resumen de riesgos 189

ÍNDICE DE FIGURAS

Figura II. 1. Servidor de aplicaciones Web Actual 33
Figura II. 2. Arquitectura comparativa aplicación web clásica y aplicación AJAX 36
Figura II. 3. Modelo clásico de aplicación Web 36
Figura II. 4. Modelo AJAX de aplicación Web 37
Figura II. 5. Patrón de diseño: Fábrica Abstracta 47
Figura II. 6. Patrón de diseño: Factory Method 49
Figura II. 7. Arquitectura base de un framework web 56
Figura II. 8. Arquitectura típica de un Framework de aplicaciones web 57
Figura III. 9. Diagrama de una aplicación JSF 70
Figura III. 10. Estructura de una aplicación JSF típica empaquetada para distribución o portabilidad 78
Figura III. 11. Modelo Vista Controlador 84
Figura III. 12. Ciclo de vida estándar Petición – Respuesta en JSF 93
Figura III. 13. Ciclo de vida estándar Petición – Respuesta en una aplicación JSF 97
Figura III. 14. Diagrama de sintaxis de los elementos de navegación 110
Figura IV. 15. Arquitectura JSF 126
Figura IV. 16. Arquitectura Struts 135
Figura IV. 17. Comparación estadística del parámetro Comportamiento 152
Figura IV. 18. Comparación estadística del parámetro Conectividad 157
Figura IV. 19. Diagrama general de resultados por cada variable 159
Figura IV. 15. Diagrama general de resultados 160
Figura V. 21. Especificación de casos de uso: Aseguramiento de precios, integración con la abastecedora 192
Figura V. 22. CU acceso al sistema. 195
Figura V. 23. CU Parametrización del sistema. 196
Figura V. 24. CU Parametrización del sistema. 196
Figura V. 25. CU Parametrización entidades del negocio. 197
Figura V. 26. CU Generación de reportes. 197
Figura V. 27. Paquetes y capa 198
Figura V. 28. Vista de despliegue 199
Figura V. 29. Vista de despliegue 200
Figura V. 30. Vista de datos 201
Figura V. 31. Vista de datos 202
Figura V. 32. Vista de datos 205

I N T R O D U C C I Ó N

Actualmente vivimos en un mundo de evolución en donde la mayoría de los procesos que se realizaban de manera manual ha cambiado pues hoy en día las entidades y empresas están automatizando todo aquello utilizando el medio global más difundido que es la Internet, a través de las aplicaciones web.

En el capítulo I Marco Referencial, se plantea el estudio de una de la herramienta que está revolucionando el mundo actual en el desarrollo de aplicaciones web como es el framework JSF, el mismo que es parte de la plataforma RIA (Rich Internet Application), permitiendo escapar de la "matriz" de tecnologías usadas actualmente para escribir aplicaciones AJAX, las cuales son difíciles de manejar y propensas a errores.

En el capítulo II Marco Teórico, se realiza un estudio teórico de los conceptos necesarios para entender completamente el funcionamiento del framework JSF para lo cual se estudia toda la terminología necesaria al lenguaje de desarrollo JAVA, lo referente a AJAX, patrones de diseño, frameworks web, arquitectura MVC, aspectos generales como ventajas, desventajas, otros frameworks web.

En el capítulo III Java Server Faces, se estudia todo lo referente al framework para desarrollo de las aplicaciones web empresarial J2EE, para luego realizar el análisis correspondiente en base a una serie de parámetros y variables demostrar la superioridad de JSF sobre otro framework que está en uso actualmente en la institución donse se aplica su funcionamiento: Struts.

En el capítulo IV: Análisis Comparativo, se estudia los 2 frameworks existentes en la empresa: Struts y JSF, para luego realizar el análisis usando diferentes parámetros y variables que permitirán demostrar la efectividad de JSF sobreelframeowk actual Struts.

En el capítulo V, Ingeniería de Software, se aplica los conocimientos obtenidos en el capítulo II y se toma en consideración la aplicación del framework JSF para la implementación de módulos de aplicación para automatizar procesos de comercialización de combustibles de EP Petroecuador.

CAPÍTULO I

1. MARCO REFERENCIAL

1.1. PROBLEMATIZACIÓN

El rápido crecimiento que ha experimentado en los últimos años el uso de aplicaciones web ha ido paralelo con el aumento en la demanda de nuevas funcionalidades que los usuarios solicitan a estas aplicaciones.

Las aplicaciones web son populares debido a lo práctico del navegador web como cliente ligero, así como a la facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software a miles de usuarios potenciales. Existen aplicaciones como los webmails, wikis, weblogs, tiendas en línea y la propia Wikipedia que son ejemplos bien conocidos de aplicaciones web.

Para garantizar el buen funcionamiento y mantenimiento de los sitios web, este debe contar con ciertos atributos y características que en conjunto forman un concepto muy importante, para alcanzar el éxito en cualquier organización, herramienta, y todo aquello que se pueda considerar como servicio. Dicho concepto es la calidad, que con atributos como, usabilidad, navegabilidad, seguridad, mantenibilidad, entre otros, hace posible por un lado la eficiencia de la solución web y por ende la satisfacción del usuario final.

Es así que se presenta JavaServer Faces (JSF), como una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE.

La Empresa pública de Hidrocarburos del Ecuador (EP Petroecuador) es una empresa estatal ecuatoriana, encarga de la explotación de hidrocarburos. Dicha empresa cuenta con procesos bien definidos y automatizados por la Subgerencia de Tecnologías de Información y Comunicaciones de la institución, pero la generación de nuevos procesos de negocio y adjuntarlos a la arquitectura existente de la empresa es necesaria debido a la creación de nuevos procesos de distribución de combustibles y a la nueva reestructuración impuesta por el estado ecuatoriano.

La existencia de varios sistemas informáticos que automatizan los procesos de negocio de distribución de combustibles por parte de EP PETROECUADOR, basados en el modelo cliente servidor, ha demostrado ir decayendo en prestaciones y

calidad, gracias a la cantidad de usuarios cliente que la institución posee, así como la generación de tráfico en la intranet de la entidad.

1.2. JUSTIFICACIÓN

Justificación Teórica

La razón principal para el desarrollo de este proyecto nace del resultado de la evaluación y posterior reflexión de las necesidades vistas durante los procesos de negocio de distribución de combustibles de EP Petroecuador, así como el gestión financiera que estos procesos implican, por lo que es posible concebir una solución informática aplicando tecnologías actuales de desarrollo web.

Actualmente, EP Petroecuador cuenta con procesos bien definidos y automatizados por la Subgerencia de Tecnologías de Información y Comunicaciones de la institución, pero la generación de nuevos procesos de negocio y adjuntarlos a la arquitectura existente de la empresa es necesaria debido a la creación de nuevos procesos de distribución de combustibles y a la nueva reestructuración impuesta por el estado ecuatoriano.

Así mismo, hay que mencionar que este trabajo consiste en el análisis del framework JavaServer Faces para la implementación de la solución web, permitiendo integrar nuevos procesos de negocio a la arquitectura actual de la empresa, basada en servidores web de IBM WebSphere y plataforma AS/400 y DB2.

JavaServer Faces (JSF) es una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE. JSF usa JavaServer Pages (JSP) como la tecnología que permite hacer el despliegue de las páginas.

Además, la solución a desarrollar hará uso de las librerías open source RichFaces, que ofrecen integración con tecnologías AJAX, y componentes desarrollados para simplificar el desarrollo sobre JavaServer Faces.

Este framework está implementado bajo el patrón de diseño MVC(Model Vista Controller) el cual es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El estilo de llamada y retorno MVC, se ve frecuentemente en aplicaciones web, donde sus principales componentes son:

- La vista, es la página HTML y el código que provee de datos dinámicos a la página.
- El modelo es el Sistema de Gestión de Base de Datos y,
- La Lógica de negocio es el controlador es el responsable de recibir los eventos de entrada desde la vista.

Justificación práctica

Al desarrollar está solución web, se busca anexar a la infraestructura actual de servidores de aplicación, nuevos módulos que automaticen varios procesos de comercialización de combustibles, así también, lograr la interconexión de estos módulos con los existentes.

Además, la solución web apunta a facilitar el acceso a la información de los clientes administradores de las aplicaciones de gestión, así como también de los clientes externos a la institución (sujetos de control).

El ámbito de la solución será:

- Módulos de autenticación de usuarios administradores y clientes definidos por la institución, basado en roles, lo que permitirá desplegar los diferentes procesos descritos en los siguientes puntos.

- Módulo de gestión de precios automáticos de productos derivados del petróleo (Aseguramiento de precios a nivel de Terminal).

    - Leer de una fuente externa de datos (hoja de cálculo, formato de archivo texto plano) datos de *precios internacionales de combustibles*, los datos de *codificación internacional de productos*, *codificación internacional de medida*.

- Transformar los productos internacionales a nacionales (en el exterior se llama: *gasolina*, en Ecuador la conocemos como *extra*)
- Estandarizar las medidas.
- Cargar rubros de impuestos nacionales.
- Calcular el precio final de la venta.
- Registrar las salidas del proceso en la base de datos *Abastecedora*.

- Integración de precios de Abastecimiento y Comercialización de Hidrocarburos.
  - Leer desde el repositorio de datos *Abastecedora*, los precios a nivel de terminal de los principales productos del sector de Hidrocarburos, aplicarles.
  - Generar mediante un asistente guiado, una fórmula para el precio de venta al cliente final, según parámetros como tipo de cliente, moneda, unidad de medida, tipo de producto.
  - Gestionar la asignación de las fórmulas para cada grupo de productos de la Comercializadora.
  - Almacenar el resultado de la evaluación de las fórmulas de precio en la base de datos *Comercial*.

1.3. OBJETIVOS

1.3.1. Objetivo General

Analizar la tecnología JavaServer Faces, como framework de desarrollo de aplicaciones Java basadas en web, para procesos de comercialización de combustibles de EP Petroecuador.

1.3.2. Objetivos Específicos

- Estudiar la tecnología JavaServer Faces, sus características, ventajas y desventajas para desarrollo de aplicaciones web en Java.
- Analizar el comportamiento y conectividad de la tecnología JSF y la tecnología STRUTS, existente en la empresa.
- Implantar los módulos de: autentificación de usuarios basado en roles, módulo de gestión de precios automáticos de productos derivados del petróleo, módulo de integración de precios de Abastecimiento y Comercialización de Hidrocarburos.

1.4. HIPÓTESIS

El desarrollo de una aplicación web, basada en el framework JavaServer Faces, mejorará el rendimiento de procesos en línea para tareas de comercialización de combustibles por parte de EP Petroecuador.

1.5. MÉTODOS Y TÉCNICAS

La presente investigación realizada y fundamentada en el paradigma cuantitativo y cualitativo de conocimientos, pretende solucionar una problemática puntual que permitirá cubrir un problema expuesto en los antecedentes. Por tal razón, los métodos y técnicas e instrumentos que proporciona la metodología de la investigación científica de vital importancia y transcendencia en el desarrollo del presente proyecto.

1.5.1. Métodos

Se utilizarán los métodos establecidos para conseguir los resultados de los objetivos planteados:

a. **Método Científico**: Este será el método que avalará nuestra investigación científica porque a su vez contempla los siguientes puntos que involucran el desarrollo de esta tesis:

- Planteamiento del problema

- Formulación de la hipótesis
- Levantamiento o recopilación de la información
- Análisis e interpretación de resultados
- Comprobación de la hipótesis
- Difusión de resultados

b. **Método Analítico - Sintético**.- Este método, como lo dice su nombre, es el análisis que se realizará de los aspectos delimitados de la presente investigación que permitirá conocer, comprender y estudiar el objeto de estudio en partes y del todo investigado en sus diferentes componentes para el planteamiento de la metodología propuesto por los autores. Además el mismo, es aplicado en la sistematización de la bibliografía, que será analizada para entenderla y describirla.

c. **Método Analítico - Crítico**.- Este método es utilizado puesto que analizamos y sintetizamos los conocimientos, basados en la sustentación de la bibliografía, plasmados en los resultados del proyecto.

d. **Método Descriptivo**.- Este método se utilizará en el momento que se describa las características de la aplicación, conjuntamente con sus interfaces gráficas, además se realizará la explicación de cada función

y procedimiento que se utilice para la realización de la aplicación y por ende el cumplimiento del objetivo general que se plantea.

1.5.2. T é c n i c a s

a) Encuesta.- Serán diseñada una serie de encuestas, para obtener información atreves de un sistema de preguntas las cuales una vez aplicadas a los empleados del departamento financiero de EP Petroecuador, especialmente dirigido al nivel estratégico de la empresa, serán utilizadas para la recopilación de la información y luego tabularla y analizarla.

b) Entrevista.- La misma que se la aplicará al Jefe el departamento financiero de la EP Petroecuador, especialmente dirigido al nivel estratégico de la empresa, a fin de conocer las principales necesidades, características y/o dificultades que se tienen en los demás sistemas de la empresa para evitarlos en la solución planteada en este trabajo.

c) Lluvia de ideas.- La lluvia de ideas será una de las principales técnicas para recolectar la información y para procesarla, por el mismo hecho que esta investigación se realizará en el departamento financiero de EP Petroecuador, especialmente con el nivel estratégico de la empresa.

CAPÍTULO II

2. MARCO TEÓRICO

2.1. INTRODUCCIÓN

En el presente capítulo se realiza un estudio teórico de los conceptos necesarios que permitan entender el funcionamiento del framework JSF, para lo cual se estudia todo lo referente a aplicaciones web, aplicaciones RIA, Ajax, framework de aplicaciones web, patrones de diseño , framework como Struts comparado con JSF.

Además se estudia las herramientas a usar para la explotación del framework JSF en ambientes de desarrollo empresarial, con el objetivo de construir aplicaciones web de forma similar a la construcción de aplicaciones de escritorio.

2.2. APLICACIONES WEB

En la actualidad, un porcentaje muy significativo de proyectos de desarrollo de software tienen que ver con las llamadas aplicaciones Web. Estas aplicaciones tienen un origen en las páginas Web simples o sitios Web con algo de contenido dinámico pero han evolucionado hasta convertirse en piezas de software bastante complejas. Esto hace posible por ejemplo no solo solicitar información sino que efectuar transacciones directamente desde un browser. Las arquitecturas cliente-servidor de los años 80's y 90's están siendo reemplazadas por soluciones de 3 capas llamadas también de cliente delgado en que la capa de presentación es manejada por un browser estándar y la capa intermedia por un servidor de aplicaciones.

En la ingeniería de software se denomina aplicación web a 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.

Las aplicaciones web generan dinámicamente una serie de páginas en un formato estándar, como HTML o XHTML, soportados por los navegadores web comunes. Se utilizan lenguajes interpretados en el lado del cliente, directamente o a través de plugins tales como JavaScript, Java, Flash, etc., para añadir elementos dinámicos a la interfaz de usuario. Generalmente cada página web en particular se

envía al cliente como un documento estático, pero la secuencia de páginas ofrece al usuario una experiencia interactiva.

2.2.1. Ventajas de las aplicaciones web

Entre las principales ventajas de las aplicaciones web, tenemos:

- Ahorra tiempo: Se pueden realizar tareas sencillas sin necesidad de descargar ni instalar ningún programa.
- No hay problemas de compatibilidad: Basta tener un navegador actualizado para poder utilizarlas.
- No ocupan espacio en nuestro disco duro.
- Actualizaciones inmediatas: Como el software lo gestiona el propio desarrollador, cuando realizamos la conexión, estamos usando siempre la última versión que haya lanzado.
- Consumo de recursos bajo: Dado que toda (o gran parte) de la aplicación no se encuentra en nuestro ordenador, muchas de las tareas que realiza el software no consumen recursos nuestros porque se realizan desde otro ordenador.
- Multiplataforma: Se pueden usar desde cualquier sistema operativo.

- Portables: Es independiente del ordenador donde se utilice, porque se accede a través de una página web (sólo es necesario disponer de acceso a Internet o a la red).

- La disponibilidad suele ser alta porque el servicio se ofrece desde múltiples localizaciones para asegurar la continuidad del mismo.

- Los virus no dañan los datos porque éstos están guardados en el servidor de la aplicación.

- Colaboración: Gracias a que el acceso al servicio se realiza desde una única ubicación es sencillo el acceso y compartición de datos por parte de varios usuarios. Tiene mucho sentido, por ejemplo, en aplicaciones online de calendarios u oficina.

- Los navegadores ofrecen cada vez más y mejores funcionalidades para crear aplicaciones web ricas (RIAs)

2.2.2. Desventajas de las aplicaciones web

Por otro lado, como desventajas de las aplicaciones web, podemos señalar:

- Son especialmente difíciles de mantener.

- Hay una presión de tiempo aún mayor.

- Los aspectos de usabilidad son cruciales.

- Muchas tecnologías disponibles que cambian día a día.
- La arquitectura es relativamente compleja.
- Deben interactuar diseñadores gráficos con programadores.
- Deben ser altamente escalables.
- Programadores generalmente poco capacitados.

2.2.3. Servidor de aplicaciones

En informática, se denomina servidor de aplicaciones a un servidor en una red de computadores que ejecuta ciertas aplicaciones.

Se trata de un dispositivo de software que proporciona servicios de aplicación a las computadoras cliente. Un servidor de aplicaciones generalmente gestiona la mayor parte (o la totalidad) de las funciones de lógica de negocio y de acceso a los datos de la aplicación. Los principales beneficios de la aplicación de la tecnología de servidores de aplicación son la centralización y la disminución de la complejidad en el desarrollo de aplicaciones.

Si bien el término es aplicable a todas las plataformas de software, hoy en día el término servidor de aplicaciones se ha convertido en sinónimo de la plataforma Java EE (antes J2EE) de Sun Microsystems.[1]

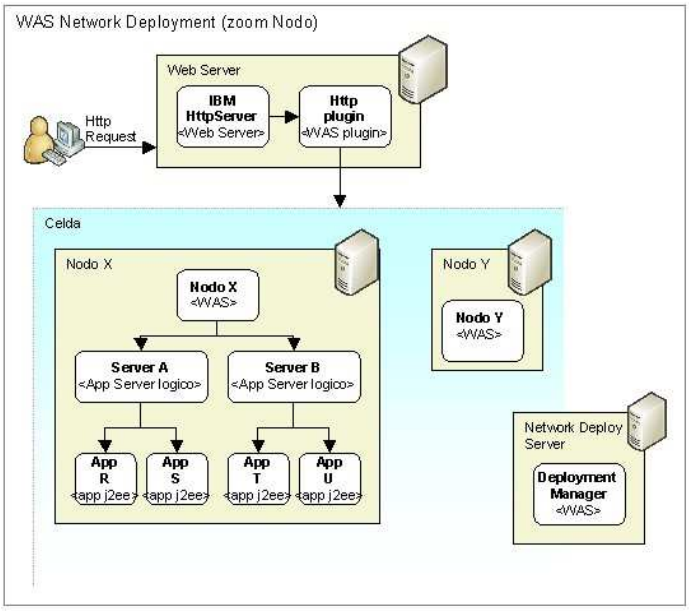


Figura II. 1. Servidor de aplicaciones Web Actual

Los servidores de aplicación típicamente incluyen también middleware (o software de conectividad) que les permite intercomunicarse con variados servicios, para efectos de confiabilidad, seguridad, no-repudio, etc. Los servidores de aplicación también brindan a los desarrolladores una Interfaz para Programación de Aplicaciones (API), de tal manera que no tengan que preocuparse por el sistema operativo o por la gran cantidad de interfaces requeridas en una aplicación web moderna.

[1] es.wikipedia.org/wiki/Servidor_de_aplicaciones

Los servidores de aplicación también brindan soporte a una gran variedad de estándares, tales como HTML, XML, IIOP, JDBC, SSL, etc., que les permiten su funcionamiento en ambientes web (como Internet) y la conexión a una gran variedad de fuentes de datos, sistemas y dispositivos.

2.3. LA TECNOLOGÍA AJAX

AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano.

De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página, además es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores dados que está basado en estándares abiertos como JavaScript y Document Object Model (DOM).

Ajax es una combinación de cuatro tecnologías ya existentes:

- **XHTML (o HTML) y hojas de estilos en cascada (CSS)** para el diseño que acompaña a la información.

- **Document Object Model (DOM)** accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones ECMAScript como JavaScript y JScript, para mostrar e interactuar dinámicamente con la información presentada.

- **El objeto XMLHttpRequest** para intercambiar datos de forma asíncrona con el servidor web. En algunos frameworks y en algunas situaciones concretas, se usa un objeto iframe en lugar del XMLHttpRequest para realizar dichos intercambios.

- **XML** es el formato usado generalmente para la transferencia de datos solicitados al servidor, aunque cualquier formato puede funcionar, incluyendo HTML pre formateado, texto plano, JSON y hasta EBML.

Como el DHTML, LAMP o SPA, Ajax no constituye una tecnología en sí, sino que es un término que engloba a un grupo de éstas que trabajan conjuntamente.

2.3.1. Modelo AJAX de Aplicaciones Web

En el gráfico II.I, II.II y II.III se aprecian las diferencias entre el modelo clásico de aplicación Web y el modelo Ajax de aplicación Web.

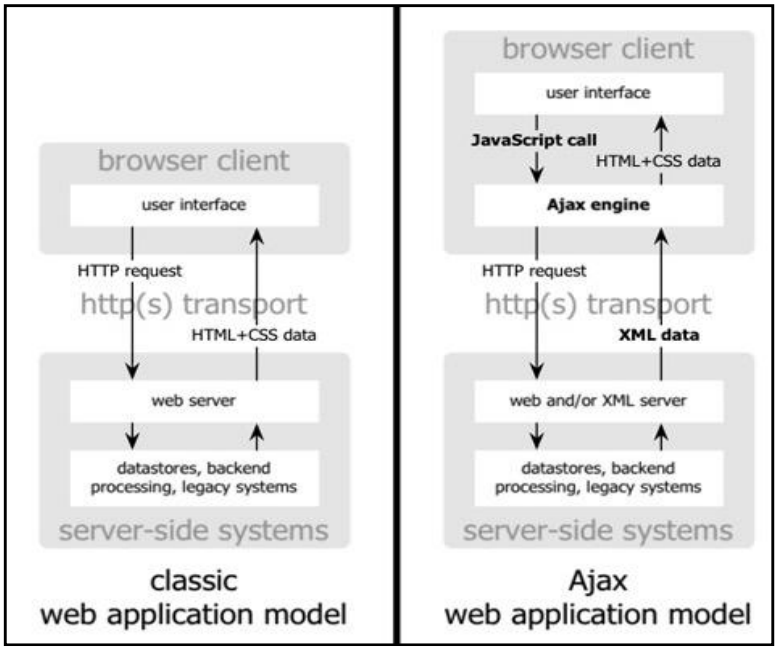


Figura II. 2. Arquitectura comparativa aplicación web clásica y aplicación AJAX

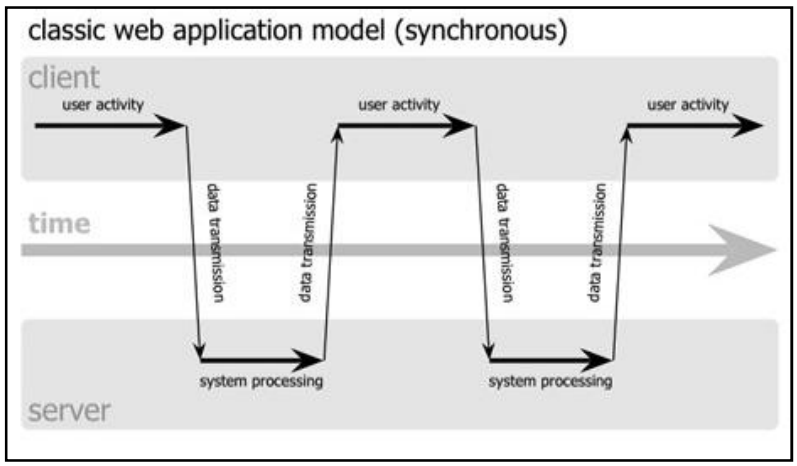


Figura II. 3. Modelo clásico de aplicación Web

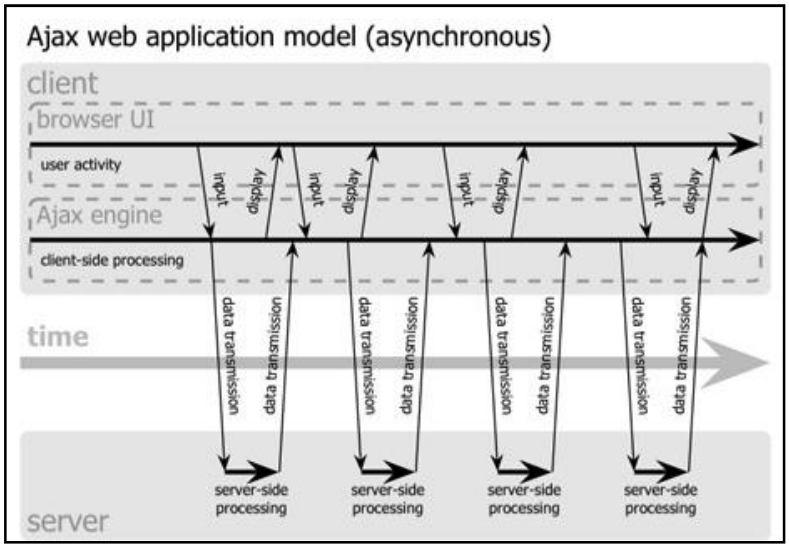


Figura II. 4. Modelo AJAX de aplicación Web

Las figuras indican que la interacción entre el servidor y el navegador (cliente) no es directa en el modelo AJAX, sino que esta se realiza a través de los componentes AJAX, lo cual genera asincronismo y disminución en el tráfico entre el navegador y el servidor.

2.3.2. Beneficios

Entre los principales beneficios al usar tecnología AJAX tenemos:

- Basado en los estándares abiertos.
- Usabilidad.
- Válido en cualquier plataforma y navegador.
- Beneficia las aplicaciones web.
- No es difícil su utilización.

- Compatible con Flash.
- Adoptado por los “grandes” de la tecnología web.
- Web 2.0.
- Es independiente del tipo de tecnología de servidor que se utilice.
- Mejora la estética de la web.

2.3.3. Ventajas

Entre las principales ventajas al usar tecnología Ajax, tenemos:

- Permite diseñar interfaces muchísimo más dinámicas acercándonos a "aplicaciones de escritorio".
- La comunicación asíncrona con el servidor permite varias cosas que reducen el "peso de la página" / "líneas de código que el cliente tiene que descargarse".
  - Campos de select, si las opciones son muchas... no es necesario transmitirlas en la primera carga de la página, se puede producir la lista de opciones cuando el cliente pulse sobre el desplegable. Otra solución es hacer una búsqueda dinámica.
  - La navegación por la aplicación mediante Ajax permite evitar al cliente descargar la cabecera de todos los "documentos", "pantallas" <html><head><tit..... etc… Se puede cambiar el contenido de

cualquier objeto del DOM dinámicamente ante los eventos que controlamos con Javascript.

- No requiere plugins o capacidades específicas de ciertos navegadores.
- Las aplicaciones son más interactivas, responden a las interacciones del usuario más rápidamente, al estilo desktop.
- Se reduce el tamaño de la información intercambiada.
  - Muchos micro-petición, pero el flujo de datos global es inferior

2.3.4. Desventajas

Entre las principales desventajas de usar tecnología Ajax, podemos encontrar:

- El cliente necesita un navegador que soporte Javascript .Hoy por hoy la mayoría de los navegadores soporta JavaScript. Internet Explorer, Mozilla, Firefox, Safari etc. Para esta desventaja se tiene una "excusa", no se está hablando de desarrollar páginas web con Ajax, sino aplicaciones web, y como toda aplicación tiene unos requisitos mínimos, y siendo este Javascript tampoco se discrimina un espectro muy amplio de usuarios.
- El mal uso de ajax/javascript puede mal emplearse sobrecargando el servidor de peticiones ej.: Si hacemos que cada milisegundo haga una consulta con una base de datos, puede colapsar el servidor.

2.4. PLATAFORMA RIA (Rich Internet Application)

Los sistemas de software han estado alrededor por varias décadas, pero sólo hace poco tiempo que comenzó a ser utilizado por millones y millones de personas en todo el mundo. Hace sólo 20 años, la mayoría de las aplicaciones de software fueron utilizadas por profesionales entrenados.

El modelo de interacción es la que puede soportar una variedad de métodos de entrada y responder intuitivamente de manera oportuna. Como regla general, para ser considerado rica la interacción con el usuario es que debe ser tan buena como la actual generación de aplicaciones de escritorio.

Esto incluye características como la prestación de los diferentes medios de interacción por información visual por ejemplo, cambiando la forma del cursor, usando las indicaciones colores.

2.4.1. Ventajas de la plataforma RIA

La plataforma RIA ofrece las siguientes ventajas:

- **No requiere instalación**.- La aplicación es descargada y ejecutada automáticamente en el navegador. El software que corre actualmente la aplicación es instalada en la máquina del cliente.

- **Las actualizaciones son automáticas**.- las versiones nuevas de la aplicación también se descargarán automáticamente con sólo volver a visitar la página de la aplicación web.

- **Plataforma independiente**.- Una aplicación rica de Internet puede ejecutarse y operar potencialmente en cada plataforma de un navegador.

- **Más seguridad**.- Las aplicaciones se ejecutan en un ambiente restringido en el navegador y por lo tanto son menos probables que sea dañada como las aplicaciones que necesitan ser instalados.

- **Más respuesta**.- Porque no todas las acciones de los usuarios requieren comunicación con el servidor, las RIA tienden a ser más sensibles que las aplicaciones web clásicas.

- **Más escalable**.- Una gran parte del trabajo computacional, así como también el cuido estatal puede ser descargado desde el cliente, por lo que el servidor puede manejar muchos más usuarios.

- **Más eficiencia de red**.- En las aplicaciones web clásicas, cada acción del usuario requiere que el servidor cargue toda la página y enviarla a la red. En el caso de las aplicaciones ricas en Internet (RIA), toda la aplicación de interfaz de usuario sólo tiene que ser comunicado una vez y todas las otras peticiones al servidor requieren sólo los datos actuales que se envían al cliente

2.4.2. Desventajas de la plataforma RIA

Como desventajas de la plataforma RIA, tenemos:

- **Requiere JavaScript o un plug-in específico**.- Porque toda la aplicación se ejecuta en él cliente a través del intérprete de JavaScript. Cuando el usuario desactiva JavaScript por completo la aplicación por lo general hace poco o nada. Obviamente es posible tener un plan alternativo para esos usuarios, pero luego se va a mantener dos aplicaciones separadas, que está lejos de ser ideal.

- **No hay acceso a los recursos del sistema**.- Como las aplicaciones AJAX se ejecutan dentro de un navegador, ellos están limitados en los recursos que pueden acceder. Por ejemplo, una aplicación AJAX no puede tener acceso al sistema de archivos del cliente.

- **Es duro de indexar completamente para los motores de búsqueda**.- Porque la mayoría de los motores de búsqueda no soportan que aplicaciones actualicen parcialmente las páginas o usen un plug-in específico tales como Flash. La mayoría de las aplicaciones dinámicas de Internet están mal indexadas por los motores de búsqueda.

- **Cuestiones de accesibilidad**.- Al hacer actualizaciones parciales de página mediante JavaScript o un plugin específico puede romper la accesibilidad. El problema más grande y más notoria es que no se pueda manejar correctamente la pantalla.

- **Depende de una conexión a Internet**.- Porque estas aplicaciones son servidas desde la web y ejecutadas en el navegador web, por lo que requieren por lo menos una conexión a Internet inicial. Pero incluso durante su uso, es necesaria una conexión a Internet para comunicarse con el servidor. Cuando la conexión no está temporalmente disponible, las RIA's no funcionará.

2.5. PATRONES DE DISEÑO

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Además, debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

2.5.1. Objetivos de los patrones de diseño

Los patrones de diseño pretenden:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.

- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

Asimismo, no pretenden:

- Imponer ciertas alternativas de diseño frente a otras.
- Eliminar la creatividad inherente al proceso de diseño.

No es obligatorio utilizar los patrones, solo es aconsejable en el caso de tener el mismo problema o similar que soluciona el patrón, siempre teniendo en cuenta que en un caso particular puede no ser aplicable. "Abusar o forzar el uso de los patrones puede ser un error".

2.5.2. Categoría de patrones:

Según la escala o nivel de abstracción:

- **Patrones de arquitectura**: Aquéllos que expresan un esquema organizativo estructural fundamental para sistemas de software.

- **Patrones de diseño**: Aquéllos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software.

- **Dialectos**: Patrones de bajo nivel específicos para un lenguaje de programación o entorno concreto.

Además, también es importante reseñar el concepto de "antipatrón de diseño", que con forma semejante a la de un patrón, intenta prevenir contra errores comunes de diseño en el software. La idea de los antipatrones es dar a conocer los problemas que acarrean ciertos diseños muy frecuentes, para intentar evitar que diferentes sistemas acaben una y otra vez en el mismo callejón sin salida por haber cometido los mismos errores.

Además de los patrones ya vistos actualmente existen otros patrones como el siguiente:

- **Interacción**: Son patrones que permiten el diseño de interfaces web.

2.5.3. Patrones Creacionales:

Los principales patrones creacionales son:

- **Abstract Factory (Fábrica abstracta)**: Permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando.

- **Builder (Constructor virtual)**: Abstrae el proceso de creación de un objeto complejo, centralizando dicho proceso en un único punto.

- **Factory Method (Método de fabricación)**: Centraliza en una clase constructora la creación de objetos de un subtipo de un tipo determinado, ocultando al usuario la casuística para elegir el subtipo que crear.

- **Prototype (Prototipo)**: Crea nuevos objetos clonándolos de una instancia ya existente.

- **Singleton (Instancia única)**: Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

- **Abstract Factory**

Contexto: Debemos crear diferentes objetos, todos pertenecientes a la misma familia. Por ejemplo: las librerías para crear interfaces gráficas suelen utilizar este patrón y cada familia sería un sistema operativo distinto. Así pues, el usuario declara un Botón, pero de forma más interna lo que está creando es un BotónWindows o un BotónLinux, por ejemplo.

El problema que intenta solucionar este patrón es el de crear diferentes familias de objetos.

El patrón Abstract Factory está aconsejado cuando se prevé la inclusión de nuevas familias de productos, pero puede resultar contraproducente cuando se añaden nuevos productos o cambian los existentes, puesto que afectaría a todas las familias creadas.

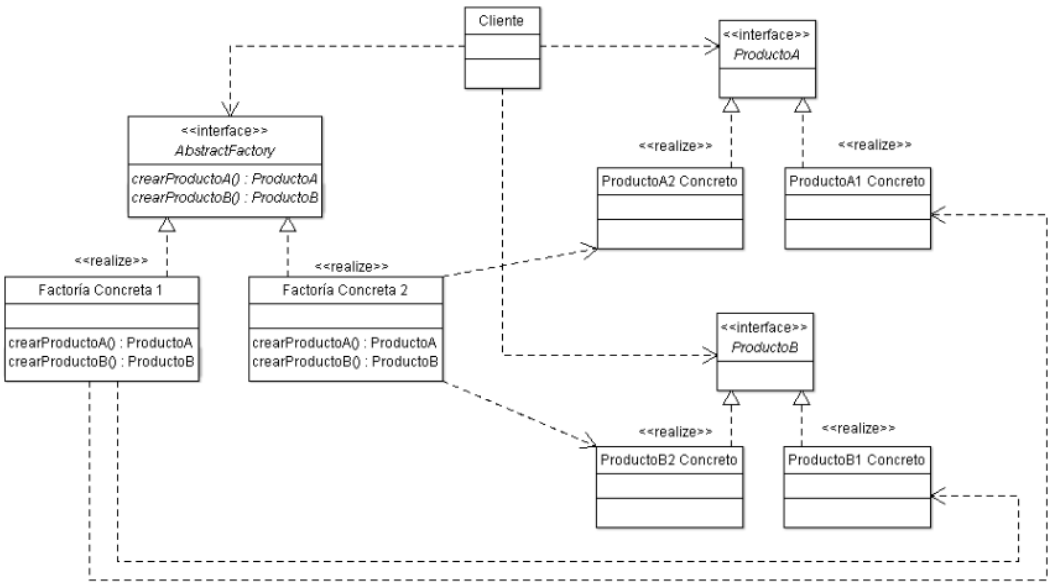


Figura II. 5. Patrón de diseño: Fábrica Abstracta

La estructura típica del patrón Abstract Factory es la siguiente:

- **Cliente**: La clase que llamará a la factoría adecuada ya que necesita crear uno de los objetos que provee la factoría, es decir, Cliente lo que quiere es obtener una instancia de alguno de los productos (ProductoA, ProductoB).

- **AbstractFactory**: Es de definición de la interfaces de las factorías. Debe de proveer un método para la obtención de cada objeto que pueda crear. ("crearProductoA()" y "crearProductoB()")

- **Factorías Concretas**: Estas son las diferentes familias de productos. Provee de la instancia concreta de la que se encarga de crear. De esta forma podemos tener una factoría que cree los elementos gráficos para Windows y otra que los cree para Linux, pudiendo poner fácilmente (creando una nueva) otra que los cree para MacOS, por ejemplo.

- **Producto abstracto**: Definición de las interfaces para la familia de productos genéricos. En el diagrama son "ProductoA" y "ProductoB". En un ejemplo de interfaces gráficas podrían ser todos los elementos: Botón, Ventana, Cuadro de Texto, Combo... El cliente trabajará directamente sobre esta interfaz, que será implementada por los diferentes productos concretos.

- **Producto concreto**: Implementación de los diferentes productos. Podría ser por ejemplo "BotónWindows" y "BotónLinux". Como ambos

implementan "Botón" el cliente no sabrá si está en Windows o Linux, puesto que trabajará directamente sobre la superclase o interfaz.

- **Factory Method**

En diseño de software, el patrón de diseño Factory Method consiste en utilizar una clase constructora (al estilo del Abstract Factory) abstracta con unos cuantos métodos definidos y otro(s) abstracto(s): el dedicado a la construcción de objetos de un subtipo de un tipo determinado. Es una simplificación del Abstract Factory, en la que la clase abstracta tiene métodos concretos que usan algunos de los abstractos; según usemos una u otra hija de esta clase abstracta, tendremos uno u otro comportamiento.

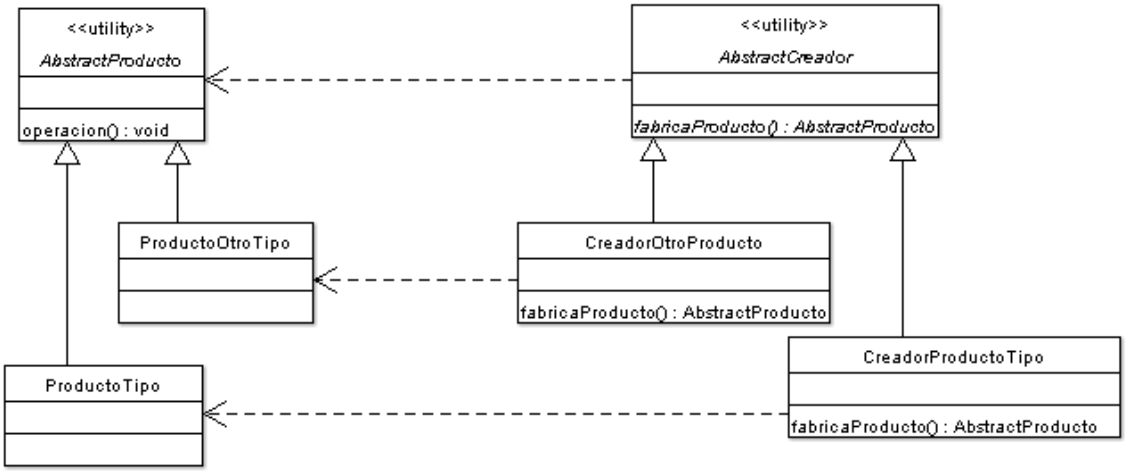


Figura II. 6. Patrón de diseño: Factory Method

Las clases principales en este patrón son el **creador** y el **producto**. El creador necesita crear instancias de productos, pero el tipo concreto de

producto no debe ser forzado en las subclases del creador, porque entonces las posibles subclases del creador deben poder especificar subclases del producto para utilizar.

La solución para esto es hacer un método abstracto (el método de la fábrica) que se define en el creador. Este método abstracto se define para que devuelva un producto. Las subclases del creador pueden sobrescribir este método para devolver subclases apropiadas del producto.

- **Singleton**

El patrón de diseño singleton (instancia única) está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto.

Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella.

El patrón singleton se implementa creando en nuestra clase un método que crea una instancia del objeto sólo si todavía no existe alguna. Para asegurar que la clase no puede ser instanciada nuevamente se regula el alcance del constructor (con atributos como protegido o privado).

La instrumentación del patrón puede ser delicada en programas con múltiples hilos de ejecución. Si dos hilos de ejecución intentan crear la

instancia al mismo tiempo y esta no existe todavía, sólo uno de ellos debe lograr crear el objeto. La solución clásica para este problema es utilizar exclusión mutua en el método de creación de la clase que implementa el patrón.

Las situaciones más habituales de aplicación de este patrón son aquellas en las que dicha clase controla el acceso a un recurso físico único (como puede ser el ratón o un archivo abierto en modo exclusivo) o cuando cierto tipo de datos debe estar disponible para todos los demás objetos de la aplicación.

El patrón singleton provee una única instancia global gracias a que:

- La propia clase es responsable de crear la única instancia.
- Permite el acceso global a dicha instancia mediante un método de clase.
- Declara el constructor de clase como privado para que no sea instanciable directamente.

2.5.4. Aplicación en ámbitos concretos

Además de su aplicación directa en la construcción de software en general, y derivado precisamente del gran éxito que han tenido, los patrones de diseño han sido aplicados a múltiples ámbitos concretos.

En particular son notorios los esfuerzos en los siguientes ámbitos:

- Patrones de interfaces de usuario, esto es, aquellos que intentan definir las mejores formas de construir interfaces hombre-máquina.
- Patrones para la construcción de sistemas empresariales, en donde se requieren especiales esfuerzos en infraestructuras de software y un nivel de abstracción importante para maximizar factores como la escalabilidad o el mantenimiento del sistema.
- Patrones para la integración de sistemas, es decir, para la intercomunicación y coordinación de sistemas heterogéneos.
- Patrones de flujos de trabajo, esto es para la definición, construcción e integración de sistemas abstractos de gestión de flujos de trabajo y procesos con sistemas empresariales.

2.6. FRAMEWORK DE DESARROLLO

Es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar.

En el desarrollo de software, un framework es una estructura conceptual y tecnológica de soporte, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Los Frameworks son diseñados con la intención de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional. Por ejemplo, un equipo que usa Apache Struts para desarrollar un sitio web de un banco, puede enfocarse en cómo los retiros de ahorros van a funcionar en lugar de preocuparse de cómo se controla la navegación entre las páginas en una forma libre de errores. Sin embargo, hay quejas comunes acerca de que el uso de frameworks añade código innecesario y que la preponderancia de frameworks competitivos y complementarios significa que el tiempo que se pasaba programando y diseñando ahora se gasta en aprender a usar frameworks.

2.6.1. Dominios de un framework:

Los frameworks para asistir el desarrollo de software, contienen típicamente un considerable repositorio de utilidades y accesorios, para la ayuda de la construcción de aplicaciones de software, pero de manera global, pueden cubrir áreas como:

- Dibujo, composición musical, composición mecánico CAD.
- Compiladores para diferentes lenguajes de programación y arquitectura de máquina.
- Modelado de aplicaciones financieras.
- Sistemas de modelamiento geográfico, espacial.
- Sistema de apoyo en la toma de decisiones.
- Reproducción multimedia y autoría.
- Aplicaciones Web.

2.6.2. Arquitectura básica de un framework:

Los frameworks para desarrollo de software, consisten en “puntos congelados” y los “puntos calientes”. Puntos congelados definen la arquitectura general de un sistema de software, es decir, sus componentes básicos y las relaciones entre ellos. Estos permanecen sin cambios

(congelados) en cualquier instancia de la estructura de aplicaciones. Los puntos calientes representan aquellas partes donde los programadores que utilizan el framework pueden añadir su propio código para agregar la funcionalidad específica para su proyecto.

Los frameworks definen lugares específicos en la arquitectura de la aplicación, donde los programadores de aplicaciones pueden hacer adaptaciones específicas para la funcionalidad deseada.

En un entorno orientado a objetos, un marco de trabajo está formado por las clases abstractas y concretas. Creación de instancias de este marco consiste en componer y subclases de las clases existentes.

Al desarrollar un sistema de software concreto con un marco de software, los desarrolladores utilizar los puntos calientes de acuerdo a las necesidades y requisitos específicos del sistema. Los frameworks de software confían en el principio de Hollywood: "No nos llames, nosotros te llamaremos." Esto significa que las clases definidas por el usuario (por ejemplo, nuevas subclases), reciben mensajes de las clases del framework predefinido. Los desarrolladores suelen manejar esta situación mediante la implementación de las interfaces de las clases padre.

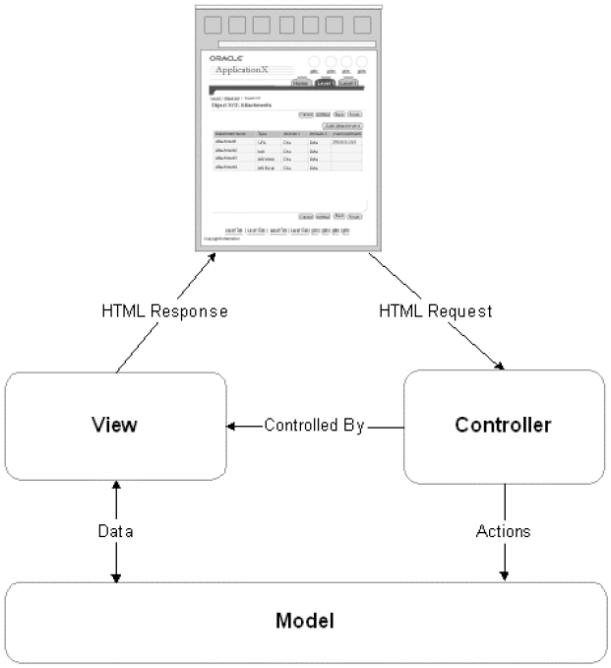


Figura II. 7. Arquitectura base de un framework web

2.7. FRAMEWORK DE APLICACIONES WEB

Un framework para aplicaciones web es un framework diseñado para apoyar el desarrollo de sitios web dinámicos, aplicaciones web y servicios web. Este tipo de frameworks intenta aliviar el exceso de carga asociado con actividades comunes usadas en desarrollos web. Por ejemplo, muchos framework proporcionan bibliotecas para acceder a bases de datos, estructuras para plantillas y gestión de sesiones, y con frecuencia facilitan la reutilización de código.

2.7.1. Arquitectura

La mayoría de frameworks de aplicaciones web están basados en el estilo de arquitectura de software: MVC. Desde el tipo de vista de arquitectura, hay 5 tipos de diseño MVC:

- Request-based, basados en la petición.
- Component-based, basados en los componentes
- Hybrid, híbridos.
- Meta, gestores de contenido
- RIA, Aplicaciones ricas de internet.

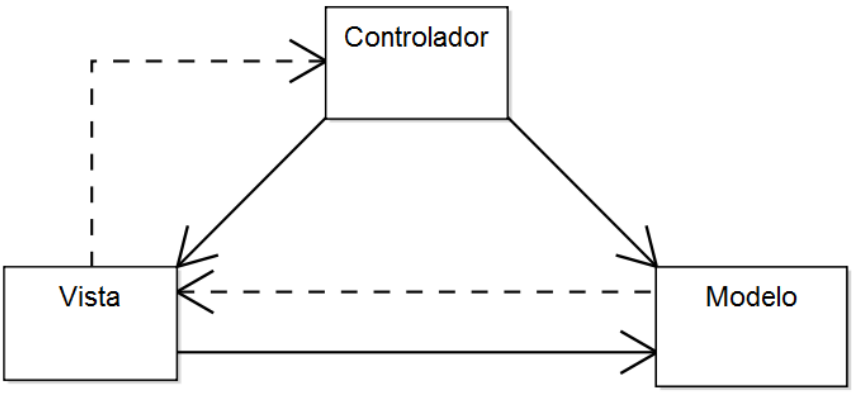


Figura II. 8. Arquitectura típica de un Framework de aplicaciones web

2.7.2. Características

- **Sistema de plantillas web.**

Las páginas web dinámicas suelen consistir en una parte estática (HTML) y una parte dinámica, que es el código HTML generado. El código que genera el código HTML se puede formar en función de variables en una plantilla, o en código. El texto que se genera puede provenir de una base de datos, de tal modo que se reduce drásticamente el número de páginas en un sitio o aplicación web.

En una plantilla, las variables del lenguaje de programación se pueden insertar sin usar código, perdiendo así la exigencia de conocimientos de programación para hacer cambios a las páginas de un sitio web. Una sintaxis específica está construida para que el servidor distinga entre etiquetas HTML y variables. Por ejemplo: JSP en la etiqueta <c:out> se utiliza para las variables de salida.

- **Caching**

Caching, o memoria caché es el almacenamiento de documentos web, con el fin de reducir el uso de ancho de banda, así como la carga del servidor, reduciendo el "retraso" percibido. Una caché de sitios web almacena copias de los documentos que atraviesan esta memoria temporal; las

solicitudes posteriores pueden ser servidas directamente de esta caché si se cumplen ciertas condiciones.

Algunos marcos de aplicación proporcionan mecanismos de almacenamiento en caché de documentos, así como para diversas etapas de preparación de la página, tales como acceso a bases de datos o la interpretación de la plantilla.

- **Seguridad**

Algunos frameworks de aplicaciones web, vienen con frameworks de autenticación y autorización, estos, permiten al servidor web identificar a los usuarios de una aplicación, y restringir el acceso a funciones basadas en algún tipo de criterio.

- **Acceso a la base de datos y mapeo**

Muchos Frameworks de aplicaciones web, han creado una API, estandarizando el acceso a las bases de datos, habilitando a la aplicación web el uso de una amplia variedad de bases de datos sin cambiar el código de la aplicación, permitiendo al desarrollador preocuparse por conceptos de alto nivel del problema.

Para mejorar el rendimiento, las conexiones a la base de datos pueden residir en un pool de conexiones. Adicionalmente, algunos framewoks

contienen algún framework orientado a objetos con herramientas de mapeo objeto relacional, el cual mapea objetos de la base de datos a objetos válidos para estructuras de programación.

Otras características de los frameworks para aplicaciones web proveen soporte a objetos transaccionales y migración de datos.

- **Mapeo URL**

Un framework facilita el mapeo de recursos URL, necesarios para nombrar los recursos en internet.

El sistema de mapeo de URL, reescribe la dirección URL de forma más "amigable", incrementando la simplicidad de un sitio y permitiendo un mejor indexado por los motores de búsqueda. Por ejemplo, el recurso: `"/page.cgi?cat=science&topic=physics"`, puede ser "traducido" a `"/page/science/physics"`, lo que provee un mejor entendimiento para el lector, y proporciona mecanismos de estructuración del sitio, para motores de búsqueda.

- **Ajax**

Las aplicaciones AJAX son excelentes para la creación de aplicaciones web que son altamente interactivas y ofrecen un mejor servicio al usuario,

mientras que es comparable a las aplicaciones de escritorio en funcionalidad, sin la molestia de descargar o instalar nada.

Ajax posibilita el incremente de la interactividad del usuario con la aplicación web, en velocidad y usabilidad.

Debido a la complejidad en el desarrollo de aplicaciones usando Ajax en Javascript, existen numerosos frameworks que simplifican el desarrollo con Ajax. Estos frameworks pueden venir embebidos en otros frameworks, como RichFaces.

- **Configuración Automática**

Algunos frameworks minimizan la configuración de aplicaciones web, a través del uso de instrospección y/o usando ciertas convenciones. Por ejemplo, muchos Java Framewoks usan Hibernate como capa de persistencia, la cual genera un esquema de la base de datos en tiempo de ejecución, esto posibilita al diseñador de la aplicación a diseñar objetos de negocio sin definir explícitamente el esquema de la base de datos.

- **Web services**

Algunos frameworks proveen herramientas para crear y proveer web services.

2.7.3. Comparación de los principales frameworks de aplicaciones Web

A continuación, se presenta un cuadro comparativo de los más notables frameworks de aplicaciones web.

- **Java**

Se detallan a continuación los principales frameworks para desarrollo de aplicaciones web en Java:

Tabla II. I: Frameworks aplicaciones web que trabajan con Java

PROYECTO	VERSIÓN ESTABLE ACTUAL	FECHA DE LIBERACIÓN	LICENCIA
Apache Struts	2.1.8.1		Apache
Google Web Toolkit	2.1.0	2010-10-28	Apache
JavaServer Faces	2.0	2009-06-29	
JBoss Seam	2.2.0 GA		LGPL
Spring	3.0.0		Apache
Apache Tapestry	4.1.6/5.1.0.5		Apache

Fuente: es.wikipedia.org

- **PHP**

A continuación se detallan los frameworks php más extendidos:

Tabla II. II: Frameworks PHP más extendidos

PROYECTO	VERSIÓN ESTABLE ACTUAL	FECHA DE LIBERACIÓN	LICENCIA
Yii	1.1.8	2011-08-11	GPL
Symfony	2.0.6	2011-11-06	MIT
Zend Framework	1.11.0	2011-08-03	BSD
CakePhp	2.0	2011-10-16	MIT

Fuente: es.wikipedia.org

- **ASP.NET**

Se detallan los frameworks presentes en la arquitectura ASP.NET

Tabla II. III: Frameworks .Net más extendidos

PROYECTO	VERSIÓN ESTABLE ACTUAL	LICENCIA
ASP.NET MVC	2.0	Ms-PL
BFC	7.40	Proprietary

Fuente: es.wikipedia.org

2.7.4. Comparación entre los principales frameworks de aplicaciones web

A continuación se detalla una breve comparación general entre los principales frameworks de aplicaciones web, independiente de la tecnología que usan.

Tabla II. IV: Comparación entre principales frameworks de aplicaciones web

PROYECTO	VERSIÓN	LENGUAJE	AJAX	MVC	I18N & L10N?	ORM	DB MIGRATION FRAMEWORK(S)	SECURITY FRAMEWORK(S)	TEMPLATE FRAMEWORK(S)	CACHING FRAMEWORK(S)	FORM VALIDATION FRAMEWORK(S)
ASP.NET MVC	3.0	.net	Yes	Yes	-	ORM-independent	-	ASP.NET Forms Auth	pluggable (default is WebForms)	Yes	Yes (client-side via plugins)
Flex	3.0	Actionscript, MXML	Not by name but similar technology	-	-	-	-	-	-	Yes	-
Ruby on Rails	3.0	Jruby	Prototype, script.aculo.us, jQuery	Yes	Yes	ActiveRecord	Yes	Plug-in	Yes	Yes	Yes
Joomla		Php 4,>	No	Yes	Yes	Yes, Cake PHP	No	Yes	Yes	Yes	No
JBoss Seam	1.6	Java	Yes	Yes	Yes	JPA, Hibernate	-	JAAS integration, Drools, Hibernate Filters, OpenID, CAPTCHA	Facelets	JbossCache, EHCache	Hibernate Validator

Apache Struts	2.0	Java	Yes	Yes	Yes	Yes	-	-	Yes	-	Yes
JavaServer Face	2.0	Java	Yes	Yes	-	free integration	-	Yes	Yes	Yes	yes
Symfony	1,4	Php 5, >	Sí	Prototype, script.aculo. us, Unobtrusive Ajax with UJS and PJS plugins	yes	Propel, Doctrine (YAML)	Vía plugin	Vía plugin	Yes	Yes	Yes

Fuente: es.wikipedia.org

Nota: la ventaja de java es que se puede integrar más fácilmente varias tecnologías, ejemplo, JSF no tiene framework de validación, pero se puede integrar `hibernate validator` solo añadiendo las librerías al `classpath`.

CAPÍTULO III

3. ESTUDIO DE LA TECNOLOGÍA JAVA SERVER FACES

3.1. INTRODUCCIÓN

El objetivo de este capítulo es analizar y aplicar el framework de desarrollo de aplicaciones web: JSF (JavaServer Faces). Todo esto para dar una solución demostrativa de la eficacia de JSF en un entorno de necesidades reales, además, se analizarán los principales conceptos de la tecnología de presentación que se ha vuelto estándar para JSF: facelets, y la implementación de componentes para JSF propia de la compañía JBoss: RichFaces.

Todo esto, con el objetivo de mitigar las necesidades empresariales, en relación a las tendencias de programación más comerciales y más exitosas.

Otro punto muy importante que se encuentra de manera implícita, es dar a conocer la experiencia que se obtuvo al aprender éstas tecnologías, considerando que este tipo de desarrollo es ajeno a la programación clásica que se lleva en la universidad, sobretodo porque no solo se aprendió JavaServer Faces, sino también Hibernate, Hibernate Validator, RichFaces, Facelets, Servlets, JavaBeans.

3.2. JAVASERVER FACES

El objetivo de la tecnología JavaServer Faces es desarrollar aplicaciones web de forma parecida a como se construyen aplicaciones locales con Java Swing, AWT (Abstract WindowToolkit), SWT (Standard Widget Toolkit) o cualquier otra API similar.

Tradicionalmente, las aplicaciones web se han codificado mediante páginas JSP (JavaServer Pages) que recibían peticiones a través de formularios y construían como respuesta páginas HTML (Hiper Text Markup Language) mediante ejecución directa o indirecta -a través de bibliotecas de etiquetas- de código Java, lo que permitía, por ejemplo, acceder a bases de datos para obtener los resultados a mostrar, o consultas típicas como inserción o modificación de registros en tablas relacionales, etc.

JavaServer Faces pretende facilitar la construcción de estas aplicaciones proporcionando un entorno de trabajo (framework) vía web que gestiona las

acciones producidas por el usuario en su página HTML y las traduce a eventos que son enviados al servidor con el objetivo de regenerar la página original y reflejar los cambios pertinentes provocados por dichas acciones.

En definitivas cuentas, se trata de hacer aplicaciones Java en las que el cliente no es una ventana de la clase JFrame o similar, sino una página HTML.

Cualquier evento realizado sobre una página JSF incurre en una carga sobre la red, ya que el evento debe enviarse a través de ésta al servidor, y la respuesta de éste debe devolverse al cliente; por ello, el diseño de aplicaciones JavaServer Faces debe hacerse con cuidado cuando se pretenda poner las aplicaciones a disposición del mundo entero a través de internet. Aquellas aplicaciones que vayan a ser utilizadas en una intranet podrán aprovecharse de un mayor ancho de banda y producirán una respuesta mucho más rápida.

3.3. CARACTERÍSTICAS PRINCIPALES

La tecnología JavaServer Faces constituye un marco de trabajo (framework) de interfaces de usuario del lado de servidor para aplicaciones web basadas en tecnología Java y en el patrón MVC (Modelo Vista Controlador).

Los principales componentes de la tecnología JavaServer Faces son:

Una API y una implementación de referencia para:

- Representar componentes de interfaz de usuario (UI-User Interface) y manejar su estado.
- Manejar eventos, validar en el lado del servidor y convertir datos.
- Definir la navegación entre páginas
- Soportar internacionalización y accesibilidad, y
- Proporcionar extensibilidad para todas estas características.
- Una librería de etiquetas JavaServer Pages (JSP) personalizadas para dibujar componentes UI dentro de una página JSP.

Este modelo de programación bien definido y la librería de etiquetas para componentes UI facilita de forma significativa la tarea de la construcción y mantenimiento de aplicaciones web con UIs en el lado servidor. Con un mínimo esfuerzo, es posible:

- Conectar eventos generados en el cliente a código de la aplicación en el lado servidor.
- Mapear componentes UI a una página de datos en el lado servidor.
- Construir una interfaz de usuario con componentes reutilizables y extensibles.

Como se puede apreciar en la figura III.9, la interfaz de usuario que se crea con la tecnología JavaServer Faces (representada por `miUI` en el gráfico) se ejecuta en el servidor y se renderiza en el cliente.

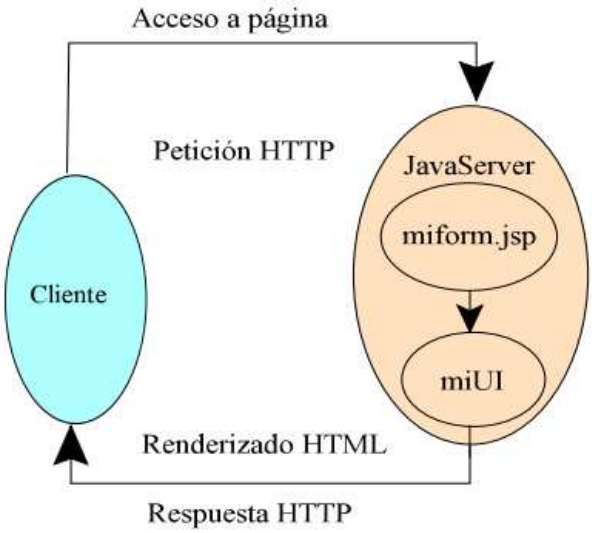


Figura III. 9. Diagrama de una aplicación JSF

La página JSP, `miform.xhtml`, especifica los componentes de la interfaz de usuario mediante etiquetas personalizadas definidas por la tecnología JavaServer Faces. La UI de la aplicación web (representada por `miUI` en la figura) maneja los objetos referenciados por la JSP / XHTML, que pueden ser de los siguientes tipos:

- Objetos componentes que mapean las etiquetas sobre la página `jsp / xhtml`.
- Oyentes de eventos, validadores y conversores registrados y asociados a los componentes.
- Objetos del modelo que encapsulan los datos y las funcionalidades de los componentes específicos de la aplicación (lógica de negocio).

3.4. BENEFICIOS DE LA TECNOLOGÍA JSF:

A continuación se explican algunos de los puntos por los que JSF es una tecnología muy interesante. La implementación de JSF está regida por los siguientes estándares:

- JSR 127 [2]
- JSR 252 [3]
- JSR 276 [4]

Una de las ventajas de que JSF sea una especificación estándar 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 éstos, soporte proporcionado, precio, evolución, etc.

JSF trata la vista (la interfaz de usuario) de una forma algo diferente a lo que estamos acostumbrados en aplicaciones web, ya que este tratamiento es mucho más cercano al estilo de Java Swing, Visual Basic o Delphi, donde la programación de la interfaz se hacer a través de componentes y está basada en eventos (pulsación de un botón, cambio en el valor de un campo, etc.).

[2] http://www.jcp.org/en/jsr/detail?id=127
[3] http://www.jcp.org/en/jsr/detail?id=252
[4] http://www.jcp.org/en/jsr/detail?id=276

JSF es muy flexible. Por ejemplo: permite crear nuestros propios componentes, y/o crear nuestros propios renderizadores para pintar los componentes en la forma que más nos convenga.

Una de las grandes ventajas de la tecnología JavaServer Faces es que ofrece una clara separación entre el comportamiento y la presentación.

Las aplicaciones web construidas con tecnología JSP conseguían parcialmente esta separación. Sin embargo, una aplicación JSP no puede mapear peticiones HTTP al manejo de eventos específicos de los componentes o manejar elementos UI como objetos con estado en el servidor. La tecnología JavaServer Faces permite construir aplicaciones web que introducen realmente una separación entre el comportamiento y la presentación, separación sólo ofrecida tradicionalmente por arquitecturas UI del lado del cliente.

Separar la lógica de negocio de la presentación también permite que cada miembro del equipo de desarrollo de la aplicación web se centre en su parte asignada del proceso diseño, y proporciona un modelo sencillo de programación para enlazar todas las piezas. Por ejemplo, personas sin experiencia en programación pueden construir páginas JSF usando las etiquetas de componentes UI que esta tecnología ofrece, al estilo de codificación de páginas html, y luego enlazarlas con código de la aplicación sin escribir ningún script.

Otro objetivo importante de la tecnología JavaServer Faces es mejorar los conceptos familiares de componente-UI y capa-web sin limitarnos a una tecnología

de script particular o un lenguaje de marcas. Aunque la tecnología JavaServer Faces incluye una librería de etiquetas JSP/facelets personalizadas para representar componentes en una página JSP/facelets, las APIs de JavaServer Faces se han creado directamente sobre el API JavaServlet. Esto brinda la posibilidad, teóricamente, de realizar algunas cosas avanzadas: usar otra tecnología de presentación junto a JSP, crear nuestros propios componentes personalizados directamente desde las clases de componentes, y generar salida para diferentes dispositivos cliente, entre otras.

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

3.5. UNA APLICACIÓN JAVASERVER FACES

En su mayoría, las aplicaciones JavaServer Faces son como cualquier otra aplicación web Java. Se ejecutan en un contenedor de servlets de Java y, típicamente, contienen:

- Componentes JavaBeans (llamados objetos del modelo en tecnología JavaServer Faces) conteniendo datos y funcionalidades específicas de la aplicación.
- Oyentes de Eventos.
- Páginas, (principalmente páginas JSP).

- Clases de utilidad del lado del servidor, como beans para acceder a las bases de datos.

Además de estos ítems, una aplicación JavaServer Faces también tiene:

- Una librería de etiquetas personalizadas para dibujar componentes UI en una página.
- Una librería de etiquetas personalizadas para representar manejadores de eventos, validadores y otras acciones.
- Componentes UI representados como objetos con estado en el servidor.

Toda aplicación JavaServer Faces debe incluir una librería de etiquetas personalizadas que define las etiquetas que representan componentes UI, así como una librería de etiquetas para controlar otras acciones importantes, como validadores y manejadores de eventos. La implementación de JavaServer Faces, de Sun proporciona estas dos librerías. La librería de etiquetas de componentes elimina la necesidad de codificar componentes UI en HTML u otro lenguaje de marcas, lo que se traduce en el empleo de componentes completamente reutilizables.

Y la librería principal (core) hace fácil registrar eventos, validadores y otras acciones de los componentes.

Como librería de etiquetas de componentes puede usarse la librería html_basic incluida con la implementación de referencia de la tecnología JavaServer Faces,

aunque también es posible definir una librería de etiquetas personalizadas que dibuje componentes propios o que proporcione una salida distinta a HTML.

Otra ventaja importante de las aplicaciones JavaServer Faces es que los componentes UI de la página están representados en el servidor como objetos con estado. Esto permite a la aplicación manipular el estado del componente y conectar los eventos generados por el cliente a código en el lado del servidor.

Finalmente, la tecnología JavaServer Faces permite convertir y validar datos sobre componentes individuales e informar de cualquier error antes de que se actualicen los datos en el lado del servidor.

Debido a la división de labores que permite el diseño de la tecnología JavaServer Faces, el desarrollo y mantenimiento de una aplicación JavaServer Faces se puede realizar muy rápida y fácilmente. Abajo tenemos un listado de los roles principales de un equipo de desarrollo típico.

Aunque en muchos equipos, los desarrolladores individuales pueden interpretar más de uno de esos roles, resulta muy útil considerar la tecnología JavaServer Faces desde varias perspectivas basadas en la responsabilidad principal que tiene cada participante:

Tabla III. V. Roles típicos en una aplicación JSF

Autores de páginas	Que utilizan un lenguaje de marcas, como HTML, para construir páginas para aplicaciones web. Cuando se utiliza la tecnología JavaServer Faces, los autores de páginas casi siempre usarán exclusivamente la librería de etiquetas.
Desarrolladores de aplicaciones	Que programan los objetos del modelo, los manejadores de eventos, los validadores, y la navegación de páginas. Los desarrolladores de aplicaciones también pueden proporcionar las clases de utilidad necesarias.
Escritores de componentes	Que tienen experiencia en programar interfaces de usuario y prefieren crear componentes personalizados utilizando un lenguaje de programación. Esta gente puede crear sus propios componentes desde cero, o puede extender los componentes estándares proporcionados por JavaServer Faces.
Vendedores de herramientas	Que proporcionan herramientas que mejoran la tecnología JavaServer Faces para hacer incluso más sencilla la construcción de interfaces de usuario en el lado servidor.

Fuente: Manual JSF

Para las aplicaciones web pueden ser muy convenientes frameworks como JSF, Struts, Spring,etc., pero éstos no servirán (a no ser a costa de un gran esfuerzo) para hacer portales. Para este segundo caso sería más adecuado usar gestores de contenidos como Lenya, OpenCMS, etc.

3.6. ESTRUCTURA BÁSICA DE UNA APLICACIÓN JSF

A continuación se muestran las piezas que son necesarias para construir una aplicación JSF:

3.6.1. Estructura de directorios para una aplicación JSF

Una aplicación JSF se corresponde con una estructura de directorios que sigue un modelo estándar, la cual se puede comprimir en un archivo con extensión WAR, para mayor comodidad y portabilidad. Esta estructura de directorios estándar es:

```
Aplicación /
              Ficheros HTML y JSP / XHTML
              WEB – INF/
                          Archivos de configuración
                          Classes/
                                Archivos.class
                          Lib/
                                Librerías
```

Con lo que la estructura del archivo war, queda así:

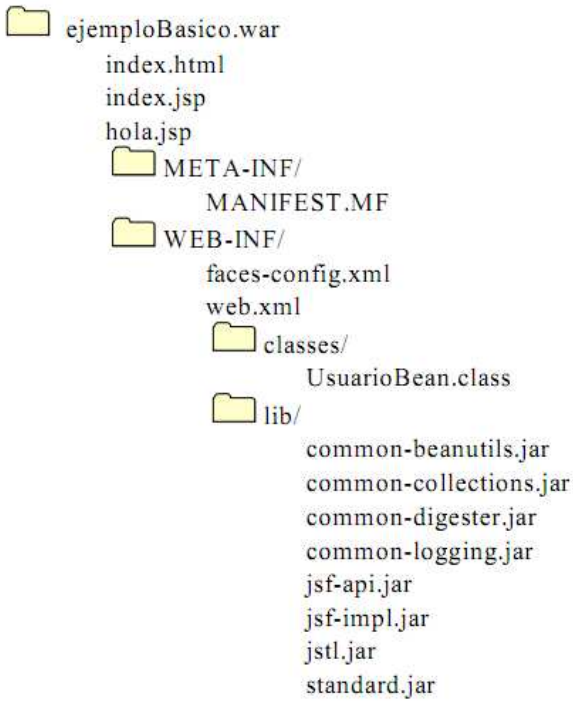


Figura III. 10. Estructura de una aplicación JSF típica empaquetada para distribución o portabilidad

Donde el directorio `META-INF` se genera automáticamente al crear el archivo `.war`.

3.6.2. Las páginas jsf

Las aplicaciones web correctamente planificadas tienen dos partes: la parte de presentación y la lógica de negocio.

La parte de presentación afecta a la apariencia de la aplicación, y en el contexto de una aplicación basada en navegadores, la apariencia está determinada por las etiquetas HTML, esto comprende marcos, tipos de caracteres, imágenes, etc. La lógica de negocio se implementa en Java y determina el comportamiento de la aplicación.

3.6.3. Ficheros de configuración de aplicación JSF: faces-context.xml, web.xml

Un archivo de configuración para la aplicación que contiene recursos del bean y las reglas de navegación. Por defecto, este archivo se denomina faces-config.xml.

- **Beans**

Un bean es el que maneja los datos de la aplicación, representación de un objeto del mundo real. Es simplemente una clase en Java en la que sus campos son accedidos siguiendo métodos getter y setter convencionales:

- Para conocer el valor de un campo llamado xxx se utiliza la función getXxx (o isXxx si es de tipo boolean)
- Para asignarle valor a un campo llamado xxx se usa la función setXxx.

En una aplicación JSF, se deben usar beans para todos los datos accedidos por una página. Los beans son los conductos entre la interfaz de usuario y la trastienda de la aplicación.

Los beans deben ser registrados en el fichero de configuración faces: `faces-config.xml`:

```
<faces-config>
  <managed-bean>
  <managed-bean-name>usuario</managed-bean-name>
   <managed-bean-class>UsuarioBean</managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
  </managed-bean>
</faces-config>
```

La etiqueta `managed-bean-class`, simplemente especifica la clase a que pertenece el `bean`, en este ejemplo: `UsuarioBean.class`.

Y finalmente tenemos la etiqueta `managed-bean-scope`, que en nuestro caso tiene el valor `session`, con lo que estamos indicando que el `bean` estará disponible para un cliente durante todo el tiempo que esté conectado al sitio web y, por tanto, mantendrá su valor a través de múltiples páginas, es decir, se está indicando en ámbito del `bean`.

Los posibles ámbitos de un `bean` pueden ser: `request, session, application, none`.

- **Reglas de navegación:**

Otra posibilidad que tiene el desarrollador de la aplicación es definir la navegación de páginas por la aplicación, como qué página va después de que el usuario pulse un botón para enviar un formulario.

El desarrollador de la aplicación define la navegación por la aplicación mediante el fichero de configuración, `faces-config.xml`, el mismo fichero en el que se declararon los `managed beans`.

Cada regla de navegación define cómo ir de una página (especificada en el elemento `from-view-id`) a otras páginas de la aplicación. El elemento `navigation-rule` puede contener cualquier número de elemento `navigation-case`, cada uno de los cuales define la página que se abrirá luego (definida por `to-view-id`) basándose en una salida lógica (definida mediante `from-outcome`).

La salida se puede definir mediante el atributo `action` del componente `UICommand` que envía el formulario.

Ahora vamos con las reglas de navegación de este ejemplo, de esta manera, indicamos a que página ir tras otra página. En este caso la navegación es simple, tras pulsar el botón Aceptar, navegamos desde `index.jsp` hasta `hola.jsp`. Estas reglas de navegación se especifican en el fichero `faces-config.xml`, como se muestra a continuación:

```
<faces-config>
  <navigation-rule>
  <from-view-id>/index.jsp</from-view-id>
   <navigation-case>
     <from-outcome>login</from-outcome>
     <to-view-id>/hola.jsp</to-view-id>
   </navigation-case>
  </navigation-rule>
</faces-config>
```

El valor de `from-outcome (login)`, indica la acción que se debe producir en la página `from-view-id (index.jsp)` para navegar al

destino representado por `to-view-id` (`hola.jsp`). La acción es un mero nombrecito que se da a los botones de un formulario; en nuestro caso, la página `index.jsp` incluía la etiqueta:

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

3.6.4. Configuración de despliegue de la aplicación

Una aplicación JSF requiere un `servlet`, llamado `FacesServlet`, el cual actual como controlador.

Esta configuración está recogida en el fichero `web.xml`, el cual se muestra a continuación, y afortunadamente, se puede usar el mismo fichero para múltiples aplicaciones JSF. De esta manera se establece el único `servlet` de nuestra aplicación es el propio del framework JSF.

```
<web-app>
  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.jsf</url-pattern>
  </servlet-mapping>
</web-app>
```

Lo importante aquí es el mapeo del `servlet`. Todas las páginas JSF son procesadas por un `servlet` especial que forma parte del código de implementación de JSF.

Para asegurarnos de que el servlet está correctamente activado cuando una página JSF es demandada, la URL posee un formato especial. En nuestra configuración, es el formato .jsf. Por ejemplo, en su navegador no se puede poner simplemente:

```
http://localhost:8080/login/index.jsp
```

si no que la dirección debe ser:

```
http://localhost:8080/login/index.jsf
```

El contenedor servlet usa la regla del mapeado servlet para activar el servlet JSF, quien elimina el sufijo faces y carga la página index.jsp.

Esto se hace así para que el framework JSF, a través de su servlet principal tome el control. Si no se hiciera de esta manera, el servidor de aplicaciones mostraría una simple página JSP como tal, y la compilación de dicha página, fuera del marco JSF, provocaría errores.

3.7. MODELO VISTA CONTROLADOR EN JSF

El patrón MVC (Modelo Vista Controlador), ver Figura III. 3, nos permite separar la

lógica de control (qué cosas hay que hacer pero no cómo), la lógica de negocio (cómo se hacen las cosas) y la lógica de presentación (cómo interaccionar con el usuario).

Utilizando este tipo de patrón es posible conseguir más calidad, un mantenimiento más fácil, perder el miedo a la falta de documentación (existe un patrón de partida por el que empezar un proyecto), etc. al margen de todo esto, una de las cosas más importantes que permite el uso de este patrón consiste en normalizar y estandarizar el desarrollo de Software.

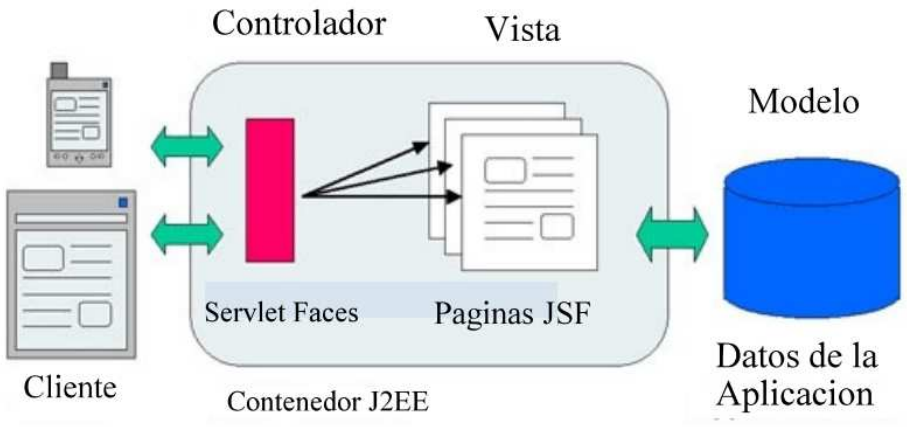


Figura III. 11. Modelo Vista Controlador

3.7.1. Ventajas Modelo Vista Controlador

- Hay una clara separación entre los componentes de un programa; lo cual nos permite implementarlos por separado.
- Hay una API muy bien definida; cualquiera que use la API, podrá reemplazar el modelo, la vista o el controlador, sin demasiada dificultad.

- La conexión entre el modelo y sus vistas (ya que puede haber varias) es dinámica: se produce en tiempo de ejecución, no en tiempo de compilación.

3.7.2. Modelo

Todas las aplicaciones software dejan a los usuarios manipular ciertos datos que proceden de una realidad sobre la que se pretende actuar, como supermercados, itinerarios de viaje, o cualquier dato requerido en un dominio problemático particular. A estos datos en estado puro, que representan el estado de la realidad se les llama modelo: modelan la parte de la realidad sobre la que se desea actuar.

Entonces, modelo es, el objeto que representa y trabaja directamente con los datos del programa: gestiona los datos y controla todas sus transformaciones. El modelo no tiene conocimiento específico de los diferentes controladores y/o vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el modelo y sus vistas, y notificar a las vistas cuándo deben reflejar un cambio en el modelo.

Como ejemplo básico, lo primero que hay que hacer es definir el modelo, que se ve reflejado en el fichero .java, donde se aprecia la clase `UsuarioBean`, que contiene los elementos necesarios para manejar los datos de la aplicación: es necesario un campo que almacene el nombre del

usuario que entró, y otro para su correspondiente contraseña, junto con sus respectivos métodos de asignación y recuperación:

```
public class UsuarioBean {

	/**
	 * Atributo nombre
	 */
	private String nombre;

	/**
	 * Atributo password
	 */
	private String password;

	/**
	 * Retorna el nombre del Usuario.
	 * @return El <k>nombre</k> del usuario.
	 */
	public String getNombre() {
		return nombre;
	}

	/**
	 * Setea el nombre del Usuario.
	 * @param nuevoValor El nombre del usuario.
	 */
	public void setNombre(String nuevoValor)
	{
		nombre = nuevoValor;
	}

	public String getPassword() {
		return password;
	}

	public void setPassword(String nuevoValor) {
		password = nuevoValor;
	}
}
```

Este modelo a utilizar en la aplicación se le comunica al sistema JSF mediante el fichero `faces-config.xml`, donde se detalla la parte de `managed-bean`, donde se aprecia un `bean` denominado usuario, que está recogido en la clase `UsuarioBean`, y con un ámbito de sesión:

```
<managed-bean>
  <managed-bean-name>usuario</managed-bean-name>
  <managed-bean-class>UsuarioBean</managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

3.7.3. Vista

La vista es el 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. Interacciona con el modelo a través de una referencia al propio modelo.

Como ejemplo, la vista está manipulada a través de las páginas JSF, es decir, mediante las los ficheros .jsp, .xhtml, o cualquier configuración de extención en web.xml. JSF conecta la vista y el modelo. Como ya se ha visto, un componente de la vista puede ligarse a un atributo de un bean del modelo, como:

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

donde se ve como se declara un campo de texto de entrada (inputText) en la vista, ese campo de texto recoge su valor de entrada en el atributo nombre de un bean denominado usuario.

De esta manera se establece el vínculo de enlace en vista y modelo.

3.7.4. Controlador

El controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el modelo. Entra en acción cuando se realiza alguna operación, ya sea un cambio en la

información del modelo o una interacción sobre la Vista. Se comunica con el modelo y la vista a través de una referencia al propio modelo.

Además, JSF opera como un gestor que reacciona ante los eventos provocados por el usuario, procesa sus acciones y los valores de estos eventos, y ejecuta código para actualizar el modelo o la vista.

Retomando el ejemplo básico, una parte del controlador la recogen las lineas de código del `.jsp`, `.xhtml`, o cualquier configuración para la extensión del fichero facelets en `web.xml` que capturan los datos del nombre de usuario, su contraseña, y el botón de Aceptar:

```
<h:inputText value="#{usuario.nombre}"/>
<h:inputSecret value="#{usuario.password}"/>
<h:commandButton value="Aceptar" action="login"/>
<h:outputText value="#{usuario.nombre}"/>
```

Por otro lado, está el control para las reglas de navegación, contenido en el fichero `faces-config.xml`, donde por ejemplo, se indica que estando `index.jsp`, si ocurre una acción denominada `login`, navegaremos a la página `hola.jsp`, esta acción comentada, es un `string` que se declara en la vista como un atributo del botón de aceptar que aparece en el formulario del ejemplo básico. El fichero `faces-config` sería el siguiente:

```
<navigation-rule>
   <from-view-id>/index.jsp</from-view-id>
   <navigation-case>
     <from-outcome>login</from-outcome>
     <to-view-id>/hola.jsp</to-view-id>
   </navigation-case>
</navigation-rule>
```

y la parte de la vista que establece la acción que activa la navegación es:

```
<h:commandButton value="Aceptar" action="login"/>
```

3.8. CICLO DE VIDA DE UNA PÁGINA JAVASERVER FACES

El ciclo de vida de una página JavaServer Faces es similar al de una página JSP: El cliente hace una petición HTTP (Hiper Text Transfer Protocol) de la página y el servidor responde con la página traducida a HTML. Sin embargo, debido a las características extras que ofrece la tecnología JavaServer Faces, el ciclo de vida proporciona algunos servicios adicionales mediante la ejecución de algunos pasos extras.

Los pasos del ciclo de vida se ejecutan dependiendo de si la petición se originó o no desde una aplicación JavaServer Faces y si la respuesta es o no generada con la fase de renderizado del ciclo de vida de JavaServer Faces. Esta sección explica los diferentes escenarios del ciclo de vida.

3.8.1. Escenarios de Procesamiento del Ciclo de Vida de una Petición

Una aplicación JavaServer Faces soporta dos tipos diferentes de respuestas y dos tipos diferentes de peticiones; la idea es que en una aplicación JSF se pueden mezclar páginas JSF y JSP (no-JSF) y, según se pidan y/o se devuelvan una u otras, tendremos diferentes respuestas

y/o peticiones:

Tabla III. VI. Escenarios de procesamiento del ciclo de vida de una petición

Respuesta Faces	Una respuesta servlet que se generó mediante la ejecución de la fase Renderizar la Respuesta del ciclo de vida de procesamiento de la respuesta.
Respuesta No – Faces	Una respuesta generada por el servlet en la que no se ha ejecutado la fase Renderizar la Respuesta. Un ejemplo es una página JSP que no incorpora componentes JavaServer Faces.
Petición Faces	Una petición al servlet que fue enviada desde una Respuesta Faces previamente generada. Un ejemplo es un formulario enviado desde un componente de interfaz de usuario JavaServer Faces, donde la URI de la petición identifica el árbol de componentes JavaServer Faces para usar el procesamiento de petición.
Petición No – Faces	Una petición al servlet que fue enviada a un componente de aplicación como un servlet o una página JSP, en vez de directamente a un componente JavaServer Faces.

Fuente: Manual JSF

La combinación de estas peticiones y respuestas resulta en tres posibles escenarios del ciclo de vida que pueden existir en una aplicación JavaServer Faces:

Escenario 1: Una petición No-Faces genera una respuesta Faces: Un ejemplo de este escenario es cuando se pulsa un enlace de una página HTML que abre una página que contiene componentes JavaServer Faces. Para construir una respuesta

Faces desde una petición No-Faces, una aplicación debe proporcionar un mapeo FacesServlet en la URL de la página que contiene componentes

JavaServer Faces. FacesServlet acepta peticiones entrantes y pasa a la implementación del ciclo de vida para su procesamiento

Escenario 2: Una petición Faces genera una respuesta no-Faces: Algunas veces una aplicación JavaServer Faces podría necesitar redirigir la salida a un recurso diferente de la aplicación Web (p.ej. una imagen sencilla) o generar una respuesta que no contiene componentes JavaServer Faces. En estas situaciones, el desarrollador debe saltarse la fase de renderizado (renderizar la respuesta) llamando a FacesContext.responseComplete. FacesContext contiene toda la información asociada con una petición Faces particular. Este método se puede invocar durante las fases aplicar valores de respuesta, procesar validaciones o actualizar los valores del modelo.

Escenario 3: Una petición Faces genera una respuesta Faces: Es el escenario más común en el ciclo de vida de una aplicación JavaServer Faces.

Este escenario implica componentes JavaServer Faces enviando una petición a una aplicación JavaServer Faces utilizando el FacesServlet. Como la petición ha sido manejada por la implementación JavaServer Faces, la aplicación no necesita pasos adicionales para generar la respuesta. Todos los oyentes, validadores y conversores serán invocados automáticamente durante la fase apropiada del ciclo de vida estándar, como se describe en la siguiente sección.

3.8.2. Ciclo de vida estándar en una aplicación JSF

La mayoría de los usuarios de la tecnología JavaServer Faces no necesitarán conocer a fondo el ciclo de vida de procesamiento de una petición. Sin embargo, conociendo lo que la tecnología JavaServer Faces realiza para procesar una página, un desarrollador de aplicaciones JavaServer Faces no necesitará preocuparse de los problemas de renderizado asociados con otras tecnologías UI. Un ejemplo sería el cambio de estado de los componentes individuales: si la selección de un componente, como por ejemplo una casilla de verificación (checkbox) afecta a la apariencia de otro componente de la página, la tecnología JavaServer Faces manejará este evento de la forma apropiada y no permitirá que se dibuje la página sin reflejar este cambio.

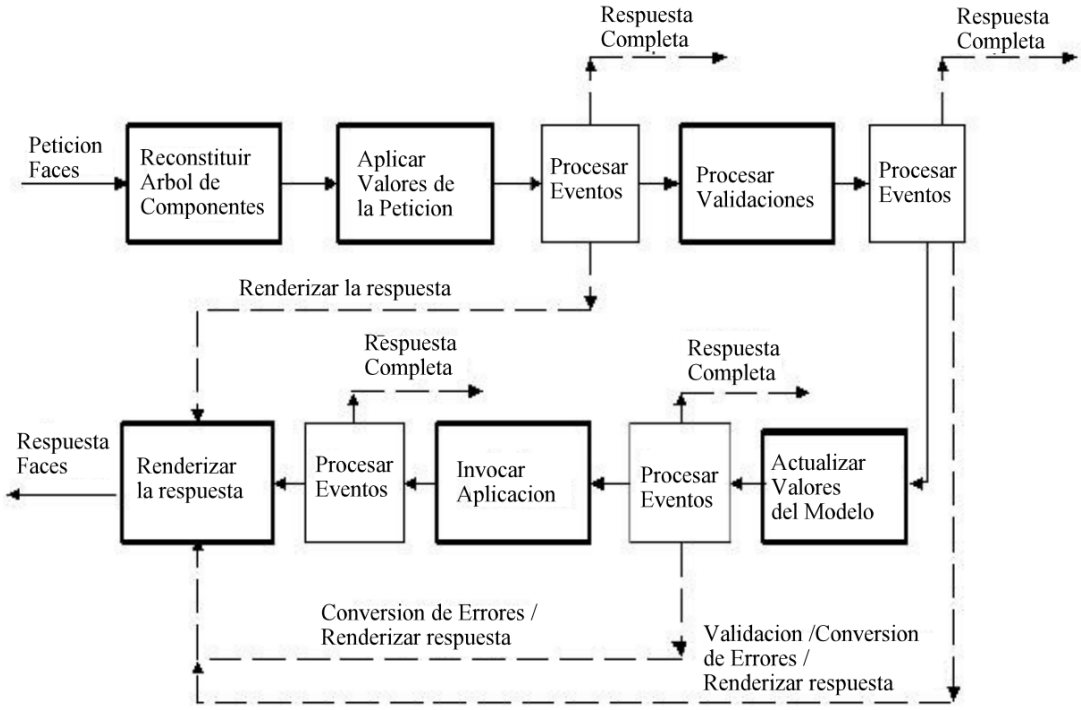


Figura III. 12. Ciclo de vida estándar Petición – Respuesta en JSF

- **Reconstruir el árbol de componentes:**

Cuando se hace una petición de una página JavaServer Faces, o cuando se produce un evento como pulsar sobre un enlace o un botón, el sistema JavaServer Faces entra en el estado reconstituir el árbol de componentes.

Durante esta fase, la implementación JavaServer Faces construye el árbol de componentes de la página JavaServer Faces, conecta los manejadores de eventos y los validadores y graba el estado en el `FacesContext`.

- **Aplicar valores de la petición**

Una vez construido el árbol de componentes, cada componente del árbol extrae su nuevo valor desde los parámetros de la petición con su método `decode`. A continuación, el valor se almacena localmente en el componente. Si la conversión del valor falla, se genera un mensaje de error asociado con el componente y se pone en la cola de `FacesContext`. Este mensaje se mostrará durante la fase renderizar la respuesta, junto con cualquier error de validación resultante de la fase procesar validaciones.

Si durante esta fase se produce algún evento, la implementación JavaServer Faces comunica estos eventos a los oyentes interesados.

En este punto, si la aplicación necesita redirigirse a un recurso de aplicación Web diferente o generar una respuesta que no contenga componentes JavaServer Faces, puede llamar a `FacesContext.responseComplete`.

En este momento, se han puesto los nuevos valores en los componentes y los mensajes y eventos se han puesto en sus colas.

- **Procesar validaciones**

Durante esta fase, el sistema JavaServer Faces procesa todas las validaciones registradas con los componentes del árbol. Examina los atributos del componente especificados por las reglas de validación y evalúa las reglas con los valores de dichos atributos. Si un valor incumple una regla, la implementación JavaServer Faces añade un mensaje de error al `FacesContext` y el ciclo de vida avanza directamente hasta la fase renderizar las respuesta para que la página sea dibujada de nuevo incluyendo los mensajes de error. Si había errores de conversión de la fase aplicar los valores a la petición, también se mostrarán.

- **Actualizar los valores del modelo**

Una vez que la implementación JavaServer Faces determina que el dato es válido, puede pasar por el árbol de componentes y actualizar los valores del modelo con los nuevos valores pasados en la petición. Sólo se actualizarán los componentes que tengan expresiones `valueRef`. Si el dato local no se puede convertir a los tipos especificados por los atributos del objeto del modelo, el ciclo de vida avanza directamente a la fase renderizar las respuesta, durante la que se dibujará de nuevo la página mostrando los errores, similar a lo que sucede con los errores de validación.

- **Invocar la aplicación**

Durante esta fase, la implementación JavaServer Faces maneja cualquier evento a nivel de aplicación, como enviar un formulario o enlazar a otra página.

En este momento, si la aplicación necesita redirigirse a un recurso de aplicación web diferente o generar una respuesta que no contenga componentes JavaServer Faces, puede llamar a `FacesContext.responseComplete`.
Posteriormente, la implementación JavaServer Faces configura el árbol de componentes de la respuesta a esa nueva página y, por último, transfiere el control a la fase Renderizar la Respuesta.

- **Renderizar la Respuesta**

Durante esta fase, la implementación JavaServer Faces invoca los atributos de codificación de los componentes y dibuja los componentes del árbol de componentes grabado en el `FacesContext`.

Si se encontraron errores durante las fases aplicar los valores a la petición, procesar validaciones o actualizar los valores del modelo, se dibujará la página original. Si las páginas contienen etiquetas `output_errors`, cualquier mensaje de error que haya en la cola se mostrará en la página.

Se pueden añadir nuevos componentes en el árbol si la aplicación incluye renderizadores personalizados, que definen cómo renderizar un componente. Después de que se haya renderizado el contenido del árbol, éste se graba para que las siguientes peticiones puedan acceder a él y esté disponible para la fase reconstituir el árbol de componentes de las siguientes llamadas.

En resumen, todas las fases presentan “subfaces“ que pueden quebrantar el flujo normal de la ejecución de una aplicación JSF:

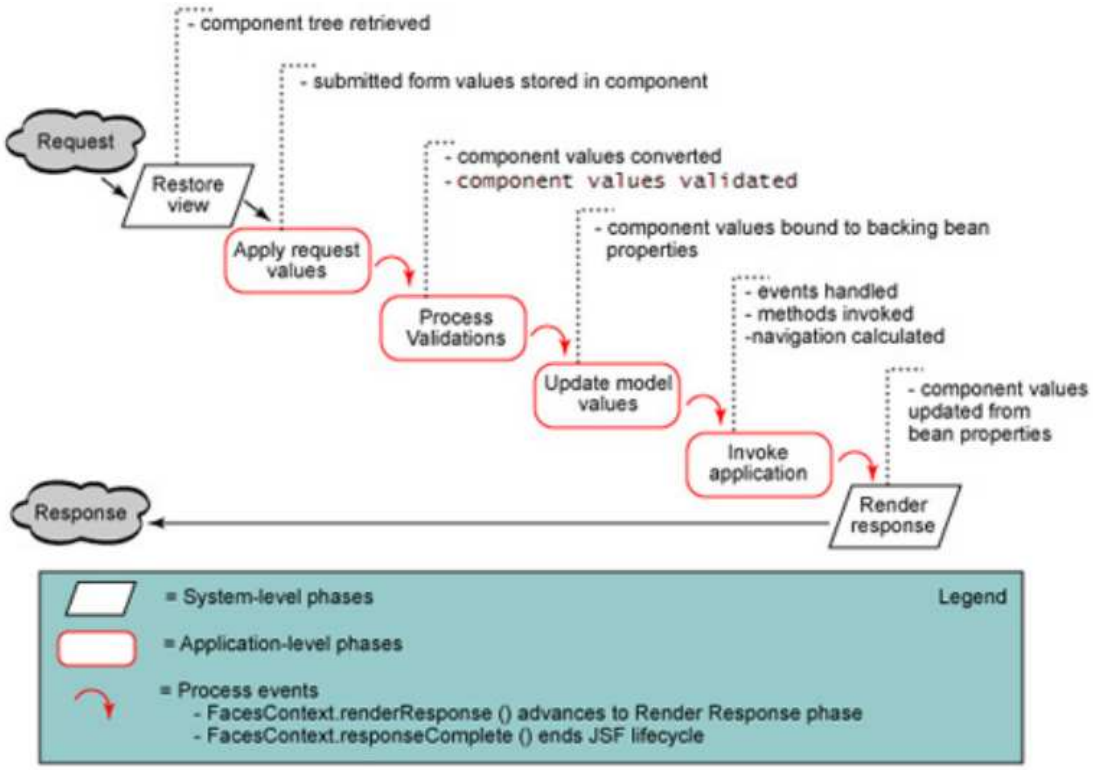


Figura III. 13. Ciclo de vida estándar Petición – Respuesta en una aplicación JSF

3.9. MANAGED BEANS

Un apartado importante en el diseño de aplicaciones web es la separación de la presentación y la lógica de negocio. JSF usa beans para lograr esta separación. Las páginas JSF se refieren a las propiedades del bean, y la lógica de programa está contenida en el código de implementación del bean. Los beans son fundamentales para programar JSF.

3.9.1. Concepto

Un Java bean es “un componente reutilizable del software, que puede ser manipulado”. Como se verá en este capítulo, los beans sirven para una gran variedad de propósito:

A primera vista, un bean parece ser similar a cualquier otro objeto. Sin embargo, los beans se manejan de una forma más concreta. Cualquier objeto se crea y se manipula dentro de un programa Java llamando a los constructores e invocando a los métodos. Sin embargo, los beans pueden ser configurados y manipulados sin programar, a través de entornos de trabajo (frameworks) o entornos de desarrollo integrados (IDE-Integrated Development Environment), que los utilizan mediante técnicas de introspección.

En el contexto de JavaServer Faces, los beans no se utilizan para nada relacionado con la interfaz de usuario: los beans se utilizan cuando se

necesita conectar las clases Java con páginas web o archivos de configuración.

Una vez que un bean ha sido definido, puede ser accedido a través de etiquetas. JSF. Por ejemplo, la siguiente etiqueta lee y actualiza el atributo `password` del `bean usuario`:

```
<h:inputSecret value="#{usuario.password}"/>
```

3.9.2. Atributos

Las características más importantes de un bean son los atributos que posee, también llamados propiedades. Cada uno de éstos tiene:

- Un nombre.
- Un tipo.
- Métodos para obtener y establecer los valores de atributo.

La especificación de los JavaBeans impone una sola exigencia en una clase bean: debe tener un constructor predeterminado, sin parámetros. Además, para que los entornos de trabajo o de desarrollo puedan acceder a sus atributos mediante introspección, una clase bean debe declarar métodos `get` y/o `set` para cada uno de ellos, o debe definir descriptores

utilizando la clave `java.beans.BeanDescriptor`.

Un método `get`… no posee parámetros mientras que un método `set`… posee un parámetro y no devuelve ningún valor. Por supuesto, una clase bean puede tener otros métodos además de los `getter` y `setter`.

3.9.3. Expresiones para valores inmediatos y directos

Muchos componentes de interfaz de usuario JSF tienen un valor de atributo que permite especificar ya sea un valor inmediato o un valor directo obtenido del atributo de un bean. Por ejemplo, se puede especificar un valor inmediato de la forma:

```
<h:outputText value="Hola mundo!"/>
```

o se puede especificar un valor directo:

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

3.9.4. Ámbitos de los beans

Para comodidad del programador aplicaciones web, un contenedor de servlets suministra diferentes ámbitos, de `petición`, de `sesión` y de `aplicación`.

Estos ámbitos normalmente mantienen beans y otros objetos que necesitan estar disponibles en diferentes componentes de una aplicación web.

Ámbito de petición: Es el de vida más corta. Empieza cuando una petición HTTP comienza a tramitarse y acaba cuando la respuesta se envía al cliente.

Por ejemplo la siguiente línea de código:

```
<f:loadBundle basename="mensajes" var="msjs"/>
```

La etiqueta `f:loadBundle` hace que la variable `msjs` solo exista mientras dura la petición. Un objeto debe tener un ámbito de este tipo sólo si lo que se quiere es reenviarlo a otra fase de procesado.

Ámbito de Sesión: El navegador envía una petición al servidor, el servidor devuelve una respuesta, y entonces ni el navegador ni el servidor tiene cualquier obligación para conservar cualquier memoria de la transacción.

Este acomodamiento simple marcha bien para recuperar información básica, pero es poco satisfactorio para aplicaciones del lado del servidor.

Por ejemplo, en una aplicación de un carrito compras, necesita al servidor para recordar los contenidos del carrito de compras.

Por esa razón, los contenedores servlet amplían el protocolo de HTTP para seguir la pista a una sesión, esto se consigue repitiendo conexiones para el mismo cliente. Hay diversos métodos para el rastreo de sesión . El método más simple es usar cookies: La pareja nombre/valor la envía el servidor a un cliente, esperando que regresen en subsiguientes peticiones.

Mientras el cliente no desactive las cookies, el servidor recibe un identificador de sesión por cada petición siguiente. Los servidores de aplicación usan estrategias de retirada, algo semejante como al URL rewriting, para tratar con esos clientes que no devuelven cookies. El URL rewriting añade un identificador de sesión a la URL, con lo cual se parece algo a esto:

```
http://ejemploBasico/index.jsp;jsessionid=64C28D1F
C...D28
```

La sesión el rastreo con cookies es completamente transparente al desarrollador web, y las etiquetas estándar JSF automáticamente reescriben URL si un cliente no usa cookies.

El ámbito de sesión permanece desde que la sesión es establecida hasta que esta termina. Una sesión termina si la aplicación web invoca el método `invalidate` en el objeto `HttpSession` o si su tiempo expira.

Las aplicaciones Web típicamente colocan la mayor parte de sus bean dentro de un ámbito de sesión.

Por ejemplo, un bean UsuarioBean puede contener información acerca de usuarios que son accesibles a lo largo de la sesión entera. Un bean CarritoCompraBean puede irse llenando gradualmente durante las demandas que levantan una sesión.

Ámbito de aplicación: Persiste durante toda la aplicación web. Este ámbito compartido entre todas las peticiones y sesiones.

Existe la posibilidad de anidar beans, para conseguir objetivos más complicados.

Considere el siguiente bean como ejemplo:

```
<managed-bean>
   <managed-bean-name>examen</managed-bean-name>
   <managed-bean-class>ExamenBackingBean</managed-bean-class>
   <managed-bean-scope>session</managed-bean-scope>
  <managed-property>
    <property-name>problemas</property-name>
        <list-entries>
                <value-class>ProblemasBean</value-class>
                <value>#{problema1}</value>
                <value>#{problema2}</value>
                <value>#{problema3}</value>
                <value>#{problema4}</value>
                <value>#{problema5}</value>
        </list-entries>
  </managed-property>
</managed-bean>
```

Ahora definimos objetos beans con nombres desde problema1 hasta problema5:

```
<managed-bean>
   <managed-bean-name>problema1</managed-bean-name>
   <managed-bean-class>ProblemaBean</managed-bean-class>
   <managed-bean-scope>none</managed-bean-scope>
  <managed-property>
    <property-name>sequence</property-name>
    <list-entries>
      <value-class>java.lang.Integer</value-class>
      <value>3</value>
      <value>1</value>
      <value>4</value>
    </list-entries>
  </managed-property>
</managed-bean>
```

Cuando el bean examen se referencia por vez primera, se dispara automáticamente la creación de los beans problema1 .. problema5, sin que haya que preocuparse por el orden en el cual se han especificado. Note que estos beans (problema1 .. problema5) no tienen ningún ámbito (none), puesto que nunca son demandados desde una página JSP.

Cuando junte ámbitos de beans, hay que asegurarse de que son compatibles, tal y como se muestra en el siguiente cuadro:

Tabla III. VII. Escenarios de procesamiento del ciclo de vida de una petición

CUANDO DEFINA UN BEAN DE ÁMBITO…	PUEDE USAR OTRO ÁMBITO DE TIPO…
None	None
Application	None, application
Session	None, application, session
Request	None, application, session, requetst

- **Configuración de un bean a través de xml**

El archivo de configuración más comúnmente usado es: `WEB-INF/faces-config.xml`.

Un bean se define con una etiqueta `managed-bean` al comienzo del fichero `faces-config.xml`. Básicamente se debe especificar nombre del bean con la etiqueta `<managed-bean-name>`, clase donde está recogida dicho bean, con la etiqueta `<managed-bean-class>` y ámbito del bean con la etiqueta `<managed-bean-scope>`, por ejemplo, para definir un bean llamado usuario, que está en la clase UsuarioBean y con un ámbito de esión, sería:

```
<managed-bean>
  <managed-bean-name>usuario</managed-bean-name>
 <managed-bean-class>UsuarioBean</managed-bean-class>
   <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

3.10. NAVEGACIÓN

Las aplicaciones JavaServer Faces usan las reglas de navegación para controlar la navegación entre páginas. Cada regla de navegación especifica cómo ir de una página a las demás dentro de la aplicación. En la arquitectura MVC, la navegación de la página es una de las responsabilidades del controlador. Las reglas de navegación de las aplicaciones JSF están contenidas en el archivo `faces-config.xml` bajo el directorio `WEB-INF`.

3.10.1. Concepto

Es la navegación de la aplicación web, de manera que se pase de una página a otra cuando se pulsa un botón o se realiza cualquier otra acción por parte del usuario.

Para empezar, existen dos tipos diferenciados de navegación: navegación estática y dinámica.

3.10.2. Navegación Estática

Considere el caso en el que un usuario rellena un formulario de una página web. El usuario puede escribir en campos del texto, puede hacer clic sobre enlaces, pulsar botones o seleccionar elementos de una lista, entre otras muchas cosas.

Todas estas acciones ocurren dentro del navegador del cliente. Cuando, por ejemplo, el usuario pulsa un botón, envía los datos del formulario y éstos son gestionados por el servidor.

Al mismo tiempo, el servidor JSF analiza la entrada del usuario y debe decidir a qué página ir para dar la respuesta.

En una aplicación web simple, la navegación es estática. Es decir, pulsar sobre un botón suele redirigir al navegador a una misma página para dar la respuesta. En este caso, simplemente, a cada botón se le da un valor para su atributo de acción (`action`), por ejemplo:

```
<h:commandButton label="Aceptar" action="login"/>
```

Esta acción desencadenante, debe concordar con la etiqueta `outcome` del fichero `faces-config.xml`, dentro de sus reglas de navegación.

En esta simple regla de navegación, se indica que tras la acción `login`, se navegará a lapágina `hola.jsp`, si esta acción ocurre dentro de la página `index.jsp`. Tenga cuidado con no olvidar la `/`, en la lineas `from-view-id` y `to-view-id.`

Puede mezclar reglas de navegación con el mismo `from-view-id`, por ejemplo:

```
<navigation-rule>
   <from-view-id>/index.jsp</from-view-id>
   <navigation-case>
     <from-outcome>login</from-outcome>
     <to-view-id>/hola.jsp</to-view-id>
   </navigation-case>
   <navigation-case>
     <from-outcome>signup</from-outcome>
     <to-view-id>/adios.jsp</to-view-id>
   </navigation-case>
 </navigation-rule>
```

En este caso, para la página `index.jsp`, si el usuario produce la acción de log in, es redirigido a la página `hola.jsp`, y si se desconecta, es redirigido a la página `adiós.jsp`.

3.10.3. Navegación dinámica

En la mayoría de aplicaciones web, la navegación no es estática. El flujo de la página no depende de qué botón se pulsa, sino que también depende de los datos que el cliente introduce en un formulario. Por ejemplo, una página de entrada al sistema puede tener dos resultados: El éxito o el fracaso.

El resultado depende de una computación, sea cual sea el nombre y la contraseña es legítima. Para implementar una navegación dinámica, el botón de aceptar debe tener un método referencia, por ejemplo:

```
<h:commandButton label="Aceptar"
        action="#{loginControlador.verificarUsuario}"/>
```

En este caso, `loginControlador`, referencia un bean, y éste debe tener un método denominado `verificarUsuario`.

Un método de referencia, en un atributo de acción, no tiene parámetros de entrada y devuelve una cadena de caracteres, que será una usada para activar una regla de navegación, por ejemplo, el método `verificarUsuario` debería parecerse a algo así:

```
public String verificarUsuario() {
    if (...)
        return "éxito";
    else
        return "fracaso";
}
```

El método devuelve un String, "éxito" o "fracaso". El manejador de navegación usa el string devuelto para buscar una regla de navegación que coincida.

De manera que en las reglas de navegación, podría encontrarse algo así:

```
<navigation-case>
     <from-outcome>exito</from-outcome>
     <to-view-id>/exito.jsp</to-view-id>
 </navigation-case>
<navigation-case>
     <from-outcome>fracaso</from-outcome>
     <to-view-id>/fracaso.jsp</to-view-id>
 </navigation-case>
```

3.10.4. Navegación avanzada

A continuación, se describen las reglas faltantes para los elementos de navegación que pueden aparecer en el fichero `faces-config.xml`. En la figura xxx se puede ver el diagrama de sintaxis de etiquetas válidas.

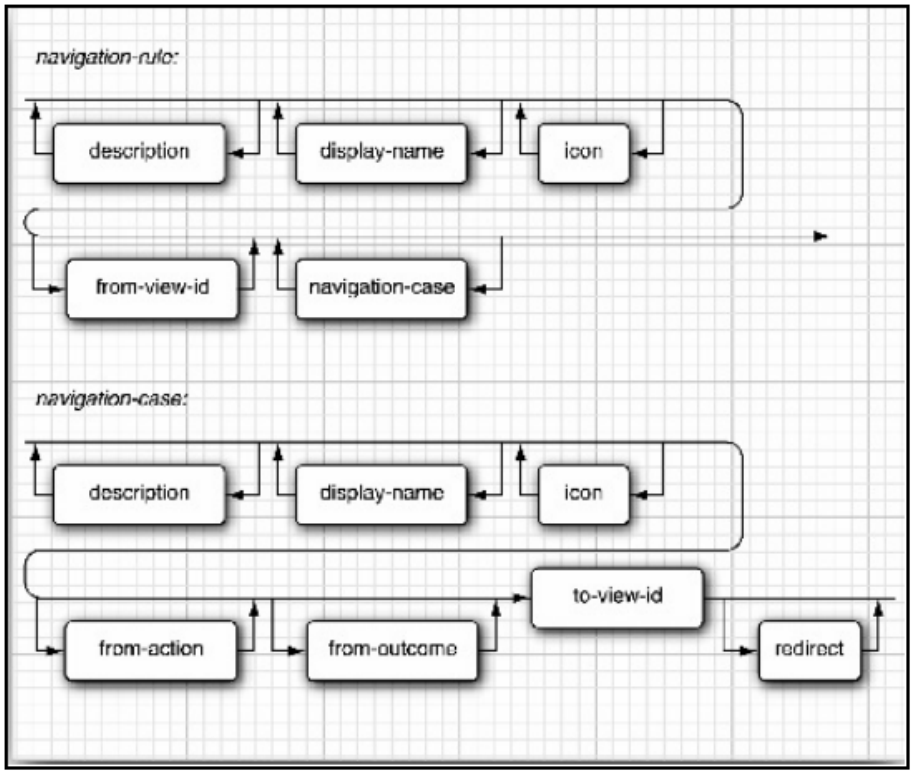


Figura III. 14. Diagrama de sintaxis de los elementos de navegación

3.10.5. Redirección

Si se añade una etiqueta `redirect`, después de `to-view-id`, el contenedor de JSP termina la petición actual y envía una redirección HTTP al cliente.

Con `<redirect>` se le dice a JSF que envíe una redireccion HTTP al usuario de una vista nueva. En el mundo de JSP, esto quiere decir que el usuario ve el URL de la página que él actualmente mira, en contra de la dirección de la página previa.

Por ejemplo, nos encontramos actualmente en la página index.jsp, y tenemos la siguiente regla de navegación:

```
<navigation-case>
     <from-outcome>exito</from-outcome>
     <to-view-id>/exito.jsp</to-view-id>
 </navigation-case>
```

La respuesta redireccionada dice al cliente que URL usar para la siguiente página. Sin redirección, la dirección original (`localhost:8080/ejemplo/index.faces`) es la misma cuando el usuario se muda de la página `index.jsp` a `exito.jsp`. Con redirección, el navegador establece la dirección nueva (`localhost:8080/ejemplo/exito.faces`).

3.10.6. Comodines

Se pueden usar los comodines, en las etiquetas `from-view-id` de las reglas de navegación; por ejemplo:

```
<navigation-rule>
   <from-view-id>/aplicacion/*</from-view-id>
   <navigation-case>
     ...
   </navigation-case>
</navigation-rule>
```

Esta regla se aplicará a todas aquellas páginas que empiecen con el prefijo aplicación. Solo se permite un *, y debe estar al final de la cadena del `from-view-id`.

3.10.7. Etiqueta from – action

Junto con la etiqueta `from-outcome`, está también la etiqueta `from-acción`. Esto puede ser útil si se tienen dos acciones separadas con la misma cadena de acción, o dos métodos de referencia de acción que devuelven la misma cadena.

Por ejemplo, suponga que tiene dos métodos, accionRespuesta y nuevoExamen, y ambos devuelven la misma cadena, “repetir”, pues, para diferenciar ambos casos de navegación, se usa el elemento `from-action`:

```
<navigation-case>
     <from-action>#{examen.accionRespuesta}</from-action>
               <from-outcome>repetir</from-outcome>
     <to-view-id>/repetir.jsp</to-view-id>
 </navigation-case>
<navigation-case>
     <from-action>#{examen.nuevoExamen}</from-action>
               <from-outcome>repetir</from-outcome>
     <to-view-id>/idex.jsp</to-view-id>
 </navigation-case>
```

3.11. ETIQUETAS BÁSICAS JSF

Los componentes UI JavaServer Faces son elementos configurables y reutilizables que componen el interface de usuario de las aplicaciones JavaServer Faces. Un componente puede ser simple, como un botón, o compuesto, como una tabla, que puede estar compuesta por varios componentes.

La tecnología JavaServer Faces proporciona una arquitectura de componentes rica y flexible que incluye:

- Un conjunto de clases `UIComponent` para especificar el estado y comportamiento de componentes UI.
- Un modelo de renderizado que define cómo renderizar los componentes de diferentes formas.
- Un modelo de eventos y oyentes que define cómo manejar los eventos de los componentes.
- Un modelo de conversión que define cómo conectar conversores de datos a un componente.
- Un modelo de validación que define cómo registrar validadores con un componente

La tecnología JavaServer Faces proporciona un conjunto de clases de componentes UI, que especifican toda la funcionalidad del componente, cómo mantener su estado, mantener una referencia a objetos del modelo, y dirigir el manejo de eventos y el renderizado para un conjunto de componentes estándar.

Estas clases son completamente extensibles, lo que significa que podemos extenderlas para crear nuestros propios componentes personalizados.

Todas las clases de componentes UI de JavaServer Faces descienden de la clase UIComponentBase, que define el estado y el comportamiento por defecto de un

UIComponent.

El conjunto de clases de componentes UI incluido en la versión 1,2 de JavaServer Faces es:

- **UICommand**: Representa un control que dispara actions cuando se activa.
- **UIForm**: Encapsula un grupo de controles que envían datos de la aplicación. Este componente es análogo a la etiqueta form de HTML.
- **UIGraphic**: Muestra una imagen.
- **UIInput**: Toma datos de entrada del usuario. Esta clase es una subclase de UIOutput.
- **UIOutput**: Muestra la salida de datos en un página.
- **UIPanel**: Muestra una tabla..
- **UISelectItem**: Representa un sólo ítem de un conjunto de ítems.
- **UISelectItems**: Representa un conjunto completo de ítems.
- **UISelectBoolean**: Permite a un usuario seleccionar un valor booleano en un control, seleccionándolo o deseleccionándolo. Esta clase es una subclase de UIInput.
- **UISelectMany**: Permite al usuario seleccionar varios ítems de un grupo de ítems. Esta clase es una subclase de UIInput.

- **UISelectOne**: Permite al usuario seleccionar un ítem de un grupo de ítems. Esta clase es una subclase de UIInput.

La mayoría de los autores de páginas y de los desarrolladores de aplicaciones no Tendrán que utilizar estas clases directamente. En su lugar, incluirán los componentes en una página usando la etiqueta correspondiente al componente. La mayoría de estos componentes se pueden renderizar de formas diferentes. Por ejemplo, un UICommand se puede renderizar como un botón o como un hiperenlace.

La implementación de referencia de JavaServer Faces proporciona una librería de etiquetas personalizadas para renderizar componentes en HTML.

Además, se puede usar conjuntamente las distintas implementaciones de etiquetas para renderizar componentes JSF, tanto con la lógica de presentación de JSP o facelets, Como se verá a continuación:

- **UIForm: Form:** representa un formulario.

```
<h:form
  ...
</h:form>
```

- **UICommand: commandButton:** un botón con una acción asociada.

```
<h:commandButton
        value="Aceptar"
        action="siguiente"
/>
```

- **UIGraphic: graphicImage:** Muestra una imagen.

```
<h:graphicImage
        value="/imagenes/duke.gif"
/>
```

- **UIInput: inputText:** Permite al usuario introducir un string.

```
<h:inputText
 value="#{formulario.nombre}"
/>
```

- **UIInput: inputSecret:** Permite al usuario introducir un string sin que aparezca el string real en el campo.

```
<h:inputSecret
 value="#{formulario.password}"
/>
```

- **UIInput: inputTextArea:** Permite al usuario introducir un string sin que aparezca el string real en el campo.

```
<h:inputTextArea  rows="3"
        cols="15"
        value="Escribir aqui.."
/>
```

Escribir
aqui..

- **UIOutput: outputText:** Muestra una línea de texto.

```
<h:outputText
 value="Hola Mundo!"
/>
```

Hola Mundo!

- **UIOutput: outputLink:** Muestra un enlace.

```
<h:outputLink
        value="http://www.hoylodejo.com/>
        <h:outputText value="Clic aquí"/>
</h:outputLink>
```

Clic aquí

- **UISelectBoolean: selectBooleanCheckbox:** Permite al usuario cambiar el valor de una elección boolean.

```
<h:selectBooleanCheckbox
value="#{cuestionario.recibirInformacion
}"/>
```

☑ Deseo recibir informacion

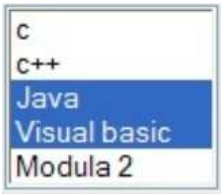
- **UISelectMany: selectManyCheckbox:** Muestra un conjunto de checkbox, en los que el usuario puede seleccionar varios.

```
<h:selectManyCheckbox
        value="#{cuestionario.aficiones}">
   <f:selectItem itemValue="musica"
        itemLabel="Musica"/>
   <f:selectItem itemValue="cine"
        itemLabel="Cine"/>
   <f:selectItem itemValue="pesca"
        itemLabel="Pesca"/>
   <f:selectItem itemValue="deporte"
        itemLabel="Deporte"/>
   <f:selectItem itemValue="lectura"
        itemLabel="Lectura"/>
</h:selectManyCheckbox>
```

☑ Musica ☐ Cine ☑ Pesca

- **UISelectMany: selectManyListbox:** Permite al usuario seleccionar varios elementos de una lista de elementos.

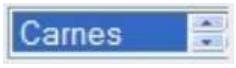
```
<h:selectManyListbox
        value="#{cuestionario.lenguajes}">
   <f:selectItem itemValue="c"
        itemLabel="c"/>
   <f:selectItem itemValue="c++"
        itemLabel="c++"/>
   <f:selectItem itemValue="java"
        itemLabel="Java"/>
   <f:selectItem itemValue="Visual basic"
        itemLabel="Visual basic"/>
   <f:selectItem itemValue="modula2"
        itemLabel="Modula 2"/>
```



```
</h:selectManyListbox>
```

- **UISelectMany: selectManyMenu:** Permite al usuario seleccionar varios elementos de una lista de elementos.

```
<h:selectManyMenu
        value="#{cuestionario.comidas}">
   <f:selectItem itemValue="carnes"
        itemLabel="Carnes"/>
   <f:selectItem itemValue="pescados"
        itemLabel="Pescados"/>
   <f:selectItem itemValue="legumbres"
        itemLabel="Legumbres"/>
   <f:selectItem itemValue="pastas"
        itemLabel="Pastas"/>
   <f:selectItem itemValue="sopas"
        itemLabel="Sopas"/>
</h:selectManyMenu>
```



- **UISelectOne: selectOneRadio:** Permite al usuario seleccionar un elemento de un grupo de elementos.

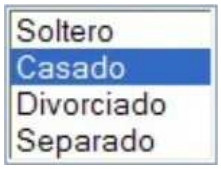
```
<h:selectOneRadio
        value="#{cuestionario.fumador}">
    <f:selectItem itemValue="si"
        itemLabel="Fumador"/>
    <f:selectItem itemValue="no"
        itemLabel="No fumador"/>
</h:selectOneRadio>
```

- **UISelectOne: selectOneMenu:** Permite al usuario seleccionar un elemento de un grupo de elementos.

```
<h:selectOneMenu
        value="#{cuestionario.sistema}">
   <f:selectItem itemValue="windows 98"
        itemLabel="Windows 98"/>
   <f:selectItem itemValue="windows xp"
        itemLabel="Windows XP"/>
   <f:selectItem itemValue="suse"
        itemLabel="Suse"/>
   <f:selectItem itemValue="red hat"
        itemLabel="Red Hat"/>
   <f:selectItem itemValue="ubuntu"
        itemLabel="Ubuntu"/>
</h:selectOneMenu>
```

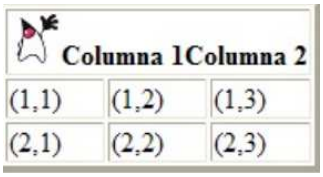
- **UISelectOne: selectOneListbox:** Permite al usuario seleccionar un elemento de una lista de elementos.

```
<h:selectOneListbox
        value="#{cuestionario.estado}">
    <f:selectItem itemValue="soltero"
        itemLabel="Soltero"/>
    <f:selectItem itemValue="casado"
        itemLabel="Casado"/>
    <f:selectItem itemValue="divorciado"
        itemLabel="Divorciado"/>
    <f:selectItem itemValue="separado"
        itemLabel="Separado"/>
</h:selectOneListbox>
```



- **UIPanel: panelGrid y panelGroup:** Muestra una tabla, con varios paneles donde se recogen diversos componentes.

```
<h:panelGrid columns="3" border="4">
   <h:panelGroup >
        <h:graphicImage url="Duke.gif"/>
        <h:outputText value="Columna 1"/>
        <h:outputText value="Columna 2"/>
   </h:panelGroup>
          <h:outputText value="(1,1)"/>
          <h:outputText value="(1,2)"/>
          <h:outputText value="(1,3)"/>
          <h:outputText value="(2,1)"/>
          <h:outputText value="(2,2)"/>
          <h:outputText value="(2,3)"/>
</h:panelGrid>
```



CAPÍTULO IV

4. ANÁLISIS COMPARATIVO

4.1. INTRODUCCIÓN

J2EE es una plataforma para desarrollo de aplicaciones empresariales, escalable, robusta, multiplataforma, madura y muy bien documentada. A pesar de esto, desde el punto de vista del programador escribir aplicaciones usando la API de Servlets, JSP y EJB es una tarea tediosa, poco amigable y de baja productividad.

Por otro lado, la aplicación resultante, en términos generales es poco estructurada, basada en componentes de baja reusabilidad y difícil de mantener.

Los frameworks J2EE facilitan el desarrollo de aplicaciones empresariales, reducen el tiempo involucrado en el proceso de desarrollo y mejoran notablemente la

calidad del software resultante. Los programadores pueden dedicarse a resolver los problemas específicos de la lógica del negocio, dejando de lado los detalles de programación de bajo nivel.

El presente capítulo tiene como objetivo fundamental, analizar y evaluar cada una de las tecnologías líderes en el desarrollo de aplicaciones web en ambiente Java Empresarial: JavaServer Faces y Struts, que han sido seleccionados para el estudio comparativo.

Un framework, es una estructura software compuesta de componentes personalizables e intercambiables que permiten desarrollar una aplicación. En otras palabras, un framework se puede considerar como una aplicación genérica incompleta y configurable a la que podemos agregarle algunas piezas para construir una aplicación concreta.

Los dos frameworks web que compararemos en este artículo, responden al patrón de diseño MVC (Modelo Vista Controlador) para web, también conocido como MVC Modelo 2. Básicamente este patrón es una guía para el diseño de arquitecturas de aplicaciones que ofrecen una fuerte interactividad con usuarios.

4.2. Determinación de las herramientas a comparar

En la actualidad existen muchos frameworks diseñados para apoyar en el desarrollo de proyectos o aplicaciones web usando el lenguaje de programación JAVA.

Los frameworks seleccionadas para el análisis son JSF y Struts, a continuación ofrecemos una descripción rápida de las mismas:

4.2.1. JavaServer Faces

Es una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE. JSF usa JavaServer Pages (JSP) como la tecnología que permite hacer el despliegue de las páginas, y en sus versiones actuales, solo soporta Faceletes, pero también se puede acomodar a otras tecnologías como XUL.

El objetivo de la tecnología JavaServer Faces es desarrollar aplicaciones web de forma parecida a como se construyen aplicaciones locales con Java Swing, AWT (Abstract WindowToolkit), SWT (Standard Widget Toolkit) o cualquier otra API similar.

Al igual que Struts, JSF pretende normalizar y estandarizar el desarrollo de aplicaciones web. Hay que tener en cuenta JSF es posterior a Struts, y por lo tanto se ha nutrido de la experiencia de este, mejorando algunas sus deficiencias. De hecho el creador de Struts (Craig R. McClanahan) también es líder de la especificación de JSF.

4.2.2. Struts

Struts Framework es un estándar para desarrollo de aplicaciones web, igual que la mayoría de frameworks para aplicaciones web, es una implementación de la arquitectura MVC.

Struts es una herramienta de soporte para el desarrollo de aplicaciones Web bajo la plataforma Java EE (Java Enterprise Edition). Struts se desarrollaba como parte del proyecto Jakarta de la Apache Software Foundation, pero actualmente es un proyecto independiente conocido como Apache Struts.

Struts permite reducir el tiempo de desarrollo. Su carácter de "software libre" y su compatibilidad con todas las plataformas en las que Java Entreprise esté disponible lo convierten en una herramienta altamente disponible.

Con la versión 2 del framework se introdujeron algunas mejoras sobre la primera versión, de cara a simplificar las tareas más comunes en el desarrollo de aplicaciones web, así como mejorar su integración con AJAX, etc.

4.3. Análisis de las herramientas

En la siguiente documentación se expresa en detalle todo lo necesario para explicar cada uno de los frameworks que se ha tomado como referencia anteriormente para el análisis comparativo.

4.3.1. JSF

- **Introducción**

A diferencia de Struts y de cualquier otro framework web J2EE, JavaServer Faces es el único que tiene una especificación creada por el Java Community Process (JCP), esto lo transforma en un estándar y como principal consecuencia todas las implementaciones, tanto las de código fuente abierto como las comerciales deben respetarla. Por otro lado, JSF 2, la versión actual del framework, forma parte de la especificación J2EE 6.0 con lo cual todas las implementaciones de servidores J2EE 6.0 lo deben soportar.

- **Características principales**

Una API y una implementación de referencia para:

- Representar componentes de interfaz de usuario (UI-User Interface) y manejar su estado.

- Manejador de eventos, validar en el lado del servidor y convertir datos.
- Definir la navegación entre páginas.
- Soportar internacionalización y accesibilidad.
- Proporcionar extensibilidad para todas estas características.
- Una librería de etiquetas JavaServer Pages (JSP) personalizadas para dibujar c omponentes UI dentro de una página JSP o facelets.

Este modelo de programación bien definido y la librería de etiquetas para componentes UI facilita de forma significativa la tarea de la construcción y mantenimiento de aplicaciones web con UIs en el lado servidor.

- **Arquitectura**

JSF está basado en el patrón MVC (Modelo Vista Controlador), nos permite separar la lógica de control (qué cosas hay que hacer pero no cómo), la lógica de negocio (cómo se hacen las cosas) y la lógica de presentación (cómo interaccionar con el usuario).

Utilizando este tipo de patrón es posible conseguir más calidad, un mantenimiento más fácil, existe un patrón de partida por el que empezar un proyecto, etc. al margen de todo esto, una de las cosas más importantes que permite el uso de este patrón consiste en normalizar y estandarizar el desarrollo de Software.

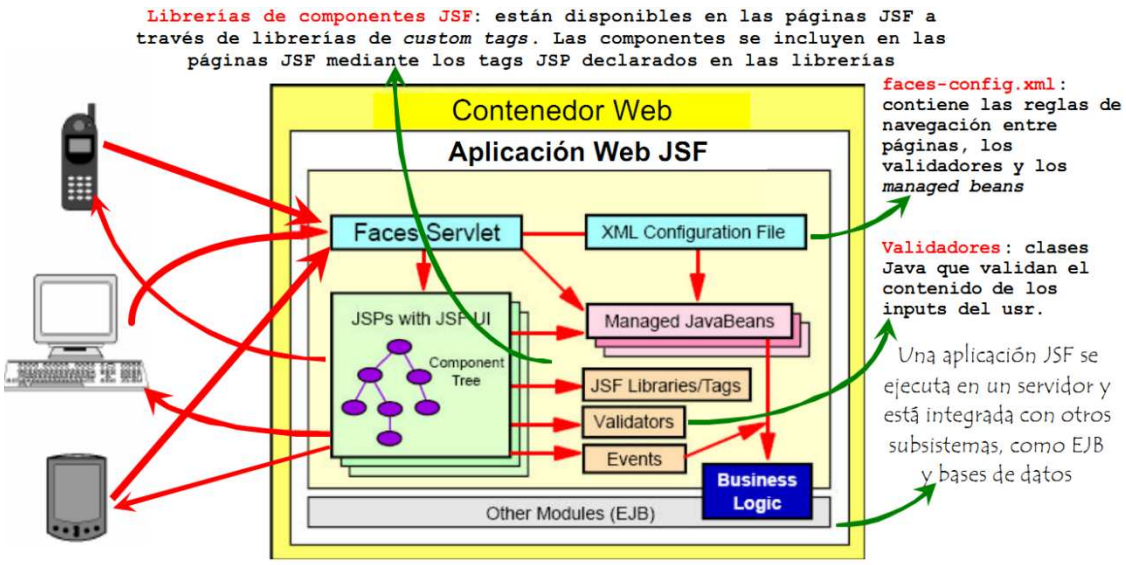


Figura IV. 15. Arquitectura JSF

- **Componentes de la tecnología JSF:**

Las páginas jsf:

Son ficheros en formato html extendido o dinámico (xhtml), que contiene el lenguaje de marcado definido en la especificación de JSF, para definir os componentes de la interfaz del usuario, formalmente, al compilar este fichero jsf, el framework convertirá la definición de componentes en lenguaje de marcado jsf a lenguaje de marcado html normal, y generará código javascript necesario, si es el caso.

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<f:loadBundle basename="jsfks.bundle.messages" var="msg"/>
<html>
 <head>
  <title>enter your name page</title>
 </head>
 <body>
   <f:view>
     <h1>
      <h:outputText value="#{msg.inputname_header}"/>
     </h1>
     <h:form id="helloForm">
      <h:outputText value="#{msg.prompt}"/>
      <h:inputText value="#{personBean.personName}" />
      <h:commandButton action="greeting" value="#{msg.button_text}"/>
     </h:form>
   </f:view>
 </body>
</html>
```

Para utilizar un componente JSF en cualquier página se requieren dos pasos:

- Incluir en la primera línea una directiva que nos indica en donde encontrar etiquetas JSF que definen elementos HTML, la segunda directiva nos dice dónde encontrar etiquetas JSF que definen elementos JSF estándar. La tercera línea carga las propiedades del archivo que contiene mensajes que se desean mostrar en la página JSP.

- Se agregan los componentes con sus correspondientes prefijos, nombres y atributos, los cuales son enlazados con sus valores a través de EL (Lenguaje de Expresiones). En el resto de las líneas del ejemplo se muestran componentes

Expression Language (EL)

Es un lenguaje de scripting que permite el acceso a componentes Java (JavaBeans) a través de JSP . Desde JSP 2.0, se ha utilizado dentro de JSP

etiquetas para separar el código Java y código JSP, y para permitir un acceso más fácil a los componentes de Java.

La evolución de EL hizo este tipo de scripts más fácil para los diseñadores de contenido web que tienen poco o prácticamente ningún conocimiento de Java.

Lax espresiones EL hacen a jsp un verdadero lenguaje de Script. Antes de EL, JSP consistía en algunas etiquetas especiales conocidas como scriptlets, dentro de las cuales el código de Java era escrito explícitamente.

Ficheros de configuración de aplicación JSF: faces-config.xml , web.xml

faces-config.xml

`Faces-config.xml` es el fichero es donde configuramos JSF. Es como el "pegamento" que une modelo, vista y controlador. El equivalente en `Struts` sería el fichero `struts-config.xml`.

Se encuentra dentro de la carpeta: `WebContent/WEB-INF`.

En este fichero por un lado declaramos los `beans` que vamos a utilizar para recoger la información de los formulario (en el `struts-config.xml` teníamos una sección similar), y por otro lado las reglas de

navegación (en el struts-config-xml se podría comparar con la definición de forwards de las acciones).

Web.xml

JSF require de un fichero de configuración central "web.xml", ubicado en el directorio WEB-INF de la aplicación web. Este fichero es similar a otras aplicaciones web que están basadas en los servlets.

En este fichero se debe especificar que "FacesServlet" es el responsable para manejar la aplicación JSF. "FacesServlet" es el controlador central de la aplicación JSF. "FacesServlet" recibe todas las solicitudes para la aplicación JSF e inicializa los componentes en el fichero jsp o facelets antes de ser desplegados.

Beans

Un apartado importante en el diseño de aplicaciones web es la separación de la presentación y la lógica de negocio. JSF usa beans para lograr esta separación. Las páginas JSF se refieren a las propiedades del bean, y la lógica de programa está contenida en el código de implementación del bean. Los beans son fundamentales para programar JSF.

A primera vista, un bean parece ser similar a cualquier otro objeto. Sin embargo, los beans se manejan de una forma más concreta. Cualquier

objeto se crea y se manipula dentro de un programa Java llamando a los constructores e invocando a los métodos. Sin embargo, los beans pueden ser configurados y manipulados sin programar, a través de entornos de trabajo (frameworks) o entornos de desarrollo integrados (IDE-Integrated Development Environment), que los utilizan mediante técnicas de introspección.

En el contexto de JavaServer Faces, los beans no se utilizan para nada relacionado con la interfaz de usuario: los beans se utilizan cuando se necesita conectar las clases Java con páginas web o archivos de configuración.

Reglas de navegación:

Las aplicaciones JavaServer Faces usan las reglas de navegación para controlar la navegación entre páginas. Cada regla de navegación especifica cómo ir de una página a las demás dentro de la aplicación. En la arquitectura MVC, la navegación de la página es una de las responsabilidades del controlador. Las reglas de navegación de las aplicaciones JSF están contenidas en el archivo faces-config.xml bajo el directorio WEB-INF.

- **Ventajas**

  - El código JSF con el que creamos las vistas (etiquetas jsp) es muy parecido al HTML estándar. Lo pueden utilizar fácilmente desarrolladores y diseñadores web.

  - JSF se integra dentro de la página JSP y se encarga de la recogida y generación de los valores de los elementos de la página, así como también en páginas facelets.

  - JSF resuelve validaciones, conversiones, mensajes de error e internacionalización (i18n)

  - JSF permite introducir javascript en la página, para acelerar la respuesta de la interfaz en el cliente (navegador del usuario).

  - JSF es extensible, por lo que se pueden desarrollar nuevos componentes a medida, También se puede modificar el comportamiento del framework mediante APIs que controlan su funcionamiento.

- **Desventajas**

  - Lenta evolución

  - Los validadores no son tan completos como en `Struts`. Esto se puede solucionar fácilmente con validadores propios (JSF da una

forma bastante intuitiva de crearlos) o si no existen validadores de terceros, ejemplo: validadores de MyFaces.

- No tiene validación del lado del cliente. Esto se puede remediar usando el framework Shale, sin embargo, a menos que la validación sea estrictamente en el cliente y no en el servidor, no se necesita más de lo que JSF ofrece.

- No soporta GET, lo cual da cierta seguridad, pero quita flexibilidad y en algunos casos "direccionabilidad", lo cual puede ser muy molesto.

- Por defecto, la implementación 1.2 de JSF es un poco lenta en relación a otros frameworks web, especialmente si se los compara con frameworks php, sin embargo, Oracle, (antes Sun) está trabajando para mejorar esto, y la implementación 2.0 de JSF es casi 30% más rápida que la versión 1.2.

- La especificación JSF no ha considerado el uso del bookmarking, o marcadores en los navegadores web.

- Cada botón o link clickeado resulta en un envío del formulario html, lo que resulta en una petición compleja si existen cantidad de componentes.

- Si se utiliza facelets, o componentes que implementan Ajax, y el navegador web no tiene habilitado el soporte para JavaScript, la aplicación JSF se vuelve innutilizable.

- JSF usa objetos de `session` para almacenar el estado de los componentes en el tiempo de vida de los mismos, en una granja de servidores es muy costoso replicar la data de la sesión.

- JSF simplifica la creación de la lógica de vista, administra de mejor manera los eventos y despacha las peticiones por el controlador, pero incremente la complejidad al ciclo de vida de una petición, muchos autores señalan que esto es falta de madurez de la tecnología.

- **Quiénes usan JSF**

- Sitios públicos en la web que han sido creados utilizando este framework.

- En el mundo financiero.

- Varias empresas que han acordado ser usuarias JSF.

- Cualquier cliente de `JavaStudio Creator`, `Oracle JDeveloper`.

4.3.2. Struts

- **Introducción**

Struts Framework es un estándar para desarrollo de aplicaciones web, el mismo que es una implementación de la arquitectura MVC.

- **Características generales de Struts**

- Aplicación de código abierto (Open Source).
- Basado en la arquitectura MVC.
- Guarda la información del ruteo de la aplicación en un archivo xml. "struts-config.xml".
- Provee por si mimo la Vista y el Controlador. El programador implementa el Modelo.

- **Arquitectura**

Las solicitudes entrantes son interceptadas por el controlador de Struts Servlet. El archivo de configuración struts-config.xml es usado por el controlador para determinar el flujo que se llevara a cabo. Dicho flujo consiste en la interacción de las siguientes transacciones:

Desde la vista hacia la acción.- El usuario ejecuta una acción como un clic o el envió de un formulario. El controlador recibe la solicitud, busca el mapeo correcto para esta petición y la reenvía a una acción. La acción llama al Modelo para ejecutar una función.

De la acción a la vista. Después de la llamada la función solicitada retorna a la clase de acción, la acción retorna un recurso a la vista y la página de resultado es desplegada hacia el usuario.

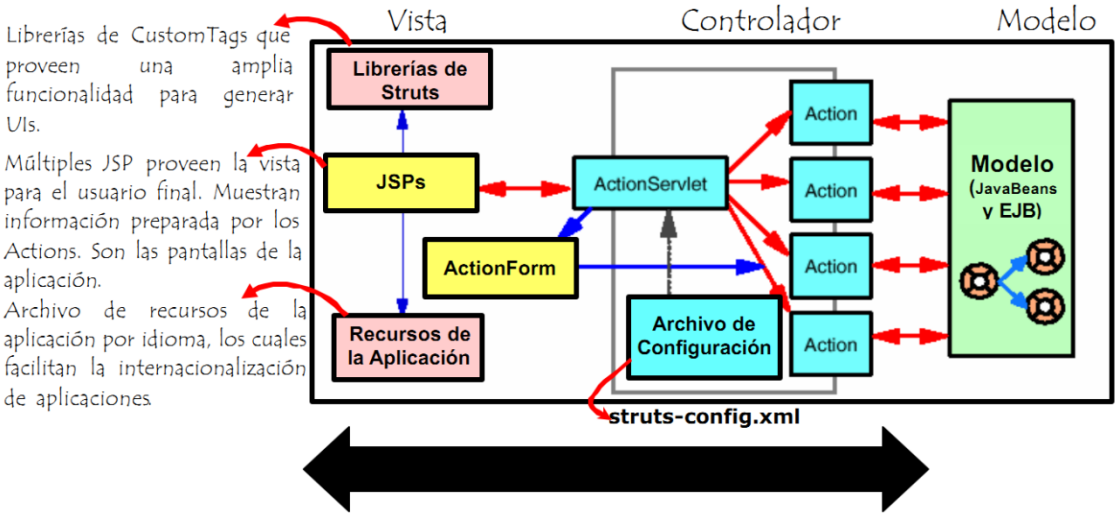


Figura IV. 16. Arquitectura Struts

- **Componentes de Struts**

El Controlador.

Encargado de recibir todas las peticiones entrantes, Su principal función es rutiar las peticiones entrantes a una clase de acción.

Archivo de configuración struts-config.xml

Este archivo contiene la información para el ruteo de peticiones de la aplicación. Este archivo debe estar situado en la carpeta "web-inf" de la aplicación.

Clases de Acción

Actúan como puentes entre las invocaciones de los clientes y los servicios de negocios. Las clases de acción procesan las peticiones y retornan objetos denominados Respuesta de Acción, los cuales son capaces de identificar que componente se debe identificar. Forman parte de la capa del Controlador.

Recursos de la Vista (View Resources)

Conformado por: páginas jsp, páginas html, Java Script, Hojas de estilo, Java Beans, Struts JSP Tags.

Formularios de Acción (ActionForms).

Permiten validar los datos enviados por el usuario antes de que estos sean procesados.

Componentes de Modelo (Model Components).

Struts Framework es capaz de soportar cualquier modelo Ejemplo:

- JavaBeans
- EJB
- CORBA
- JDO

- **Ventajas**

  - Uso de Etiquetas. Promueve el código reusable y permite abstraer el código java de las páginas jsp.
  - Librería de Etiquetas. Proporciona un conjunto grande de etiquetas para distintas aplicaciones.
  - Código Abierto. Cuentas con todas las ventajas del código abierto. Como la modificación del código fuente de struts framework.
  - JDO

- **Desventajas**

  - Cambios Frecuentes. Struts Framework esta sujeto a cambios constantes debido a que se encuentra en un proceso de maduración.

- Dichos cambios no aseguran que los programas escritos en antiguas versiones sean soportados en las nuevas versiones.

- Alcance Limitado. Soporte para determinados contenedores de aplicaciones web.

- No abstrae completamente al desarrollador del funcionamiento del protocolo http.

- Aunque se adapta a las incorporaciones de diferentes bibliotecas de tags no está diseñado para facilitar la creación de componentes propios.

- Las vistas quedan atadas al dispositivo en el cual se renderizan. No facilita el armado de vistas independientes del dispositivo.

4.4. Determinación de los parámetros de comparación

El objetivo de este artículo es determinar los parámetros que nos servirán para comparar los 2 más importantes frameworks J2EE para desarrollo de aplicaciones web de código abierto: Struts y JavaServer Faces (JSF), y establecer pautas que faciliten la elección a la hora de decidir cuál es el framework más adecuado para la implementación de un proyecto determinado.

4.4.1. Parámetro 1: Comportamiento

El presente parámetro trata sobre las características de eficiencia de los frameworks a analizar, su comportamiento en el ambiente existente en la empresa, así como sus características principales para diferenciar su comportamiento en un aplicativo web existente en la empresa.

4.4.2. Parámetro 2: Conectividad

Este parámetro mide la capacidad del framework para integrarse con otras tecnologías, las mismas pueden estar o no en el ambiente de la empresa.

Se analizan diferentes tipos de frameworks de aplicación, como complemento para los frameworks estudiados, dando a notar que Struts o JSF por sí solos pueden dar una aplicación web robusta, pero que se pueden integrar con otras tecnologías para agilizar el trabajo u obtener mayor calidad.

4.5. Determinación de las variables para los parámetros de Comparación

En la siguiente tabla tenemos las variables que se usarán en el estudio:

Tabla IV. VIII: Determinación de las variables de los parámetros de comparación

VARIABLE	STRUTS
Comportamiento	Configuración del framework.
	Lógica de navegación.
	Validación y conversión de datos.
	Líneas de código generadas.
	Binding entre Java y el html.
	Flexibilidad del controlador y manejo de eventos.
	Lógica de navegación.
	Soporte Ajax.
Conectividad	Tiempo de respuesta.
	Consumo de ancho de banda.
	Frameworks de persistencia.
	Caching frameworks.
	Internacionalización y localización.

Fuente: Autor

4.6. Análisis comparativo

En esta sección se va mostrar el estudio del framework JSF bajo la perspectiva del modelo MVC en aplicaciones empresariales J2EE, como estándar para el desarrollo de aplicaciones Web, la comprobación se hará por medio de cuadros comparativos en los que se va a determinar las variables de cada uno de los

parámetros tomado en cuenta para el respectivo análisis, seguidos de una valoración, interpretación y calificación del criterio evaluado por parte del autor.

Para obtener resultados cuantitativos y cualitativos que permitan una selección adecuada de uno de los frameworks analizados para desarrollar aplicaciones web empresariales, se realizará la calificación de cada uno de los parámetros de comparación que se basa en la siguiente escala:

Tabla IV. IX. Escala de calificación para parámetros de comparación

REGULAR	BUENO	MUY BUENO	EXCELENTE
< 70%	>= 70% y <80%	>= 80% y <90%	>= 90%

Fuente: Autor

La siguiente tabla permitirá saber la valoración cuantitativa para de las variables de los parámetros seleccionados:

Tabla IV. X. Escala valoración Cuantitativa

1	2	3
No se cumple	Se cumple parcialmente	Se cumple
Malo	Bueno	Excelente
Deficiente	Poco eficiente	Muy eficiente
Nada	Poco	Mucho
No	Más o menos	Si

Fuente: Autor

Donde cada una de las variables va a ser evaluada sobre el **máximo** que es **3** y cada uno de los ítems de la interpretación incluye la siguiente nomenclatura **(x, y) / w** en donde cada letra significa lo siguiente:

x	Representa el puntaje que obtiene el framework Struts.
y	Representa el puntaje que obtiene el framework JSF.
w	Representa la base del puntaje sobre la cual se está calificando el parámetro.

La calificación definitiva de la herramienta en base a cada parámetro de comparación se obtiene sumando los puntajes obtenidos del análisis, utilizando las siguientes formulas:

$\mathbf{P}_{\mathbf{Struts}} = \sum(x), \quad \mathbf{P}_{\mathbf{JSF}} = \sum(y), \quad \mathbf{P}_{\mathbf{c}} = \sum(w)$

Calificación de Struts: $(C_c - Struts) = (P_{Struts} / P_c) * 100\%$

Calificación de JSF: $(C_c - JSF) = (P_{JSF} / P_c) * 100\%$

En donde:

$\mathbf{P}_{\mathbf{Struts}}$	Puntaje acumulado por `Struts` en el parámetro analizado.
$\mathbf{P}_{\mathbf{JSF}}$	Puntaje acumulado por `JSF` en el parámetro analizado.
$\mathbf{P}_{\mathbf{c}}$	Puntaje sobre el que se califica el parámetro analizado.
$\mathbf{C}_{\mathbf{c}}$ **- Struts**	Porcentaje de la calificación total que obtuvo Struts en el parámetro.
$\mathbf{C}_{\mathbf{c}}$ **- JSF**	Porcentaje de la calificación total que obtuvo Struts en el parámetro.

4.6.1. Parámetro 1: Comportamiento

Las siguientes variables han sido escogidas en función de la conducta de los 2 frameworks evaluados, parámetros que deberán estar presentes tanto en Struts como en JavaServer Faces.

4.6.4.1. Determinación de las variables

a. Configuración del framework.

b. Lógica de navegación.

c. Validación y conversión de datos.

d. Líneas de código generadas.

e. Binding entre Java y el html.

f. Flexibilidad del controlador y manejo de eventos.

g. Soporte Ajax.

h. Independencia del motor gráfico.

- **Valorizaciones**

A continuación se describe las consideraciones a evaluar en cada uno de las variables ya determinadas en el parámetro comportamiento:

a) Variable Configuración del framework

La configuración previa de un framework es el paso inicial para su funcionamiento, se evaluará si los frameworks ofrecen una configuración transparente y el grado de dificultad que presenta.

b) Variable Lógica de navegación

La navegación es una característica clave de Struts y de JSF. Ambos frameworks tienen un modelo de navegación declarativo y definen la navegación usando reglas dentro de un archivo de la configuración XML.

Existen dos tipos de mecanismos de navegación: navegación estática, cuando una página redirecciona directamente a la siguiente y navegación dinámica, cuando cierta acción o lógica determina cuál es la siguiente página. JSF y Struts soportan ambos tipos de navegación.

La navegación en Struts está basada en objetos ActionForward, son quiénes definen "los lugares a donde ir o pasar el control una vez completado el Action". Son los "links" de la aplicación.

La navegación JSF es manejada por objetos listeners de eventos, que procesan los eventos generados por las componentes de IU contenidas en las páginas. Estos listeners realizan algún procesamiento y luego devuelven un resultado lógico, que es usado por el sistema de navegación para seleccionar la siguiente página a mostrar.

c) Variable Validación y conversión de datos

En Struts la validación se hace validando al objeto ActionForm completo, que representa todos los campos del formulario de entrada. En cuanto a la conversión de datos, usa la estándar de JavaBeans.

Por otro lado, JSF permite validar individualmente cada componente del formulario. Se puede validar usando los validadores estándares, creando métodos validadores en los backing beans o creando clases validadoras especiales.

La conversión de datos, también es de granularidad más fina ya que es posible asociarle conversores específicos a las componentes. La distribución estándar de JSF provee conversores de los tipos de datos más comunes como fechas y monedas, además soporta regionalización. De la misma manera que los conversores, es posible crear conversores especiales.

d) Variable Líneas de código generadas

Su significado varía de un lenguaje de programación a otro, pero también dentro de un mismo lenguaje de programación. Este indicador permitirá evaluar la cantidad de líneas de código que se llevan para realizar un determinado proceso con Strus y JSF, con la finalidad de establecer el framework que menos líneas de código genera.

e) Variable Binding entre Java y el html

Este indicador permitirá evaluar el comportamiento de los frameworks en estudio acerca de cómo es la configuración del framework entre el controlador y la lógica de vista, así como también, evaluar el conjunto de tags de presentación que serán los que permiten al componente gráfico enlazarse con el controlador, quien a su vez mantiene el estado de los componentes de la aplicación.

f) Variable flexibilidad del controlador y manejo de eventos

Struts y JSF implementan el patrón Front-Controller, que centraliza el manejo de peticiones provenientes de clientes. Este indicador permitirá evaluar la dificultad de cada uno de los frameworks para implementar el manejo de eventos en una aplicación web.

g) Variable Soporte Ajax

Esta variable permitirá evaluar el nivel de integración y la dificultad que se pueda encontrar al incluir Ajax en la aplicación web, en el mundo Java, es conocido que el trabajo con Ajax lleva consigo añadir dificultad para el

desarrollador, se evaluará cómo se comportan Strus y JSF con frameworks Ajax.

h) Variable Independencia del motor gráfico

Struts y JSF en principio es independiente del motor de visualización aunque generalmente se elija jsp para mostrar las vistas en Strust y Facelets para JSF. Esta variable medirá la integración de los Struts y JSF con varios frameworks o librerías de presentación.

Tabla IV. XI. Resumen variables parámetro Comportamiento

VARIABLE	STRUTS	JSF
Configuración del framework.	Mucho	Poco
Lógica de navegación	Bueno	Excelente
Validación y conversión de datos.	Bueno	Excelente
Líneas de código generadas.	Muy Eficiente	Poco eficiente
Binding entre Java y el html.	Buena	Excelente
Flexibilidad del controlador y manejo de eventos.	Buena	Excelente
Soporte Ajax.	Mala	Excelente
Independencia del motor gráfico.	Buena	Excelente

Fuente: Autor

4.6.4.2. Interpretación de la valorización

Las escalas definidas en la tabla IV.3. se han aplicado a las variables valorizadas como se indica en la tabla IV.4, interpretándolas de la siguiente manera

- La configuración previa para usar el framework es extrema importancia, si bien, tanto Struts como JSF basan su configuración en archivos XML, y siguen el patrón estándar establecido por J2EE, sin embargo, el fichero `struts.xml` posee más "verbose" que el fichero `faces-config.xml`. **(3, 2) / 3**

  Aunque este proyecto cubre la versión de `Struts 1.2` y `JSF 1.2`, cabe mencionar que las versiones recién liberadas de `Struts` y JSF han simplificado su configuración, enmarcando `JSF 2.0` una característica nueva: no es necesario escribir explícitamente la configuración, JSF "descubre" automáticamente la configuración de la aplicación.

- Ambos frameworks tienen un modelo de navegación declarativo y definen la navegación usando reglas dentro de un archivo de la configuración XML. Existen dos tipos de mecanismos de navegación: navegación estática y dinámica.

  En `Struts` típicamente una petición se corresponde con una acción y una vez finalizada se aplica una regla de navegación. Sin embargo en JSF las acciones se codifican por componente y de esta manera es posible que una página que contiene múltiples componentes defina diferentes acciones por cada una de ellas y a su vez compartan la misma regla de navegación. **(2, 3) / 3.**

- En Struts la validación se hace validando al objeto ActionForm completo, que representa todos los campos del formulario de entrada.

  Por otro lado, JSF permite validar individualmente cada componente del formulario. Se puede validar usando los validadores estándares, creando métodos validadores en los backing beans o creando clases validadoras especiales (validaciones útiles para casos genéricos).

  La conversión de datos en Struts usa el estándar de Java Beans, y se puede extender el framework para crear validadores personalizados basados en la interface StrutsTypeConverter; por otro lado JSF implementa conversores comunes, como fechas y monedas, e implementa regionalización, de igual manera se puede extender el framework. **(2, 3) / 3.**

- La generación de código html y JavaScript, por parte de Struts es más "limpia", JSF necesita mantener el estado de cada uno de sus componentes, por lo que mayor cantidad de código, por otro lado, Struts se limita a generar código html necesario para enlazar la lógica de vista con el controlador. **(3, 2) / 3.**

- Referente al binding entre Java y html: Struts, por el lado del controlador tiene un Servlet que se encarga de mantener la información de los componentes, y esta información es entregada a las etiquetas TAGs que se encargan de presentar la información en componentes html estándar. Cada etiqueta se vincula con su par en

java mediante el nombre de la etiqueta, que debe ser el mismo que el del bean en Java.

Por otra parte JavaServer Faces ofrece la misma forma de enlazar el componente de la lógica de vista con los objetos del controlador, además puede enlazar llamadas a métodos en objetos del controlador, lo que lo haría que JSF mapee de manera más natural las datos de la interfaz a los objetos de negocio. **(2, 3) / 3.**

- Struts y JSF implementan el patrón Front-Controller, que centraliza el manejo de peticiones provenientes de clientes. En el caso de Struts, es posible dividir la aplicación en módulos, podrían coexistir distintos objetos que procesen los requerimientos y los errores de manera particular, enchufados al controlador central. En el caso de JSF, los puntos extensión tienen una granularidad más fina, ya que cada una de las componentes que conforman una página JSF pueden tener asociados comportamientos customizados, entre ellos validaciones, conversiones y procesamientos de eventos.

  JSF agrega muchos beneficios al controlador único, proveyendo la capacidad de manejar múltiples eventos sobre una página, mientas que Struts puede manejar un único evento por página. **(2, 3) / 3.**

- Struts 1.2 como framework, no soporta Ajax, las librerías gráficas de Struts no permiten ningún manejo de tecnología Ajax, sin embargo, Ajax puede ser utilizado a nivel de la jsp, asignando acciones tal como si se lo hiciere en un documento html, lo que a la

final le agrega dificultad al desarrollo, por lo que el desarrollador debería implementar código Javascript.

Por otro lado, JSF sí soporta Ajax, con librerías tags específicas para el uso de tecnología Ajax (a4j) lo que libera al desarrollador de la tediosa labor de implementar código Javascript. **(1, 3) / 3.**

- Struts cuenta con un conjunto de custom tags que facilitan la creación de formularios HTML para entrada de datos y que interactúan con los objetos del framework Struts. Provee las mismas funcionalidades que HTML, pero le facilitan al programador la creación de los formularios y la visualización de los errores. Además, la distribución de Struts tiene integrado el framework Tiles también de código fuente abierto, que a través de plantillas, extiende las capacidades provistas por Struts para la vista. De esta manera, se mejora el look&feel de las aplicaciones. Pero la independencia de la capa de vista llega hasta ahí, solo renderizará html como motor gráfico, mientras que Jsf, al cumplir las mismas funciones de Struts con facelets, que es superior a tiles, y la integración con librerías de componentes gráficos de terceros, como RichFaces, IceFaces, MyFaces, etc, integra un renderizado mejorado, presentando la posibilidad de gestionar el render de la página para algún tipo distinto de salida, como pueden ser dispositivos móviles, pdf, svg, entre los principales. **(2, 3) / 3.**

- **Calificación**

$P_c = \sum(w) = 3+3+3+3+3+3+3+3 = 24$

$P_{Struts} = \sum(x) = 3 + 2 + 2 + 3 + 2 + 2 + 1 + 2 = \mathbf{17}$

C_c - Struts: $(P_{Struts} / P_c) * 100 = (17/24)*100\% = \mathbf{70,83\%}$

$P_{JSF} = \sum(y) = 2 + 3 + 3 + 3 + 3 + 3 + 3 + 3 = \mathbf{23}$

C_c - JSF: $(P_{JSF} / P_c) * 100 = (23/24)*100\% = \mathbf{95,83\%}$

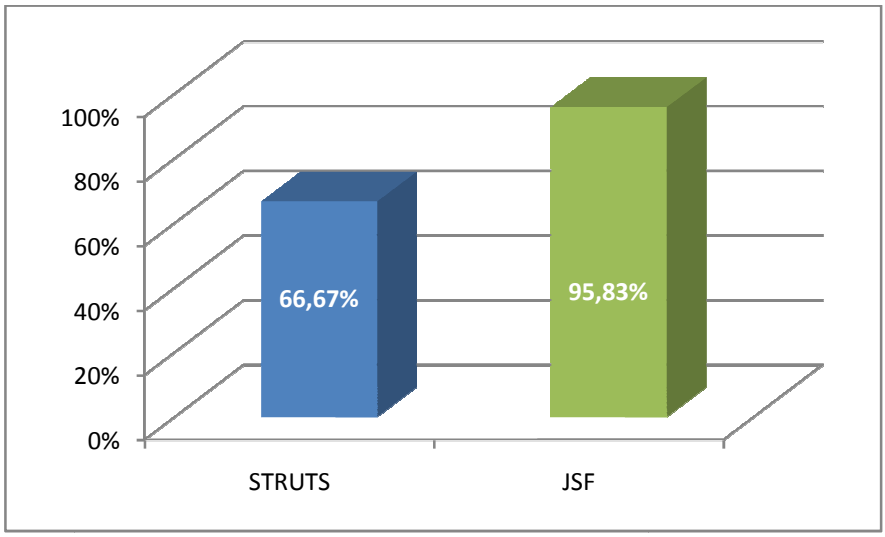


Figura IV. 17. Comparación estadística del parámetro Comportamiento

4.6.2. Parámetro 2: Conectividad

4.6.4.3. Determinación de las variables

a. Tiempo de respuesta.

b. Consumo de ancho de banda.

c. Frameworks de persistencia.

d. Caching frameworks.

e. Internacionalización y localización.

- **Valorizaciones**

a. Variable tiempo de respuesta.

Esta variable permitirá conocer el tiempo de respuesta de la aplicación web, tanto en Struts, como JSF, para determinar cuál es el framework que posee mayor velocidad en su ejecución.

b. Variable consumo de ancho de banda.

Esta variable permitirá evaluar cada framework en relación al ancho de banda que consume una petición típica de una aplicación web.

c. Variable frameworks de persistencia.

Este indicador permitirá conocer el nivel de conectividad o acople con frameworks de existencia compatibles con `Struts` y JSF, permitiendo al desarrollador mapear los objetos de la base de datos hacia objetos de negocio en lenguaje `Java`.

d. Variable caching frameworks.

Esta variable permitirá conocer el nivel de conectividad o acople de los frameworks en estudio con frameworks de caché de objetos, para mantener el estado entre objetos entre las peticiones y usuarios, si la página ya se encuentra en memoria caché, la presenta, caso contrario se realiza la petición al servidor.

e. Variable internacionalización y localización.

Esta variable permitirá conocer el soporte para internacionalización tanto de Struts como de JSF, los 2 extienden la internacionalización de una aplicación web J2EE.

Tabla IV. XII. Resumen variables parámetro Conectividad

	STRUTS	**JSF**
Tiempo de respuesta.	Excelente	Excelente
Consumo de ancho de banda.	Muy Eficiente	Poco eficiente
Frameworks de persistencia.	Más o menos	Sí
Caching frameworks.	Más o menos	Sí
Internacionalización y localización.	Sí	Sí

Fuente: Autor

4.6.4.4. Interpretación de la valorización

- En lo relacionado al tiempo de respuesta, Encontramos que tanto Struts como JSF devuelven una petición prácticamente al mismo tiempo, esto debido a que la lógica tanto para Struts como para JSF está basado en beans de negocio estándar en Java. **(3, 3) / 3.**

- Debido la cantidad de código generado por JSF en contraste con las líneas generadas por Struts, el consumo de ancho de banda es ligeramente mayor en aplicaciones JSF. **(3, 2) / 3.**

- En lo relacionado a frameworks de persistencia, ORM como Hibernate, iBatis(parcialmente soportado por los 2), EJB, los 2 framework ofrecen soporte, sin embargo JSF, al ser una implementación más nueva que Struts ofrece mecanismo de soporte a frameworks de persistencia más sofisticados que Struts, de igual manera ofrece soporte a frameworks de validación de entradas a nivel de lógica de aplicación que Struts no posee. Pero con esto, no queremos decir que Struts no posea soporte para frameworks de persistencia, pues sí lo tiene, pero la conectividad entre capas es de menor medida. **(2, 3) / 3.**

- Respecto a frameworks de cache, tenemos una situación parecida a lo que sucede con frameworks de persistencia: JSF al ser más joven, se integra de mejor manera con frameworks de caché, Struts sí tiene soporte para frameworks de caché, pero exige una mayor

configuración para el acople o conectividad con dichos frameworks, tarea que JSF envuelve de mejor manera, casi automática.

(2, 3) / 3.

Respecto a internacionalización y localización, tanto Struts como JSF soportan el modelo i18n & I10n, estándar en una aplicación web Java empresarial, el descriptor de la aplicación web: web.xml, resuelve el fichero .properties específico para cada lenguaje, informándole en la configuración qué idioma se desea mostrar.

(3, 3) / 3.

- **Calificación**

$P_c = \sum(w) = 3 + 3 + 3 + 3 + 3 = 15$

$P_{Struts} = \sum(x) = 3 + 3 + 2 + 2 + 3 = \mathbf{13}$

C_c - Struts: $(P_{Struts} / P_c) * 100 = (13/15)*100\% = \mathbf{86,66}\%$

$P_{JSF} = \sum(y) = 3 + 2 + 3 + 3 + 3 = \mathbf{14}$

C_c - JSF: $(P_{JSF} / P_c) * 100 = (14/15)*100\% = \mathbf{93,33}\%$

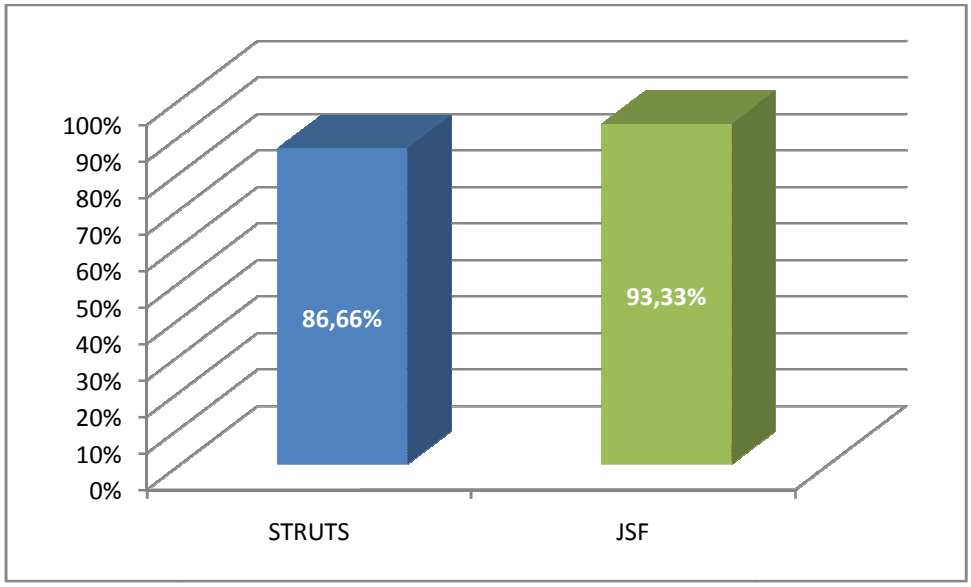


Figura IV. 18. Comparación estadística del parámetro Conectividad

4.7. Análisis de los resultados

Los puntajes finales y el porcentaje que se ha obtenido una vez que se ha analizado los frameworks Struts y JSF mediante las variables evaluadas, se obtienen de la siguiente manera:

Puntaje Total del Análisis: $(P_T) = \sum (P_c)$

Puntaje Total de Struts: $(PT_{Struts}) = \sum (P_{Struts})$

Puntaje Total de JSF: $(PT_{JSF}) = \sum (P_{Struts})$

Porcentaje Total de Struts: $(\%\ Struts) = (PT_{Struts}/P_T) * 100\%$

Porcentaje Total de JSF: $(\%\ JSF) = (PT_{JSF}/P_T) * 100\%$

A continuación, se muestra la matriz general de los resultados obtenidos:

Tabla IV. XIII. Tabla general de resultados

PARÁMETRO	VARIABLE	STRUTS	JSF
Comportamiento	Configuración del framework	3	2
	Lógica de navegación.	2	3
	Validación y conversión de datos.	2	3
	Líneas de código generadas.	2	3
	Binding entre Java y el html.	2	3
	Flexibilidad del controlador y manejo de eventos.	2	3
	Soporte Ajax.	1	3
	Independencia del motor gráfico.	2	3
Conectividad	Tiempo de respuesta.	3	3
	Consumo de ancho de banda.	3	2
	Frameworks de persistencia.	2	3
	Caching frameworks.	2	3
	Internacionalización y localización	3	3
TOTALES		**29**	**37**

Fuente: Autor

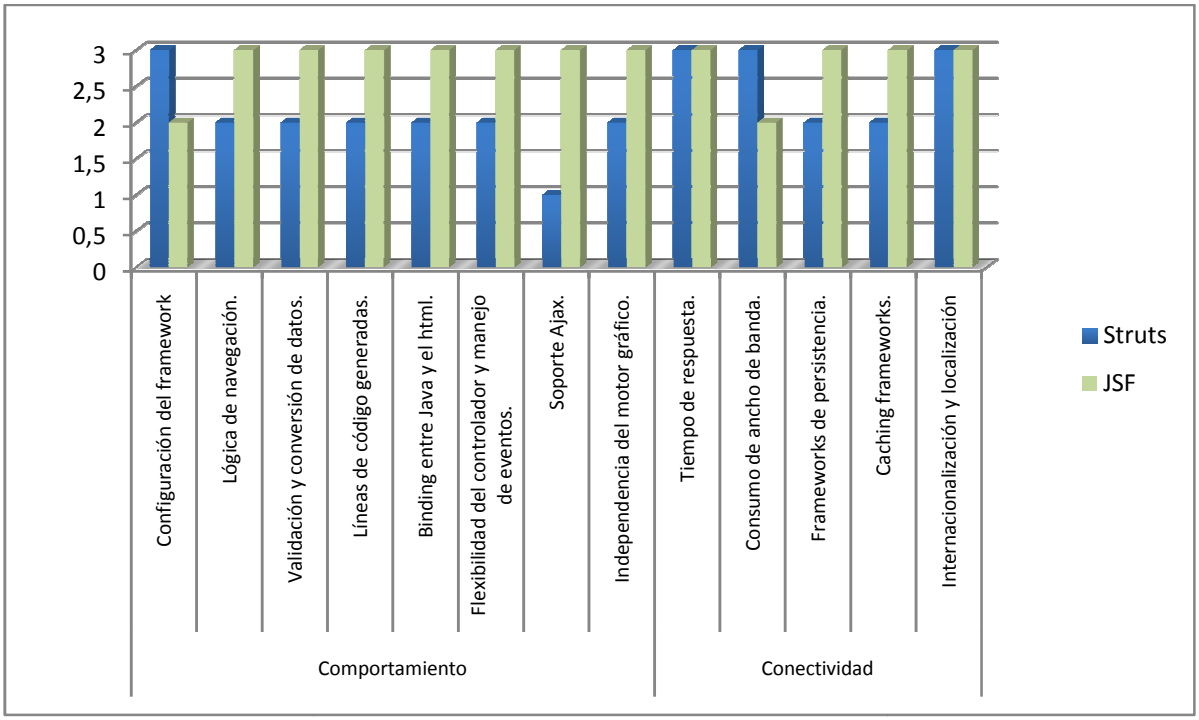


Figura IV. 19. Diagrama general de resultados por cada variable

P_T = 15+24 = 39

PT_{Struts} = 17+13 = 30

PT_{Jsf} = 23+14 = 37

(% Struts) = (30/39) * 100 = **76,92%** Equivalente a **Bueno**.

(% JSF) = (37/39) * 100 = **94,87%** Equivalente a **Excelente**.

Resultados que podemos representar de la siguiente manera:

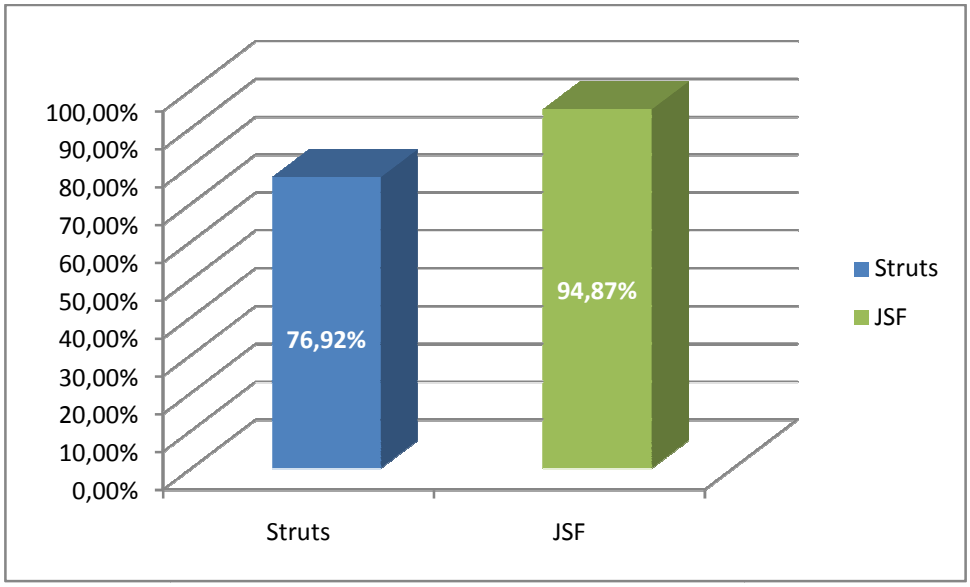


Figura IV. 20. Diagrama general de resultados

Después de haberse realizado el análisis comparativo correspondiente de los respectivos parámetros con sus variables, como se pude observar en la Figura IV.5, se ha determinado que para desarrollar una aplicación web empresarial sobre la arquitectura J2EE, el framework JavaServer Faces es el que tiene una máxima calificación de **94,87**% correspondiente a **excelente** por lo que se determina que es el framework más adecuado para desarrollar la aplicación web empresarial en ambiente J2EE; por otro lado: Struts obtuvo una calificación de **76,92**% que equivale a **bueno**.

Finalmente, en base a los resultados expuestos, se puede interpretar lo siguiente:

- Los frameworks Struts y JavaServer Faces han demostrado ser los más idóneos para el desarrollo web empresarial en J2EE, sin embargo, la evolución de los frameworks web, apunta hacia una abstracción sobre el protocolo en que estos trabajan: http, y en esto, JavaServer Faces

cumple una mejor función al ocultar al desarrollador los detalles de implementación, a veces engorrosos y repetitivos del protocolo sobre el cuál funciona la red internet.

- La configuración del framework, sea cual sea el que se elija, siempre es un paso decisivo para la elección o rechazo del mismo. Una configuración extensa puede resultar en un mantenimiento engorroso a futuro, en esta parte, JSF 1.2 posee menos configuración que Struts, con menos ficheros de configuración y con menos parámetros a considerar; cabe recalcar que las versiones comparadas fueron la 1.2 de cada framework, las versiones actuales, especialmente JSF 2.0, posee un motor de introspección en las clases de la aplicación que es capaz de configurar reglas de navegación y binding java - html casi automáticamente.

- En lo que se refiere a lógica de navegación, Struts y JSF manejan los 2 tipos de navegación: estática y dinámica, pero JSF ofrece un mecanismo más lógico de navegación, más limpio y entendible; Struts acarrea un poco más de "verbose" en la configuración; los 2 frameworks basan su lógica de navegación en ficheros de configuración xml.

- En lo que se refiera a validación y conversión de datos, JSF supera a Struts al proporcionar soporte directo o nativo a validación a nivel de bean, inclusive, su integración con frameworks de validación como Hibernate Validator, hace que la tarea de validar para tipos nativos de Java (integer, String, etc) sea prácticamente automática, por otro lado

Struts no ofrece dicha integración con frameworks de validación, pero sí validación a nivel de bean.

- Las líneas de código que generan tanto Struts como JSF, superan en gran medida a una aplicación web sin usar ningún tipo de framework, aquí, Struts genera menor cantidad de líneas de código, esto es por su limitada integración con otros frameworks, tanto de validación, persistencia, presentación, además, JSF genera más líneas de código pues implementa código JavaScript en la lógica de presentación, Struts no lo implementa a menos que se inserte código en la capa de vista.

- El enlace o binding entre el negocio y la capa de presentación, en Java: bean y lógica de vista, está dado por ficheros de configuración xml, donde se mapean los objetos y sus parámetros, su alcance en la aplicación y su comportamiento, el mapeo de Struts de estas clases, no es tan "lógica", lo cuál puede ser un poco frustrante al inicio, JSF proporciona etiquetas xml más naturales, y entendible para realizar la configuración de binding entre la vista y el controlador.

El futuro próximo está marcado por las interfaces ricas, una interactividad sin necesidad de realizar consultas permanentes al servidor ni realizar recargas completas (Ajax) con la posibilidad que desde el cliente se puedan acceder a diferentes servidores a la vez, lo que marca un cambio en la concepción del navegador web, que pasa de ser una interfaz sin inteligencia ni estado (salvo

por las cookies) a otra con la posibilidad de procesar la información y mantener un estado propio, cosa que lo está logrando JSF.

Después de haber realizado el análisis comparativo y según los resultados obtenidos de todos los parámetros con sus respectivas variables, se puede concluir que para la implementación de una aplicación web empresarial Java (J2EE) JavaServer Faces es el framework más indicado con una puntuación del 94,87%; por otro lado, Struts ha alcanzado una puntuación sobre el 76,92%.

4.8. Demostración de la Hipótesis

Para poder realizar la demostración de la hipótesis de este trabajo investigativo se va a probar que con el uso de la tecnología JavaServer Faces se mejorará el rendimiento de procesos en línea para tareas de comercialización de combustibles por parte de EP Petroecuador.

Se tomará en consideración muchas variables que inciden en los tiempos de respuesta que se dan actualmente en el módulo de Aseguramiento de precios a nivel de terminal, tanto en la implementación con el framework existente en la empresa: Struts, así como el mismo módulo, pero implementado en JavaServer Faces.

4.8.1. Tiempos de respuesta usando Struts

Cabe recalcar que en este parámetro a más de considerar el tiempo que se lleva al servidor en procesar una petición y enviar una respuesta al cliente, se deberá tomar en consideración variables como tipo de cliente, carga del servidor de aplicaciones, número de clientes conectados usando el módulo.

Las mediciones se efectuarán con el software Webserver Stress Tool, configurándolo para peticiones de 20 usuarios, con llamadas a las páginas que más lógica demandan, detrás del servidor de aplicaciones Java: Websphere (ver anexo 3).

- Carga de información desde archivo externo:

  En este proceso, el módulo de aseguramiento de precios a nivel de terminal solicita la carga de información al sistema desde un fichero de Microsoft Excel, en formato 2003 o 2007, con información fuente para el cálculo de las fórmulas de precios, tiempo: 11 segundos.

- Validación, conversión, estandarización de nombres y medidas:

  En este punto, el asistente guiado, luego de la carga de información desde un fichero externo, evalúa que el fichero cumpla el formato específico, realiza la estandarización de nombres y unidades de

medida, presenta información de resumen del proceso, tiempo: 11 segundos.

- Cálculo de precios:

  El asistente guiado, luego de ingresar parámetros de fecha de origen de los precios base para el cálculo, y las fechas de vigencia para los precios a calcular, evalúa cada fórmula para cada producto previamente asignado. Tiempo: 5 segundos.

- Generación de estructura de precios PCO.

  En este punto, luego del cálculo de precios a partir de las fórmulas, el asistente guiado arma la *estructura de precios*, que es una estructura de da datos que será almacenada en la base de datos *Abastecedora*, que es donde residen los precios calculados, con sus componentes como IVA, castigos, etc. Tiempo consumido: 6 segundos.

- Almacenamiento de estructura de precios

  El asistente solicitará la confirmación del usuario para cargar la estructura de datos de los precios calculados para efectuar el almacenamiento en los objetos de la base de datos *Abastecedora*, con lo cual, el proceso de aseguramiento de precios llega a su fin; tiempo para almacenar estructura; tiempo: 14 segundos.

- Generación de reporte pdf de estructura de precios.

En cada punto, el asistente guiado solicita la impresión de la vista actual para permitirle avanzar en el asistente guiado, la generación del fichero pdf consume un promedio de 7 segundos.

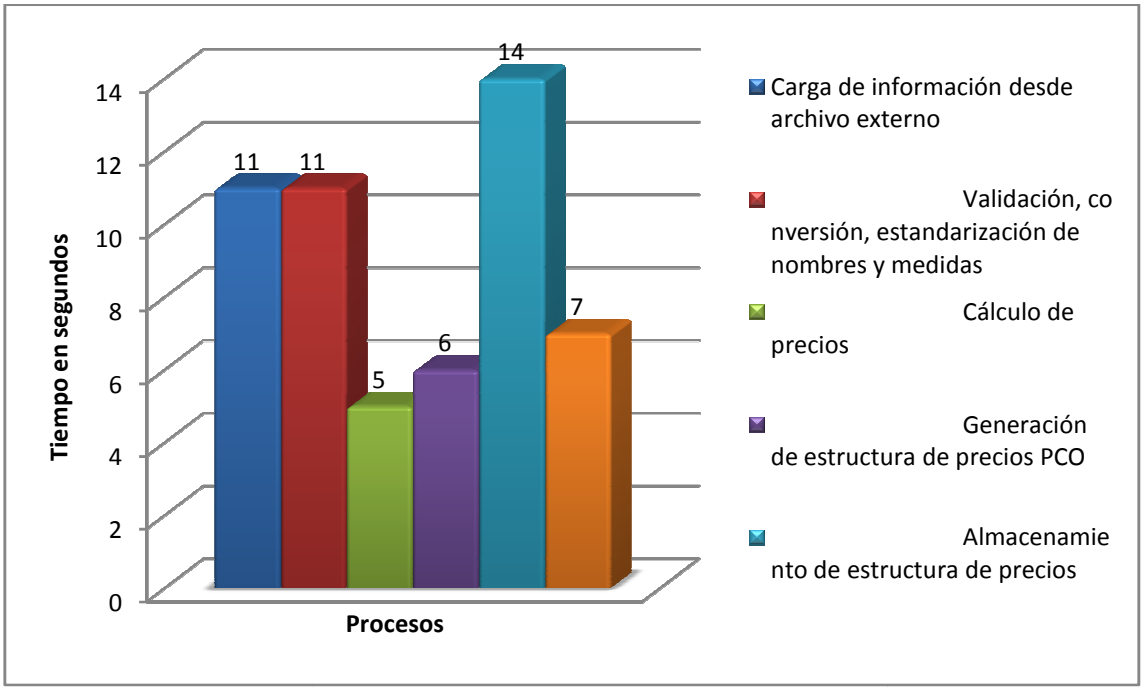


Figura IV. 16. Tiempos de respuesta módulo Aseguramiento de precios, Struts

4.8.2. Tiempos de respuesta usando JSF

Al igual que el punto anterior, se va a considerar el módulo de aseguramiento de precios a nivel de Terminal, esta vez usando el framework JSF.

- Carga de información desde archivo externo:

  En esta ocasión, el componente usado para subir un fichero externo a la aplicación es un componente enriquecido, de la librería RichFaces,

lo que permite el uso de Ajax en la aplicación, reduciendo la carga de red, y el proceso se ve disminuido en su tiempo de respuesta: 7 segundos.

- Validación, conversión, estandarización de nombres y medidas:

  En este punto, la lógica del programa es la misma que en `Struts`, no se ve afectada por el uso de librerías de componentes gráficos o de persistencia o de validación, el tiempo, sin embargo disminuye, debido a que la generación de la lógica de vista ya está precompilada: 7 segundos.

- Cálculo de precios:

  El asistente guiado, luego de ingresar parámetros de fecha de origen de los precios base para el cálculo, y las fechas de vigencia para los precios a calcular, evalúa cada fórmula para cada producto previamente asignado. En esta ocasión, el tiempo de respuesta se ve afectado, esto debido a la cantidad de información que deberá ser cargada en la lógica de vista, debido al uso de componentes gráficos enriquecidos (rich) Tiempo: 8 segundos.

- Generación de estructura de precios PCO.

  En esta ocasión, el proceso disminuye, el framework de chaché de JSF permite mantener en sesión los datos del paso anterior, con lo que la generación de la estructura de datos es más rápida: 2 segundos.

- Almacenamiento de estructura de precios

  El tiempo de espera se ve reducido, por la misma razón del uso de buffer de memoria caché de JSF; tiempo: 5 segundos.

- Generación de reporte pdf de estructura de precios.

En este punto, prácticamente se mantiene el tiempo, debido a que el proceso hace uso de las mismas librerías generadoras de archivos pdf: 7 segundos.

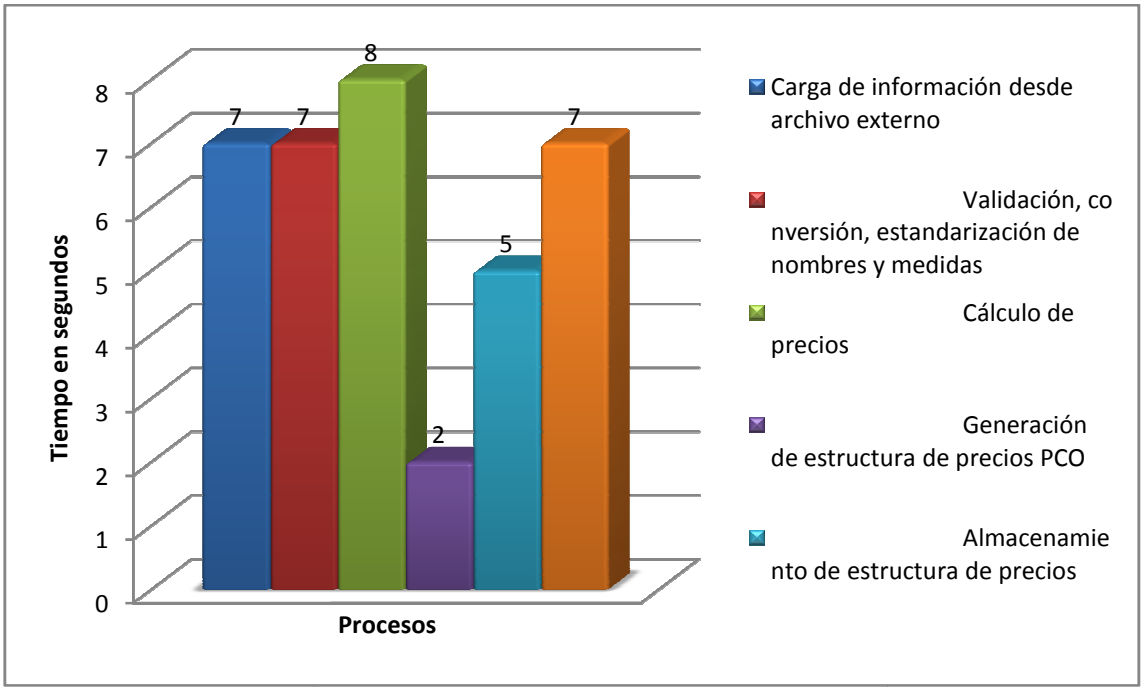


Figura IV. 17. Tiempos de respuesta módulo Aseguramiento de precios, JSF

4.8.3. Análisis matemático

Este análisis se lo realizará empleando fórmulas matemáticas verificando los tiempos de respuestas que se han obtenido tanto sin el uso de la tecnología existente en la empresa: Struts, y la tecnología JSF para luego sacar conclusiones que servirán para demostrar la veracidad o no de la hipótesis, las fórmulas que se van a emplear son las siguientes:

Puntaje Total del Análisis: $(P_T) = \sum (P_{Struts} + P_{JSF})$

Puntaje Struts: P_{Struts}

Puntaje JSF: P_{JSF}

Tiempo ahorrado: $P_{Struts} - P_{JSF}$

- Carga de información desde archivo externo:

$P_T = \sum (P_{Struts} + P_{JSF})$
$P_T = 11 + 7$
$P_T = 18$

$P_{Struts} = (P_{Struts} * 100\%) / P_T$
$P_{Struts} = (11 * 100\%) / 18$
$P_{Struts} = 61,11\ \%$

$P_{JSF} = (P_{JSF} * 100\ \%) / P_T$
$P_{JSF} = (7 * 100\%) / 18$
$P_{JSF} = 38.89\ \%$

$Tiempo\ ahorrado = P_{Struts} - P_{JSF}$
$Tiempo\ ahorrado = 61,11\% - 38,89\%$
$Tiempo\ ahorrado = \mathbf{22.22\%}$

- Validación, conversión, estandarización de nombres y medidas:

$$P_T = \sum (P_{Struts} + P_{JSF})$$
$$P_T = 11 + 7$$
$$P_T = 18$$

$$P_{Struts} = (P_{Struts} * 100\%) / P_T$$
$$P_{Struts} = (11 * 100\%) / 18$$
$$P_{Struts} = 61,11\ \%$$

$$P_{JSF} = (P_{JSF} * 100\ \%) / P_T$$
$$P_{JSF} = (7 * 100\%) / 18$$
$$P_{JSF} = 38.89\ \%$$

$$\text{Tiempo ahorrado} = P_{Struts} - P_{JSF}$$
$$\text{Tiempo ahorrado} = 61,11\% - 38,89\%$$
$$\text{Tiempo ahorrado} = \mathbf{22.22\%}$$

- Cálculo de precios:

$$P_T = \sum (P_{Struts} + P_{JSF})$$
$$P_T = 5 + 8$$
$$P_T = 13$$

$$P_{Struts} = (P_{Struts} * 100\%) / P_T$$
$$P_{Struts} = (5 * 100\%) / 13$$
$$P_{Struts} = 38,46\ \%$$

$$P_{JSF} = (P_{JSF} * 100\ \%) / P_T$$
$$P_{JSF} = (8 * 100\%) / 13$$
$$P_{JSF} = 61,54\ \%$$

$$\text{Tiempo ahorrado} = P_{Struts} - P_{JSF}$$
$$\text{Tiempo ahorrado} = 38,46\% - 61,54\%$$
$$\text{Tiempo ahorrado} = \mathbf{-\ 23,08\%}$$

- Generación de estructura de precios PCO.

$$P_T = \sum (P_{Struts} + P_{JSF})$$
$$P_T = 6 + 2$$
$$P_T = 8$$

$$P_{Struts} = (P_{Struts} * 100\%) / P_T$$
$$P_{Struts} = (6 * 100\%) / 8$$
$$P_{Struts} = 75,00\ \%$$

$$P_{JSF} = (P_{JSF} * 100\ \%) / P_T$$
$$P_{JSF} = (2 * 100\%) / 8$$
$$P_{JSF} = 25,00\ \%$$

$$\text{Tiempo ahorrado} = P_{Struts} - P_{JSF}$$
$$\text{Tiempo ahorrado} = 75,00\% - 25,00\%$$
$$\text{Tiempo ahorrado} = \mathbf{50,00\%}$$

- Almacenamiento de estructura de precios

$$P_T = \sum (P_{Struts} + P_{JSF})$$
$$P_T = 14 + 5$$
$$P_T = 19$$

$$P_{Struts} = (P_{Struts} * 100\%) / P_T$$
$$P_{Struts} = (14 * 100\%) / 19$$
$$P_{Struts} = 73,68\ \%$$

$$P_{JSF} = (P_{JSF} * 100\ \%) / P_T$$
$$P_{JSF} = (5 * 100\%) / 19$$
$$P_{JSF} = 26,32\ \%$$

$$\text{Tiempo ahorrado} = P_{Struts} - P_{JSF}$$
$$\text{Tiempo ahorrado} = 73,68\% - 26,32\%$$
$$\text{Tiempo ahorrado} = \mathbf{47,36\%}$$

- Generación de reporte pdf de estructura de precios.

$$P_T = \sum (P_{Struts} + P_{JSF})$$
$$P_T = 7 + 7$$
$$P_T = 14$$

$$P_{Struts} = (P_{Struts} * 100\%) / P_T$$
$$P_{Struts} = (7 * 100\%) / 14$$
$$P_{Struts} = 50,00\ \%$$

$$P_{JSF} = (P_{JSF} * 100\ \%) / P_T$$
$$P_{JSF} = (7 * 100\%) / 14$$
$$P_{JSF} = 50,00\ \%$$

$$\text{Tiempo ahorrado} = P_{Struts} - P_{JSF}$$
$$\text{Tiempo ahorrado} = 50,00\% - 50,00\%$$
$$\text{Tiempo ahorrado} = \mathbf{0,00\%}$$

A continuación se interpretará los resultados obtenidos en el módulo de aseguramiento de precios a nivel de terminal:

- Se puede observar que para el proceso de carga de información desde fichero externo usando JSF sobre Struts, se obtiene un ahorro de 22,22%.
- Para el proceso de validación, conversión, estandarización de nombres y medidas, el ahorro en tiempo de proceso alcanzado es del 22,22%.
- No existió mejoramiento en el proceso de cálculo de precios, al contrario, se aumentó el tiempo de respuesta en 23,08%.
- Para el proceso de generación de estructura de precios PCO, el proceso se vio beneficiado en un 50%.

- Se puede observar también que el proceso de almacenamiento de la estructura de precios generada en el punto anterior, se ha reducido el tiempo de espera en un : 47,36%.

- Para el proceso genérico de generación de reportes pdf, no ha existido mejora considerable en el tiempo de respuesta: 0,00%.

De los resultados obtenidos, se puede concluir que: luego de haber realizado todos los cálculos correspondientes para la demostración de la hipótesis, y analizando proceso por proceso del estudio efectuado, se puede concluir que el desarrollo de una aplicación web, basada en el framework JavaServer Faces mejora el rendimiento de procesos en línea, al reducir tiempos de espera, para procesos de comercialización de combustibles de EP Petroecuador.

CAPÍTULO V

5. Diseño y desarrollo de la aplicación

5.1. Introducción

En el presente capítulo se pone en marcha la parte aplicativa del proyecto, es decir, los módulos de la solución web para procesos de comercialización de combustibles de EP Petroecuador, usando el framework de aplicaciones web J2EE JSF.

5.2. Metodología para el desarrollo del proyecto

Para el desarrollo de la construcción de los módulos de aplicación, se sigue el Proceso Unificado de Rational, como estándar en la empresa dado que establece claros procesos para todo el ciclo del desarrollo del proyecto.

Esta metodología fue desarrollada desde el inicio del proceso de incepción, hasta llegar a las etapas de interacción con el usuario y documentación del proyecto.

El proceso iterativo e incremental de RUP está caracterizado por la realización en paralelo de todas las disciplinas de desarrollo a lo largo del proyecto, con lo cual la mayoría de los artefactos son generados muy tempranamente en el proyecto pero van desarrollándose en mayor o menor grado de acuerdo a la fase e iteración del proyecto.

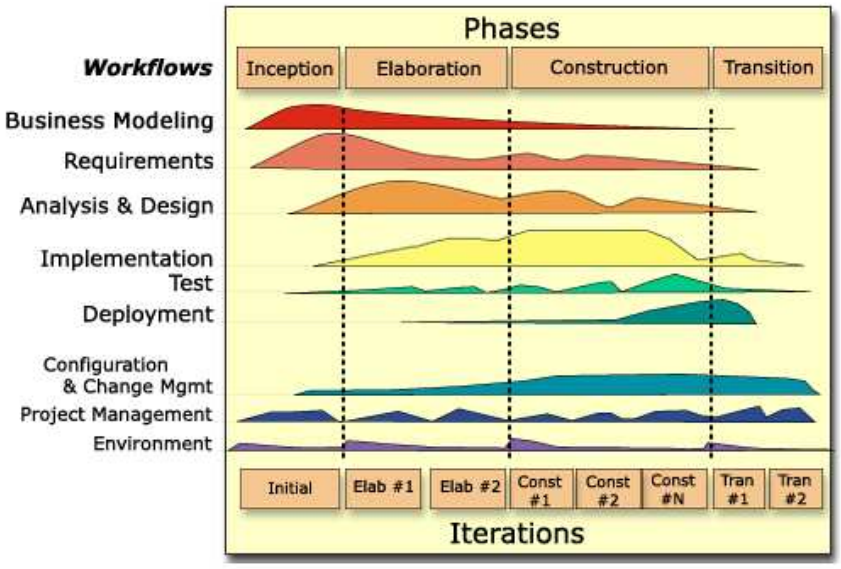


Figura V. 18. Proceso Unificado de Rational

En las siguientes secciones se describen los procesos realizados para cada fase del proyecto y sus entregables, cabe recalcar que se ha resumido considerablemente el modelo RUP para este trabajo, debido a que la cantidad de entregables para el

proyecto es extensa, parte importante se puede encontrar en la sección de anexos.

5.2.1. Fase de incepción

- **Plan de desarrollo de software**

  El propósito del Plan de Desarrollo de Software es recoger toda la información necesaria para el control del proyecto. Describe el enfoque al desarrollo de software y el plan de alto nivel generado y usado por líderes para gestionar el proyecto.

- **Definiciones, siglas y abreviaturas**

  Ver artefacto: Glosario de términos.

- **Entregables del proyecto**

  Los entregables del proyecto serán los establecidos a continuación:

  1) **Plan de desarrollo de Software**

  Es el presente documento.

  2) **Visión**

  Este documento define la visión del producto desde la perspectiva del cliente, especificando las necesidades y características del producto. Constituye una base de acuerdo en cuanto a los requisitos del sistema.

3) Riesgos

La identificación de los riesgos que estarán implicados en el desarrollo de esta aplicación, los que serán considerados por el líder del proyecto en el momento del desarrollo; permitiendo su mitigación o la ejecución de la actividad de contingencia necesaria. Para esto, es importante que los riesgos se prevean con anterioridad.

Estos riesgos serán gestionados de acuerdo a las estrategias y supervisión que se plantee el grupo de desarrollo para cada riesgo.

4) Glosario

Es un documento que define los principales términos usados en el proyecto. Permite establecer una terminología consensuada.

5) Modelo General de Casos de Uso

Diagrama general de casos de uso.

6) Especificación de casos de uso

Documento que define la especificación de casos de uso, su explotación a mayor nivel.

- **Plan de proyecto**

En esta sección se presenta la organización en fases e iteraciones y el calendario del proyecto.

Se describe el plan de fases del proyecto:

Tabla V. XIV: Plan de fases

FASE	ITERACIONES	COMIENZO	FINALIZACIÓN	ESTIMACIÓN
Inicio	1	17-Ago-2010	03-Sep-2010	3 semanas
Elaboración	2	30-Ago-2010	24-Sep-2010	4 semanas
Construcción	4	27-Sep-2010	23-Dic-2010	12 semanas
Transición	2	03-Ene-2010	28-Ene-2010	4 semanas

- **Objetivos de las iteraciones**

Los hitos que marcan el final de cada fase se describen en la siguiente tabla.

Tabla V. XV: Objetivo de las iteraciones

DESCRIPCIÓN	ITERACIÓN	HITO
Fase de Inicio		En esta fase desarrollará los requisitos del producto desde la perspectiva del usuario. Los principales casos de uso serán identificados y se hará un refinamiento del Plan de Desarrollo del Proyecto. La aceptación del cliente / usuario del artefacto

		Visión y el Plan de Desarrollo marcan el final de esta fase.
Fase de Elaboración		En esta fase se analizan los requisitos y se desarrolla un prototipo de arquitectura (incluyendo las partes más relevantes y críticas del sistema). Al final de esta fase, todos los casos de uso correspondientes a requisitos que serán implementados en la primera fase de Construcción deben estar analizados y diseñados.
Fase de Construcción		Durante la fase de construcción se terminan de analizar y diseñar todos los casos de uso. También se aplican las pruebas y se valida con el cliente / usuario. Se comienza la elaboración de material de apoyo al usuario. Se implementan todos los casos de uso.
Fase de Transición		En esta fase se preparara la implantación del aplicativo de manera adecuada, incluyendo el entrenamiento de los usuarios. El hito que marca el fin de esta fase incluye, la entrega de toda la documentación del proyecto con los manuales de instalación y todo el material de apoyo al usuario, la finalización del entrenamiento de los usuarios y el empaquetamiento del producto.

- **Visión**

  - **Alcance**

El propósito de este documento es recolectar, analizar y definir necesidades de alto nivel y características de los distintos procesos de comercialización de combustibles identificados por el nivel estratégico de la empresa. Está enfocado en las capacidades que necesitan los interesados y los usuarios, y por qué esas necesidades existen.

- **Definición del problema**

El problema se define en la siguiente tabla:

Tabla V. 1: Definición del problema

El problema de	La falta de automatización de ciertos procesos de negocio identificados por el nivel estratégico de la empresa, han ocasionado la falta de seguimiento, gestión y control de la información derivados de dichos procesos, alrededor de procesos de autenticación de usuarios aplicaciones web, generación automática de precios, integración de información de precios. - La autenticación de usuarios, para módulos basados en web, es diferente a la autenticación de usuarios en procesos batch o clientes AS, lo que dificulta el seguimiento o control de los usuarios y sus tareas dentro de los aplicativos. - La generación automática de precios (aseguramiento de precios) de productos derivados del petróleo ha generado tiempos de espera prolongados, así como errores durante el proceso de conversión o estandarización de medidas, carga de rubros de impuestos nacionales, lo que se reflejaría en el cálculo del precio de venta, ya que este proceso se lo realiza de forma manual.
Afecta a	- Clientes de EPP. - Administradores, nivel estratégico de la empresa. - Usuarios externos a EPP. Funcionarios DIRNEA. - Funcionarios de EPP.
el impacto de ello es	Falta de control de control y seguimiento de procesos identificados.
Una solución exitosa debería	Proveer al usuario de una herramienta, o conjunto de herramientas, de uso intuitivo, amigable, que permita mantener un control y seguimiento de los procesos identificados. Logrando obtener en línea y en todo momento, desde cualquier punto con acceso a internet, información de calidad, requerida por los distintos niveles del ámbito estratégico de la empresa y sus clientes.

- **Definición de posicionamiento del producto:**

Se define a continuación el posicionamiento del producto:

Tabla V. 2: Definición del problema

Para	Clientes de EPP. Funcionarios de la siguientes instituciones: EPP Usuarios externos.
Quiénes	Podrán realizar actividades de gestión, seguimiento, control de los diferentes procesos identificados en un ambiente web, intuitivo y de fácil manejo.
El (nombre del producto)	Diseño y construcción de procesos de negocio dispersos para apoyo a la gestión de EPP y anexarlos a SCIWeb
Que	Permitirá la autenticación de usuarios basados en roles, transformación automática de precios de productos derivados del petróleo, integración de información de precios Abas-Comer, gestionar los procesos de gestión de pagos y prepago.
A diferencia de	No darle a usuarios de utilitarios internos, seguimiento o control mediante autenticación vía web. Mantener de manera manual el proceso de aseguramiento de precios.
Nuestro producto	Permitirá gestionar la autenticación de usuarios a módulos web, basado en roles. Permitirá el aseguramiento automático de precios de combustibles y afines.

- **Resumen de los usuarios:**

Se resume los usuarios de los módulos de aplicación:

Tabla V. XVI: Resumen de los usuarios

NOMBRE	DESCRIPCIÓN	RESPONSABILIDADES
Sujetos de control y Autenticación basada en roles		
Sujetos de control	Usuarios internos Administradores del sistema	Los usuarios serán validados en los aplicativos mediante servicios web, registrando el acceso y rol del usuario para con el aplicativo.
Generación automática de precios: Aseguramiento de precios		
Comercializadora	Usuarios internos de EPP, Administradores.	Disponen de criterios de evaluación y validarán el proceso de conversión y cálculo de los ficheros externos que traen la información de los productos derivados del petróleo.

- **Ambiente de los usuarios**

El sistema constará de varios módulos, estos módulos se integrarán al ambiente web SCIWeb, por lo que formará parte del principal paquete en línea. Los usuarios ingresarán al sistema identificándose por medio de un usuario y contraseña y tras este paso dispondrán de permisos según el rol identificado en el sistema. Este sistema está regido por estándares de cualquier aplicación web, por tanto los usuarios estarán familiarizados con su entorno.

Los reportes serán generados y visualizados en formato de documento portable: PDF, lo cual también resultará familiar, permitiendo guardar el documento o su impresión en cualquier momento.

- **Necesidades clave de los interesados**

A continuación se describen las necesidades de los afectados:

Tabla V. XVII: Necesidades de los interesados

NECESIDAD	PRIORIDAD	EXPECTATIVA	SOLUCIÓN ACTUAL	SOLUCIÓN PROPUESTA
Permitir el acceso a aplicativos web, basándose en roles asignados a los diferentes usuarios internos	Alta	Mejorar la atención de los usuarios mediante el módulo de autenticación de usuarios basado en roles.	Acceso indiscriminado a aplicativos, cada aplicativo presenta una ventana de login implementada fuera del sistema general.	Interfaces web que permitirán el acceso a aplicativos web, usando servicios web, asegurando las opciones del módulo según rol de usuario.
Permitir el aseguramiento automático de precios de derivados de petróleo.	Alta	Evitar la generación de errores y pérdida de información durante el proceso de cálculo de precios, y su total control.	Proceso semiautomático, con hojas de Excel.	Interfaces web que permitan la subida/carga de información de una fuente externa, para procesarla y emitir reportes con la información solicitada (cambio a moneda local, conversión de unidades, carga de rubros locales, cálculo del precio final)

- **Alternativa y competencias**

No se ha identificado alternativas o competencias.

- **Perspectiva del producto**

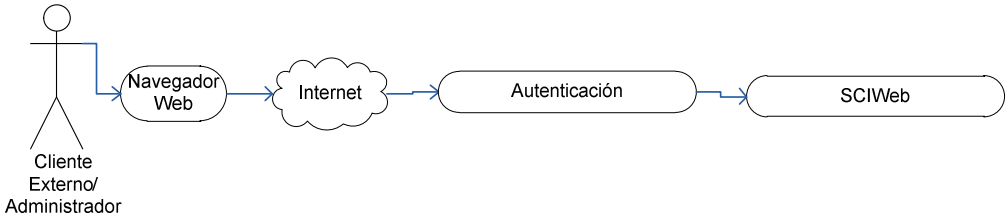


Figura V. 19. Perspectiva del producto

Este sistema será un producto en línea, que utilizará el sistema de seguridades estándar empleado por todos los aplicativos construidos en EPP basados en la misma arquitectura J2EE.

- **Características del producto**

Esta sección define y describe las características del Sistema. Las características son las capacidades de alto nivel del sistema que solucionará a cada una de las necesidades planteadas anteriormente.

1. Autenticación de usuarios basados en roles.

   CU01. Autenticación de usuarios basados en roles: este módulo permite la autenticación de usuarios contra el sistema, para aplicativos web, basado en perfiles. Adicionalmente el módulo hará uso de servicios web que validarán el proceso.

2. Asistente automático de precios (Aseguramiento de precios)

   CU02.- Asistente automático para el aseguramiento de precios: este módulo se enfoca en el aseguramiento de precios de productos derivados combustibles, el cuál procesará un fichero externo, en formato de Microsoft Excel, asegurando los procesos de cálculo y en consecuencia los valores resultantes.

- Transformar los productos internacionales a nacionales (en el exterior se llama: gasolina, en Ecuador la conocemos como extra, etc.)

- Estandarizar las medidas.

- Cargar rubros de impuestos nacionales.

- Calcular el precio final de la venta.

- Registrar las salidas del proceso en la base de datos de facturación de combustibles.

3. **CU03.-** Integración de información Comercializadora - Abastecedora:

- Integración de procesos de información de precios de la base de datos *Abastecedora* y *Comercialización*.

    - **Otros requisitos del producto**

La interfaz de usuario será compatible con Internet Explorer 7 o superior.

- **Riesgos**

    - **Introducción**

La identificación de los riesgos que estarán implicados en el desarrollo de esta aplicación, los que serán considerados por el líder del proyecto en el momento del desarrollo; permitiendo su mitigación o la ejecución de la actividad de contingencia necesaria. Para esto, es importante que los riesgos se prevean con anterioridad.

Estos riesgos serán gestionados de acuerdo a las estrategias y supervisión que se plantee el grupo de desarrollo para cada riesgo.

- **Propósito**

Identificar los riesgos, típicamente en la fase de inicio y hacerles frente. Para ello trataremos de evitarlos, o asumirlos. En este último caso habrá que tratar de mitigar el riesgo y definir un plan de contingencia paralelo que se presente efectivamente.

- **Alcance**

El presente proyecto formará parte del sistema central SCIWeb, por lo que dependerá completamente de su arquitectura, además, hará consumo de librerías, o servicios existentes en la organización..

- **Visión general**

La Poca especificación en la recolección de los requerimientos por parte de los usuarios, se constituirá en el principal riesgo de la aplicación, así como la incertidumbre del equipo de trabajo referente a la tecnología usada para implantar la solución.

- **Riesgos**

  - **R1: Mala especificación de requerimientos**

Magnitud o rango del riesgo:

Riesto Técnico (RT) Prioridad Alta.

Descripción

Especificación no completa de requerimientos por parte del nivel estratégico, o continua modificación de los mismos.

Impactos

- Retraso en fase de elaboración y construcción.
- Calidad de la solución.
- Retraso del análisis y la implementación.

Indicadores

- Dificultad al momento de analizar la Visión del producto,
- Falta de aprobación de la Especificación de los Casos de Uso.

Estrategia de mitigación

- Correcta ejecución en la ingeniería de requerimientos.
- Aplicación de técnicas de recolección de datos, entrevistas, encuestas, criterio personal, etc.
- Revisión con los expertos del negocio sobre la especificación de los Casos de Uso.

- **R2: Personal de desarrollo inexperto**

Magnitud o rango

Riesgo Técnico (RT) Prioridad Media.

Descripción

Personal o equipo de desarrollo no está capacitado en la plataforma y/o arquitectura de la solución propuesta.

Impactos

- Retraso en las fases de elaboración, construcción, transición.
- Baja calidad del producto de software.

Indicadores

Baja calidad de artefactos de la fase de Incepción (Visión, riesgos, Plan de desarrollo de Software)

Estrategia de mitigación

Proporcionar ejemplos y realización de guías a los nuevos participantes del equipo de trabajo.

Se ha resumido el documento de riesgos en el siguiente cuadro, a diferencia de RUP recomienda redactar a renglón seguido:

Tabla V. XVIII: Resumen de riesgos

CLASIFICACIÓN DEL RIESGO/ MAGNITUD	IMPACTO & DESCRIPCIÓN DEL RIESGO	ESTRATEGIA DE MITIGACIÓN Y/O PLAN DE CONTINGENCIA
7	La Versión V1 podrían no estar disponible fecha planificada, por cuanto el personal de Servicios Administrativos para esa fecha sale de vacaciones.	Monitor progress against the schedule & milestones.
		Incrementar el esfuerzo para completa a tiempo sobre una base regular esta versión preliminar.
5	Incompatibilidad del framework de trabajo (RichFaces) con la versión del WAS 60. Esto podría afectar el rendimiento, estabilidad y funcionamiento del sistema	Sugerir urgentemente la actualización de la versión del WAS de 6.0 a 6.1 al área de Ingeniería y Procesamiento.
5	Compartir el WAS con otros aplicativos empresariales.	Desarrollar tempranamente prototipos para probar rendimiento.
4	No adaptabilidad del Sistema de Seguridades de la empresa con framework de Struts al sistema que será implementado con el framework RichFaces.	Generación de Diagrama de Componentes para encontrar solución posible.
3	Incompatibilidad con los navegadores de Internet y las configuraciones en las máquinas clientes.	Dirección y prueba de compatibilidad durante la fase de elaboración.
3	El estudiante tiene relativamente poca experiencia con el Rational Unified Process (RUP) y Técnicas de Orientación a Objetos, así como en el framework de trabajo (RichFaces). Esto podría conducir a una reducción de la eficiencia y la calidad de los productos más pobres	Calendarizar autolecciones de entrenamientio sobre RUP.
		Establecer sesiones de asesoría soporte en el framework de RichaFaces por parte de otros equipos de trabajo.
		Garantizar que todo el diseño y código sea inspeccionado.

- **Casos de Uso Generales**

A continuación se diagrama los casos de uso generales, los mismos que serán explotados en el siguiente entregable.

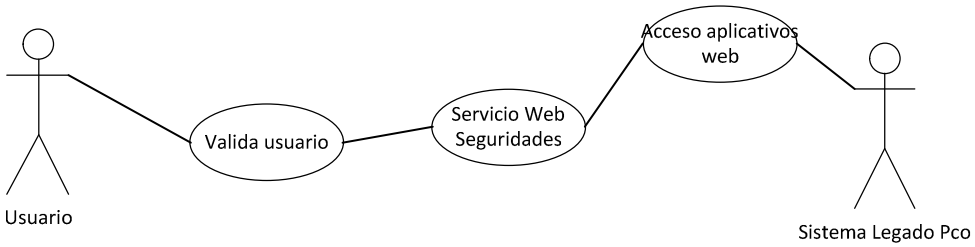


Figura V. 20. Casos de uso generales, módulo autenticación web

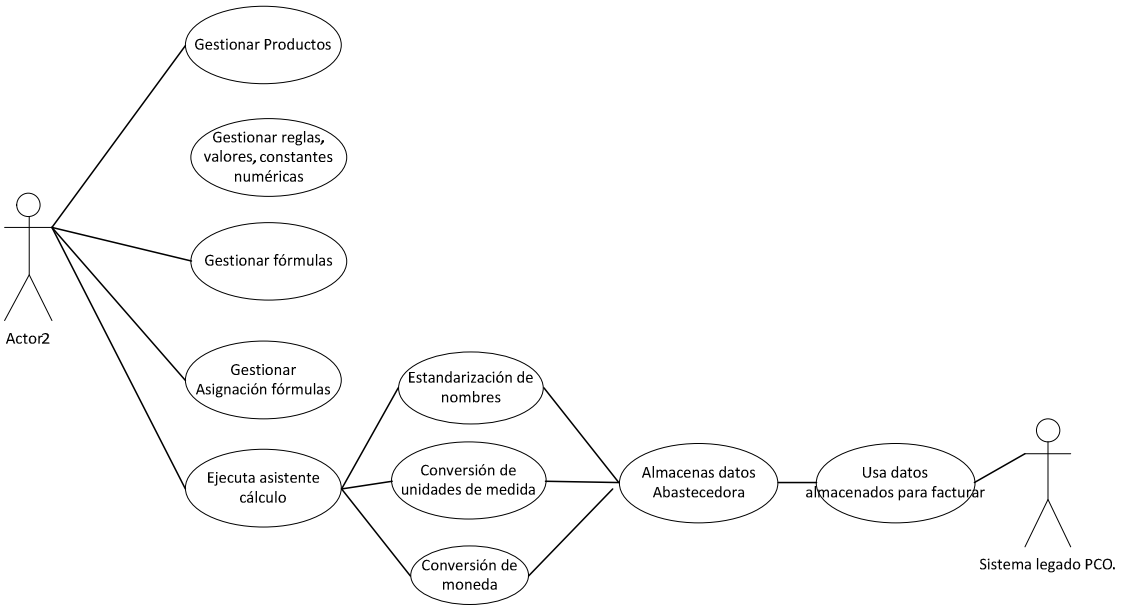


Figura V. 21. Casos de uso generales, módulo aseguramiento de precios e integración con abastecedra

- **Especificación de casos de uso**

  - **Aseguramiento de precios**

    Este caso de uso lo realiza un sujeto de la Comercializadora: usuarios internos de EPP, Administradores, con poder de decisión y validación de resultados.

    Le permite la ejecución de un asistente guiado para la carga del fichero Microsoft Excel, de una fuente externa, como base para el proceso de aseguramiento de precios.

    Los precios de los derivados combustibles son calculados en base a datos de distintos orígenes, uno de ellos se obtiene a través de una Hoja de Microsoft Excel, que es originada en el Exterior, la posibilidad de que este archivo sufra cambios, y por ende se produzcan errores en el cálculo de los precios da lugar a la necesidad de asegurar de la mejor manera posible dichos procesos de cálculo y en consecuencia los valores resultantes.

  - **Flujo de Eventos**

    1. El programa solicitará la carga del fichero externo en formato de Microsoft Excel.

    2. Cuando los datos han sido cargados exitosamente, se visualiza una lista con los siguientes datos: CÓDIGO INTERNACIONAL, FECHA, HIGH, LOW, CLOSE como indicadores del precio internacional.

    3. A continuación, se obtiene una lista de precios cargados, transformados a las unidades de medida utilizadas en el país.

4. Se realiza el cálculo en base a los periodos estipulados por el usuario.

5. Los datos obtenidos del cálculo, son transformados en una estructura de datos que está establecida por el sistema de la abastecedora.

6. Usuario observa y valida los datos.

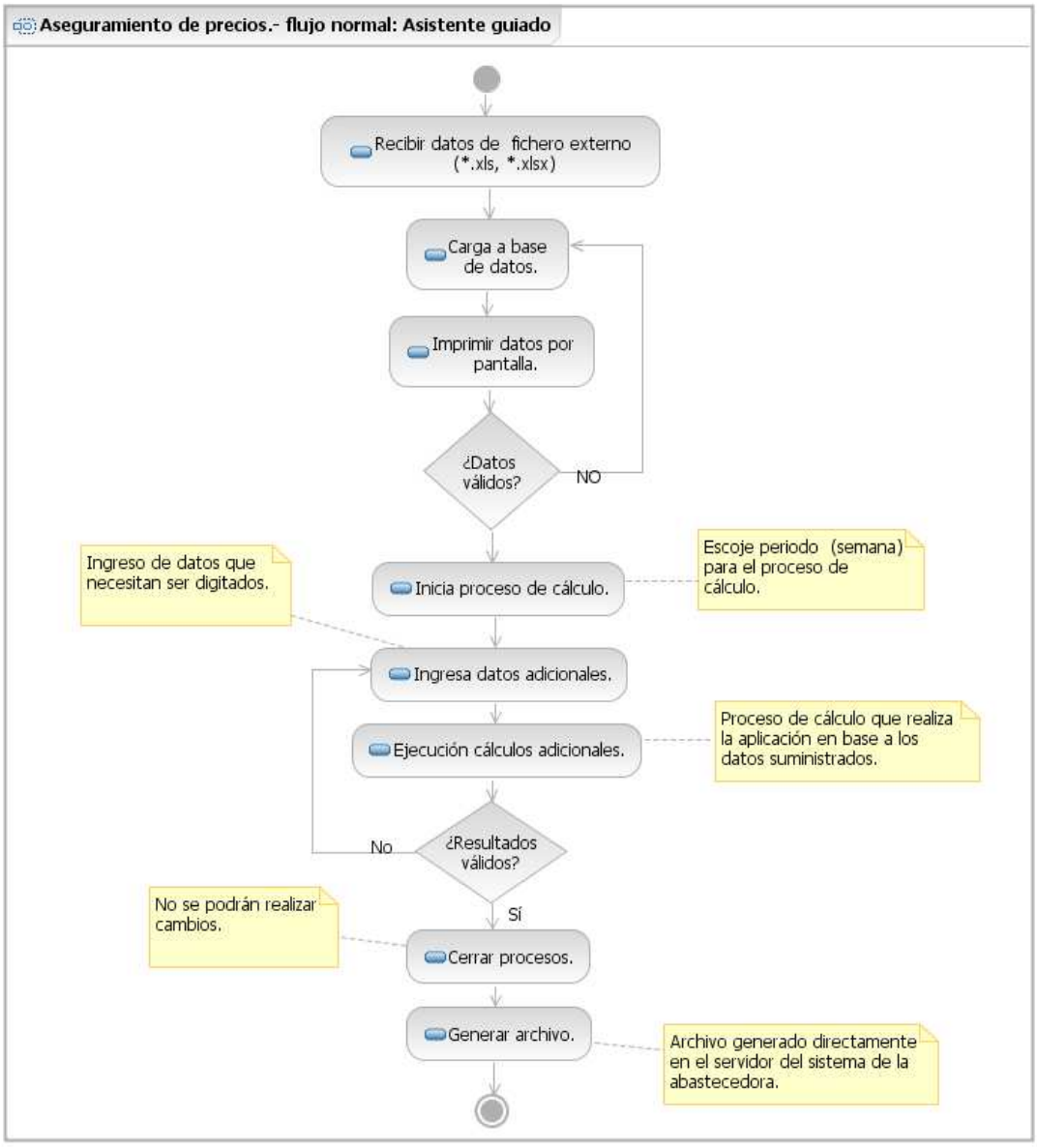


Figura V. 21. Especificación de casos de uso: Aseguramiento de precios, integración con la abastecedora

- **Flujos alternativos**

  - En el punto 1 de 2.2.1:

  Archivo fuente externo, con formato Microsoft Excel no presenta el formato especificado por EPP, el sistema emitirá un mensaje de error al respecto.

  - En el punto 4 de 2.2.1:

  Periodo no válido, rango de fechas erróneo, el sistema emitirá un mensaje de error al respecto.

- **Requerimientos especiales**

  - Fichero externo de Microsoft Excel

  El fichero deberá cumplir el formato establecido por la comercializadora.

  - Datos

  El fichero deberá cumplir con los requerimientos de datos íntegros.

  - Precondiciones

  El fichero deberá cumplir el formato establecido por la comercializadora.

- Postcondiciones

Cálculos realizados pasan a generarse en el sistema de la abastecedora

5.2.2. Fase de elaboración

- **Arquitectura de software**

  - **Propósito**

Este documento proporciona una visión global de la arquitectura del sistema, usando un número de diferentes puntos de vista de arquitectura para representar diferentes aspectos del sistema. Su objetivo es captar y transmitir las decisiones importantes de arquitectura que se han hecho en el sistema.

  - **Referencia arquitectónica**

Este documento presenta la arquitectura como una serie de vistas; vista de casos de uso, vista lógica, vista de implementación y vista de despliegue. Todas estas vistas sobre un modelo Modelamiento de Lenguaje Unificado (UML) desarrollado usando Rational Software Architect.

- **Vistas de casos de uso**

Una descripción de los Casos de uso esta descrita en las Especificación de Casos de Uso.

La Vista de Casos de Uso es una entrada importante al conjunto de escenarios y/o casos de uso que son el enfoque de una iteración. Este describe el conjunto de escenarios y/o casos de uso que representan funcionalidad central importante. Esto también describe el conjunto de escenarios y/o casos de uso que tiene una cobertura arquitectural sustancial (que ejercita muchos elementos arquitecturales) o que estresa o ilustra un punto específico delicado de la arquitectura.

Los casos de uso son los siguientes:

- Acceso al Sistema.

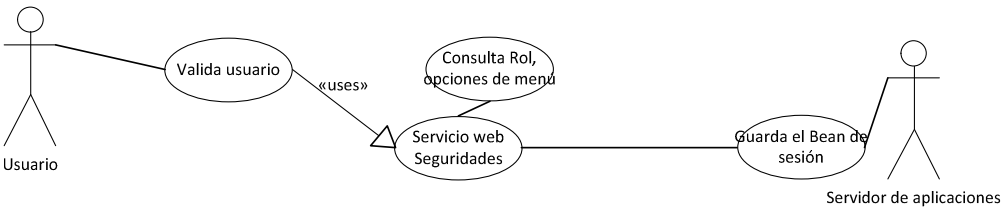


Figura V. 22. CU acceso al sistema.

- Aseguramiento de precios

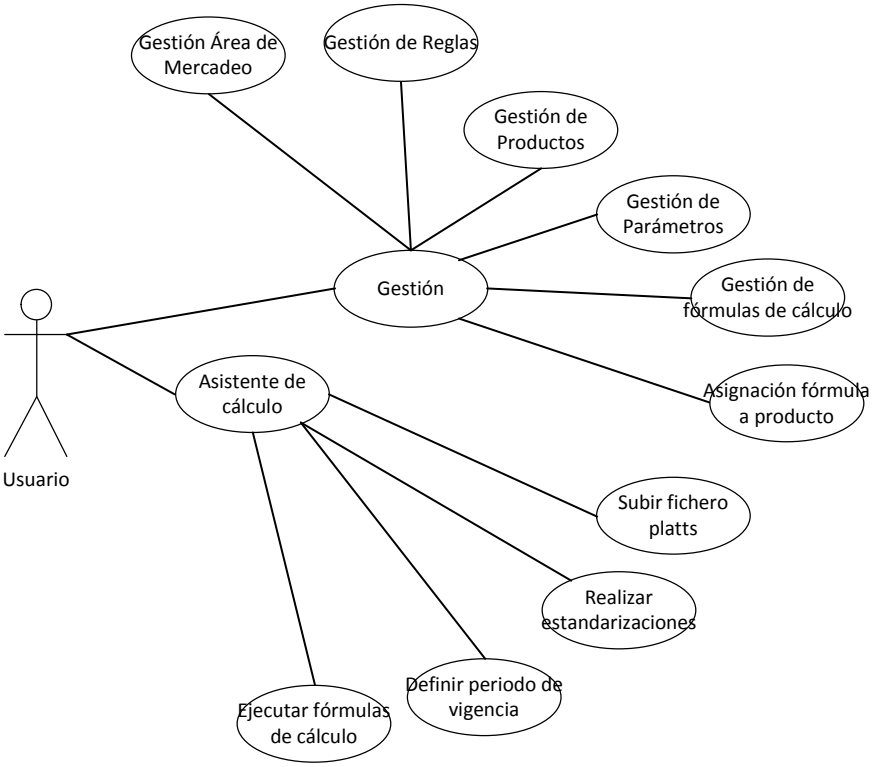


Figura V. 23. CU Parametrización del sistema.

- Parametrización del Sistema.

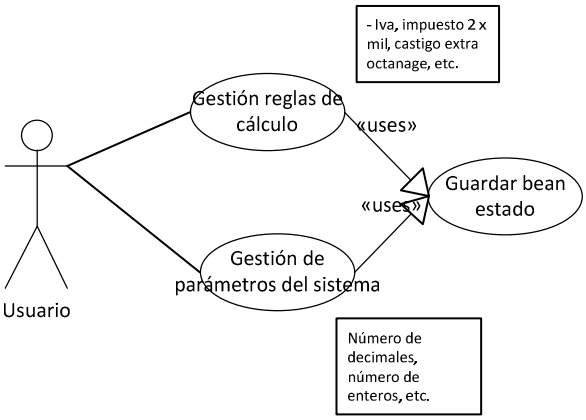


Figura V. 24. CU Parametrización del sistema.

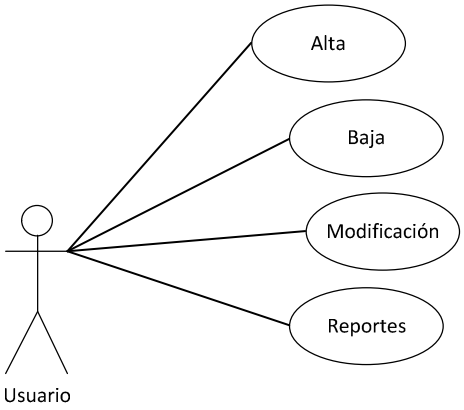


Figura V. 25. CU Parametrización entidades del negocio.

- Reportes.

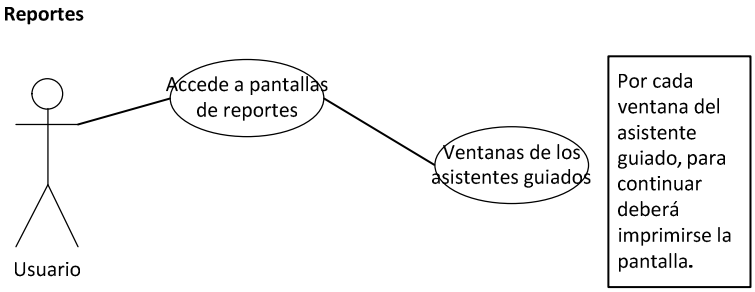


Figura V. 26. CU Generación de reportes.

- **Paquetes y capas de subsistema**

Los paquetes fundamentales son los siguientes:

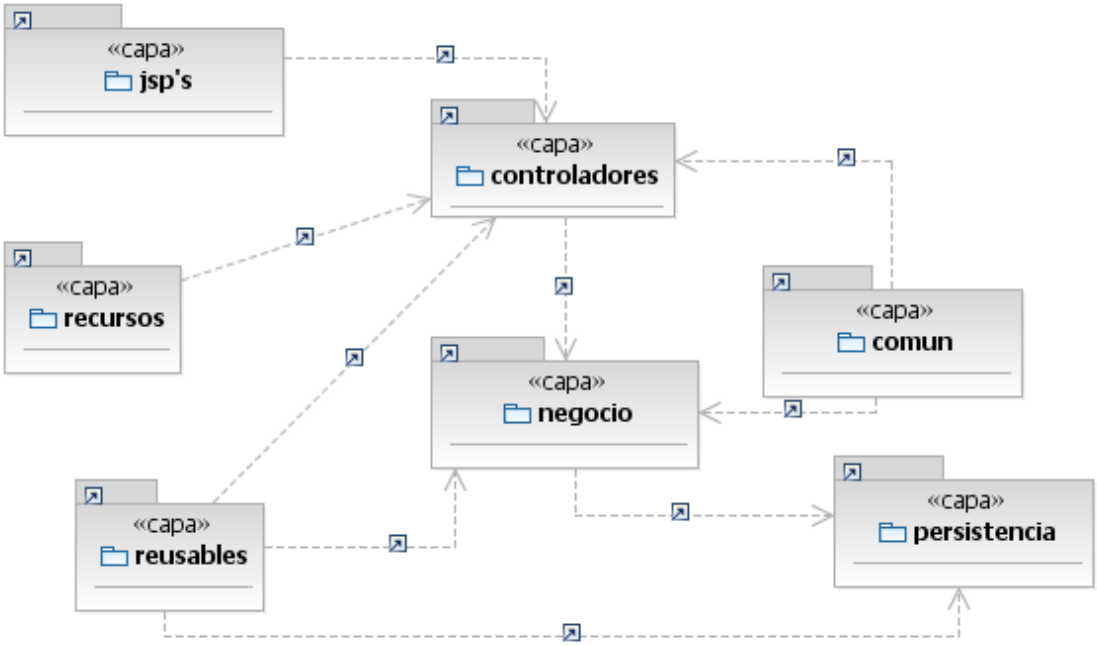


Figura V. 27. Paquetes y capa

- **Vista de Despliegue**

Se muestra a continuación el diagrama general en el cual el software se despliega y se ejecuta:

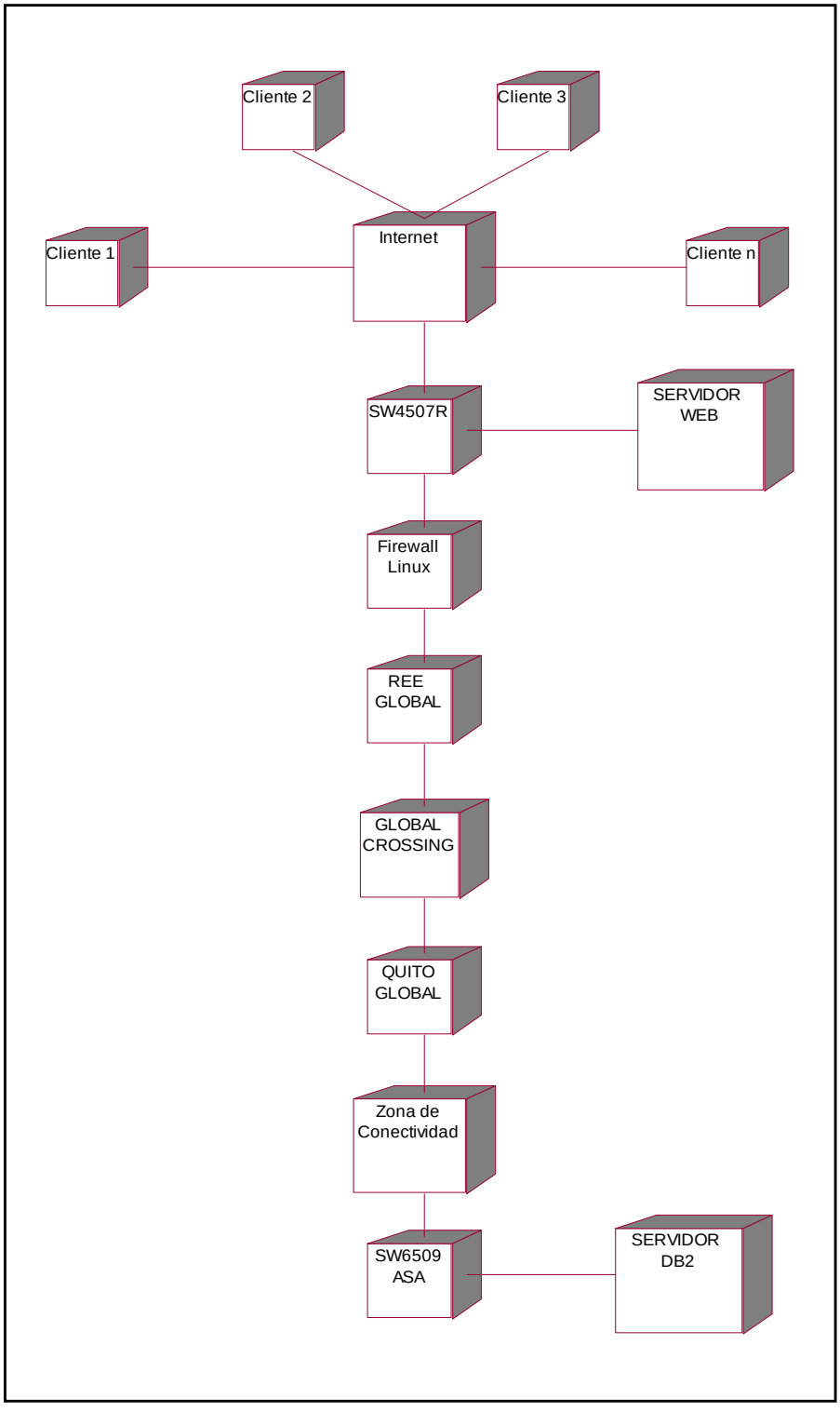


Figura V. 28. Vista de despliegue

- **Capas**

Diseño arquitectónico por capas usadas en el aplicativo:

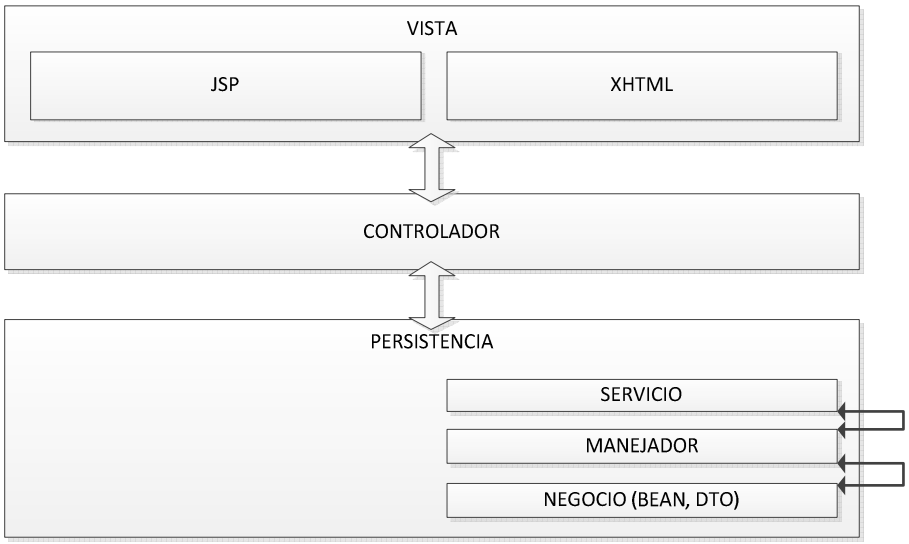


Figura V. 29. Vista de despliegue

- **Vista de datos**

Modelo conceptual de la base de datos:

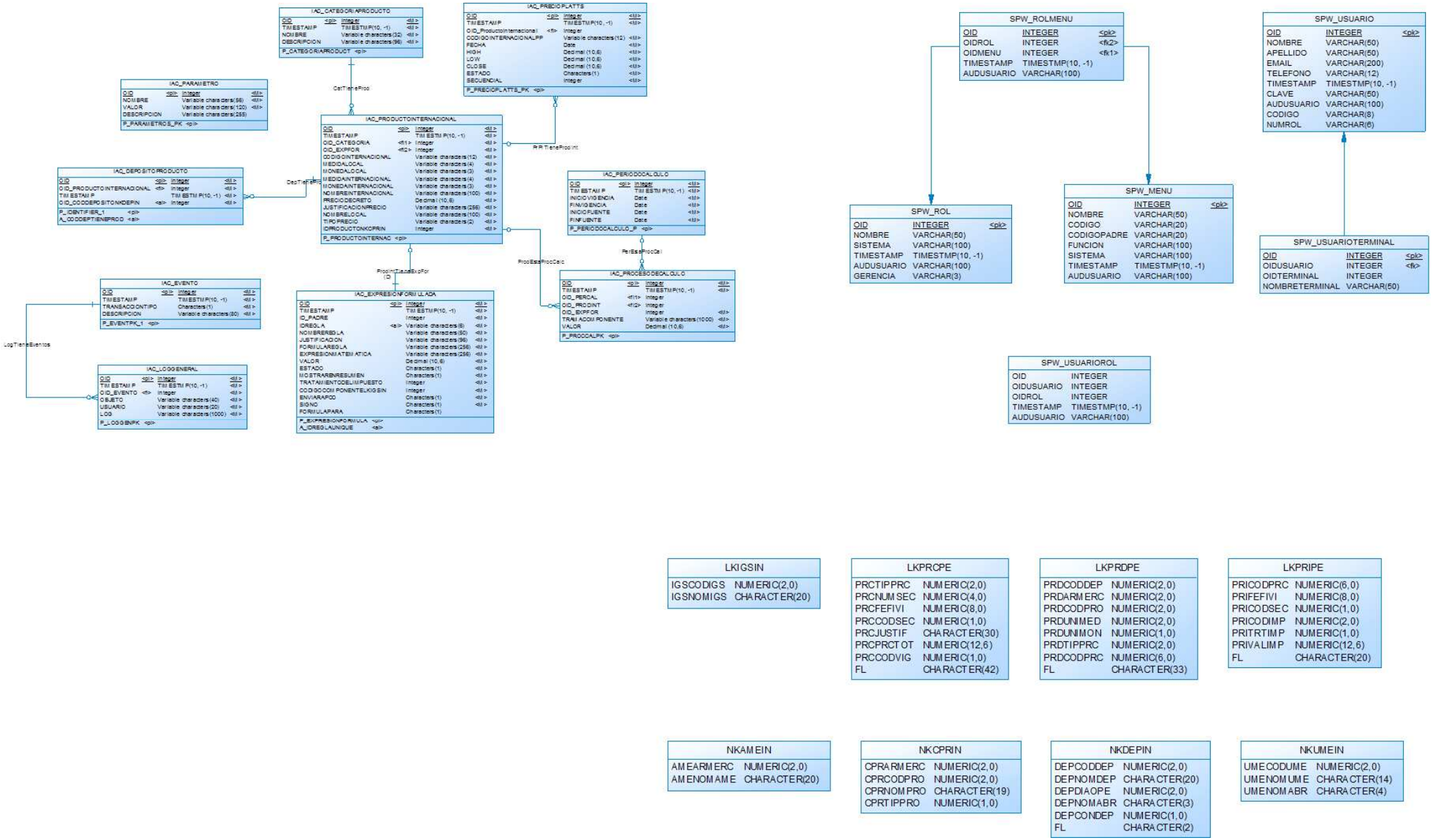
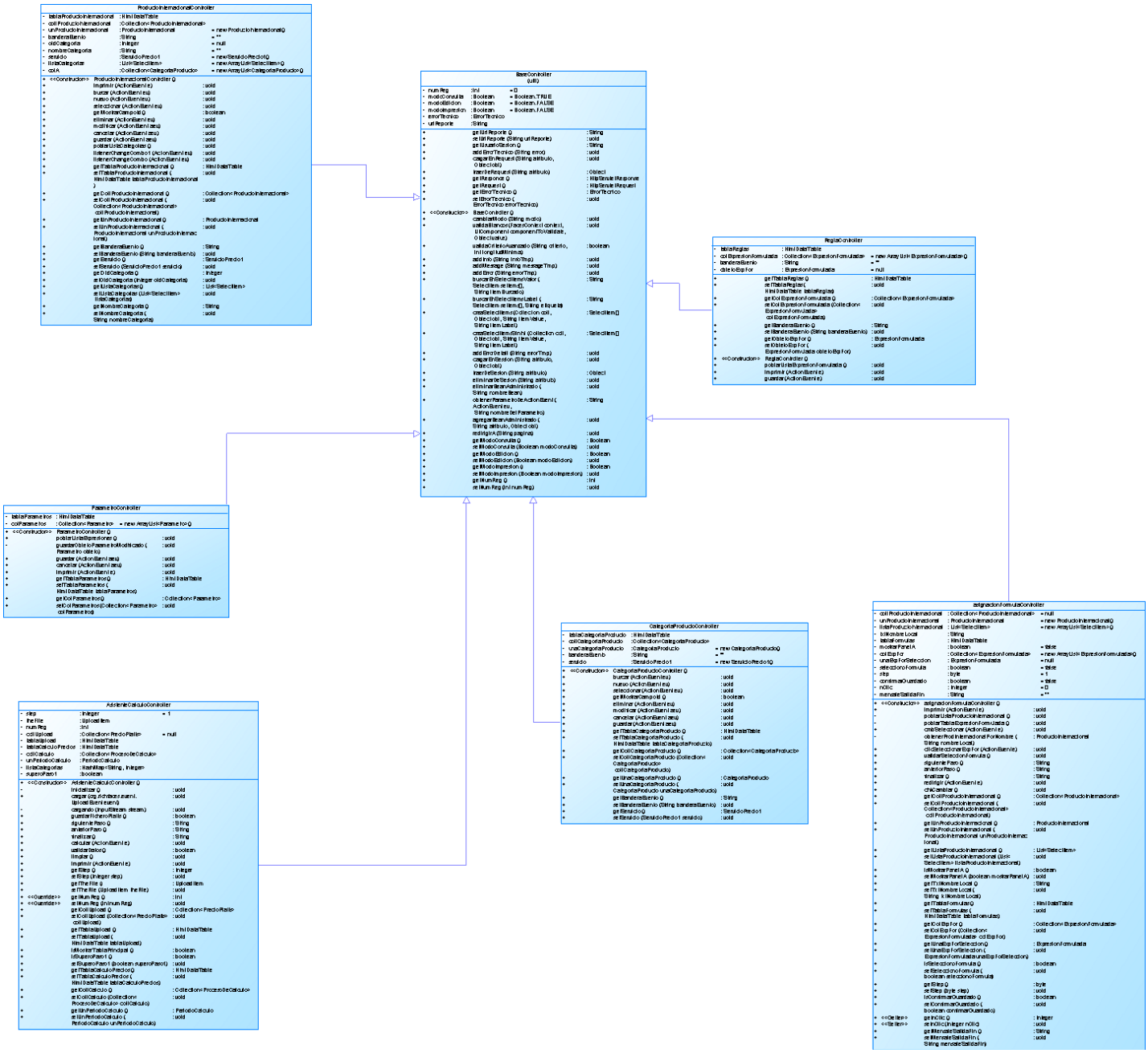


Figura V. 30. Vista de datos

- **Vista de objetos**

Capa controlador, vista general:



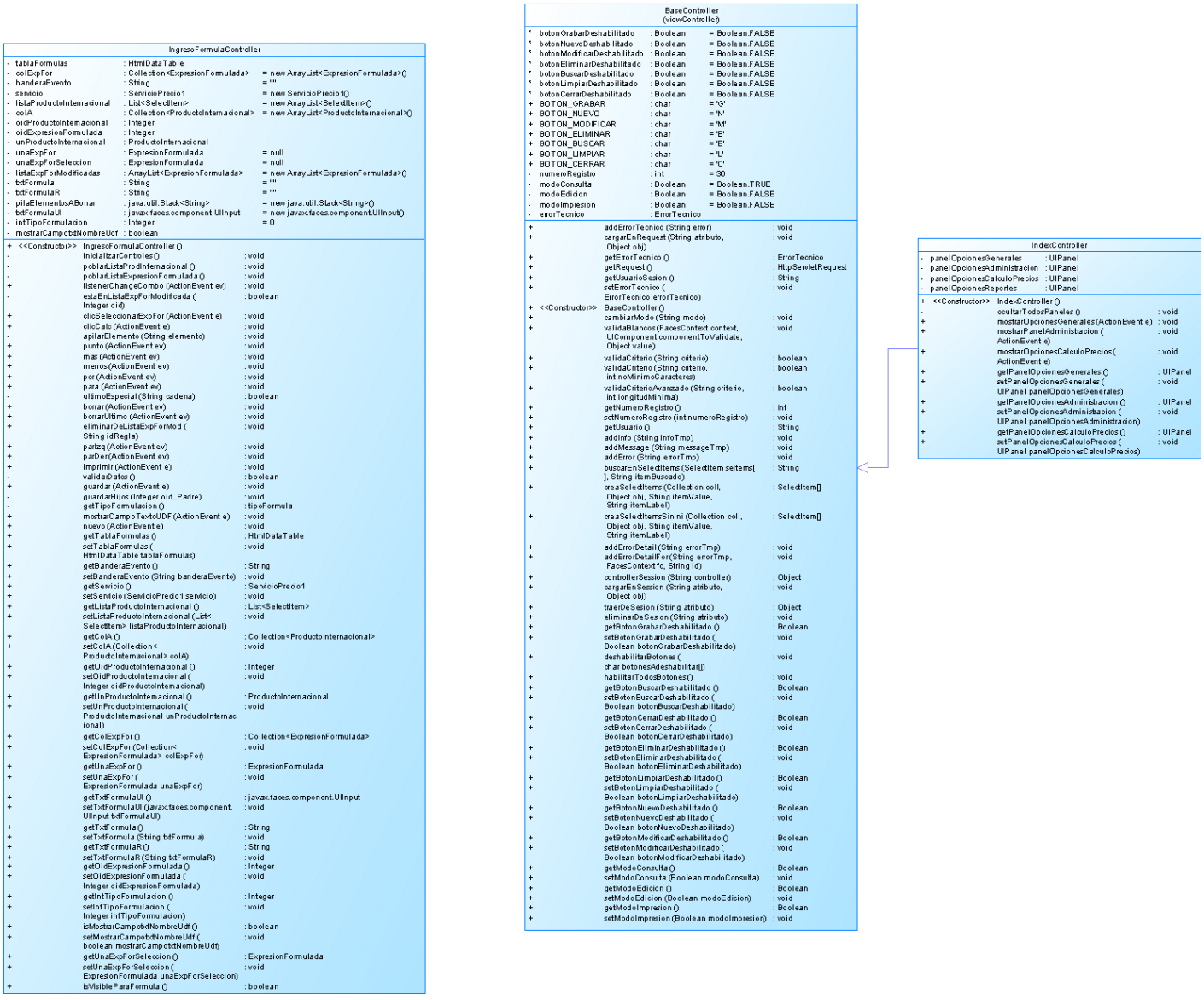
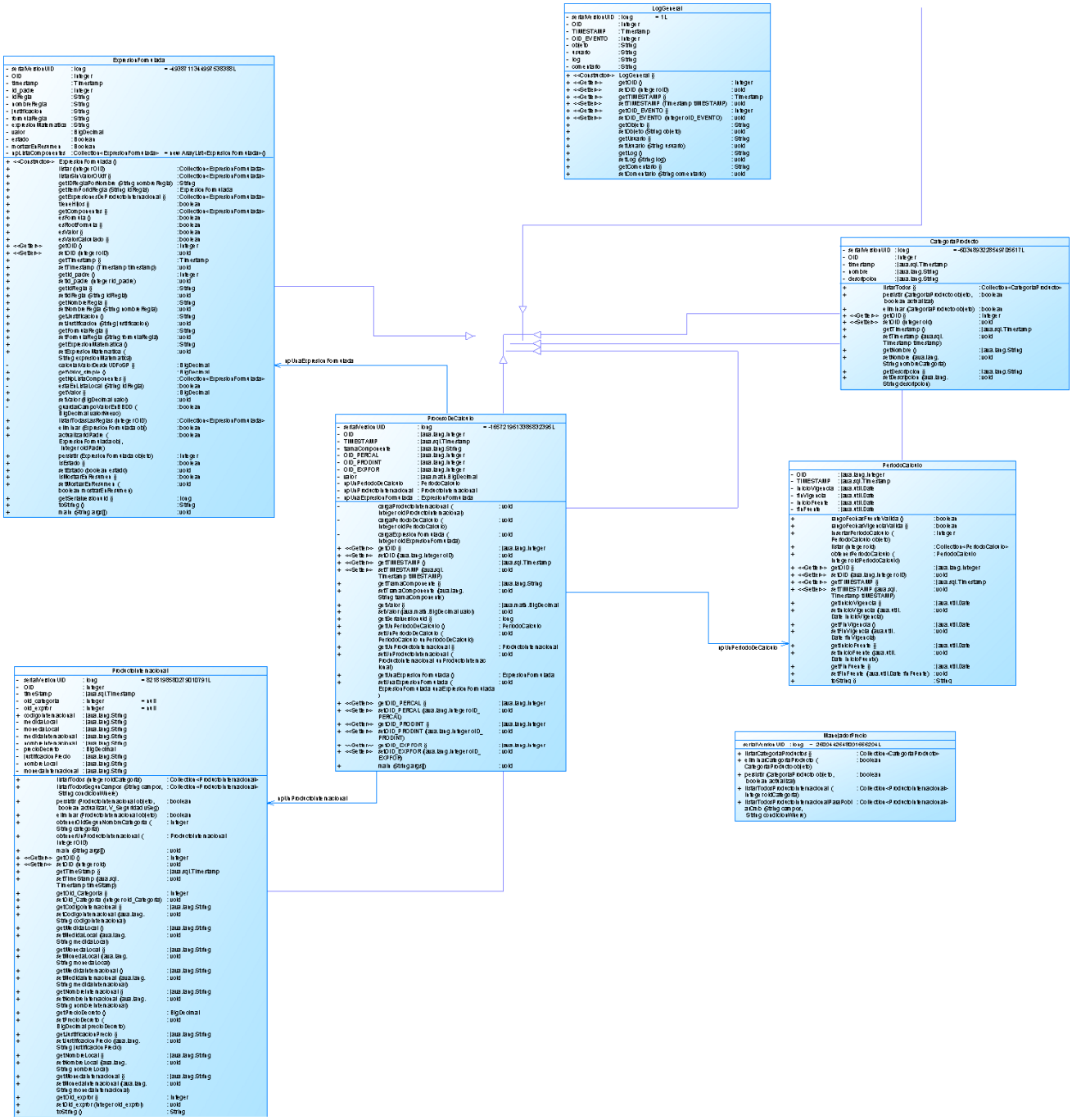


Figura V. 31. Vista de objetos, controlador


LogGeneral
ExpresionFormateada
CategoriaProducto
PeriodoDeCalculo
PrioridadDeLaNacional
ManejadorPrecio
usUnaExpresionFormateada
usUnPeriodoDeCalculo
usUnaPrioridadDeLaNacional

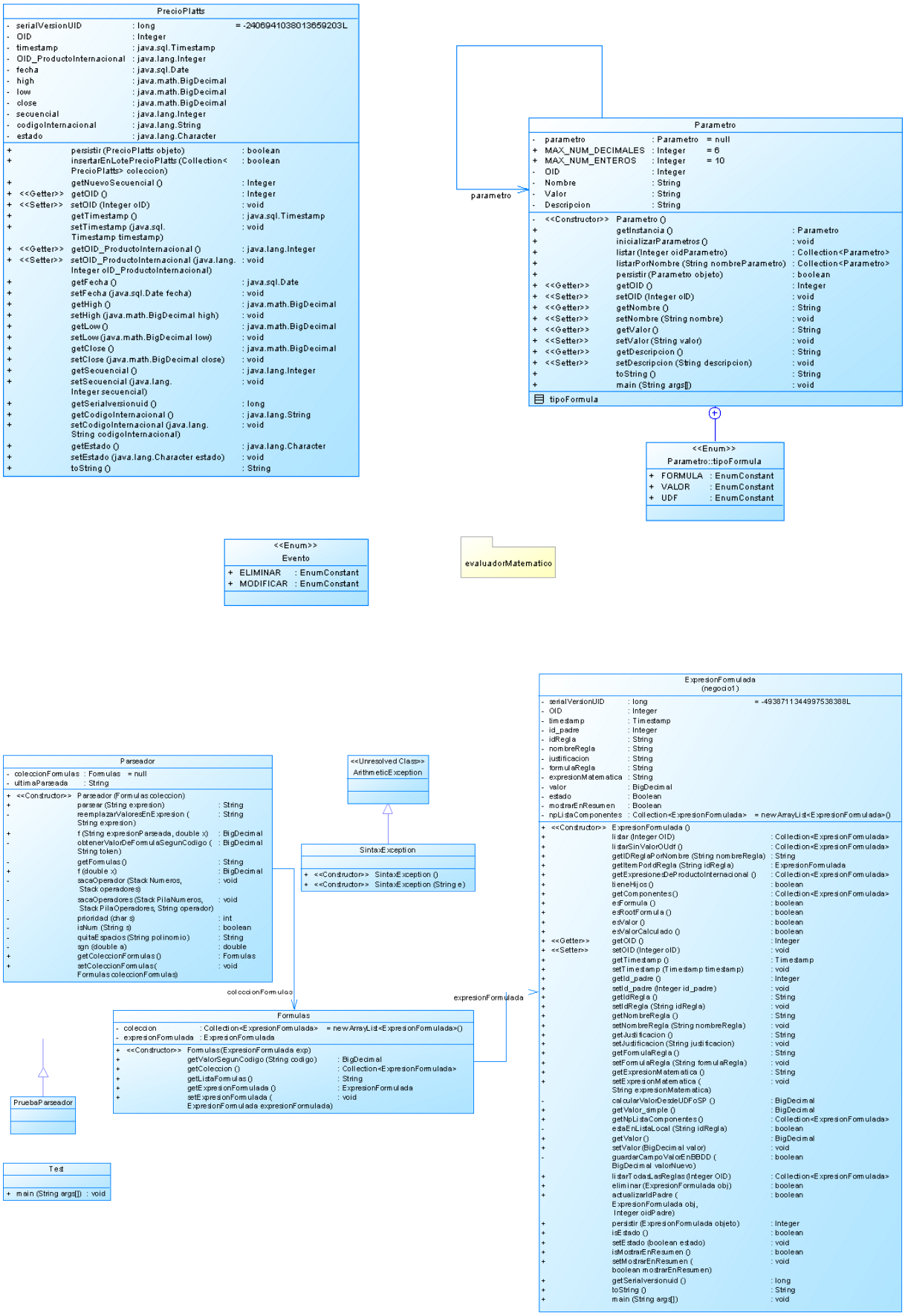


Figura V. 32. Vista de objetos, negocio

5.2.3. Fase de Construcción

- **Documento de orden de trabajo de ingeniería de software**

Este documento describe las actividades que han de ser realizadas así como las salidas esperadas, se hace referencia a la descripción de actividades a ejecutar así como los documentos o productos (código) que serán producidos.

- **Evaluación estatus del proyecto**

Este documento contiene un reporte del progreso del trabajo, resultados de hitos completados y una lista de acciones para corregir cualquier desviación de la ejecución del proyecto.

- **Completar el análisis, diseño, desarrollo y pruebas**

El propósito de generar todas estas tareas es el de completar toda la funcionalidad requerida del sistema en cuestión.

- **Generación de versiones**

Con esta actividad se pretende generar versiones del software, alpha, beta u otras pruebas de liberación, cada vez más estables.

5.2.4. Fase de Transición

Para la fase de transición se deben tener contemplados los siguientes entregables:

- Producto

  El producto es el propósito. El esfuerzo del proyecto entero acoplado para crear un producto que provea un beneficio al cliente.

- Material de soporte para el usuario final:

  Ver anexo 2, 3

CONCLUSIONES

Como resultado del desarrollo del presente documento de tesis en donde se realizó como primer punto el análisis de conceptos de la tecnología JSF, se lo comparó con el framework más famoso de aplicaciones web empresariales en Java: Struts, para justificar la implementación de módulos de aplicación para EP Petroecuador, se muestra las siguientes conclusiones:

- La evolución de los frameworks web marca una tendencia clara a la abstracción del protocolo en el que se sustenta (`http`) para beneficiarse de los modelos basados en controles y componentes que tanto éxito tuvieron en el ámbito del escritorio.

- Las diferentes aproximaciones de los frameworks muestran que, en la mayoría de estos, se atacan los mismos problemas: navegación, internacionalización, manejo de errores, validación de entradas, escalabilidad, etc.

- `Struts` permiten un manejo más directo de los datos que se procesan mientras que los más modernos como JSF, buscan la abstracción casi total del protocolo pero a cambio de generar modelos de componentes extensibles y orientados a eventos.

- JSF es un conjunto de herramientas para el desarrollo de aplicaciones Web, donde se busca encapsular todos los elementos necesarios para que el desarrollo de aplicaciones Web se vea simplificado, permitiendo un desarrollo parecido al de componentes locales como Java Swing, AWT (`Abstract Window Toolkit`), SWT (`Standard Widget Toolkit`) o cualquier otra API similar.

- La arquitectura de JSF, separa claramente las capas del modelo tradicional de aplicaciones web empresariales: MVC, permitiendo desarrollar por separado cada capa, y presenta un api extensible, funcional, para desarrollar componentes a la medida, tanto de la vista como del controlador.

- JSF, interactúa con una gran cantidad de componentes Open Source, lo que lo hace muy extensible, su API se interconecta, o interacciona con diferentes tipos de frameworks, de persistena (`Hibernate`, `ibatis`, etc) o de validación (`hibernate validator`, `bean validator`), o frameworks de caché de aplicación (`django`).

- La capa de presentación presenta una rica arquitectura, soporta varias tecnologías de presentación, como `jsp` o `facelets`, para este documento, se trabajó con `facelets`, como motor de presentación, ya que evita la compilación y construcción del objeto `Servlet` al momento de usar el motor de presentación de `jsp`.

- Se mostró la superioridad de JSF sobre `Struts` mediante un análisis comparativo que analizó parámetros de comportamiento y conectividad, con resultados que marcan la una implementación más actual de JSF, contra `Struts` que se está rezagado en su evolución.

- Los 2 frameworks analizados buscan solucionar el problema de la separación de funciones, un tema que traía más de un dolor de cabeza a los equipos en los que había tanto desarrolladores como diseñadores gráficos. No todos lo solucionan de la mejor manera. Sin embargo, JSF se muestra mayor amigable y accesible para el diseñador web que ya tiene experiencia con etiquetas `html`.

- El futuro actual y próximo está marcado por las interfaces ricas, una interactividad sin necesidad de realizar consultas permanentes al servidor ni

realizar recargas completas con la posibilidad que desde el cliente se puedan acceder a diferentes servidores a la vez, lo que marca un cambio en la concepción del navegador web, que pasa de ser una interfaz sin inteligencia ni estado (salvo por las cookies) a otra con la posibilidad de procesar la información y mantener un estado propio.

- Como reflexión final, expresar que lo mejor de conocer varios frameworks es saber que no existe uno que sea la solución a todos los problemas sino que cada uno fue desarrollado con objetivos diferentes y es necesario ver cuál de todos se alinea mejor con los objetivos de nuestro proyecto evaluando ventajas y desventajas de cada uno.

RECOMENDACIONES

A partir del desarrollo del presente trabajo se dará algunas recomendaciones respecto del mismo, que los lectores deberán tomar en cuenta para la correcta utilización del framework JSF que ha sido objeto de este estudio:

- Se recomienda que el desarrollador entienda el funcionamiento general del modelo MVC para la implementación de una aplicación web empresarial con JavaServer Faces, MVC es el patrón de arquitectura base sobre la cual se implementa cualquier framework web en la actualidad.

- Se recomienda también que el desarrollador comprenda el ciclo de vida de una aplicación JSF, aunque no es necesario que lo comprenda en su totalidad, sin duda, le ayudará a desarrollar y depurar de mejor manera la aplicación.

- Referente a la lógica de vista, JSF puede renderizarse usando JSP, que a su vez producirá el código HTML de la aplicación, sin embargo, esto implica generar un Servlet por cada página JSP que la aplicación contenga, la recomendación es usar FACELETS, como motor de presentación, ya que se evita la generación de los Servlet antes mencionados, además, permite la integración de librerías de componentes Rich, las mismas que tienen soporte Ajax "out of the box".

- Un buen diseño arquitectónico garantiza estabilidad y flexibilidad en la aplicación, lo que se refleja en un valor agregado en el aplicativo, ya que sin un buen diseño, el consumo elevado de memoria de la máquina virtual Java, conlleva a tiempos muertos en el procesador, lo que se traduce en degradación del tiempo de respuesta.

- Las conexiones a la base de datos son recursos finitos y costosos, y el API de jdbc es conocido por su alto desempeño, así como su alto consumo de recursos, se recomienda la implementación de estategias de "caching" o cacheo de información, utilizando frameworks de persistencia, y/o la implementación de un pool de conexiones en el servidor de base de datos.

- Se recomienda también, tener en cuenta el consumo de memoria de los beans que se implementen en ámbito de session, ya que la memoria consumida por estos tipos de objetos no se libera rápidamente, lo que puede llevar a la jvm a tiempos elevados de recolección de basura, o lo que es peor, desbordamiento de memoria.

- De igual manera, eliminar las sesiones lo más pronto posible, implementando botones de cerrar sesión que eliminen los beans administrados, o mediante la implementación de filtros JSF, que controlarán la navegación en la aplicación, eliminando los beans que ya no están siendo visitados.

- Evite el uso de binding entre componentes visuales html y el bean controlador, debido a que estos objetos consumen gran cantidad de memoria, evite el uso excesivo de este tipo de implementación.

    - La optimización de una aplicación JavaServer Faces mediante el archivo de configuración de la aplicación web: web.xml, puede mejorar el consumo de ancho de banda y de memoria, puede iniciar personalizando los siguientes parámetros:

        - javax.faces.STATE_SAVING_METHOD
        - org.apache.myfaces.COMPRESS_STATE_IN_SESSION
        - org.apache.myfaces.SERIALIZE_STATE_IN_SESSION
        - org.apache.myfaces.NUMBER_OF_VIEWS_IN_SESSION

- facelets.DEVELOPMENT.

- El desarrollo de la aplicación web en Java, puede verse muy beneficiado al usar un IDE configurado de manera correcta, con herramientas específicas para el framework, se recomienda el uso de Eclipse Helios o superior, con los plugins de desarrollo de aplicaciones Web: JBoss Tools 3.x.x, puesto que tienen soporte visual para JSF, así como para facelets, RichFaces, y para archivos configurables xml; el mismo paquete ofrece herramientas para usarse con Struts.

- En general, tenemos que expresar que lo mejor de conocer varios frameworks es saber que no existe uno que sea la solución a todos los problemas sino que cada uno fue desarrollado con objetivos diferentes y es necesario ver cuál de todos se alinea mejor con los objetivos de nuestro proyecto evaluando ventajas y desventajas de cada uno.

RESUMEN

Se estudió y analizó la tecnología JavaServer Faces (JSF) para el desarrollo de aplicaciones web para procesos de comercialización de productos de la Subgerencia de Comercialización de Hidrocarburos del Ecuador, en la ciudad de Quito.

Para el desarrollo de la investigación se hizo uso del método Científico y del método Descriptivo, con el fin de levantar y recopilar información, analizar e interpretar los resultados que permitirán la comprobación de la hipótesis; así como también se describirá características de la tecnología JavaServer Faces.

Para la demostración de la hipótesis se empleó herramientas como técnicas de entrevistas a los empleados de la Subgerencia de Comercialización de la Empresa de Hidrocarburos del Ecuador (EP Petroecuador), así como también software de benchmarking y pruebas de stress sobre, los procesos automatizados, con lo cual se puede concluir que con el desarrollo de la aplicación web usando la tecnología JavaServer Faces (JSF), se ha podido mejorarlos procesos en línea en un 18%.

Analizado y comparando las tecnologías para desarrollo de aplicaciones web más importantes en Java: Struts y JSF, se eligió JSF como el más adecuado, porque alcanzó el mayor puntaje: 94,87%, en comparación con el 76,92% obtenido por Struts.

La tecnología JSF es de gran ayuda para los desarrolladores Java, porque permite un desarrollo semejante al desarrollo típico de aplicaciones de escritorio, reduciendo con ello tiempo en el desarrollo de la aplicación, y se constituye en la aproximación más cercana al modelo: Modelo Vista Controlador (MVC).

SUMMARY

We studied and analyzed the JavaServer Faces (JSF) technology, for the development of web applications for marketing Manager Office for the Ecuador's oil company in Quito - Ecuador.

For the development of the research a study was conducted using the scientific and descriptive method, in order to lift and collect information, also, we analyze and interpret the results that will allow the hypothesis demonstration, and also describe the features of JavaServer Faces.

In order to hypothesis demonstration, we used tools such as interview techniques to employees of the Marketing Manager Office of Oil Company of Ecuador (EP Petroecuador), as well, we used benchmarking software and stress test on automated processes, Thus it can be concluded that the web application development using JavaServer Faces (JSF) has been improved on-line process by 18%.

Also, we analyzing and comparing the more important technologies for web application development in Java: Struts and JSF, JSF was chosen as the most accurate, because it reached the highest score: 94.87%, compared with 76.92% reached by Struts.

JSF technology is helpful for Java developers because it allows a development similar to the typical development of desktop applications, also JSF reduces application’s development time, and it is the closest approximation to the model: Model View Controller (MVC).

Glosario

JDBC	Java Database Connectivity, API de Java que permite hacer conexiones a bases de datos para manipular la información que están contienen.
Query	Consultas sobre una base de datos o sobre una bodega de datos. Permiten manipular la información de las bases de datos y las bodegas.
RDBMS	Manejador de bases de datos relacional.
Página web	Página de Internet, es un documento electrónico adaptado para la Web, pero normalmente forma parte de un sitio Web. Su principal característica son los hipervínculos de una página, siendo esto el fundamento de la Web.
Sitio Web	Es un conjunto de páginas web, típicamente comunes a un dominio de Internet o subdominio en la World Wide Web en Internet.
Aplicación Web	En la ingeniería de software se denomina aplicación web a 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.
Servidor Web	Un servidor web es un programa que está diseñado para transferir hipertextos, páginas web o páginas HTML (HyperText Markup Language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música. El programa implementa el protocolo HTTP (HyperText Transfer Protocol) que pertenece a la capa de aplicación del modelo OSI.
Protocolo	Es un conjunto de reglas usadas por computadoras para comunicarse unas con otras a través de una red. Un protocolo es una convención o estándar que controla o permite la conexión, comunicación, y transferencia de datos

	entre dos puntos finales.
JavaServer Faces	Es una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE.
Struts	Struts es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC bajo la plataforma Java EE (Java Enterprise Edition).
Spring	Es un framework de código abierto de desarrollo de aplicaciones para la plataforma Java.
Hibernate	Hibernate es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java (y disponible también para .Net con el nombre de NHibernate)
JBoss	JBoss es un servidor de aplicaciones J2EE de código abierto implementado en Java puro. Al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo para el que esté disponible Java. Es una división de la empresa Red Hat, dedicada al desarrollo de middleware Java.
Eclipse	Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores.
WebSphere	WebSphere es una familia de productos de software propietario de IBM, aunque el término se refiere de manera popular a uno de sus productos específicos: WebSphere Application Server (WAS).
DB2	DB2 es un motor de base de datos relacional que integra XML de manera nativa, lo que IBM ha llamado pureXML, que permite almacenar documentos completos dentro del tipo de datos xml para realizar

operaciones y búsquedas de manera jerárquica dentro de éste, e integrarlo con búsquedas relacionales.

Bibliografía

1.- **CRAIG L**., UML y patrones., 2ª edición., Madrid – España., Prentice Hall., 2003., Pp. 40 - 120.

2.- **MARTIN SIERRA A**., AJAX en J2EE., México – México., Editorial Alfaomega Grupo Editor S.A de CV., 2008., Pp 129 – 150.

3.- **MARTIN SIERRA A.**, Programador JAVA 2., Editorial Alfaomega Grupo Editor S.A de CV. México, P. 234.

4.- **API, REFERENCIA JSF**

http://java.sun.com/javaee/javaserverfaces/reference/api/

2010/08/14

5.- **COMPARACIÓN STRUTS Y JSF**

websphere.sys-con.com/read/46516.htm?CFID=61124&CFTOKEN

2011/08/20

6.- **COMPARATIVA DE FRAMEWORKS WEB**

http://www.javahispano.org/storage/contenidos/frameworks_web.pdf

2011/08/30

7.- **FRAMEWORKS RIA BASADOS EN AJAX**.

http://java.sun.com/javaee/javaserverfaces/reference/api/

2010/08/12

8.- **INGENIERÍA INFORMÁTICA EMPRESARIAL**

http://s3.amazonaws.com/ppt-download/presentaciongwt-014-phpapp02.pdf

2011/09/15

9.- **INTRODUCCIÓN A RICHFACES**

adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=richFacesJsfIntro

2011/08/24

10.- **JAVABEANS TRAIL**

http://download.oracle.com/javase/tutorial/javabeans/index.html

2011-07-25

11.- **JSF, IMPLEMENTACIÓN CASO PRÁCTICO VIDEOCONFERENCIAS**

http://web.uvic.ca/~rafaelr/tesis1.2.pdf

2011/08/21

12.- **MÉTODO CIENTIFICO**

http://www.hondurassilvestre.com/Reportajes/MetodoCientifico.htm

2010/08/12

13.- **METODOS DE INVESTIGACION**

http://unam.edu.es/pub/inetinfo/12843

2010/08/14

14.- **PRACTICAL RICHFACES**

4shared\practical-richfaces.9781430210559.44872.pdf

2011-07-05

15.- **RICHFACES API**

http://boss.org/richfaces

2011/08/24

16.- **RICHFACES DEVELOPER GUIDE**

http://docs.jboss.org/richfaces/latest_3_3_X/en/devguide/html_single/

2011/08/24

17.- **RUP**

http://www.slideshare.net/msch/utilizando-metodologia-rup-parte1

2011/07/20

18.- **STRUTS Y JAVASERVER FACES, CARA A CARA**

http://www.ing.unp.edu.ar/wicc2007/trabajos/ISBD/109.pdf

2011/08/21

19.- **UTILIZANDO JSF**

exes.es/TemariosExes/Programacionydesarrollo/JAVA20SERVER20FACES.pdf

2011/08/20

Anexos

Índice de anexos:

Anexo 1: Obtención de valores para comprobación de hipótesis.
Anexo 2: Manual de Usuario
1. Introducción
1.1. Propósito
1.2. Alcance
2. Accediendo al sistema
3. Funcionamiento general de la solución web
1.2.1. Requerimientos
1.2.2. Interfaz
1.2.3. Navegabilidad
1.2.4. Mensajes informativos:
1.2.5. Ventanas o paneles modales
1.2.6. Impresión
4. Administrando al sistema
1.1. Sub-Módulo de Gestión
1.1.1. Gestión de parámetros
1.1.2. Gestión de reglas
1.1.3. Gestión de productos
1.1.4. Gestión de fórmulas
1.1.5. Gestión de áreas de mercadeo
1.2. Sub-Módulo Asistente de cálculo
Anexo 3: Fórmulas de cálculo
1.1. Requisitos:
1.2. Proceso de cálculo:
Servicio Web para seguridades, Configuración Was, configuración Eclipse, entorno war.
- Configuración previa de eclipse
- Pasos para levantar el proyecto
- Creación de la clase para el Servicio Web:
- Creación del Servicio Web:
- Para probar el servicio web:
- Para subir el WS al WAS:

Para desactivar la carga de las librerías internas JAX-WS del WAS.
Para agregar los proyectos o librerías de las cuales depende nuestro WS:

Anexo 1: Obtención de valores para comprobación de hipótesis.

Introducción

La recolección de datos como tiempo de respuesta del servidor frente a los frameworks comparados en este trabajo: `Struts` y JSF, fue desarrollada bajo parámetros de observación comunes, es decir, los mismos procesos fuer

Objetivo

Obtener los indicadores de tiempo de respuesta, usando tecnología Struts y JSF.

Descripción del entorno

El entorno de pruebas es el siguiente:

- Servidor:

  Servidor de aplicaciones WAS 6.0. IBM iSeries, versión 5,4.

  Procesador: Power 6, 4 núcleos.

  Memoria: 24 Gb.

  DB2: UDB for iSeries, v 8.x.x.

  Entorno de red bajo LAN/1000 Mbps.

- Cliente:

  PC compatible x86.

  Windows XP_sp3, 2 Gb Ram. Core 2 Duo 2100 Mhz.

  Navegador Web: Mozilla Firefox 8.0

Herramientas a utilizar

Toda la recolección de información se realizará con software agente que se ejecutan en el lado del cliente, debido a que no tenemos acceso al servidor, además, al ser el servidor una plataforma cerrada, no se tiene conocimiento de herramientas propias de benchmark o performance que no sean parte de IBM o del Sistema Operativo de iSeries.

Webserver Stress Tool

Software especializado en la simulación de peticiones HTTP, generando hasta 10000 usuarios simultáneos. Se lo usa para evaluar el desempeño del servidor bajo condiciones normales y de carga excesiva, para asegurarse de la disponibilidad y velocidad de los servicios que el servidor de aplicaciones mantiene.

http://www.paessler.com/webstress

upTime

Software que monitorea la disponibilidad, capacidad y performance de un datacenter.

http://www.uptimesoftware.com/

Recolección de datos

Se instaló la versión trial de `uptime`, con total funcionalidad por 30 días.

- Se configuró las opciones estándar para monitoreo de aplicaciones: CPU Usage, Free Memory, Network Ussage, Estadísticas de disco, Sistema de archivos.

- Mediate el uso del aplicativo WebStress Tools, se configuró una prueba con los siguientes parámetros:

Número de usuarios: 200.

Tiempo de prueba: 10 minutos.

Procesos (en este caso páginas jsf y Struts calificadas):

categoriaProductoABM.jsf

ingresoFormula.jsf

asistenteCalculo.jsf (Subir fichero, efectuar proceso de cálculo)

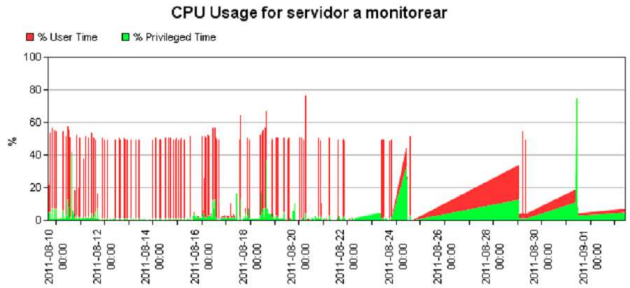
index.jsf (login)

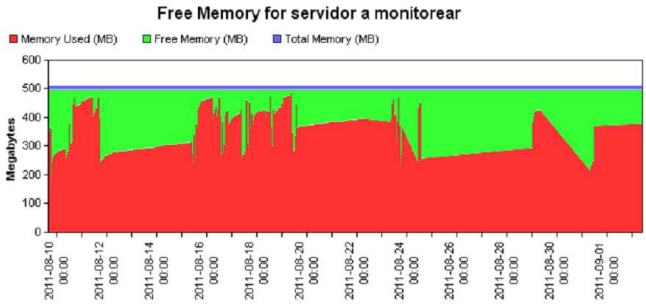
index.jsf (menú principal)

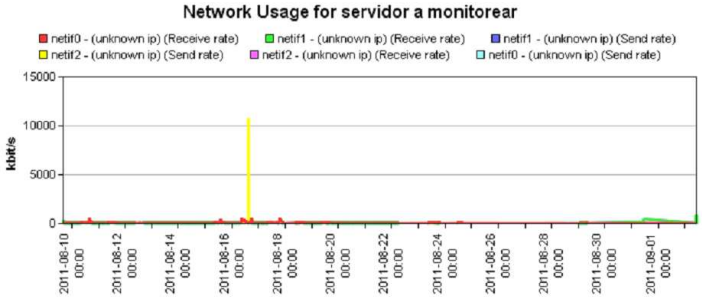
- Se efectuaron las pruebas por un periodo de 5 días, para cada página o procedimiento JSF, se recolectaron los tiempos de respuesta, se promedió el total de días, y se generó la siguiente tabla:

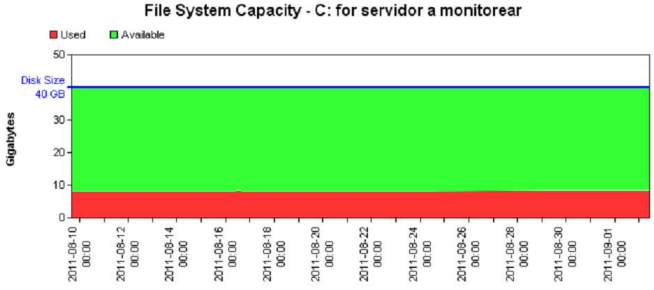
MEDICIÓN	STRUTS	JSF
Carga de información desde archivo externo:	11	7
Validación, conversión, estandarización de nombres y medidas:	11	7
Cálculo de precios:	5	8
Generación de estructura de precios PCO.	6	2
Almacenamiento de estructura de precios	14	5
Generación de reporte pdf de estructura de precios.	7	7

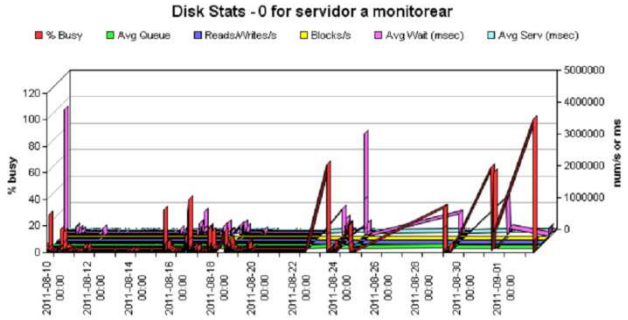
Salida del agente upTime:










Disk Stats - 0 for servidor a monitorear
% Busy
Avg Queue
Reads/Writes/s
Blocks/s
Avg Wait (msec)
Avg Serv (msec)
% busy
num/s or ms
0
20
40
60
80
100
120
0
1000000
2000000
3000000
4000000
5000000
2011-08-10 00:00
2011-08-12 00:00
2011-08-14 00:00
2011-08-16 00:00
2011-08-18 00:00
2011-08-20 00:00
2011-08-22 00:00
2011-08-24 00:00
2011-08-26 00:00
2011-08-28 00:00
2011-08-30 00:00
2011-09-01 00:00

Anexo 2: Manual de Usuario

Contenidos:

Manual de Usuario

1. Introducción

1.1. Propósito

El presente documento sirve de manual al usuario como iniciarse en el manejo de los módulos de autenticación, aseguramiento de precios a nivel de terminal, e integración de precios con la comercializadora.

1.2. Alcance

Este manual indica al usuario como ingresar a un aplicativo web, gestionar parámetros de cálculo para el asistente guiado, y ejecutar asistente de aseguramiento e integración de precios.

2. Accediendo al sistema

El acceso se lo hace desde la página de inicio del Portal Web. En la sección Servicios a Funcionarios.

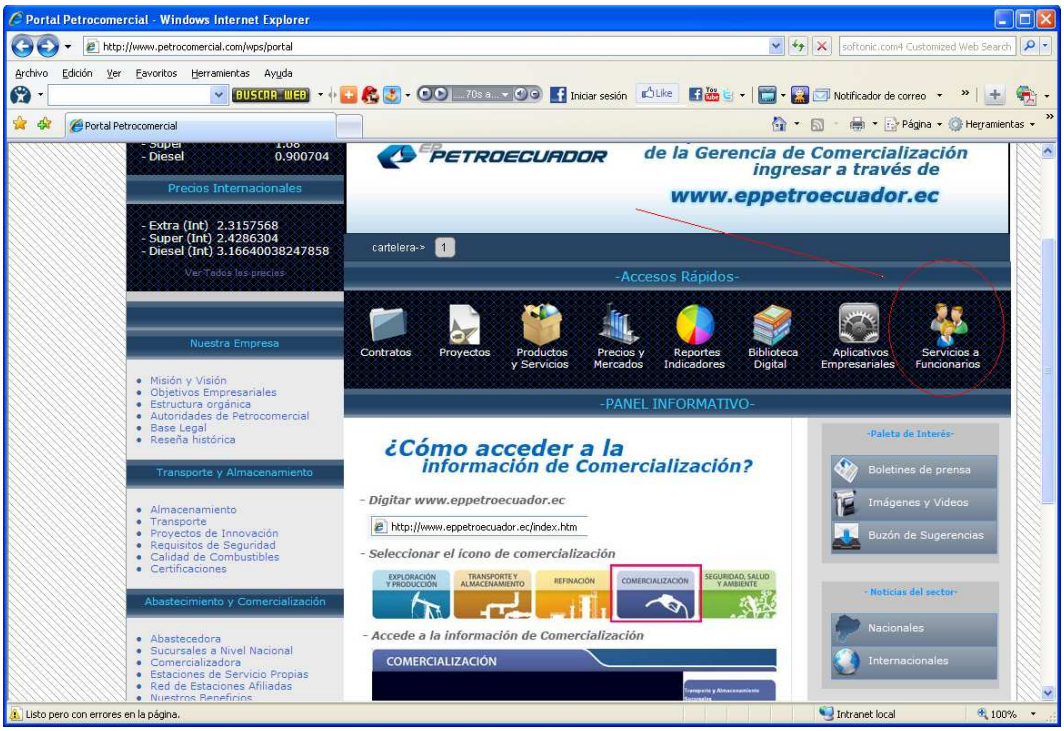


Ilustración I: Acceso al sistema

O digitando directamente la dirección del aplicativo:

http://7.103.56.157:8080/AseguramientoDePreciosWeb/paginas/index.jsf

Nota: la dirección del servidor puede variar, pues el utilitario es interno y no se tiene salida de la intranet.

Proceso de log-in:

- Ingresamos: Nombre de usuario, contraseña, seleccionamos aplicativo.
- Clic en `Validarse`…
- Ahora, seleccionamos el `rol` del usuario, clic en `Ingresar`.

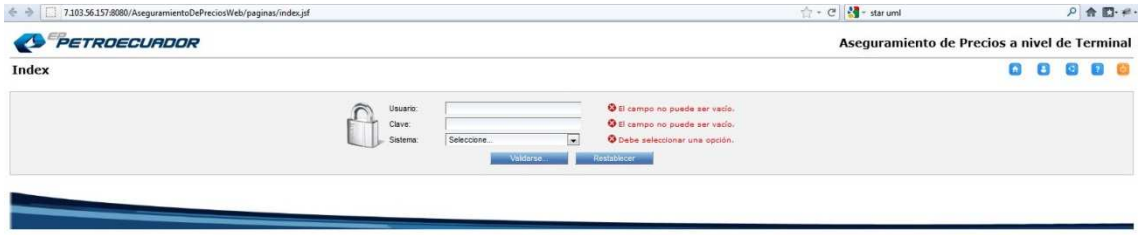


Ilustración II: Datos de acceso aplicativo web

Finalmente, si los datos son validados correctamente, se presenta el menú principal del aplicativo seleccionado, en este caso: aseguramiento de precios:

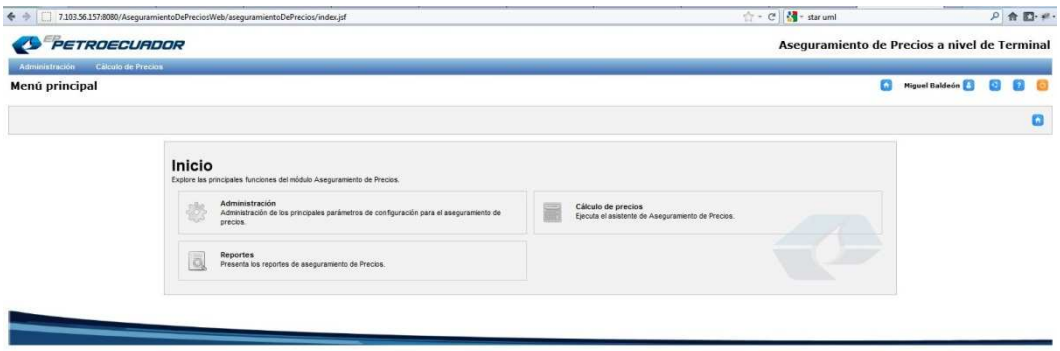


Ilustración III: Datos de acceso aplicativo web

3. Funcionamiento general de la solución web

A continuación se detallan aspectos básicos del funcionamiento de la aplicación web.

1.2.1. Requerimientos

- Cualquier navegador web actual, con soporte para `Javascript` y `cookies`, que deberán estar habilitadas.

- Se recomienda `Mozilla Firefox 5` o superiores , `Chrome 9` o superior.

- Internet Explorer versión 8 y anteriores no están soportados.

1.2.2. Interfaz

La interfaz de la solución está formada por elementos estándar en una página web: botones, botones con íconos, hipervínculos, etc.

A continuación, se muestra una pantalla genérica con sus principales opciones de menú y accesibilidad:

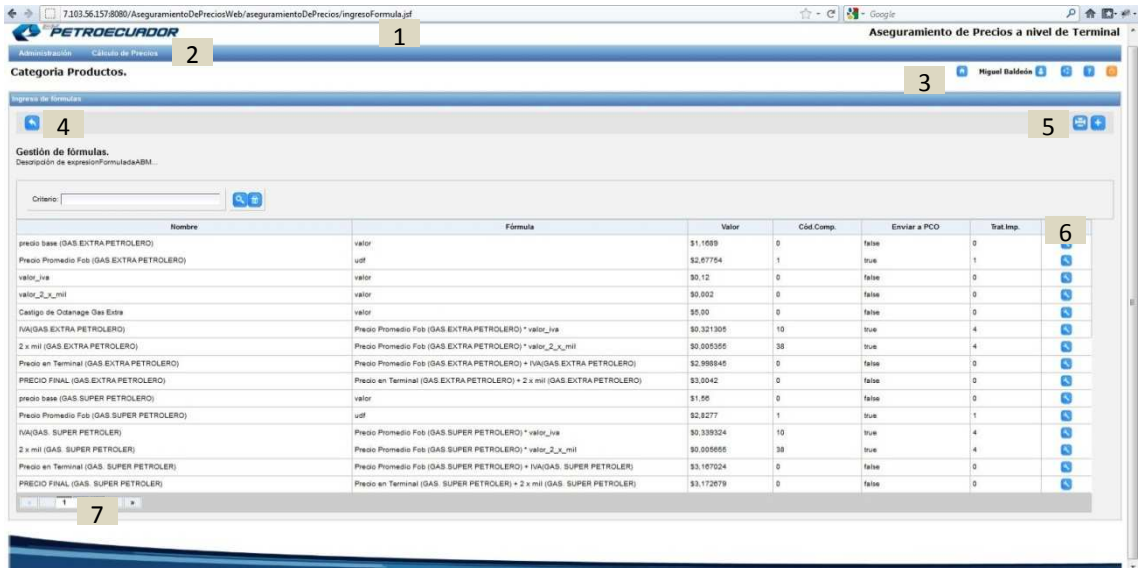


Ilustración IV: Datos de acceso aplicativo web

Donde:

1. Barra de títulos, logos, título principal
2. Barra de menús: contiene acceso a las principales opciones del sistema.
3. Barra de herramientas: botones generales de aplicativo, cada aplicativo presentará los mismos botones de aplicación, en esta barra es donde se encuentra el botón “Salir” del sistema.
4. Botones de navegación internos del aplicativo.
5. Botones de comandos, internos del aplicativo: aquí tenemos botones de opción propios del aplicativo: Impresión, búsquedas, agregar, eliminar, etc.
6. Grid, malla, o tabla con los datos de la vista actual.
7. Botones de paginación.

1.2.3. Navegabilidad

Para navegar dentro del aplicativo, podemos usar los botones que representan cada opción, por ejemplo: clic en el botón  para guardar los cambios en la vista actual; clic sobre el botón  para agregar un ítem en la vista actual; clic sobre  para eliminar un item, generalmente se encuentran en los grid o tabla de datos;  para modificar un item, generalmetne se encuentran en los grid o tabla de datos.

Para regresarse entre pantallas, se recomienda no usar el botón de Retroceder del navegador web, ya que JSF no actualiza los componentes del sistema que

están por detrás de la página web, en su lugar, se recomienda usar el los botones de navegación que aparecen en todos los aplicativos:

y el botón 'Retroceder' presente en las pantallas que admiten esta operación:  .

1.2.4. Mensajes informativos:

La salida de los mensajes informativos, o mensajes de error se encuentra de la siguiente manera: En la parte superior de la vista actual, debajo de la barra de logotipo y título.

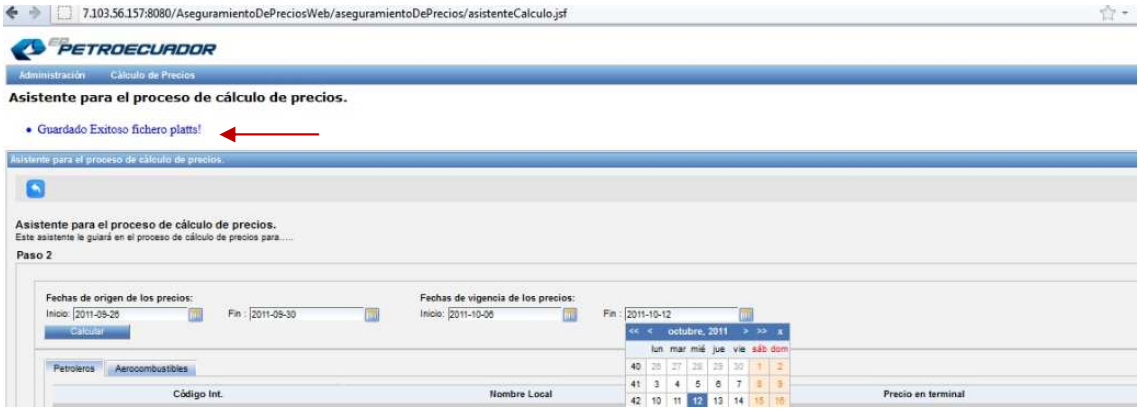


Ilustración V: Mensajes de información

Ilustración VI: Mensajes de error

Ilustración VII: Mensajes de error a nivel de formulario

1.2.5. Ventanas o paneles modales

El aplicativo cuenta con ventanas o paneles modales, que consisten en un panel que sobresale de la página web y que bloquea las opciones que se encuentra detrás de este panel. Este panel permite la entrada de información extra, que es recogida generalmente por el formulario o página web que está detrás del panel modal.

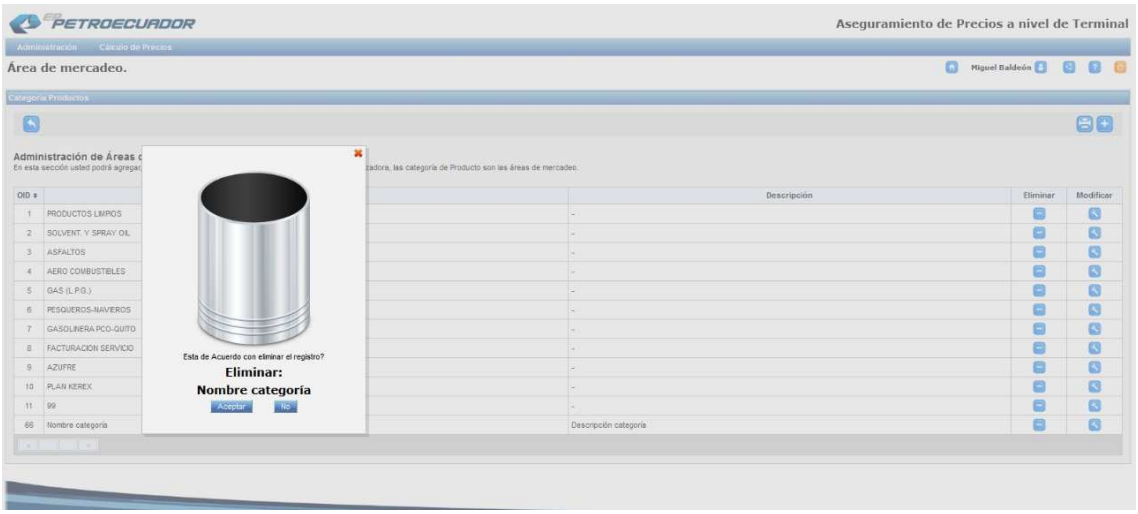


Ilustración VIII: Panel modal eliminar

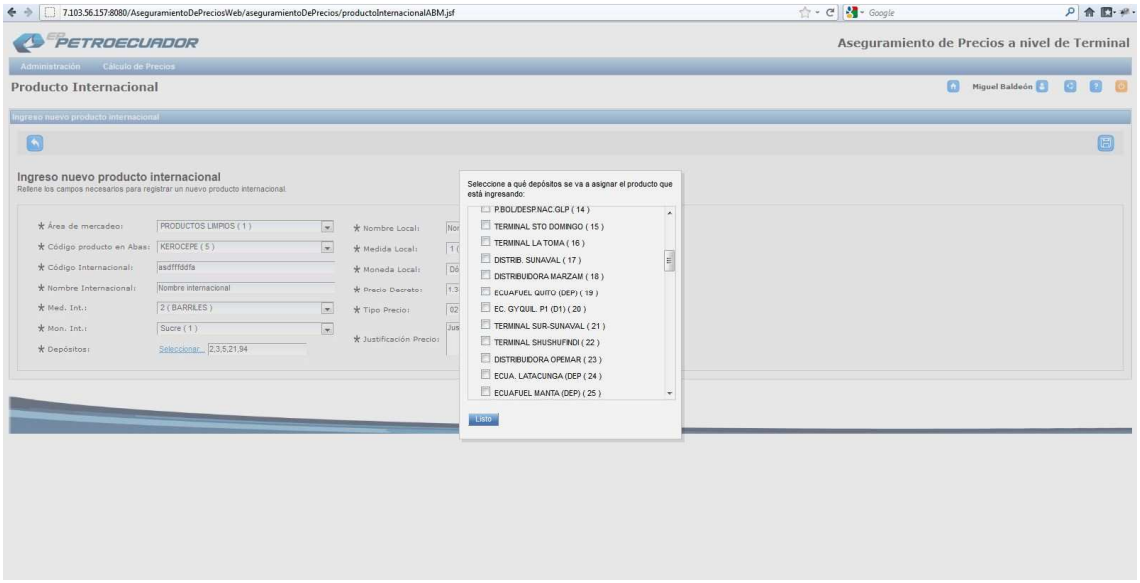


Ilustración IX: Panel modal de ingreso información extra.

1.2.6. Impresión

El aplicativo cuenta con ventanas o paneles modales, que consisten en un panel.

4. Administrando al sistema

En esta sección indicamos los distintos menús que tiene el módulo de aseguramiento de precios a nivel de terminal.

Este es el menú principal del sistema:

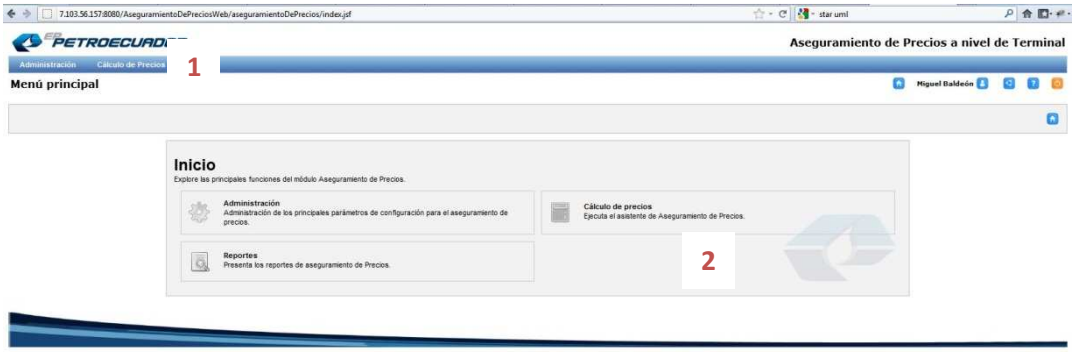


Ilustración X: Menú principal

Podemos acceder al sistema por **(1)**: Menús de aplicación; **(2)** Iconos de aplicación.

El resumen del sitio puede resumirse de la siguiente manera:

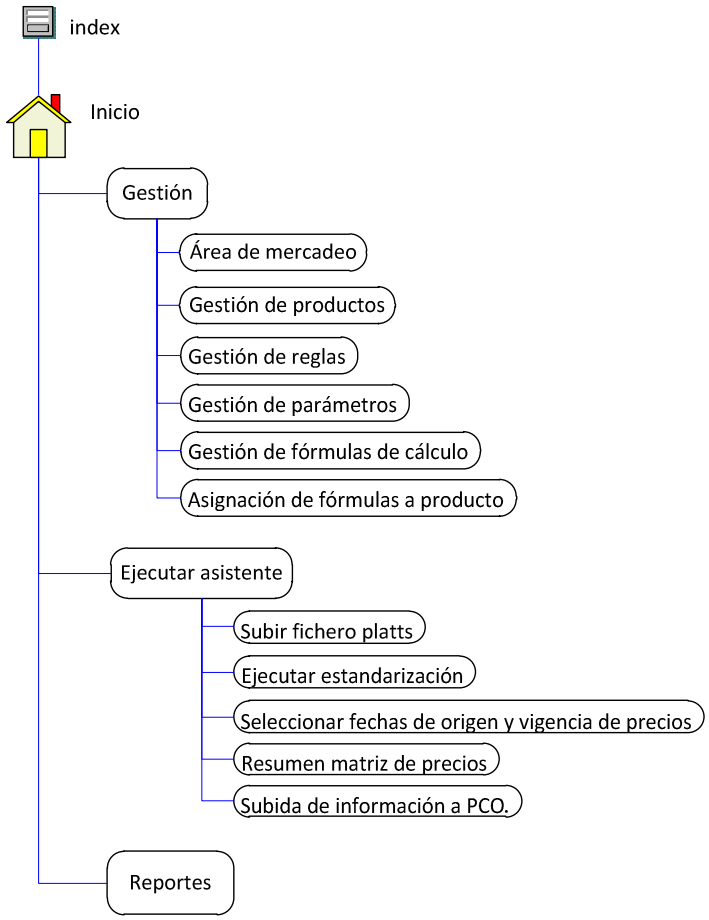


Ilustración XI: Datos de acceso aplicativo web

El módulo está divido en 2 sub partes: `Gestión` y `Asistente de cálculo`:

1.1. Sub-Módulo de Gestión

En esta sección se encuentra la carga, configuración, o gestión de las variables involucradas en el proceso de cálculo, como son: Área de mercadeo, Gestión de productos, Gestión de reglas, Gestión de parámetros, Gestión de fórmulas de cálculo, Asignación fórmula a producto.

1.1.1. Gestión de parámetros

Los parámetros del sistema son valores numéricos que son usados en la evaluación de la fórmula para un producto determinado.

Son parámetros: número de decimales, número de enteros, etc.

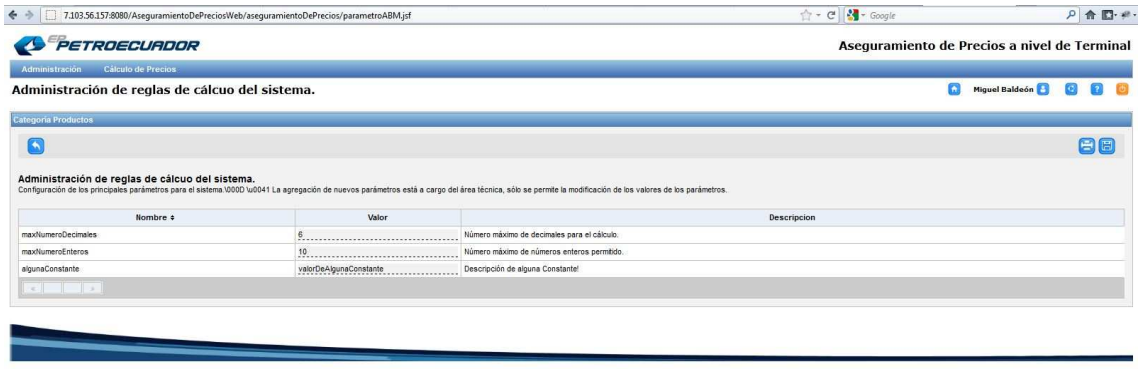


Ilustración XII: Vista modo consulta Gestión de parámetros.

Operaciones:

Solo puede modificar el valor de un parámetro, dando doble clic sobre el valor del parámetro e ingresando el nuevo valor, para luego presionar el botón Guardar: 

1.1.2. Gestión de reglas

Las reglas son constantes numéricas que se usarán para el cálculo o evaluación de las fórmulas para el cálculo de precios.

Es un sub módulo con soporte ABM (Alta, baja, modificación) para las reglas involucradas en el cálculo.

Las reglas pueden ser: valor del iva, valor de impuestos de octanaje según tipos de derivados del petróleo, impuesto 2 x mil, impuestos o castigos en general.

Al ingresar a esta sección desde el menú principal se observa la pantalla de consulta por defecto.

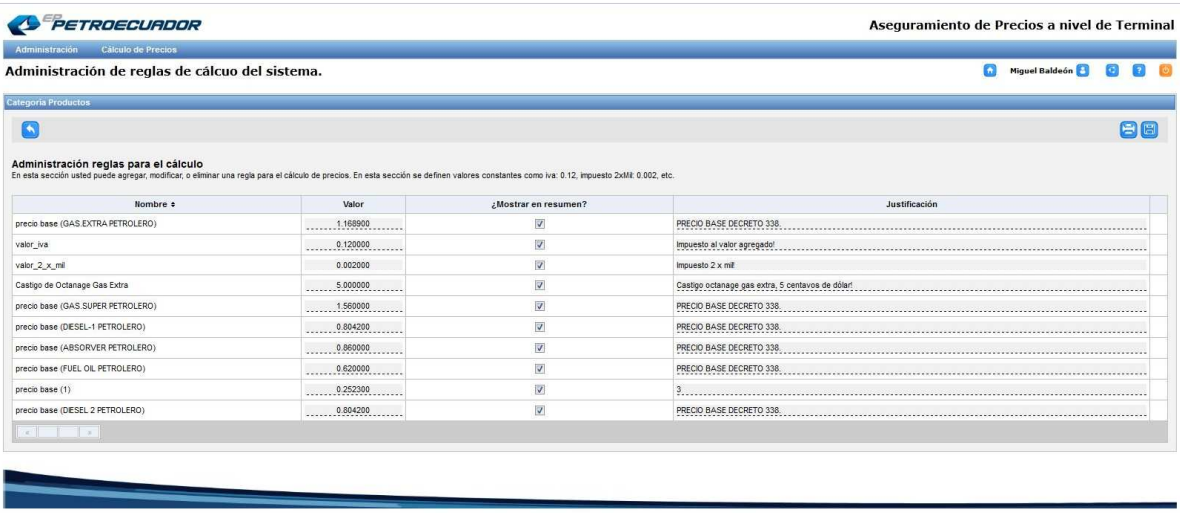


Ilustración XIII: Vista modo consulta Gestión de reglas.

Operaciones:

Para cambiar el valor de una regla:

- Doble clic sobre el valor de la regla e inserte el nuevo valor.

- Presione el botón guardar 

1.1.3. Gestión de productos

Los productos de la comercializadora son los objetos o entidades a los que se les aplicará una fórmula para obtener el precio de comercialización a nivel de terminal.

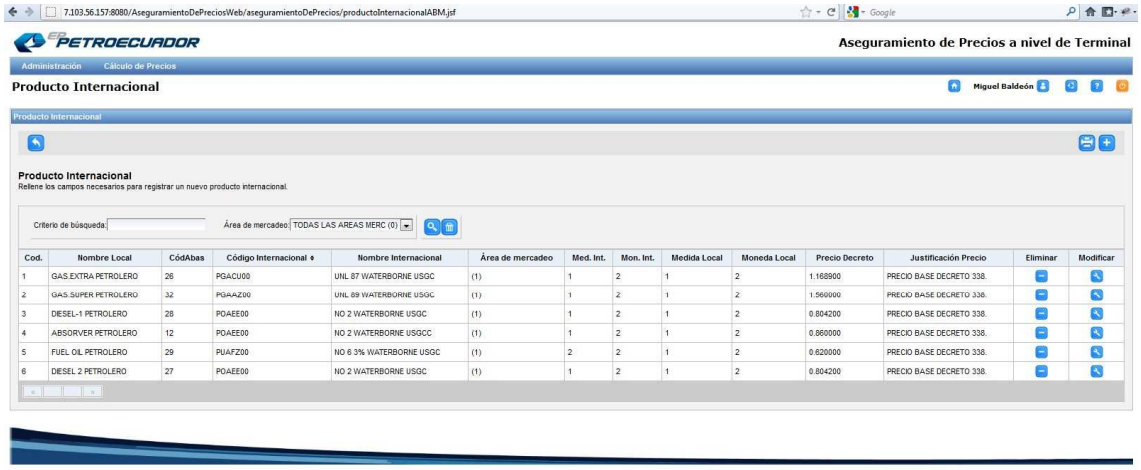
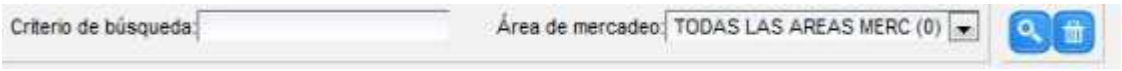


Ilustración XIV: Vista de consulta gestión de productos

Operaciones:



`Criterio de búsqueda`: criterio que representará la búsqueda según el `Área de mercadeo seleccionada`.

Agregar: Agrega un producto internacional:

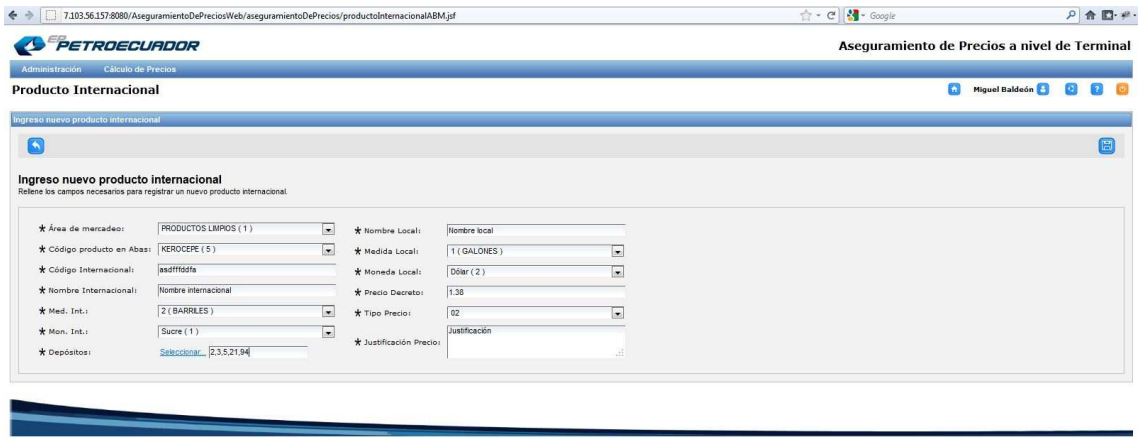


Ilustración XV: Agrega un producto internacional

Imprimir: permite la impresión de la lista de productos.

1.1.4. Gestión de fórmulas

En esta sección se gestionan las fórmulas de cálculo de precio para ser asignadas a un producto.

La pantalla inicial es la siguiente:

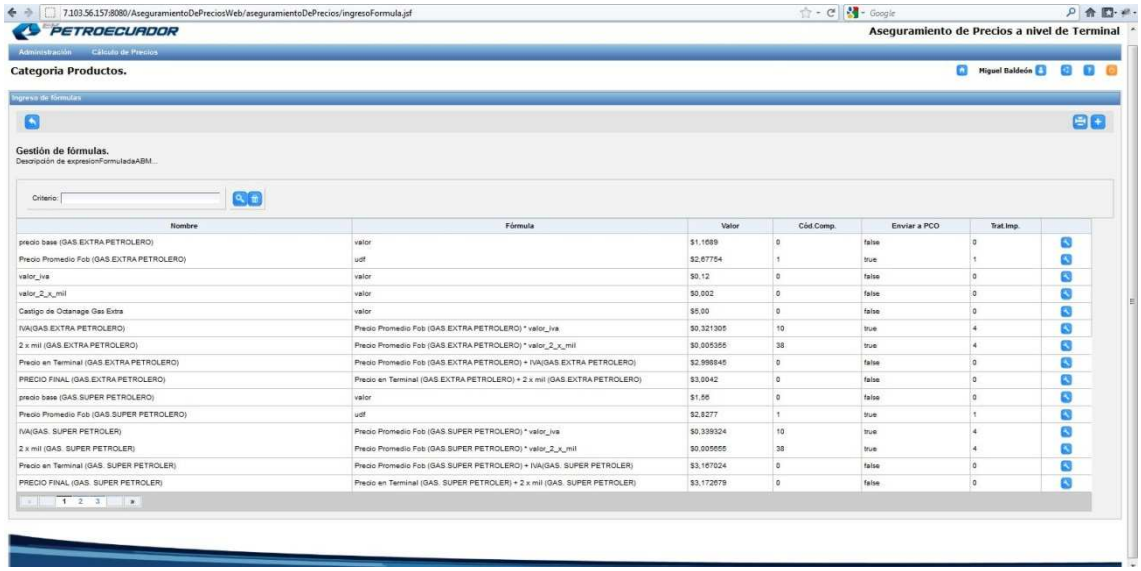


Ilustración XVI: Modo consulta, edición de fórmulas

Operaciones:

Búsqueda:

Criterio de búsqueda, cualquier expresión alfanumérica, con los botones de iniciar búsqueda  o limpiar cuadro de texto: 

Agregar / Modificar:

Desde cada fila mostrada en la tabla, usted puede editar una fórmula de precio. O puede agregar una nueva fórmula de precio:

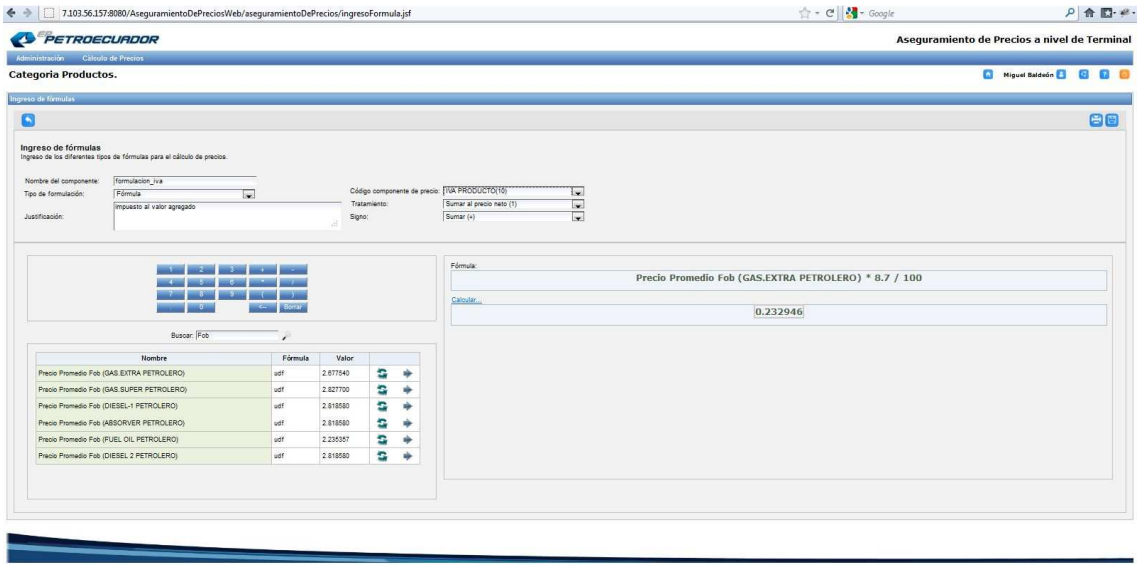


Ilustración XVII: Modo edición, fórmulas de precio

Para consultar cómo se conforma una fórmula de precio, ver Anexo 3.

1.1.5. Gestión de áreas de mercadeo

En esta sección se administra las áreas de mercadeo en las que se mueven los distintos productos.

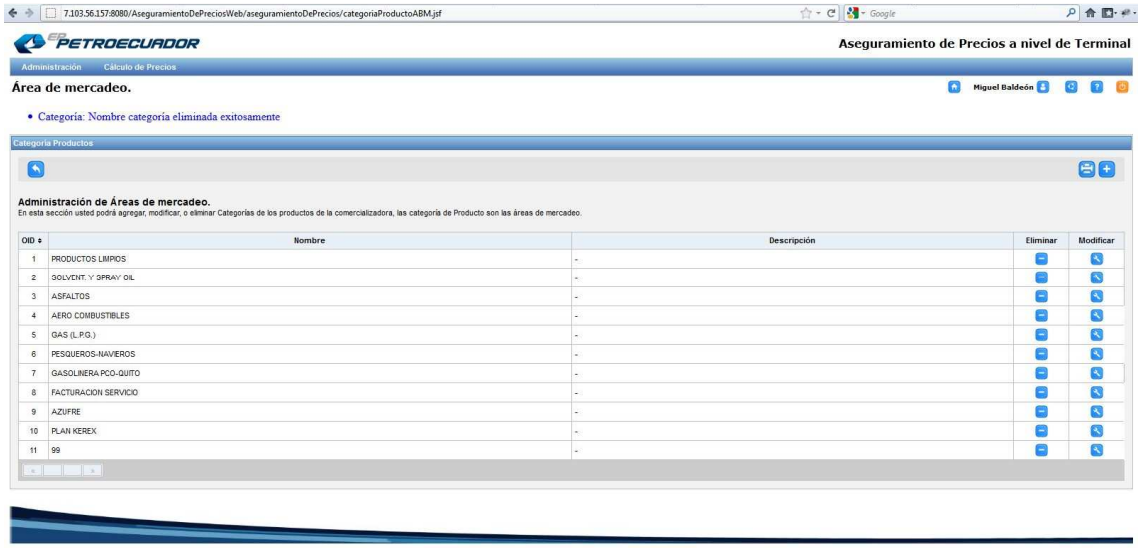


Ilustración XVIII: Modo consulta: administración áreas de mercadeo

Operaciones:

Agregar

Agrega una nueva área de mercadeo.

Ilustración XIX: Modo consulta: administración áreas de mercadeo

1.2. Sub-Módulo Asistente de cálculo

Esta sección trata acerca del sub módulo Asistente de cálculo, el cuál evalúa las funciones matemáticas asignadas a un producto, para obtener el valor del precio de venta a nivel de terminal.

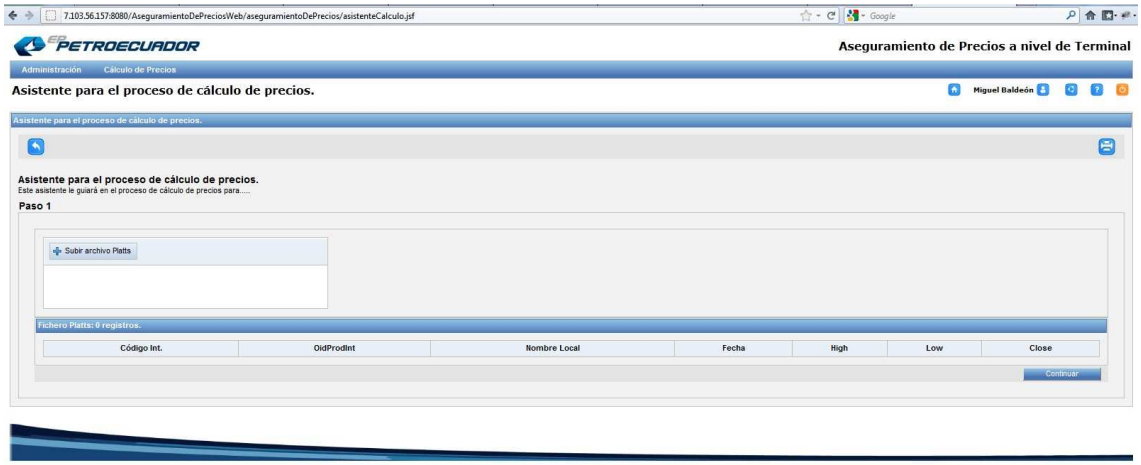


Ilustración XX: Asistente de cálculo, paso 1

Como podemos ver en la Ilustración XX, en este paso, se procede a la carga del fichero externo en formato Microsoft Excel, el cual contendrá el resumen de los valores de los indicadores de los precios internacionales de los productos derivados del petróleo y energía.

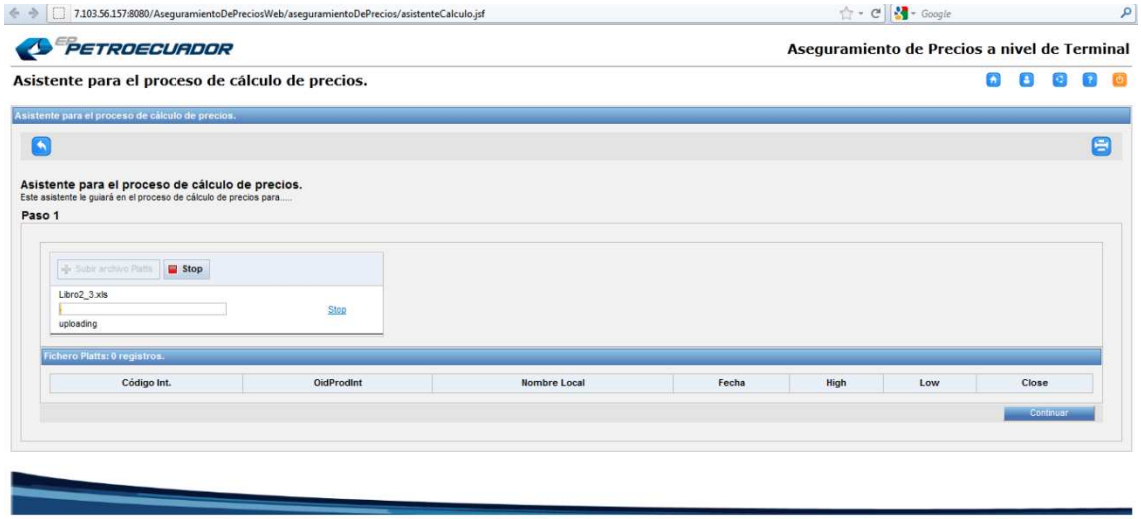


Ilustración XXI: Subida del fichero platts al sistema

Para acceder al paso 2, se debe imprimir la vista actual: 

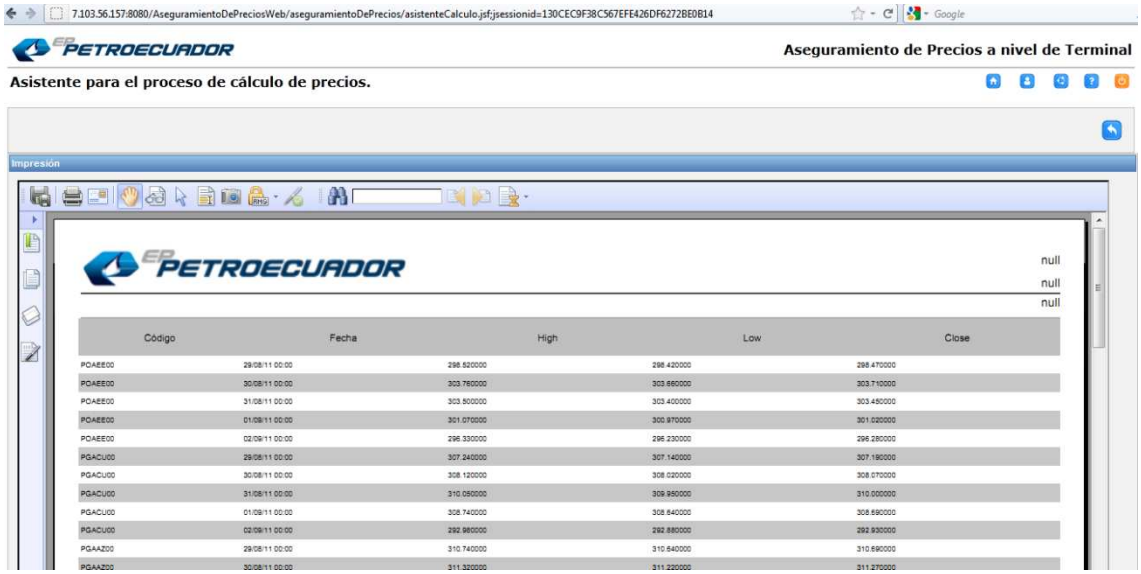


Ilustración XXII: Impresión paso 1

Paso 2:

Ingresamos fechas de origen de precios y vigencia de los precios que se van a calcular, luego, procedemos a dar clic en el botón Calcular: , obtenemos la siguiente vista:

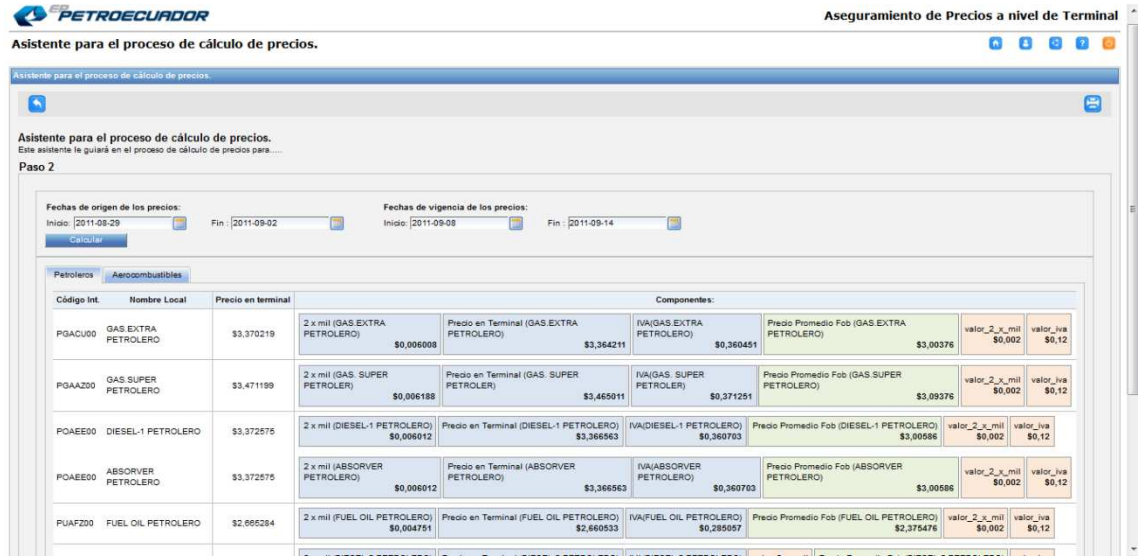


Ilustración XXIII: Impresión paso 2

Ilustración XXIV: Impresión paso 2, continuación

En el paso 3, luego de presionar Continuar: Continuar, el asistente mostrará la matriz de precios, que deberá ser impresa para poder continar al paso 4.

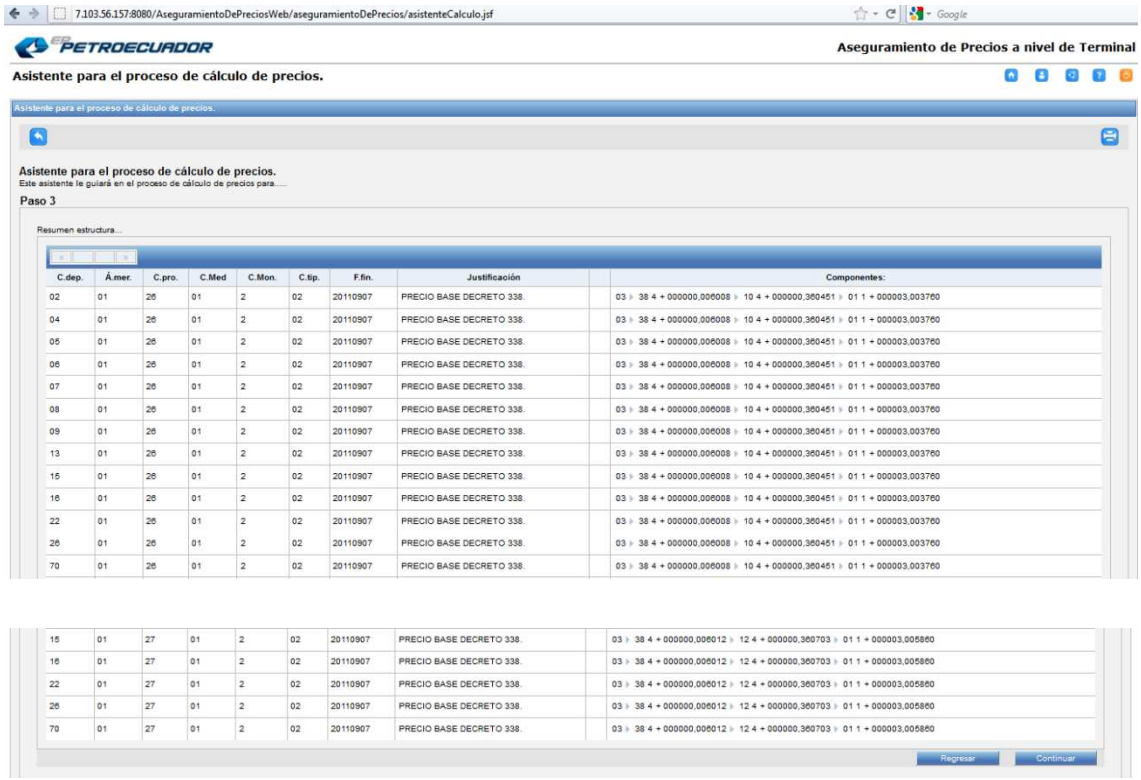
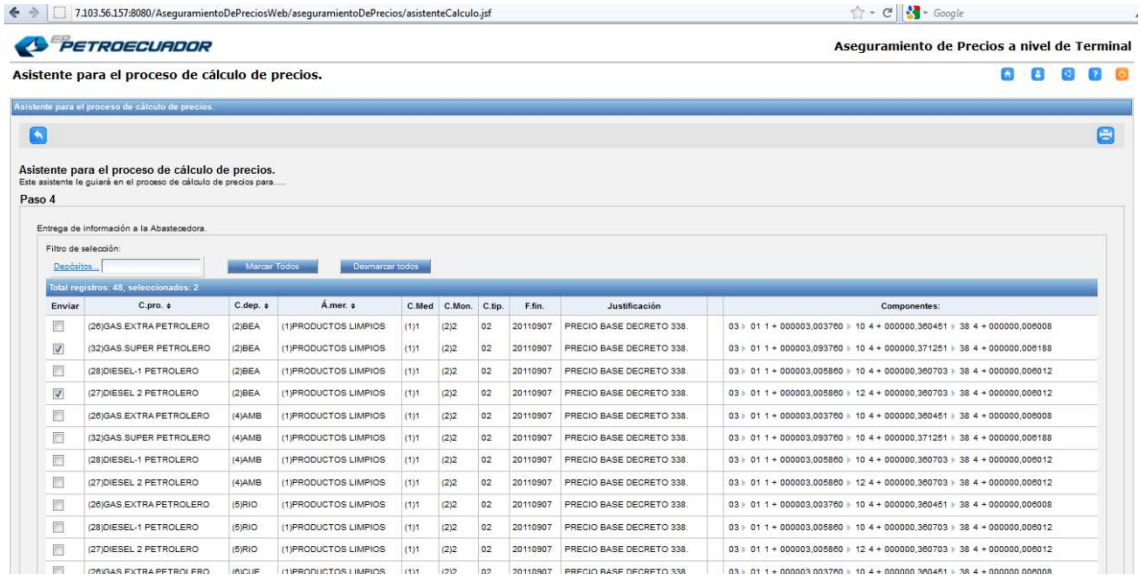


Ilustración XXV: Paso2, matriz de precios

Al presionar Continuar, llegamos al paso 4, donde se presenta la pantalla final del asistente, en la que elegimos los productos que deseamos actualizar en el sistema de la Abastecedora, a nivel de terminal, los productos elegidos son enviados al sistema PCO, como una matriz de precios, que se replican hacia todos los puntos de control de la abastecedora.



Para finalizar este asistente, debemos aceptar las condiciones y se habilitará el botón Finalizar:

El sistema informará cuántos productos han sido ingresados y regresará a la pantalla del menú inicial.

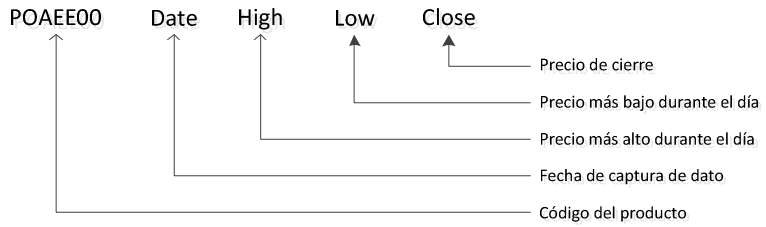
Anexo 3: Fórmulas de cálculo

Proceso general para formar un precio:

Este grupo de productos se basa en los precios de los últimos 5 días laborables de la semana anterior a la actual, de Lunes a Viernes, obtenidos desde Platts on the web.

1.1. Requisitos:

Obtener el fichero Platts en formato Microsoft Excel, con el siguiente encabezado:



El precio High, Low, Close está dando en centavos por galón.

El grupo de productos petroleros es el siguiente:

Código internacional	Nombre internacional	Nombre local
POAEE00	NO 2 WATERBORNE USGC	Diesel
PGACU00	UNL 87 WATERBORNE USGC	Diesel 1
PGAIX00	SUP 93 WATERBORNE USGC	Diesel 2
PJABM00	JET KERO 54 WATERBORNE USGC	
PGAAZ00	UNL 89 WATERBORNE USGC	Gasolina Super
PUAFZ00	NO 6 3% WATERBORNE USGC	Fuel Oil

1.2. Proceso de cálculo:

1. Precio promedio FOB:

```
Promedio(5 últimos días laborables) / 100
```

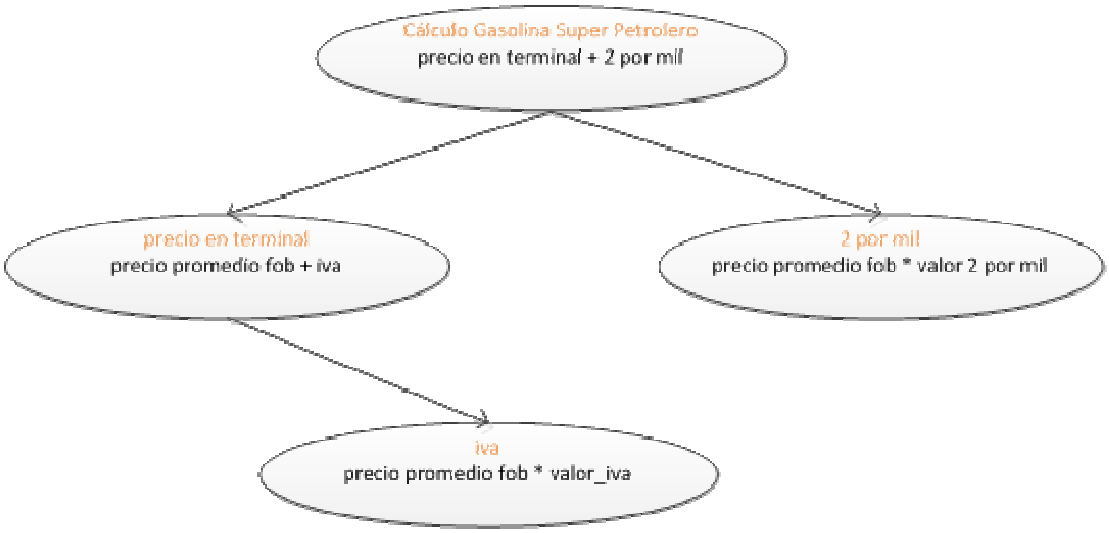
2. Precio en Terminal:

   A `precio promedio FOB` le sumamos el `IVA`:

```
Precio en terminal= Precio promedio FOB * 1.12
```

   Para aprobar este precio, debe cumplirse la siguiente condición, tomada del *Decreto Ejecutivo 338*: el precio en terminal,

Ejemplo de estructura de precios:



Nota: El proceso de cálculo de fórmulas es confidencial, propiedad de EP Petroecuador, por lo que no se puede mostrar información en detalle.

Anexo 4: Web Service de seguridades y su configuración con JSF en eclipse:

Implementación de Servicios Web bajo JAX-WS para WAS 7
Configuración Servidores IBM Was 7.0

Configuración entorno de trabajo con Eclipse

para JSF

Manual Técnico.0

Contenido:

SERVICIO WEB PARA SEGURIDADES **253**
NOTAS: ..... ¡ERROR! MARCADOR NO DEFINIDO.

- **Configuración previa de eclipse** **253**
- **Pasos para levantar el proyecto** **254**
- **Creación de la clase para el Servicio Web:** **254**
- **Creación del Servicio Web:** **255**
- **Para probar el servicio web:** **258**
- **Para subir el WS al WAS:** **258**
  Para desactivar la carga de las librerías internas JAX-WS del WAS. 259
  Para agregar los proyectos o librerías de las cuales depende nuestro WS: 260
- **Siguientes pasos:** ..... ¡Error! Marcador no definido.

Glosario:

Completar, es más técnico…

WS	Web Service.
Web Service	Es una pieza de software que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones.
WAS	Websphere Application Server, servidor de aplicaciones de altas prestaciones de IBM.
CXF	Es un framework openSource para la construcción y desarrollo de servicios web basados en el estándar de SUN JAX-WS Y JAX-RS.
Axis	Otro framework openSource para la construcción y desarrollo de servicios web basados en el estándar de SUN JAX-WS Y JAX-RS.
JAX-WS	Java Api por XML Web Services, de Sun a partir de la versión 1.5 del JDK, basado en anotaciones para la construcción de WS.

Notas tesis, módulo SeguridadesWS

Tomado de: EPP, CXF documentación,WS documentación general, IBM developerWorks Web Service y WAS7.

Servicio Web para seguridades, Configuración Was, configuración Eclipse, entorno war.

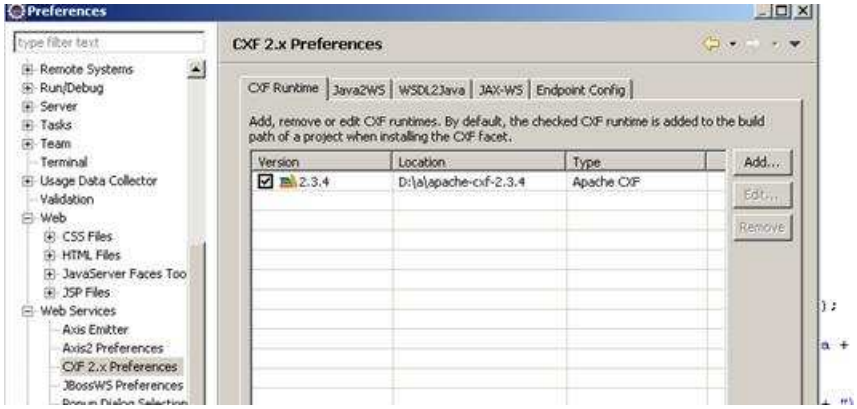
- Configuración previa de eclipse

1. Asegurarse de que exista el JDK 1.5 o superior en el entorno de desarrollo, si bien no es indispensable, ayuda al memento de compilar los servicios con Apache CXF: el Framework recomendado para hacer los servicios, implementa los estándares de JAX-WS.
2.
3. Descargar Apache CFX: cxf.apache.org/.

   - Para el WAS 7, se usó la versión: apache-cxf-2.3.4, aunque WAS ya trae su propia implementación de JAX-WS, pero su utilización se limita al entorno de desarrollo Rational, se podría hacer con Eclipse, o cualquier IDE, pero este no es el caso de este tutorial.

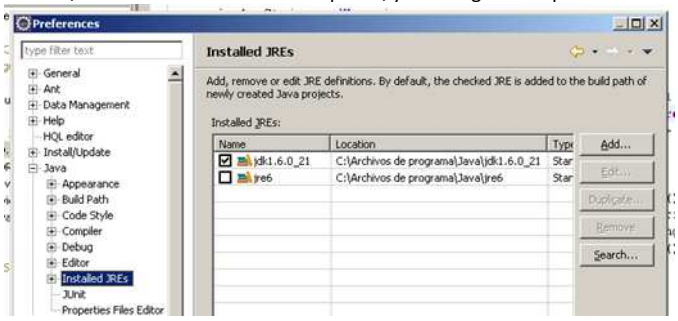
   - Al no usar la implementación JAX-WS propia de IBM WAS, se pierden algunas funcionalidades propias del WAS, que se salen del alcance de este documento, sin embargo, la especificación del framework CXF usada en este tutorial parece superar a la implementación que viaja dentro del WAS.

4. Agregar a los runtimes de WebServices en Eclipse: Windows->Preferences->WebServices->CXF 2.x preferences.



5. Asegurarse que el entorno de Eclipse use el JDK 1.5 o superior, el generador de CXF necesita de tools del JDK (al realizar este tutorial, sí compila los WS con JRE 6, pero hay errores en el trace de la generación de los WS, sin embargo funcionaron, pero para evitar esos errores se usó el JDK en vez del JRE.)

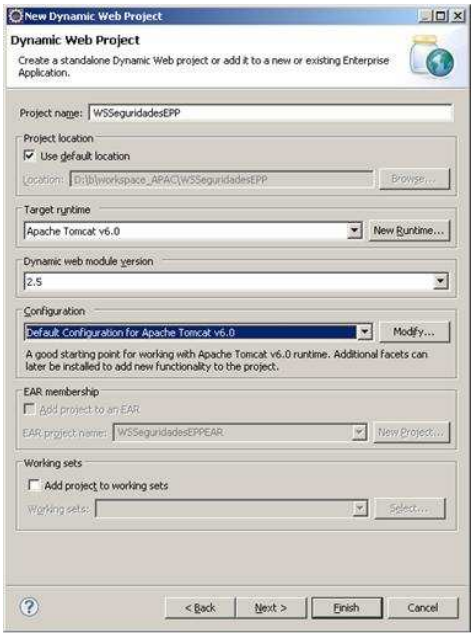
   Para configurar el entorno, instalamos el JDK 1.5 o superior, y lo configuramos por defecto en Eclipse:



- Pasos para levantar el proyecto

1. Crear un `Dynamic Web Project` en `Eclipse`.
2. El `Runtime` de la aplicación web será `Apache Tomcat 6.0.26`. (al momento de levantar este documento).
3. El JRE para el servidor apache, será un `JDK 1.5` o superior (`JDK 1.6.0_21` al momento de levantar este documento):

4. La versión del descriptor web será la `2.5`, necesaria para que funcione en WAS `7.0.0.9`, y la configuración por defecto de Apache `Tomcat v.6.0`.



- Creación de la clase para el Servicio Web:

1. Creamos una clase estándar `Java`, y creamos métodos que reciban o retornen cualquier tipo de dato, el framework CXF realizará la conversión a objetos y esquemas xml para ser transportados por cualquier protocolo web, en este caso SOAP `1.1`(Configuración estándar de CXF) a través de HTTP.

2. Para este tutorial, se agregó al `classpath` del proyecto el proyecto PEC_`Seguridades`.

3. Se hizo 3 métodos públicos para el servicio web, uno que retorna un objeto `V_Seguridades`, otro que retorna una colección de objetos `V_Seguridades`, y otra que retorna una cadena de texto:

```
public class EPPSeguridades {

        /**
         * Prueba retorno objeto V_Seguridad.
         * @return
```

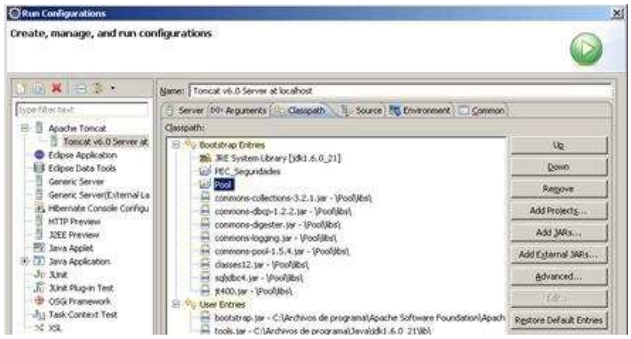
```
	 */
	public pec.seg.sacco.dto.V_Seguridad getVistaSeguridad(String parametro){
		pec.seg.sacco.dto.V_Seguridad col = null;
		System.out.println("Intentando obtener conexión!, se ha pasado el parámetro de
prueba: " + parametro);
		try {
			col = new pec.seg.sacco.dto.V_Seguridad().bbbb();
		} catch (Throwable e) {
			e.printStackTrace();
		}
		return col;
	}

public Collection<V_Seguridad> getRoles(String parametro){
		Collection<V_Seguridad> col = null;
		System.out.println("Intentando obtener conexión!, se ha pasado el parámetro de
prueba: " + parametro);
		try {
			col = new V_Seguridad().buscaTodosRoles();
		} catch (Throwable e) {
			e.printStackTrace();
		}
		return col;
	}

	public String getMensaje(String parametro){
		System.out.println("Intentando retornar el mensaje!");
		return "Este es un mensaje: " + parametro;
	}
}
```

- Creación del Servicio Web:

Notas Previas:

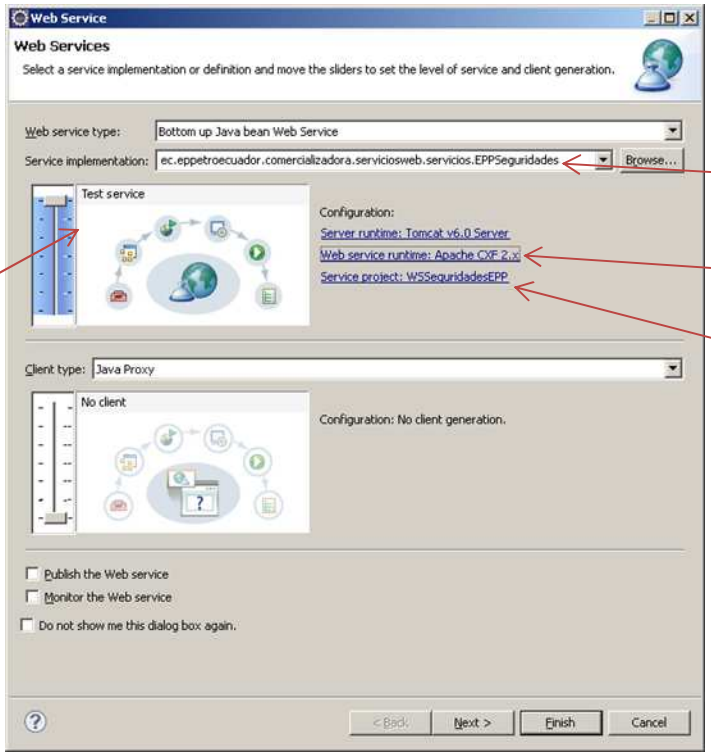
- Antes de generar el servicio, agregar a la configuración de `Ejecución` o `Debugging` los proyectos necesarios para la clase del servicio web generada en el punto anterior.

  Ejemplo:



1. Clic con el botón secundario del mouse en la clase que implementará el Servicio Web, luego clic en la opción Web Services->Create Web Service.

2. Asegurarse que los siguientes campos concuerden con los siguientes valores, y clic en `Next`:



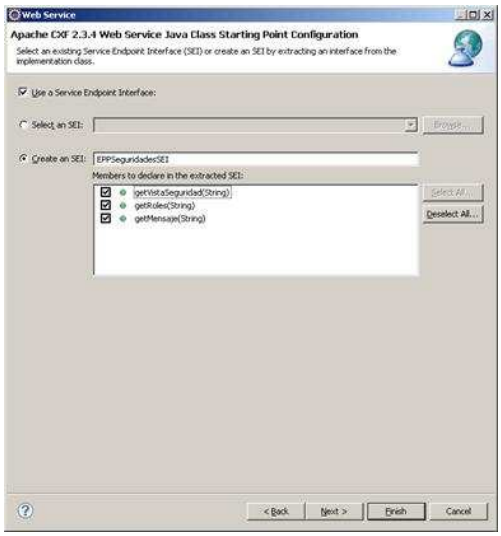
Clase que implementa el WS

Framework para el WS

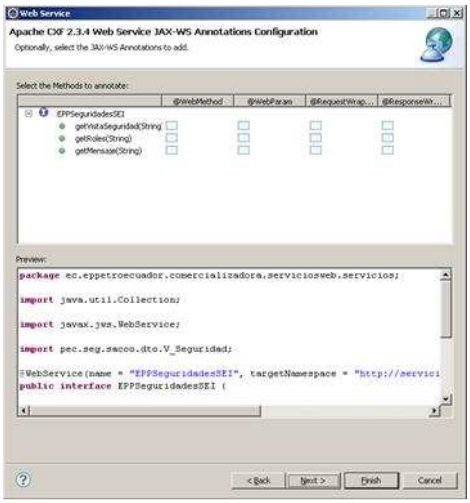
Proyecto que contendrá la

Cumple todo el ciclo para generar el

3. Escogemos los métodos que deseamos publicar en el WS, y una clase para el SEI (Service EndPoint Interface), clic en Next.



4. En esta pantalla se puede generar las anotaciones para generar el WS, dejamos los valores por defecto, clic en Next.



5. Generación del descriptor del servicio, escogemos un nombre y los valores por defecto que nos da el asistente, clic en `Next`.

6. Empieza la generación del WS, en consola se podrá ver la traza de ejecución de las herramientas del framework para crear el WS.

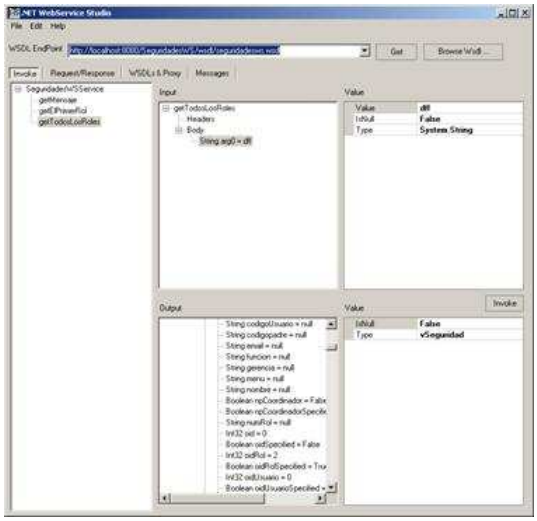
7. Clic en `Start server` para publicar el servicio en `Tomcat`, la traza de publicación del proyecto web se verá en consola.

8. En las siguientes pantallas aceptamos los valores por defecto, o hacemos clic en `Finish`.

- Para probar el servicio web:

Por alguna razón que no entiendo, no puedo consumir el WS dentro del tool de Eclipse: `WSExplorer,` que se lanza al terminar el asistente de generación del WS, por lo que se usó `.Net WebServiceStudio`.

1. Ejecutamos el cliente: `.Net WebServiceStudio (http://webservicestudio.codeplex.com/)`
2. Pegamos la ruta del fichero descriptor del `WS`, clic en `Get`.
3. Escogemos el método, le proveemos los parámetros, en caso de ser necesario, y finalmente clic en `Get`.
4. El resultado se verá en `Output`, además, el `trace` del servicio Web se verá en la consola `output` de `Eclipse`.



- Para subir el WS al WAS:

Notas:

- La versión De `WebSphere Application Server` es la `7.0.0.9`.
- A continuación se empaquetara el `WS` en una aplicación empresarial. En la documentación del WAS, se recomienda desactivar la carga de la propia implementación de `JAX-WS` y permitir la carga de las librerías `JAX-`

WS usadas en el proyecto, en este caso las librerías de la implementación Apache CXF, la más recomendada al momento de escribir este tutorial.

Para desactivar la carga de las librerías internas JAX-WS del WAS.

- En el War de nuestra aplicación, en el fichero META-INF/MANIFEST.MF (podemos modificarlo con Eclipse antes de subirlo a un EAR) agregamos el parámetro:

```
DisableIBMJAXWSEngine: true
```

- En nuestro proyecto Web, cambiamos la ruta del descriptor wsdl para que apunte a la dirección del servidor donde se va a publicar:

  Modificamos la línea:

```
<soap:address
location="http://localhost:8080/SeguridadesWS/services/SeguridadesWSPort"/>
```

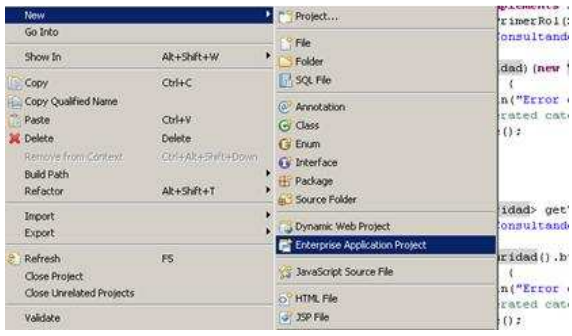
  Por:

```
<soap:address
location="http://localhost:8080/SeguridadesWS/services/SeguridadesWSPort"/>
```

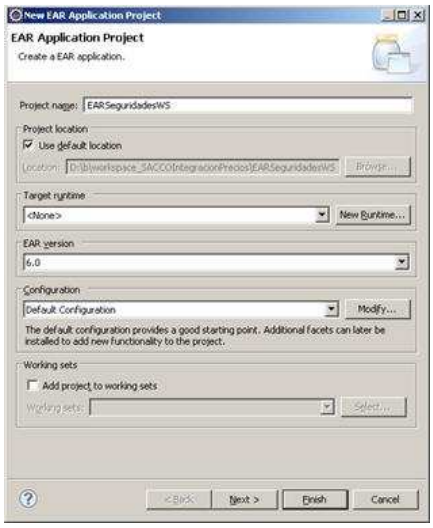
- Al momento de subir nuestra aplicación al WAS, el servidor de IBM GENERA EL WS según el descriptor de CXF, por lo que si tenemos proyectos de los cuales dependa nuestro Web Services, proyectos agregados al classpath, de nuestra aplicación, debemos agregarlos también al Deployment Assembly de nuestro EAR, ver: Para agregar los proyectos o librerías de las cuales depende nuestro WS más adelante en este documento.

Pasos:

1. Creamos una nueva aplicación empresarial, colocamos el nombre que deseemos:



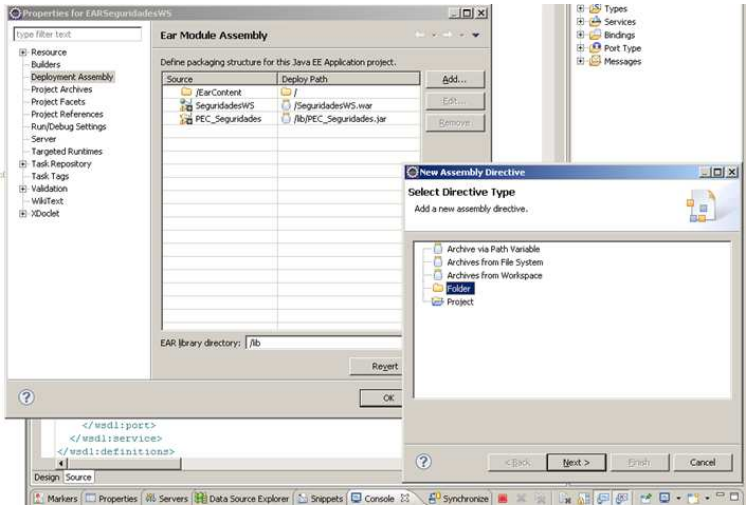
2. Nos aseguramos que la descripción del EAR sea la versión 6.0, los demás parámetros aceptamos por defecto.



3. Escogemos el proyecto que será embebido en nuestra aplicación empresarial, en este caso `SeguridadesWS`, clic en `Finish`.

Para agregar los proyectos o librerías de las cuales depende nuestro WS:

4. Clic con el botón secundario del mouse sobre el proyecto EAR y escogemos `Propiedades`.
5. En la opción `Deployment Assembly`, añadimos los proyectos o librerías necesarias, al momento de exportar, estos proyectos o librerías serán ensamblados en un `jar` válido.



6. Exportamos nuestra aplicación como `Enterprise Archive:` EAR, el tamaño del proyecto EAR encapsulado ocupará aproximadamente 25 Mb, debido a que toda la librería necesaria de CXF es embebida en la aplicación.
7. Subimos al WAS como cualquier aplicación normal, aceptando los parámetros por defecto de instalación de un EAR.