ESCUELA POLITÉCNICA DEL EJÉRCITO

DPTO. DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

GUÍA DE DESARROLLO DEL FRAMEWORK, PARA EL DISEÑO ÁGIL DE APLICACIONES WEB, EN LA EMPRESA KRUGER CORPORATION. PROTOTIPO: SISTEMA PARA EL PLAN DE CAPACITACIÓN INTERNA DE EMPLEADOS

Previo a la obtención del Título de:

INGENIERO EN SISTEMAS E INFORMÁTICA

POR: MUÑOZ ONOFA ANDRÉS ESTEBAN

SANGOLQUÍ, 24 DE ABRIL DEL 2012

ESCUELA POLITÉCNICA DEL EJÉRCITO

INGENIERÍA EN SISTEMAS

AUTORIZACIÓN

Yo, Andrés Esteban Muñoz Onofa con cédula de identidad No. 1717160210, autorizo a la biblioteca Alejandro Segovia de la Escuela Politécnica del Ejército se publique la Tesis “Guía de desarrollo del framework ZK, para el diseño ágil de aplicaciones web, en la empresa KRUGER CORPORATION. Prototipo: Sistema para el plan de capacitación interna de empleados” realizada por mi persona.

Sangolquí, 27 de Abril del 2012

Andrés Muñoz
1717160210

CERTIFICACIÓN

Certifico que el presente trabajo fue realizado en su totalidad por el Sr. MUÑOZ ONOFA ANDRÉS ESTEBAN como requerimiento parcial a la obtención del título de INGENIERO EN SISTEMAS E INFORMÁTICA

Sangolquí, Abril del 2012

_____

ING. RAMIRO DELGADO.

DEDICATORIA

A mi madre y a mis dos padres Jorge y Carlos por construir mi esencia y ser mi apoyo incondicional durante toda mi vida.

Andrés Esteban Muñoz Onofa

AGRADECIMIENTOS

A todas las personas que directa o indirectamente han colaborado con este trabajo, a toda mi familia especialmente a mi madre por estar siempre a mi lado y brindarme siempre su sabiduría y a ese ser especial que llego a mi vida para llenarla de dulzura, amor y momentos espectaculares. Gracias.

Andrés Esteban Muñoz Onofa

ÍNDICE DE CONTENIDOS

RESUMEN 1

ABSTRACT 2

CAPÍTULO 1: INTRODUCCIÓN 3

1.1 Contextualización del Problema **3**

1.1.1 Labor de la empresa: 3

1.1.2 Área específica: 3

1.2 Formulación del Problema: **3**

1.3 Delimitación Espacial **4**

1.4 Delimitación Temporal **4**

1.5 Sistematización del problema **5**

1.6 Objetivos **5**

1.6.1 Objetivo General: 5

1.6.2 Objetivos Específicos: 6

1.7 Justificación e importancia: **6**

1.8 Alcance **9**

1.9 Situación actual de la arquitectura web utilizada en la empresa **10**

1.9.1 Tecnologías en uso 10

1.9.1.1 JSF 2.X 10

1.9.1.2 Richfaces 4.x 11

1.9.1.3 Primefaces 3.x 11

1.9.1.4 Eclipse 3.6 ____ 12
1.9.1.5 JBOSS 6.0 ____ 12
1.9.2 Organización de los proyectos ____ 12
1.9.3 Los Controllers y Datamanagers ____ 13
1.9.3.1 Organización del contenido web ____ 15
1.9.4 Organización de las carpetas ____ 16
1.9.4.1 Archivo web.xml ____ 20
1.9.5 Proyectos utilitarios ____ 21
1.9.5.1 Proyecto prjUtilitarioJSF ____ 21
1.9.5.2 Beans. ____ 22
1.9.5.3 Converters ____ 22
1.9.5.4 Proyecto prjResources ____ 23
1.9.5.5 Componentes por composición ____ 23

CAPÍTULO 2: MARCO TEÓRICO ____ 26

2.1 Fundamentos esenciales del Framework ZK ____ **26**

2.1.1 Qué es el Framework ZK? ____ 26
2.1.2 XHTML ____ 27
2.1.3 XUL ____ 28
2.1.4 ZULM ____ 28

2.2 Arquitectura del Framework ZK ____ **28**

2.2.1 Tecnología Ajax embebida ____ 28
2.2.2 Arquitectura centrada en el servidor ____ 30
2.2.3 Capa de presentación ____ 31
2.2.4 Un framework dirigido por eventos ____ 32
2.2.5 Fases de una petición Ajax ____ 32

2.3 Configuración del Framework ZK 34

2.3.1 Introducción a la configuración del Framework ZK 34

2.3.2 Instalación de Java Runtime Environment 34

2.3.3 Instalación del Contenedor de Servlets Java 35

2.3.3.1 Instalación de Apache Tomcat utilizando el instalador 35

2.3.3.2 Instalación de Apache Tomcat descomprimiendo archivo .zip 36

2.3.3.3 Desplegar y probar el ejemplo zkdemo-all.war 36

2.3.4 Librerías Relacionadas 37

2.3.5 Configuración del archivo web.xml 38

2.3.6 Configuración del archivo zk.xml 40

2.4 Componentes del framework ZK 42

2.4.1 Conceptos de componente 42

2.4.2 El Ciclo de Vida de un Componente 43

2.4.3 El Ciclo de Vida de una Página ZULM 43

2.4.3.1 Fase de Inicialización de la Página 43

2.4.3.2 Fase de creación de componentes 44

2.4.3.3 Fase de procesamiento de eventos 45

2.4.3.4 Fase de representación 45

2.4.3.5 Actualización de las páginas 45

2.4.3.6 Fase de Procesamiento de solicitudes 46

2.4.3.7 Fase de procesamiento de eventos 46

2.4.3.8 Fase de representación 46

2.4.4 Recolección de Basura 47

2.4.5 Atributos de los componentes 47

2.4.5.1 El atributo id 47

2.4.5.2 Los atributos if y unless 48

2.4.5.3 El atributo forEach 48

2.4.5.4 El atributo use 49

2.4.5.5 Espacios ID 51
2.4.6 Expresiones de Lenguaje (EL) 52
2.4.7 Eventos 52
2.4.8 Espacios de nombres en XML y ZUML (*namespaces*) 54
2.4.9 Creación de cuadros de diálogo modal 55
2.4.10 Dojo 56

CAPÍTULO 3: APLICACIÓN DE LA METODOLOGÍA 57

3.1 Guía de desarrollo y programación, utilizando el framework ZK. 57

3.1.1 Introducción al desarrollo de aplicaciones con el framework ZK. 57
3.1.1.1 Objetivo de la guía de desarrollo del framework ZK. 57
3.1.1.2 A quién está dirigida esta guía 58
3.1.1.3 Perspectiva de desarrollo en el framework ZK. 58
3.1.1.4 Perspectiva del desarrollo de componentes en el framework ZK. 59
3.1.2 Directrices de tecnologías usadas en el framework ZK. 60
3.1.2.1 MVC vs. Zscript 60
3.1.2.1.1 Cuándo se debe usar MVC? 60
3.1.2.1.2 Cuándo se debe usar Zscript? 61
3.1.2.2 MVC Extractor. 61
3.1.2.3 Data Binding. 61
3.1.2.3.1 Cuándo usar Data Binding? 61
3.1.2.4 ZULM vs. Richlet vs. JSP 62
3.1.2.4.1 Cuándo usar ZULM? 62
3.1.2.4.2 Cuándo utilizar Richlet? 63
3.1.2.4.3 Cuándo utilizar JSP? 63
3.1.2.5 Bookmarks vs Multiple Pages 64
3.1.2.6 Namespaces nativos vs Componentes XHTML 65

3.1.2.6.1 Cuándo usar namespaces nativos 65
3.1.2.6.2 Cuándo usar componentes XHTML? 65
3.1.2.7 Include, Macro, Composite y Templating 66
3.1.2.7.1 Cuándo usar include? 66
3.1.2.7.2 Cuándo usar componentes macro? 66
3.1.2.7.3 Cuándo usar componentes por composición? 67
3.1.2.7.4 Cuándo usar Plantillas o Templating? 67
3.1.3 Preparación del entorno de desarrollo 68
3.1.3.1 Instalación de Eclipse 68
3.1.3.2 Instalación de ZK Studio. 69
3.1.3.3 Creación de un proyecto ZK en Eclipse 73
3.1.4 Programación bajo el framework ZK 76
3.1.4.1 Creación de páginas 77
3.1.4.1.1 Árbol de componentes. 78
3.1.4.1.2 Escritorio, Página y Componente 79
3.1.4.1.3 Formación de las páginas ZUL 81
3.1.4.1.4 Programación del Lenguaje de Expresiones EL 85
3.1.4.2 Scripts en ZULM 85
3.1.4.2.1 Insertar código Script del lado del servidor. 85
3.1.4.2.2 Zscript 86
3.1.4.3 Evaluación iterativa 88
3.1.4.3.1 El atributo forEach 88
3.1.4.3.2 El objeto Each 89
3.1.4.4 Anotaciones 90
3.1.4.4.1 Anotaciones en ZULM 90
3.1.4.4.2 Propiedades de las anotaciones 90
3.1.4.5 Cómo re direccionar a otras páginas 91
3.1.4.6 Cómo programar Richlets 92

3.1.4.6.1 Implementación de un Richlet 92
3.1.4.7 Implementación de operaciones CRUD bajo el patrón MVC 94
3.1.4.7.1 Creación de la clase CRUDComposer.java 96
3.1.4.7.2 Creación de la página crud.ZUL 101
3.2 Guía de desarrollo de componentes gráficos. 104
3.2.1 Introducción a los componentes de ZK 104
3.2.1.1 Objetivo de la guía de desarrollo de componentes gráficos 104
3.2.1.2 A quién está dirigida esta guía 105
3.2.1.3 Qué es un Componente ZK 105
3.2.1.3.1 Componente 105
3.2.1.3.2 Widget 105
3.2.1.4 Cómo trabaja un Componente ZK 106
3.2.2 Creación de un Componente ZK 107
3.2.2.1 JavaScript en los componentes ZK 107
3.2.2.2 Implementación de un Componente 108
3.2.2.3 Implementación de un Widget 110
3.2.3 Creación de archivos de configuración 112
3.2.4 Manejo de eventos 112
3.2.5 Comunicación Cliente-Servidor 113
3.2.5.1 Escuchas de lado del Servidor 114
3.2.6 Encapsular en un archivo Jar 116
3.3 Guía de estilos en las páginas. 117
3.3.1 Introducción a los estilos en las páginas del framework ZK. 117
3.3.1.1 Objetivo de la guía 117
3.3.1.2 A quien esta dirigida 117
3.3.2 Conceptos de estilos en el framework ZK. 117
3.3.2.1 Sclass 117
3.3.2.2 Zclass 118

3.3.2.3 Personalización 119
3.3.2.4 Sobrescribir estilos 120
3.4 Comparación entre el framework ZK y el framework JSF 2.0. 121
3.4.1 Introducción 121
3.4.2 Arquitecturas 122
3.4.2.1 Arquitectura JSF 123
3.4.2.2 Arquitectura ZK 126
3.4.3 Características comunes 127
3.4.3.1 Modelo Vista Controlador 127
3.4.3.2 Validación ycontrol de errores 127
3.4.3.3 Internacionalización 128
3.4.3.4 Componentespersonalizados 129
3.4.3.5 Uso de Ajax 130
3.4.4 Componentes 131
3.4.4.1 Componentes JSF 131
3.4.4.2 Componentes ZK 133
3.4.5 Pruebas de rendimiento 134
3.4.6 Tabla de resumen 140
3.4.7 Conclusiones de la comparación 141
CAPÍTULO 4: PROTOTIPO DESARROLLADO CON EL FRAMEWORK ZK 142
4.1 Descripción técnica del prototipo 142
4.1.1 Proyecto de modelo 143
4.1.2 Proyecto cliente 144
4.1.3 Proyecto web 145
4.2 Manual de usuario 146
4.2.1 Introducción 146

4.2.2 Ingreso al sistema _____ 146

4.2.3 Administración de empleados _____ 148

4.2.4 Administración de planes de carrera y eventos. _____ 152

4.2.5 Lector de noticias _____ 153

CAPÍTULO 5: CONCLUSIONES Y RECOMENDACIONES _____ 155

5.1 Conclusiones _____ **155**

5.2 Recomendaciones _____ **157**

LISTADO DE TABLAS

Tabla 3.1: Listado de componentes de Richfaces 4. 133

Tabla 3.2: Listado de componentes ZK. 134

Tabla 3.3: Tabla comparativa entre el framework ZK y el framework JSF 140

LISTADO DE FIGURAS

Figura 1.1: Agrupación de paquetes de un proyecto JSF. 13

Figura 1.2:Agrupación de carpetas en un proyecto web con JSF. 16

Figura 1.3: Agrupación de páginas por módulos del sistema. 16

Figura 1.4: Organización de la carpeta resources 17

Figura 1.5: Organización de la carpeta templates 18

Figura 1.6: Distribución de los espacios en la página pageLayout1 18

Figura 1.7: Distribución de los espacios en la página pageLayout2 19

Figura 1.8: Distribución de los espacios en la página pageLayout3. 19

Figura 1.9: Organización de la carpeta WEB-INF. 20

Figura 1.10: Estructura del proyecto prjResources. 23

Figura 2.1: Demostración en vivo muestra lo fácil que es manipular los componentes de ZK. 27

Figura 2.2: ZK loader, ZK UA y ZK cliente trabajando en una petición Ajax 29

Figura 2.3: Arquitectura del framework ZK. 30

Figura 2.4: Flujo de una petición Ajax, en el framework ZK 33

Figura 3.1: Opciones de descarga que proporciona la página de Eclipse. 69

Figura 3.2: Opciones de instalación de las librerías de ZK Studio. 69

Figura 3.3: Ventana de Eclipse Marketplace. 70

Figura 3.4: Ventana de descargas de Eclipse 71

Figura 3.5: Ventana de instalación de descargas de Eclipse 72

Figura 3.6: Ventana de bienvenida del framework ZK dentro de Eclipse 72

Figura 3.7: Ventana de selección de proyecto ZK. 73

Figura 3.8: Ventana de selección de proyectos web. 74

Figura 3.9: Ventana para agregar librerías externas a un proyecto web. 75

Figura 3.10: Ventana para actualizar un proyecto en Eclipse. 76

Figura 3.11: Ventana para agregar una nueva página ZUL. 77

Figura 3.12: Representación de un Escritorio, Página y Componentes en un navegador web. 79

Figura 3.13: Opciones de creación de componentes en la interfaz de usuario. 95

Figura 3.14: Página que permite realizar operaciones CRUD. 95

Figura 3.15: Ventana de creación de clases Java. 96

Figura 3.16: Librerías necesarias para la ejecución de la clase Composer. 97

Figura 3.17: Implementación de una clase base como componente. 109

Figura 3.18: Implementación grafica de un widget. 111

Figura 3.19: Comunicación entre un widget y un componente Java. 113

Figura 3.20 : Estructura básica de un proyecto ZK, empaquetado en un archivo Jar 116

Figura 3.21: Componente ZK personalizado su aspecto CSS 119

Figura 3.22: Relación de las clases de estilo de un componente ZK. 120

Figura 3.23: Ciclo de vida de una petición JSF 124

Figura 4.1: Distribución del proyecto *prjModeloPlan.* 143

Figura 4.2: Distribución del proyecto *prjClientePlan.* 144

Figura 4.3: Distribución del proyecto *prjWebProject* 145

Figura 4.4: Link de ingreso al sistema desde la página de Kruger Corporation. 146

Figura 4.5: Pantalla de ingreso al sistema. 147

Figura 4.6: Menú principal del prototipo desarrollado con el framework ZK. 147

Figura 4.7: Botón de ingreso a la administración de empleados. 148

Figura 4.8: Filtro de búsqueda de empleado por departamento 149

Figura 4.9: Filtro de búsqueda de empleado por perfil. 149

Figura 4.10: Filtros de edición de un empleado. 150

Figura 4.11: Figura pastel de tecnologías utilizadas por los empleados. 151

Figura 4.12: Pantalla emergente para agregar un plan de carrera a un empleado. 151

Figura 4.13: Botón de ingreso a la pantalla de administración de carreara. 152

Figura 4.14: Pantalla de administración de carreras profesionales. 152

Figura 4.15: Calendario de eventos. 153

Figura 4.16: Botón de ingreso a la pantalla de noticias 154

Figura 4.17: Pantalla de noticias utilizando el framework ZK. 154

LISTADO DE CÓDIGO

Código 2.1: Programación del archivo web.xml en el framework ZK. 39

Código 2.2: Programación del archivo zk.xml 41

Código 2.3: Ejemplo del atributo id 48

Código 2.4: Ejemplo de atributo *if* y *unless* 48

Código 2.5: Ejemplo de atributo *forEach* 49

Código 2.6: Ejemplo de componente ListBox. 49

Código 2.7: Asociar un componente a una clase para utilizar sus métodos 50

Código 2.8:. MyClass.java 50

Código 2.9: MyWindow.java 51

Código 2.10: Ejemplo de EL. 52

Código 2.11: Controlador de eventos con un atributo definido 53

Código 2.12: Controlador de eventos con un método definido 53

Código 2.13: Controlador de eventos. 54

Código 2.14: Mezcla de componentes ZUL y ZHTML 55

Código 2.15: Un ejemplo del uso de Dojo para crear una lista de ojo de pescado 56

Código 3.1: Ejemplo de código embebido Zscript. 87

Código 3.2: Richlet implementado en una clase Java 93

Código 3.3: Vinculación de una clase Composer con una página ZUL. 96

Código 3.4: Clase CRUDComposer.java. 101

Código 3.5: Creación de un componente listbox. 102

Código 3.6: Creación de un formulario dentro de un componente *grid* 103

Código 3.7: Creación botones dentro de un componente *hbox* 103

Código 3.8: Página crud.zul 104

Código 3.9: Interacción entre el usuario, el componente ZK y la aplicación 106

Código 3.10: Invocación de un método de un componente ZK. 107

Código 3.11: Ejemplo de creación de una clase en JavaScript. 108

Código 3.12: Implementación de un componente.....110

Código 3.13: Archivo JavaScript que representa la implementación de un widget 111

Código 3.14: Archivo lang-addon.xml 112

Código 3.15: Implementación de un evento en el componente propio SimpleLabel. 113

Código 3.16: Comunicación Cliente-Servidor en un componente. 114

Código 3.17 : Agregar un evento mediante la función *addClientEvent* 115

Código 3.18 : Utilización de un evento propio. 115

Código 3.19: Utilización de la clase sclass. 118

Código 3.20: Sobrescribir clases CSS que contienen los componentes 121

Código 3.21: Configuración de un properties en el archivo faces-config.xml 128

Código 3.22: Gmap.zul. 136

Código 3.23: Gmapx.jspx.....137

Código 3.24: LocatorBean.java 139

Código 3.25: Configuración del archivo faces-config.xml para agregar un bean. 139

RESUMEN

El presente trabajo muestra la investigación de la arquitectura, los componentes, la configuración e instalación del framework ZK. Además se desarrollaron tres guías de programación y diseño del framework y un prototipo de aplicación construido con el mismo. Se ha documentado ejemplos prácticos reales para la construcción de proyectos dentro de la plataforma JEE que utilicen en su capa web el framework ZK.

Se eligió el framework ZK, por la necesidad de investigar nuevas soluciones tecnológicas, que puedan proveer frameworks ágiles para el desarrollo de aplicaciones web, que sea intuitivo al codificarse, fácil de aprender para los programadores inexpertos, que tenga poca configuración y que responda rápidamente ante cualquier cambio o nuevo requerimiento. ZK ofrece un entorno de desarrollo rápido, y una línea de aprendizaje prácticamente plana .Soporta todos los principales patrones de desarrollo, tales como MVC o MVVMy tiene características ayudan a mejorar la cultura organizacional de desarrollo web en los sistemas que se vayan a ejecutar en un futuro en Kruger Corporation.

Adicional a la investigación del framework se realizó una comparación con el framework JSF (Java Server Faces) en su versión 2. Se comparó tomando en consideración las principales características.

ABSTRACT

This thesis shows the investigation of the architecture, components, configuration and installation of the ZK framework. As results of this work three guides programming and design was developed and also a prototype application was built with it. It has been documented real practical examples for the construction of projects within the JEE platform to use in the web tier framework ZK.

ZK framework was chosen because of need to research new technology solutions that provide flexible frameworks for developing web applications, that is intuitive to coded, easy to learn for novice programmers and respond quickly to any changes or requirements of the application. ZK provides a rapid development environment, and a nearly flat learning curve. ZK supports all major development patterns such as MVC. Many features of the framework match with above needs and help to improve the culture Web development organizational systems that are executed in the future in Kruger Corporation.

In addition tothe investigationof the framework,a comparison was madewithJava ServerFacesframeworkin version 2. Main characteristics were compared with ZK and JSF frameworks

CAPÍTULO 1: INTRODUCCIÓN

1.1 Contextualización del Problema

1.1.1 Labor de la empresa:

Kruger Corporation a lo largo de los años se ha consolidado como una de las empresas más reconocidas de tecnología a nivel nacional, y que cada día se abre importante campo a nivel internacional. De esta manera Kruger se constituye como una importante y reconocida empresa de desarrollo de software, creación y transmisión tecnológica.

1.1.2 Área específica:

Para el desarrollo de este proyecto de investigación, se trabajó con el departamento de desarrollo de software específicamente alojado en el cliente Supermaxi, ya que es ahí donde se encuentran arquitectos de software conocedoresdel tema, que darán las directrices necesarias para encaminar la investigación de acuerdo a las necesidades tecnológicas ágiles de cambio e implementación. Se trabajó en conjunto bajo una supervisión y aprobación de la gerencia técnica de la empresa de igual forma.

1.2 Formulación del Problema:

Actualmente Kruger Corporation desarrolla sus sistemasempresariales utilizando frameworks como JSF[1] (Java Server Faces) y STRUTS[2] para su capa web en la

[1] Java Server Faces http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html

plataforma JEE[3] (Java Enterprise Edition). Estos frameworks proveen características importantes que permiten un desarrollo web robusto, dinámico y estable. Sin embargo el desarrollo no es completamente ágil si se quisiera implementar una solución web en muy poco tiempo y con un alcance mediano, los componentes que proveen dichos frameworks en ocasiones no cumplen con los requerimientos funcionales que necesita la aplicación.

Es ahí donde se ve la necesidad de investigar nuevas soluciones tecnológicas, que puedan proveer frameworks ágiles para el desarrollo de aplicaciones web, que sea intuitivo al codificarse, fácil de aprender para los programadores inexpertos, que tenga poca configuración y que responda rápido ante cualquier cambio o requerimiento de la aplicación.

1.3 Delimitación Espacial

La investigación y el desarrollo de un prototipo del framework ZK se lo llevará a cabo en mi domicilio y las instalaciones del cliente Supermaxi de la empresa Kruger.

1.4 Delimitación Temporal

La investigación y el desarrollo de un prototipo del framework ZK se realiza en base a las necesidades tecnológicas web que requieren los sistemas empresariales desde que salió la especificación Java EE 6 en el año 2009.

Se investigó el framework ZK, a partir de su versión 5.

---

[2] Apache Struts http://es.wikipedia.org/wiki/Apache_Struts
[3] Java Platform, Enterprise Edition

1.5 Sistematizacióndel problema

Será posible con la investigación del framework ZK, obtener un marco referencial teórico y práctico, para adoptar dicho framework como estándar de arquitectura web en las aplicaciones empresariales basadas en la plataforma JEE de la empresa Kruger Corporation.

Será posible con la investigación del framework ZK, y el desarrollo de un prototipo de sistema basado en dicho framework, hacer más ágil e intuitivo para el programador el desarrollo de las aplicaciones web y que sean iguales o más robustas que las aplicaciones que utilizan frameworks conocidos como JSF.

Será posible con la investigación del framework ZK, obtener un desarrollo web de las aplicaciones del lado del servidor basado en componentes y orientada a eventos, que dinamice más el desarrollo web en aplicaciones empresariales y provea características visuales más llamativas e intuitivas para el usuario.

1.6 Objetivos

1.6.1 Objetivo General:

Realizar una guía de desarrollo del framework ZK, mediante la investigación dela arquitectura, funcionalidady componentes del mismoy el desarrollode un prototipo de aplicación basado en dicho frameworkque como funcionalidad principal permita controlar el plan de capacitación interno de los empleados en Kruger Corporation, con la finalidad de proporcionar una alternativa tecnológica para el diseño y la

arquitectura de la capa web a los desarrolladores de software de la empresa en mención.

1.6.2 Objetivos Específicos:

- Proporcionar la documentación pertinente para el desarrollo de componentes, la programación y la construcción de vistas del framework ZK, que sirva como herramienta a la gerencia técnica, a los programadores y arquitectos de software en Kruger Corporation, para facilitar la toma de decisiones al momento de elegir un framework para el desarrollo web de aplicaciones empresariales.
- Desarrollar una aplicación utilizando el framework ZK que contenga la arquitectura, funcionalidad y componentes que el framework provee y que funcionalmente controle el plan de capacitación interno de Kruger Corporation.
- Realizar un análisis comparativo entre el framework ZK y el framework JSF que permita decidir cuándo y porqué usar uno u otro framework si se requiere desarrollar un nuevo sistema, requerimiento o aplicación empresarial.

1.7 Justificación e importancia:

Los sistemas empresariales dentro de la plataforma JEE, consisten en la integración e interacción de varios componentes que independientemente, pueden desempeñar una labor específica para el correcto funcionamiento del sistema. La capa web o de

presentación debe ser un componente fundamental en el desarrollo de dichos sistemas empresariales, ya que es la cara del sistema al cliente.

Es por eso que se necesita desarrollar sistemas empresariales, que presenten una interfaz de usuario amigable, intuitiva, dinámica, en la que el usuario interactué con el sistema de manera activa, y que facilite la labor diaria del mismo, pero de igual manera, que dicha interface sea ágilmente desarrollada por el programador, respondiendo rápidamente a cambios de requerimientos.

Para lograr esto, el camino más óptimo adoptado por los arquitectos, desarrolladores y programadores de software es la implementación de un framework web.

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. Los más antiguos como STRUTS permiten un manejo más directo de los datos que se procesan mientras que los más modernos como JSF, ZK o GWT[4](Google Web Toolkit), buscan la abstracción casi total del protocolo, a cambio de generar modelos de componentes fácilmente extensibles y orientados a eventos.

Cada framework busca solucionar objetivos generales. Por ejemplo, mientras que ROR[5](Ruby On Rails), permite crear aplicaciones de manera muy ágil, pero como resultadose tiene una sencillez extrema y falta de flexibilidad, otros como COCOON[6] y JSF fueron diseñados para no depender del dispositivo de presentación, lo que otorga varios puntos a las aplicaciones que buscan verse bien en dispositivos

[4] http://code.google.com/webtoolkit/
[5] http://es.wikipedia.org/wiki/Ruby_on_Rails
[6] http://cocoon.apache.org/1363_1_1.html

móviles como celulares palms o tablets. ASP.NET [7] por su parte permite una productividad interesante pero limita el trabajo a la herramienta que provee su fabricante quedando atado a las futuras decisiones de la compañía.

Por lo expuesto anteriormente se puede decir que el futuro próximo del desarrollo web ágil, está marcado por las interfaces ricas de usuario, con interactividad, sin necesidad de realizar consultas permanentes al servidor, con la posibilidad que desde el cliente se pueda 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, a otra con la posibilidad de procesar la información y mantener un estado propio.

Una vez expuestos los puntos más relevantes, se plantea la investigación de un nuevo framework web llamado ZK, ya que ofrece un excelente rendimientoen el desarrollo de aplicaciones empresariales robustas, pero a su vez ágiles de desarrollar. Esto se demuestra al constatar oficialmente que empresas como SONY, SUN, IBM, ROCHE, TOYOTA, ADOBE entre muchas otras más, utilizan este framework para el desarrollo de sus sistemas y aplicaciones.

ZK ofrece un entorno de desarrollo rápido, y una línea de aprendizaje prácticamente plana.ZK soporta todos los principales patrones de desarrollo, tales como MVC, así como muchas más características propias del framework que satisfacen las necesidades anteriormente mencionadas y pueden ayudar a mejorar la cultura organizacional de desarrollo web en los sistemas que se vayan a ejecutar en un futuro en Kruger Corporation.

[7] http://es.wikipedia.org/wiki/ASP.NET

También se plantea el desarrollo de un prototipo de sistema desarrollado con dicho framework, donde se aplique todo en torno a la investigación realizada y se abstraiga todos los conceptos que se estén por definir en la investigación.

1.8 Alcance

El alcance del presente trabajo de investigación abarca la descripción del framework ZK en su plataforma de componentes gratuitos, es decir todas las librerías que provee dicho framework conlicenciamientoGPL(General Public License) y LGPL[8](Lesser General Public License).

Dentro del alcance también está el desarrollar un sistema prototipo,que utilice para el desarrollo de la capa web el presente framework a investigar.

La investigación comprende los siguientes aspectos:

- Instalación del framework.
- Configuración del framework.
- Componentes gratuitos del framework y personalización de los mismos.
- Arquitectura y componentes internos del framework.
- Ciclo de vida de una petición Ajax dentro del framework.
- Programación de dichos componentes.
- Construcción de las vistas de la capa web.
- Desarrollo de componentes propios.
- Comparación con el framework JSF.

[8] http://www.zkoss.org/license/extension

El sistema desarrollado, tiene como funcionalidad más relevante el controlar el plan de capacitación interno en Kruger Corporation. Entre las funcionalidades más relevantes del prototipo que se va a desarrollar:

- Crear un plan de carrera para cada empleado.
- Administrar la carrera profesional de cada empleado dentro de la empresa.
- Crear cursos de capacitación de acuerdo al área de trabajo.
- Se creara los eventos a los cuales asistirán empleados.
- Se podrá planear la asistencia de los empleados a dichos cursos.
- Visualización de los cursos o eventos vigentes a futuro.

1.9 Situación actual de la arquitectura web utilizada en la empresa

1.9.1 Tecnologíasen uso

1.9.1.1 JSF 2.X

Java Server Faces es el frameworkadoptado para el desarrollo de la arquitectura web. Este framework es el estándar que brinda la especificación JEE 6 para construir interfaces de usuario. Es un framework que reside en el lado del servidor, orientado a eventos y basado en componentes características que también tiene el framework ZK.

La tecnología de JSF consiste en los siguientes puntos:

- Un API (Application Programming Interface) para representar los componentes y la gestión de su estado, manejo de eventos, validación del lado del servidor y también del lado del cliente, comunicación entre componentes, conversión de datos, definición de la navegación entre las

páginas que en la versión 2 de JSF es casi implícita, internacionalización y acceso directo a los datos de una base de datos.

- Librerías de etiquetas para agregar componentes en las páginas web y para conectar los componentes del lado del servidor.

1.9.1.2 Richfaces 4.x

Richfaces es un conjunto de librerías desarrolladas por JBoss Community que implementan el framework JSF y que sirven como complemento al mismo, para potenciarlo y optimizarlo de acuerdo a las necesidades de desarrollo que se quieran implementar. Consiste en la inclusión de componentes basados en la tecnología ya nombrada y que extienden la funcionalidad de los componentes nativos de JSF. Richfaces facilita la utilización del framework JSF ya que abstrae en sus librerías toda la tecnología de este framework, por ejemplo la utilización de Ajax o la paginación en base.

1.9.1.3 Primefaces 3.x

Es otra implementación de la tecnología JSF.

Primefaces consiste en un conjunto de librerías que residen en un servidor de aplicaciones. Primefaces provee también componentes que enriquecen la interface de usuario y optimizan la utilización del framework. Cabe recalcar que los componentes Primefaces se utilizan como una alternativa de componentes ya que los componentes principales en la arquitectura web actual son de Richfaces.

1.9.1.4 Eclipse 3.6

Es un IDE (Integrate Development Evironment) de código abierto, que puede multiplataforma, desarrollado por la empresa Eclipse Foundation. Este programa sirve para el desarrollo y modelamiento de muchas aplicaciones que pueden ser de diferentes ámbitos y escritas en diferentes lenguajes. Está programado en Java, por lo que se especializa en la construcción de aplicaciones escritas en dicho lenguaje.. Actualmente se utiliza Eclipse en su versión 6 y se añade las librerías de JBoss Tools. Estas librerías hacen que se pueda trabajar de mejor manera dentro del IDE[9] (Entorno de Desarrollo Integrado) con tecnologías propias de la especificación JEE.

1.9.1.5 JBOSS 6.0

Es un servidor de aplicaciones para la plataforma JEE de código abierto y que está hecho al igual que Eclipse en Java.

Este servidor de aplicaciones puede ser utilizado en cualquier ambiente donde corra una JVM (Java Virtual Machine) por lo que es multiplataforma al igual que el lenguaje en el que está hecho.

1.9.2 Organización de los proyectos

El nombre de los paquetes hace referencia al sistema y funcionalidad que se desea incluir en este, tomando en cuenta un prefijo específico que dependerá de la

[9]http://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado

compañía para la cual se esté realizando la implementación, entonces el estándar es:

[*prefijo_pais*].[*tipo_organizacion*].[*id_organizacion*].[*prefijo_sistema*].[*tipo_proyecto*].[*modulo*].[sub-modulo/funcionalidad]…

Ejemplo: ec.com.smx.sic.web.articulos.catalogos

Para un proyecto web realizado con JSF, luego del paquete que agrupa el último nivel de funcionalidad se deberá crear por lo menos dos paquetes adicionales llamados: controller y datamanager. Como se muestra en la Figura 1.1

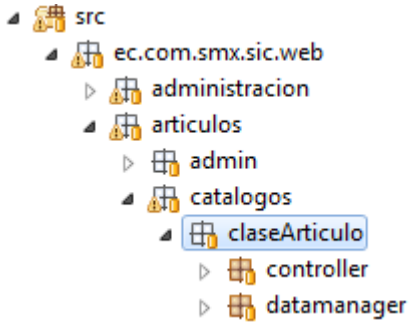


Figura 1.1: Agrupación de paquetes de un proyecto JSF.

Fuente: El autor.

1.9.3 Los Controllers y Datamanagers

Los *Controllers* o Controladores, desde el punto de vista de JSF son *ManagedBeans* donde se ubicarán todos los procesos principales para una pantalla, normalmente el *scope* o alcance de estos beans será *Request* o *View.* Como ya se sabe el alcance *Request* solo mantiene el estado del bean el tiempo que dura una petición y es muy corto.

El alcance *View* mantiene el estado del bean mientras no se navegue a otra vista o pantalla, esto es muy útil porque nos evita o disminuye el uso de variables de sesión.

Los *Datamanagers* o Manejadores de datos también son *ManagedBeans* pero estos tienen un alcance de *Session*, es decir el estado de esté bean se mantendrá de una vista o pantalla a otra. Es recomendable guardar en sesión solo lo que se necesite, no es recomendable tener objetos muy pesados ni procedimientos o métodos en estos beans.

Todo controlador debe obligatoriamente extender una clase llamada *CommonController* ya que esta contiene métodos genéricos que permiten un correcto funcionamiento de las pantallas. Además deberán tener una propiedad asociada a su *Datamanager*, por otra parte todos los *Datamanager* deberán extender de una clase llamada *CommonDataManager.*A continuación se muestra un ejemplo de código de un controlador.

```
@ManagedBean
@RequestScoped
publicclass AdminClaseArticuloController extends CommonController{

	@ManagedProperty(value="#{adminClaseArticuloDataManager}")
	private AdminClaseArticuloDataManager adminClaseArticuloDataManager;

	@Override
	publicCommonDataManager getDataManager() {
		returnadminClaseArticuloDataManager;
	}

	/**
	 * Método usado para inicializar
	 */
	@Override
	publicvoid initialize() {
		//AQUI LA INICIALIZACION DE LA PANTALLA
@ManagedBean
@SessionScoped
publicclass AdminClaseArticuloDataManager extends CommonDataManager{


	@Override
```

```
public String getIdDataManager() {
	return"adminClaseArticuloDataManager";
}
```

El nombre de todo controlador debe tener el sufijo *Controller*, y en el casos de los Datamanagers el sufijo *DataManager.*

Como se puede observar en el ejemplo anterior todo controlador debe implementar dos métodos: getDataManager() e initialize().

- getDataManager (): Obtendrá el objeto *Datamanager* asociado al controlador.
- Initialize (): Permite inicializar el controlador, cabe señalar que este método inicializa solo una vez el controlador.

Hay casos donde se tiene un controlador para varias pantallas, en estos casos en lugar de usar el método *initilize* es preferible usar un método con la anotación *PostConstruct*, tomando en cuenta que se debe controlar la inicialización por cada pantalla.

Por otra parte todo *Datamanager* deberá implementar el método *getIdDataManager()* este método retornará el identificador del bean que por omisión es el nombre de la clase con la primera letra minúscula a menos que se especifique lo contrario en la anotación *@ManagedBean.* Este id será usado para eliminar el datamanager de sesión cuando el usuario cierre la ventana de trabajo.

1.9.3.1 Organización del contenido web

El contenido web está ubicado dentro de la carpeta donde se ubican las páginas xhtml, archivos de configuración, recursos como imágenes, JavaScript, hojas de estilo y librerías locales.

1.9.4 Organización de las carpetas

En la Figura 1.2, se muestra la organización de las carpetas en el proyecto:

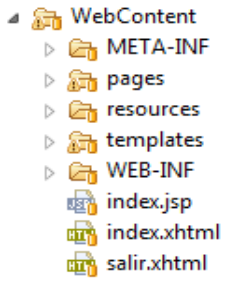


Figura 1.2: Agrupación de carpetas en un proyecto web con JSF.

Fuente: El autor.

Las páginas ubicadas en la raíz de la carpeta WebContentson páginas genéricas que controlan el ingreso y la salida del sistema.

La carpeta pages contendrá las páginas xhtml del proyecto, dentro de esta se ubica una estructura jerárquica que agrupa los diferentes módulos y por cada módulo sus opciones o funcionalidades, como se muestra en la Figura 1.3.

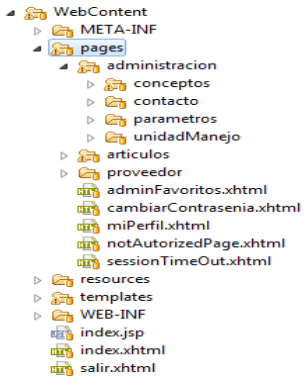


Figura 1.3: Agrupación de páginas por módulos del sistema.

Fuente: El autor.

La carpeta resources debe contener todos los recursos del contenido web, por ejemplo componentes por composición (componentes), imágenes (images), JavaScript (js) y hojas de estilo en cascada (CSS), cabe señalar que el nombre resources es un estándar establecido para aplicaciones JSF 2.x, es decir que el motor de JSF buscará los recursos en esa ruta. También se puede agregar esta carpeta dentro de una archivo JAR, en este caso deberá ir dentro de la carpeta META-INF, esto es muy interesante porque permite agrupar archivos comunes que luego pueden ser usados en muchos proyectos web. Como muestra la Figura 1.4.

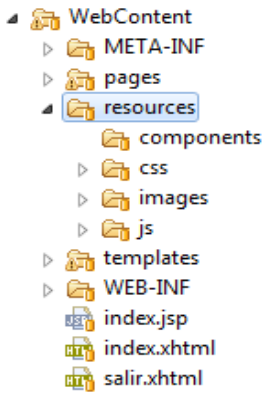


Figura 1.4: Organización de la carpeta resources

Fuente: El autor.

La carpeta templates contiene segmentos genéricos que todas las pantallas del sistema deberán tener. La carpeta main contiene la plantilla de la vista principal y sus segmentos. La carpeta pages contiene tres plantillas base que pueden ser usadas por cualquier vista y los segmentos genéricos de cada una. La Figura 1.5 muestra las estructura de esta carpeta:

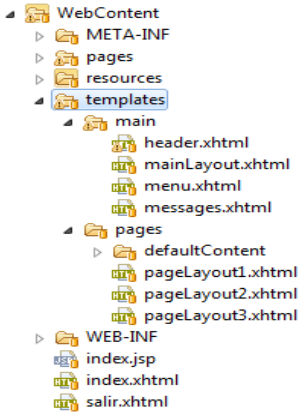


Figura 1.5: Organización de la carpeta templates

Fuente: El autor.

Para usar una de estas plantillas se debe hacer uso de los componentes de *facelets*, por ejemplo la página que vaya a usar la plantilla dos (pageLayout2.

Las figuras 1.6, 1.7 y 1.8 muestran la estructura de las plantillas.

- pageLayout1

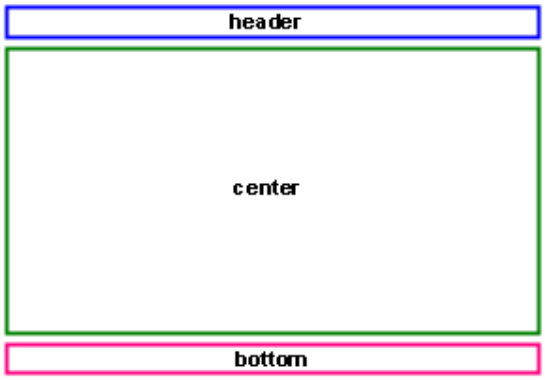


Figura 1.6: Distribución de los espacios en la página pageLayout1

Fuente: El autor

- pageLayout2

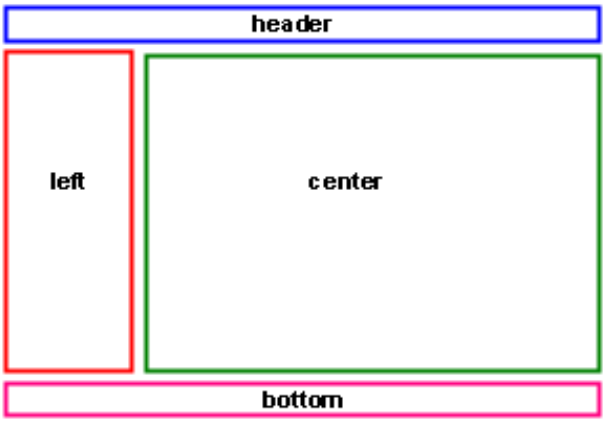


Figura 1.7: Distribución de los espacios en la página pageLayout2

Fuente: El autor.

- pageLayout3

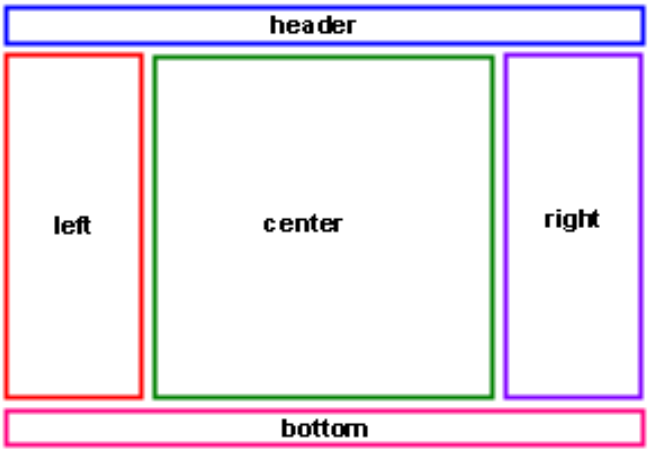


Figura 1.8: Distribución de los espacios en la página pageLayout3.

Fuente: El autor.

Dependiendo de la plantilla que se use se deberá definir el contenido de la página.

Finalmente en la carpeta WEB-INF se guardan los archivos de configuración del proyecto web los principales son: web.xml y faces-config.xml.

Esto se observa en la Figura 1.9

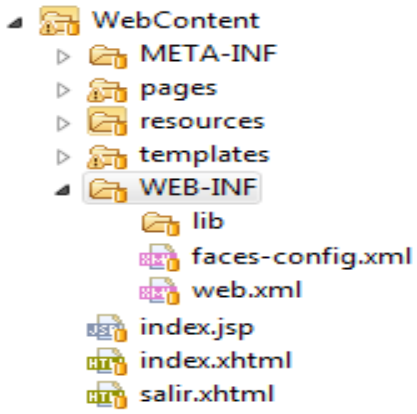


Figura 1.9: Organización de la carpeta WEB-INF

Fuente: El autor.

1.9.4.1 Archivo web.xml

En el archivo web.xml se recomienda que tenga la siguiente configuración:

Parámetros de contexto.

- `javax.faces.FACELETS_SKIP_COMMENTS`

Este parámetro se deberá asignar a **true** para que los comentarios en las páginas no sean interpretados por el lenguaje de expresiones.

- `javax.faces.INTERPRET_EMPTY_STRING_SUBMITTED_VALUES_AS_NULL`

Este parámetro se deberá asignar a **true** para que los componentes inputText que son String sean enviados en nulo y no como cadenas vacías, cuando estos no tengan valores.

- `javax.faces.PROJECT_STAGE`

Este parámetro se deberá asignar a *Production*ya que en este ambiente el framework hace un mejor uso de los recursos.

- `javax.faces.DATETIMECONVERTER_DEFAULT_TIMEZONE_IS_SYSTEM_TIMEZONE`

Este parámetro se deberá asignar a **true** para que no haya problemas de conversión en las fechas.

- `com.sun.faces.expressionFactory`

Este parámetro se deberá asignar con el valor `com.sun.el.ExpressionFactoryImpl` para que funcione la implementación del lenguaje de expresiones de SUN.

A continuación una parte del código del archivo web.xml

```
<context-param>
	<param-name>javax.faces.PARTIAL_STATE_SAVING</param-name>
	<param-value>true</param-value>
</context-param>
```

1.9.5 Proyectos utilitarios

Actualmente hay dos proyectos utilitarios los cuales dan soporte a las aplicaciones JSF, desde diferentes puntos de vista.

1.9.5.1 Proyecto prjUtilitarioJSF

Este proyecto contiene clases utilitarias y objetos JSF comunes que dan soporte a las aplicaciones JSF. En este proyecto está configurado un archivo genérico *faces-*

config.xml en la carpeta META-INF con definiciones comunes.Entre los objetos JSF comunes se tiene:

1.9.5.2 Beans.

Son clases que cumplen un propósito funcional específico, y es genérico para varios proyectos como validadores, convertidores entre otros. Por ejemplo:

- *ec.com.smx.framework.jsf.commons.beans.LogAuditoriaBean* Usado para controlar la visualización del histórico de cambios registrados en las tablas de auditoría del framework. Los controladores que quieran usar la funcionalidad de este bean deberán instanciarlo manualmente.
  - *ec.com.smx.framework.jsf.commons.session*
  - *SessionControllerBase*

  Usado para controlar el ingreso de un usuario a un sistema.

  - *SessionDataManagerBase*

  Usado para almacenar datos generales como el usuario, la compañía, el sistema y menú de opciones.

1.9.5.3 Converters

Ubicados en el paquete `ec.com.smx.framework.jsf.converters`. Aquí está el `DateFormatConverter` usado para dar formato específico a una fecha. También se puede usar el convertidor común que tiene JSF pero hay que tomar en cuenta el *TimeZone* que este configurado mediante el parámetro:

`javax.faces.DATETIMECONVERTER_DEFAULT_TIMEZONE_IS_SYSTEM_TIMEZONE`

Ubicados en el paquete ec.com.smx.framework.jsf.validators. Validadores comunes para la Cédula, RUC, Email, EAN (European Article Number, usado para la codificación de artículos), etc.

1.9.5.4 Proyecto prjResources

En este proyecto se encuentran los recursos compartidos por las aplicaciones JSF, todos están ubicados dentro de la carpeta META-INF/resources. Además también está configurado el archivo de propiedades para el skin personalizado en el caso de usar RICHFACES.

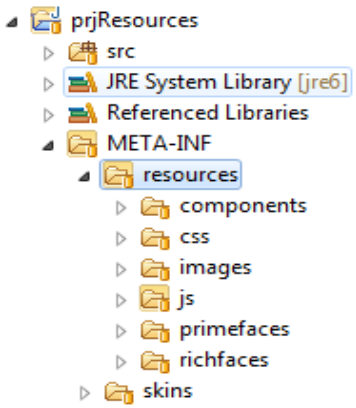


Figura 1.10: Estructura del proyecto prjResources.

Fuente: El autor.

1.9.5.5 Componentes por composición

Los componentes por composición están en la carpeta *components*, cada uno es la agrupación de varios componentes nativos, la idea principal es que puedan

reutilizarse y de esta forma disminuir el tiempo de desarrollo ya que cada uno tiene incluido los estándares de diseño.

Para hacer uso de estos componentes se debe agregar la siguiente referencia en cada página:

```
xmlns:s="http://java.sun.com/jsf/composite/components"
```

A continuación una descripción de los componentes actualmente creados.

1. **carrousel**. Despliega una colección de imágenes.
2. **ouputImage**. Muestra una imagen (*h:graphicImage*) precedida de una etiqueta (*h:outputLabel*).
3. **outputMessage**. Muestra un mensaje (*h:outputText*) precedido de una imagen (*h:graphicImage*) dentro de una panel (*rich:panel*) con un estilo personalizable.
4. **outputText.** Permite mostrar texto (*h:outputText*) precedida de una etiqueta (*h:outputLabel*).
5. **validationInputDate.** Está formado por un componente calendario (*rich:calendar*) precedido de una etiqueta (*h:outputLabel*), además tiene un indicador de error en caso de que la validación del componente falle.
6. **validationInputSecretText.** Está formado por un componente input (*h:inputSecret*) precedido de una etiqueta (*h:outputLabel*), además tiene un indicador de error en caso de que la validación del componente falle.
7. **validationInputText.** Está formado por un componente input (*h:inputText*) precedido de una etiqueta (*h:outputLabel*), además tiene un indicador de error en caso de que la validación del componente falle.

8. **validationInputTextArea.** Está formado por un componente input (*h:inputTextarea*) precedido de una etiqueta (*h:outputLabel*), además tiene un indicador de error en caso de que la validación del componente falle.
9. **validationSelectOneMenu.** Está formado por un componente input (*h:selectOnerMenu*) precedido de una etiqueta (*h:outputLabel*), además tiene un indicador de error en caso de que la validación del componente falle.
10. **validationSelectOneRadio.** Está formado por un componente input (*h:selectOneRadio*) precedido de una etiqueta (*h:outputLabel*), además tiene un indicador de error en caso de que la validación del componente falle.

CAPÍTULO 2: MARCO TEÓRICO

2.1 Fundamentos esenciales del Framework ZK

2.1.1 Qué es el Framework ZK?

ZK es un framework dirigido por eventos y basado en componentes, para desarrollar aplicaciones web sin JavaScripty basadas en Ajax, que permite al programador disminuir la codificación, el tiempo de desarrollo y lograr interfaces ricas para el usuario.

A diferencia de otrosframeworks que utilizan Ajax, en el framework ZK no es necesario tener ningún conocimientode JavaScriptpara desarrollaraplicaciones basadas en Ajax, ya que el motor de ZKgenera automáticamenteel código JavaScript. Para desarrollaruna aplicación web conZK, se necesita sabersólo un poco sobre HTML.Para simplificar el desarrollodeaplicaciones web, ZKutiliza el lenguaje marcadoZUMLpara la creación de páginas web concomponentesZK.

Para tener una mejor apreciación de la vista en general del framework, se puede ingresar a la página (http://www.zkoss.org/zkdemo/grid). Esta demostración en vivo,exploradiferentes ejemplosde los componentes deZKque se construyencon Ajaxy ofrecen unaprogramación en tiempo real.

La Figura 2.1 muestra la ilustración deun componenteZK:Gráfico, la distribución de la gráfica de pastel se actualizaráautomáticamente al cambiarlos valores de lascategorías de programación

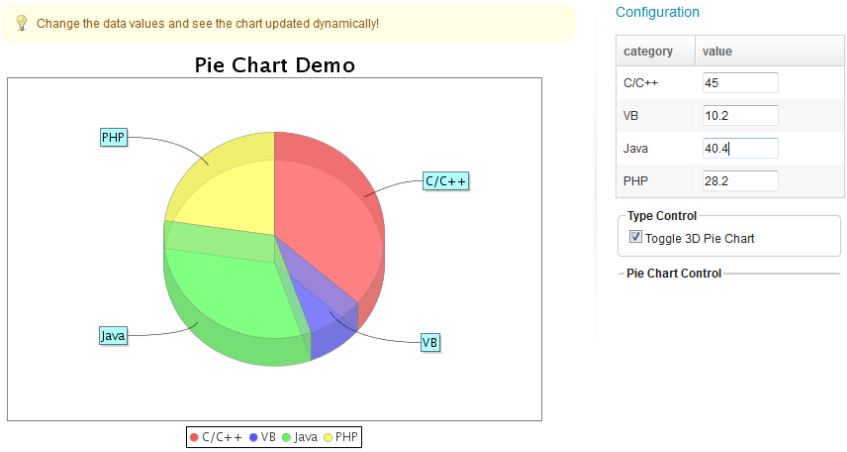


Figura 2.1: Demostraciónen vivomuestralo fácil que esmanipular los componentes deZK.

Fuente: http://www.zkoss.org/zkdemo/chart/pie_chart

2.1.2 XHTML

XHTML es una abreviación de *Extensible HyperText Markup Language.* XHTML es muy similar a HTML, de hecho XHTML es HTML escrito en con la sintaxis de XML. Aquí detallaré las principales diferencias entre estos dos lenguajes de marcado:

1. La notación (mayúscula o minúscula) de un elemento o atributo es importante en XHTML es decir como en lenguaje Java, en XHTML toma en cuenta las mayúsculas y minúsculas en su escritura.
2. Los elementos sin contenido deben ser cerrados en XHTML de la siguiente manera (
) .

2.1.3 XUL

XUL es la abreviación para *XML User Interface Markup Language.* Este lenguaje no se lo ha creado con el framework ZK. Es originalmente definido por el equipo de Mozilla. La página del proyecto es (http://www.mozilla.org/projects/xul/). Este lenguaje fue creado con el propósito de tener una plataforma y un lenguaje independiente para definir interfaces de usuario. Por lo tanto, el desarrollador ZK se encuentra beneficiado por la gran comunidad que existe alrededor de XUL, y puede usar los widgets[10] XUL.

2.1.4 ZULM

ZULM es la abreviación para *ZK User Interface Markup Language.* ZULM está basado en XUL. Existen algunas extensiones para el estándar XUL.Este lenguaje de marcado, tiene una línea de aprendizaje casi plana para los desarrolladores ya que es muy parecido a escribir las páginas en XHTML.

2.2 Arquitectura del Framework ZK

2.2.1 Tecnología Ajax embebida

El funcionamiento basado en Ajaxdel frameworkZK,se realiza portres partes importantes, como se muestra en La Figura 2.2;el ZK loader, el motor de actualización asíncrona ZK UA, y el motor delcliente ZK.ElZK loader y el motor de actualización asíncrona ZKAU,están compuestos porun conjunto deservlets de Java,

[10] http://es.wikipedia.org/wiki/Widget

y el motor de cliente ZK se compone decódigosJavaScript.La Figura 2.2 ilustrael mecanismocuando elZK loaderrecibe una solicitud deURLpor primera vez.

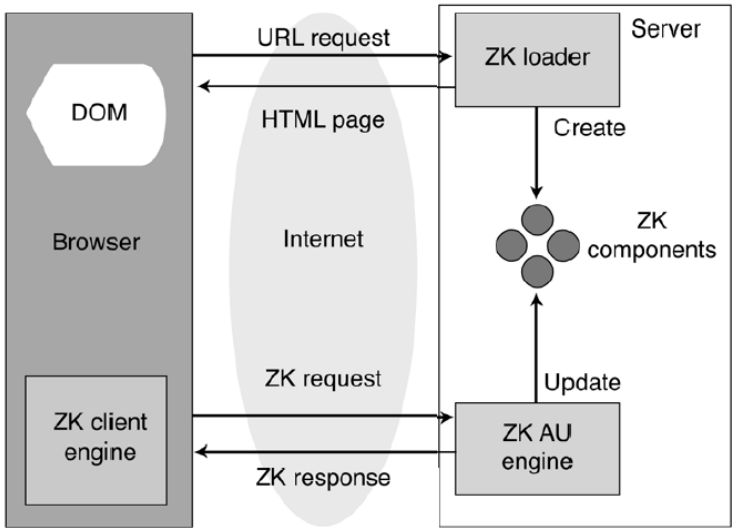


Figura 2.2: ZK loader, ZK UA y ZK cliente trabajando en una petición Ajax

Fuente: Henri Cheng & Robie Cheng ZK Ajax without JavaScript Framework, 2008

El mecanismo funciona de la siguiente manera:

1. El ZK loader interpreta una solicitud URL y genera la correspondiente página HTML, incluyendo los estándares HTML, los estilos CSS, el código JavaScript, y los componentes ZK en el lado del servidor.
2. El ZK loader envía la página HTML al motor del cliente ZK. El motor del cliente ZK reside en el lado del cliente para monitorear los eventos JavaScript encolados en el browser.
3. Si los eventos JavaScript se activan, el motor de cliente ZK envía esos eventos es decir las peticiones Ajax devuelta al motor de actualización asíncrona ZK AU sobre el servidor.El motor de actualización asíncrona ZK AU

recibe las peticiones Ajax, actualiza los componentes ZK, y envía una respuesta Ajax de vuelta al lado del cliente.

4. Por último, el motor del cliente ZK recibe la respuesta y actualiza el contenido correspondiente en el browser y el árbol de componentes.

El motor declienteZKse compone deuna gran cantidad decódigo JavaScript quese encarga de recibirlos eventos yactualizar elcontenido de las páginasweb, de esta manera es posible lograr una arquitectura muy similar a la que se tiene en las aplicaciones de escritorio construidas con la biblioteca gráfica de java Swing[11].

2.2.2 Arquitectura centrada en el servidor

La Figura 2.3 muestra como trabajan juntos los componentes internos y externos del framework ZK en una petición Ajax.

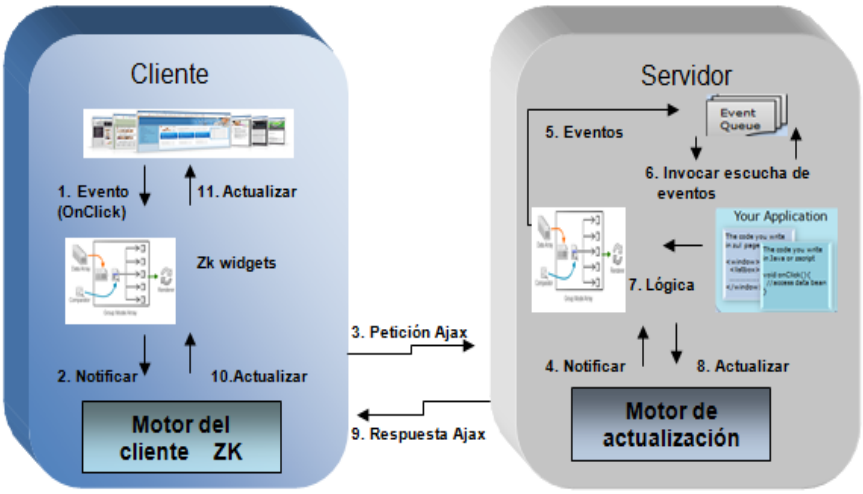


Figura 2.3: Arquitectura del framework ZK.

Fuente: El autor.

[11] http://en.wikipedia.org/wiki/Swing_%28Java%29

Las aplicacionesZK se ejecutan en el servidor y pueden tener acceso a los recursos y servicios del mismo,construirla interfaz de usuariocon componentes, interactuar conla actividad del usuario, y luegomanipular los componentespara esta interfaz, tal como se explica en La Figura 2.3. Todose lleva a caboen el servidor.La sincronización delos estados delos componentesentre el navegadory el servidorse realiza automáticamente por el motor de actualización de ZKy es transparente a la aplicación. Cuando está ejecutándose del lado del servidor,accede fácilmente acualquier tecnología Java. Las actividades del usuario, incluyendo las peticiones Ajax y la modificación de datos en el servidor (CRUD), se abstraendelos objetos de evento.

En la Figura 2.3 se muestra las 7 fases de una petición Ajax en el framework.

2.2.3 Capa de presentación

El Framework ZK está diseñado para ser lo más ligero posible, por este motivo el framework se focaliza más en la capa de presentación. La misma que no requiere de ninguna otra tecnología especifica back-end para funcionar correctamente. Aparte el framework deja al desarrollador la opción de elegir cualquier middleware como Java DataBase Connectivity (JDBC), Hibernate, Java Mail, Enterprise Java Beans (EJBs) o Spring entre otros. Todas estas tecnologías funcionan perfectamente con el framework. De esta manera se puede concluir que utilizando el framework ZK se construye aplicaciones dinámicas e interactivas en el cliente con la posibilidad de ser integrada con tecnologías familiares en el servidor.

2.2.4 Un framework dirigido por eventos

Comola mayoría de losframeworks Ajax, la función del servidores pasiva,ya quesólo es responsable del suministro yaceptación delos datosdespués de recibirlas peticiones del cliente. La comunicación entre los componenteses bastante compleja yrequiereuna gran cantidad deprogramación JavaScript, aparte existe un gran problema de incompatibilidad entreJavaScripty navegadores. Por el contrario,en la solución deZK, todos los componentesson creadosen el servidor, lo que hacemás fácilla comunicación entre ellosya que se puedeacceder a estos componentes directamenteen el servidor.

La forma en la que los componentes se comunicanentre sí,es por eventos, lo que significa quela interacciónpuede ser provocada porlas actividadesde un usuarioen el cliente oeventos enviados desdeotros componentes.

2.2.5 Fases de una petición Ajax

En la Figura 2.4, explicaré cuál es el flujo de una petición Ajax en el framework ZK.

A continuación se detallará cada etapa descrita en la Figura 2.4.

1. El usuario ejecuta un click sobre un botón que cambiará el título de algún componente en la pantalla.
2. El evento onClick de JavaScript se dispara.
3. El motor del cliente de ZK, atrapa este evento.
4. El motor de cliente de ZK, envia este comando al servidor a travez de un *XMLHttpRequest.*

5. En el lado del servidor, el motor de actualización asíncrona recibe el comando enviado desde el cliente, y a su vez envía el evento onClick al escucha de eventos registrado para dicho componente.
6. El listener o escucha actualiza el atributo título del componente, es decir ejecuta la acción programada en el evento onClick.
7. El lintener para el evento onClick notifica al motor de actualización asíncrona, que la ejecución del evento ha sido terminada.
8. El motor de actualizacion asíncrona prepara una respuesta Ajax y la envía al navegador por medio de unXMLHttpResponse.
9. El motor del cliente ZK, recibe la respuesta y actualiza el arbol de componentes DOM de la página.

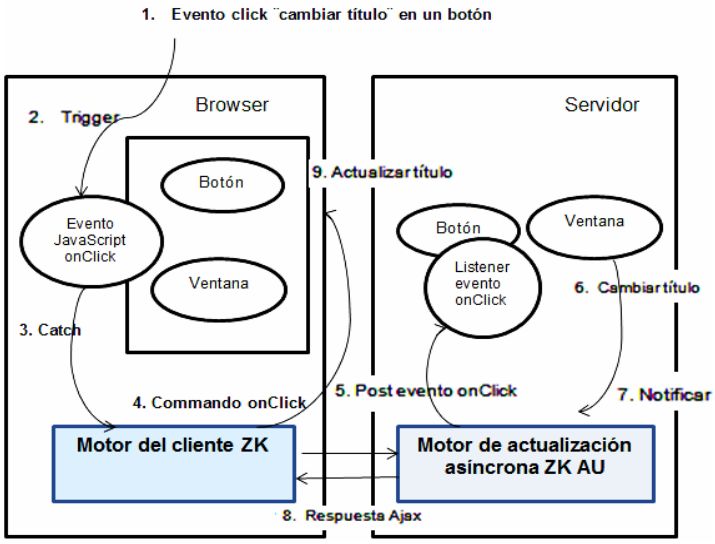


Figura 2.4: Flujo de una petición Ajax, en el framework ZK

Fuente: El autor.

2.3 Configuración del Framework ZK

2.3.1 Introducción a la configuración del Framework ZK

El framework ZK se ejecuta como un conjunto de servlets, dentro de un contenedor de servlets Java. El objetivo principal de la configuración del framework, es preparar el ambiente en donde las aplicaciones ZK se desarrollaran y ejecutarán. Para esto, se requiere la instalación de ciertos programas fundamentales para el funcionamiento del framework, esto incluye la instalación de Java Runtime Environment, la instalación de un contenedor de servlets Java y un servidor para desplegar los archivos .war (Se asume que el sistema operativo es superior a un Windows 2000).

2.3.2 Instalación de Java Runtime Environment

El Java Runtime Environment (JRE) es necesario para que pueda correr el framework y cualquier otra tecnología Java, ya que contiene las librerías del lenguaje en sí. Si no se ha instalado todavía cualquier JRE, se podría descargar desde la página oficial de Oracle (http://java.com/es/download/) e instalar en cualquier computadora de manera gratuita. Se sugiere instalar la última versión de JRE para el correcto funcionamiento del framework. Se sugiere instalar también una versión actual para el contenedor de servlets (por ejemplo Tomcat 6 o superior). Adicionalmente, para el desarrollo de aplicaciones Java es fundamental instalar el Java Development Kit (JDK). Este kit de desarrollo, ya incluye las librerías del JRE, también incluye un compilador y un depurador para código Java. Es fácilmente instalable en cualquier plataforma, ya sea Windows, Linux o Solaris entre otras.

Para instalar JDK se debe ir a la página oficial de Oracle (http://www.oracle.com) y elegir el ambiente del sistema operativo donde se desea instalar Java. Una vez elegido el ambiente se debe seguir las instrucciones de instalación.

Es recomendable, una vez instalado todo el ambiente Java, realizar un test online para determinar si se ha instalado correctamente las librerías. Esto se lo puede realizar en la siguiente dirección (http://www.java.com/es/download/installed.jsp)

2.3.3 Instalación del Contenedor de Servlets Java

El segundo paso en la configuración e instalación del framework ZK es el instalar un contenedor de servlets o servidor de aplicaciones si se quisiera. El framework deja a criterio del desarrollador con cual servidor de aplicaciones se desea trabajar. A continuación presentaré una guía de instalación para el servidor de servlets Apache Tomcat ya que es uno de los contenedores de servlets más estables y conocidos. Para la guía de instalación se utilizara la versión 6.0.23.

2.3.3.1 Instalación de Apache Tomcat utilizando el instalador

Se puede instalar Tomcat utilizando Windows Service Installer siguiendo los siguientes pasos:

- Descargar el Windows Service Installer (apache-tomcat-6.0.23.exe) desde (http://tomcat.apache.org/download-55.cgi#6.0.23).
- Una vez descargado elarchivo, haga dobleclic en el iconopara iniciar elprograma de instalación y siga las instrucciones enla pantallapara finalizar la instalación. Una cosa sedebe hacer esescribirel número de puertoHTTP(por defecto

es8080)de su configuración yel directoriodonde está instalado elTomcat($TOMCATmás adelante).Además, recuerdeel nombrey la contraseña deladministrador configuradoen la instalación deTomcat, ya que es posible quelos necesitepara la web implementación de aplicaciones

2.3.3.2 Instalación de Apache Tomcat descomprimiendo archivo .zip

Otra alternativa para instalar el servidor Apache Tomcat, es haciéndolo manualmente, es decir sin la ayuda de un asistente:

1. Descargar el archivo ZIP (apache-tomcat-6.0.23.zip) desde

(http://tomcat.apache.org/download-55.cgi#6.0.23).

2. Descomprimir el archivo en cualquier parte de su computador, es recomendable que se descomprima en una ruta fácil de acceder.

3. Ir a la ruta donde se instaló el JDK, en una ventana del explorador de Windows.

4. Desde Inicio Menú ➢Control Panel ➢Sistema, presionar en opciones avanzadas. Después presionar en el botón de Variables de Entorno; presionar el botón de Nuevo en la opción de variables de sistema; tipo JAVA_HOME y copiar la ruta donde se ha instalado el JDK. Por último aceptar los cambios.

2.3.3.3 Desplegar y probar el ejemplo zkdemo-all.war

En esta parte de la configuración, se probara que el entorno de desarrollo esté funcionando correctamente, para ello se probara levantando una aplicación de ejemplo que nos provee la página del framework gratuitamente y que contiene todas

las librerías (archivos JAR) referentes al framework y que son necesarias para que el ejemplo se despliegue correctamente. A continuación los pasos para levantar la aplicación:

Descargar la última versión del archivo demo de ZK (zk-demo-x.x.x.zip) desde la página (http://www.zkooss.org/download/).

Descomprimir el archivo descargado desde la página de ZK, y encontrar el archivo zkdemo-all.war

Correr el archivo zkdemo-all.war en Tomcat copiando el archivo dentro del directorio $TOMCAT/webapps/. El resto del trabajo, incluido la descompresión del archivo y la compilación del mismo lo manejara el servidor.

Activar el contenedor de servlets haciendo click en Inicio ➤ Programas ➤ Apache Tomcat ➤ Monitor Tomcat. Se puede observar un ícono de Tomcat en la bandeja de iconos. Se debe dar click derecho en el ícono y seleccionar Empezar Servicio. El cuadrado rojo en el ícono se pone verde cuando el servidor ha iniciado correctamente.

Para probar que se ha desplegado correctamente la aplicación de ejemplo, se debe abrir el navegador y visitar la siguiente dirección (http://localhost:8080/zkdemoall).

2.3.4 Librerías Relacionadas

El archivo zkdemo-all.war incluye todos los archivos JAR necesarios en el directorio WEB-INF/lib. Dentro del archivo .WAR. La siguiente lista es una introducción a cada archivo JAR y su función:

- bsh.jar: BeanShell interprete de código Java
- commons-el.jar: Implementación de Apache para interpretar el Lenguaje de Expresiones (EL).
- commons-fileupload.jar: Implementación de Apache para cargar archivos.
- commons-io.jar: Implementación de Apache para la transmisión de I/O (usado para la carga de archivos).
- dojoz.jar: Componentes relacionados Dojo Ajax.
- fckez.jar: Componentes relacionados con FCKeditor editor HTML.
- gmapsz.jar: Componentes relacionados con Google Maps.
- zcommon.jar: Librería común para ZK.
- zhtml.jar: Componentes relacionados con XHTML.
- zk.jar: Código kernel ZK.
- zkplus.jar: Código de integración con Acegi Security, Spring y Hibernate.
- zul.jar: Componentes relacionados con el lenguaje de marcado XUL.
- zweb.jar: Código relacionado con utilidades Web.

2.3.5 Configuración del archivo web.xml

El archivo web.xml se encuentra en la carpeta WEB-INF/ y describe cómo una aplicación web debe ser desplegada. Si se desea construir una aplicación con el framework ZK, se tiene que configurar dicho archivo correctamente, con el fin de que se relacionen bien los componentes del framework como los servlets, los oyentes (listeners), y los filtros.

El Código 2.1 muestra un ejemplo de un archivo web.xml:

```
<listener>
<description>Se utiliza para limpiar variables cuando se destruye una
sesion</description>
<display-name>ZK Session Cleaner</display-name>
<listener-class>org.zkoss.zk.ui.http.HttpSessionListener</listener-class>
</listener>
<servlet>
      <description>ZK loader for ZUML pages</description>
      <servlet-name>zkLoader</servlet-name>
      <servlet-class>org.zkoss.zk.ui.http.DHtmlLayoutServlet</servlet-
class>
      <!-- Debe especificar la URI del motor de actualizacion
(DHtmlUpdateServlet). Esta debe ser igual a <url-pattern> para el motor de
actualizacion.-->
      <init-param>
            <param-name>update-uri</param-name>
            <param-value>/zkau</param-value>
            </init-param>
      <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
      <servlet-name>zkLoader</servlet-name>
      <url-pattern>*.zul</url-pattern>
</servlet-mapping>
<servlet-mapping>
      <servlet-name>zkLoader</servlet-name>
      <url-pattern>*.zhtml</url-pattern>
</servlet-mapping>
<!-- Optional. For richlets. -->
<servlet-mapping>
      <servlet-name>zkLoader</servlet-name>
      <url-pattern>/zk/*</url-pattern>
</servlet-mapping>
<servlet>
      <description>El motor de actualizacion asíncrona del framework
ZK</description>
      <servlet-name>auEngine</servlet-name>
      <servlet-class>org.zkoss.zk.au.http.DHtmlUpdateServlet</servlet-
class>
</servlet>
<servlet-mapping>
      <servlet-name>auEngine</servlet-name>
      <url-pattern>/zkau/*</url-pattern>
</servlet-mapping>
```

Código 2.1: Programación del archivo web.xml en el framework ZK.

Fuente: El autor.

El archivo web.xml descrito anteriormente incluye las configuraciones de un oyente o escucha (<listener></listener>) y dos servlets importantes llamados DHtmlUpdateServlet y DHtmlLayoutServlet. El DHTML layout servlet es el encargado

del diseño y se lo llama zkLoader, es responsable de cargar una página ZULM y de la creación de componentes basados en definiciones de la página, cuando el contenedor de servlets recibe las solitudes enviadas desde el browser. El DHTML update servlet llamado auEngine, es responsable de manejar los eventos y peticiones, las cuales son Ajax XMLHttpRequest.

Es importante señalar que la update-uri del zkLoader debe ser acompañada por la url-pattern del auEngine, de lo contrario la aplicación no trabajara correctamente. Como complemento los desarrolladores pueden apoyarse en richlets para construir las aplicaciones web con Java puro. Un richlet es una pequeña porción de código que crea todos los componentes necesarios para responder una petición del usuario. Sin embargo, los richlets están creados para desarrolladores que tienen un amplio conocimiento del framework ZK. Para poder distinguir las páginas creadas con richlets de las creadas con lenguaje ZULM, se necesita definir un nuevo parámetro como URL para que el ZK loader pueda distinguir una petición hecha con un richlet en específico. En El Código 2.1, el ZK loader va a manejar todas las peticiones del usuario utilizando el URL del /zk/. Por otra parte, para especificar el richlet a utilizar para las peticiones de una URL determinada, es necesario configurarlo en el archivo zk.xml.

2.3.6 Configuración del archivo zk.xml

Para complementar el archivo web.xml, se debe configurar el archivo zk.xml, el mismo que nos permitirá personalizar nuestra aplicación ZK. El Código 2.2 muestra

cómo sería la configuración de un richlet para una página web que indique que la sesión ha caducado.

El Código 2.2 muestra un ejemplo de un archivo zk.xml:

```
<richlet>
<richlet-class>org.zkoss.zkdemo.test.TestRichlet</richlet-class>
<richlet-url>/test</richlet-url>
</richlet><!-- Opcional -->
<session-config>
<!—Un URL puede causar a la página que se vuelva a recargar -->
<timeout-uri>/timeout.zul</timeout-uri>
</session-config>
<!-- Opcional -->
<listener>
<description>Monitor de estadística </description>
<listener-class>org.zkoss.zk.ui.util.Statistic</listener-class>
</listener>
```

Código 2.2: Programación del archivo zk.xml

Fuente: El autor.

Los richlets deben trabajar en conjunto con el patrón de URL (url-pattern) definido para el richlet en el archivo web.xml, con el fin de determinar qué clase de richlet deben tramitar las peticiones del patrón URL especificado. En el ejemplo zkdemo-all.war, las peticiones de la dirección (http://localhost:8080/zkdemo-all/zk/test) serán manejadas por la URL especificada org.zkoss.zkdemo.test.TestRichlet clase.

El session-config define los atributos de la sesión de una aplicación web ZK. Por ejemplo el timeout-uri es un atributo que define la respuesta de una página web cuando una sesión ha agotado el tiempo de sesión. En el caso que no se configure este atributo, el navegador volverá a cargar la misma página.Un oyente o listener, se utiliza para configurar una opción de clase personalizada callback, que requiere el

uso de Java y por lo tanto deben ser implementadas por los desarrolladores. En este caso el monitor de estadística declarado.

2.4 Componentes del framework ZK

Una vez entendida la arquitectura del framework y realizada la configuración del mismo, es necesario explicar el funcionamiento y la composición básica de sus componentes y cómo se puede escribir páginas con dichos componentes.

2.4.1 Conceptos de componente

Un componente es un objeto de la interfaz de usuario (UI), tal como un botón, una etiqueta o un árbol. El componente define la representación gráfica y comportamiento lógico de un elemento en la interfaz de usuario.

Modificando los atributos del componente o manipulando su estructura, se puede controlar la presentación visual de una aplicación en el cliente. Todos los componentes de ZK implementan la interfaz *org.zkoss.zk.ui.*

Una página es una colección de componentes. Las páginas son miembros de la clase *org.zkoss.zk.ui.* La clase Page, y sus componentes contenidos en la página, serán mostrados en ciertas posiciones en el navegador.

Una página es creada automáticamente cuando el ZK loader interpreta una página ZULM. Un escritorio que hereda de la clase *org.zkoss.zk.ui.Desktop,* es una colección de páginas para servir a la misma solicitud URL.Se define entonces a un componente como la sinergia de dos mundos ya que además de ser un objeto de

Java en el servidor, un componente tiene una parte visual en el navegador embebido en las páginas. Es decir, cuando un componente se une a una página, su parte visual se crea en el navegador. Y cuando un componente se elimina de una página, la parte visual se retira.

2.4.2 El Ciclo de Vida de un Componente

El framework ZKsuponealgunas secuenciasde procesamiento,con respecto alas páginas decarga y actualización, que puedan afectar ala forma de escribirlas aplicacionesZK. En esta sección,se examinaráel ciclo de vidade los componentesZKtener una mejor comprensióndel mecanismodel framework ZK.

2.4.3 El Ciclo de Vida de una Página ZULM

El ZK loader carga una página ZULM en cuatro fases:

1. Inicialización de la Página.
2. Creación de los componentes.
3. Procesamiento de eventos.
4. Representación de los componentes.

2.4.3.1 Fase de Inicialización de la Página

Enesta fase, ZKejecuta las instruccionesde procesamiento inicial, llamadas *init*. Si no están definidas estas instrucciones, esta fasese pasa por alto. Para cadainstrucción de procesamientoinicial conun atributo de clase, se construyeuna instancia de laclase especificaday se llama al método *doInit ()*, por ejemplo, *<? claseinit*

*="MyInit"?>*específicaa la claseMyInitcomo iniciadorde la página, es decir lo que la página va a hacer dependiendo de los requerimientos de la misma.

Otra forma de definir estas instrucciones es especificar un archivo*zscript*, para ejemplo, <?initzscript= "/ mi/init.zs">. A continuación, el archivo zscriptserá interpretadaen lapágina Fase inicial.

2.4.3.2 Fase de creación de componentes

En esta fase, el gestor de ZK interpreta una página ZUML, y crea e inicializa los componentes de acuerdo a las especificaciones en la página. Los pasos en esta fase son los siguientes:

1. Para cada elemento, el gestor de ZK, examina si se tiene atributos y aplica la configuración en caso que estos atributos tengan valor. Si no, el componente y todos sus elementos secundarios son ignorados.
2. Si se especifica el atributo *forEach* con una colección de elementos, ZK realiza los pasos del tercero al séptimo para cada elemento de la colección.
3. El ZK loader crea un componente basado en el nombre del elemento o la clase especificada en el atributo de uso.
4. Los miembros de la clase del componente se inicializan uno por uno, en el orden que atributos se especifican en la página ZUML.
5. El ZK loader interpreta los elementos anidados y se repite todo el procedimiento.

6. El ZK loader invoca el método *afterCompose ()* si el componente implementa la interfaz *org.zkoss.zk.ui.ext.AfterCompose.*
7. Después de crear todos los subcomponentes o elementos anidados, se envía el evento *onCreate* con el fin de inicializar el contenido de los elementos adicionales. En el evento *onCreate,* los eventos son registrados por los componentes.

2.4.3.3 Fase de procesamiento de eventos

En esta fase, ZK llama un oyente para cada evento de la cola de eventos. Un hilo independiente se comienza a invocar para cada uno de los oyentes, por lo que pueden ser suspendidos sin afectar el procesamiento de otros eventos. Durante el proceso, un detector de eventos podría desactivar otros eventos.

2.4.3.4 Fase de representación

Después de que todos los eventos son procesados, ZK representa los componentes en una página HTML normal y la muestra en el navegador. Para mostrar un componente se llama el método *redraw ()*.

El método *redraw ()* no alterará el contenido del componente.

2.4.3.5 Actualización de las páginas

Se necesitan de tres fases para que el motor de actualización asíncrona llamado ZK AU procese las solicitudes enviadas desde los clientes ZK:

1. Procesamiento de solicitudes.
2. Procesamiento de eventos.
3. 3. Representación.

Las solicitudes para el mismo escritorio se procesan de forma secuencial. Las solicitudes de diferentesescritorios se procesan en paralelo.

2.4.3.6 Fase de Procesamiento de solicitudes

Dependiendo de las especificaciones de la petición, el motor de ZK AU puede actualizar el contenido de los componentes implicados en la solicitud. Después, envía los eventos correspondientes a la cola de eventos.

2.4.3.7 Fase de procesamiento de eventos

Esta fase es la misma que la fase de procesamiento de eventos de carga de una página ZUML. que los procesos deeventos, uno por uno y utiliza un hilo independiente para cada uno.

2.4.3.8 Fase de representación

Luego de que todos los eventos son procesados, ZK hace que los componentes sean actualizados, generalas respuestas correspondientes y envía estas respuestas al cliente. Una vez en el cliente, se actualiza el árbol de componentes de la página con las respuestas enviadas desde el servidor.

Se puede redibujar toda la representación o solo actualizarun atributo de un componente, eso depende de la lógica que se utilizó en el evento.

2.4.4 Recolección de Basura

A diferencia de muchos componentes basados en interfaces gráficas de usuario, ZK no tiene el método *destroy* () o *close* () para suscomponentes. Al igual que en la especificación*World Wide Web Consorcio* (W3C) DOM, uncomponente se elimina desde el navegador cuando ya no existe más en la página. Una vez que un componente se separa de una página, ya no es gestionado porZK, esto si la aplicación no mantiene ninguna referencia a ella. De esta manera, la memoria ocupada por el componentese eliminará por medio del *Garbage Collector* alojado en la Java Virtual Machine (JVM).

2.4.5 Atributos de los componentes

2.4.5.1 El atributo id

Para hacer referencia a un componente en el código Java o expresiones EL, se debe asignar un identificador utilizando el atributo id. En el Código2.3, un identificador de "etiqueta" se asigna a la etiqueta para que el motor de ZK pueda identificar este componente Label en el servidor.

```
<windowtitle="Vote"border="normal">
Te gusta ZK?
<labelid="label"/>
<separator/>
<buttonlabel="Yes"onClick="label.value = self.label"/>
<buttonlabel="No"onClick="label.value = self.label"/>
</window>
```

Código 2.3: Ejemplo del atributo id

Fuente: Henri Cheng & Robie Cheng ZK Ajax without JavaScript Framework, 2008

2.4.5.2 Los atributos if y unless

Estos atributos se utilizan para definir una declaración de verdadero-falso para determinar si se crea o no un componente. En el ejemplo mostrado en El Código2.4, una expresión EL se utiliza para determinar si el parámetro es verdadero o falso. Estas dos etiquetas se creará si elvalor de la votación es verdadera, de lo contrario, ninguno de ellos se crearán. Si ambos atributos se especifican, el componente no se creará si los dos atributos no son verdaderos.

```
<window><labelvalue="Vote 1"if="${param.vote}"/><labelvalue="Vote
2"unless="${!param.vote}"/></window>
```

Código 2.4: Ejemplo de atributo *if* y *unless*

Fuente: Henri Cheng & Robie Cheng ZK Ajax without JavaScript Framework, 2008

2.4.5.3 El atributo forEach

El atributo forEach se utiliza para determinar cuántos componentes se crearán. Si especifica una colección de objetos de este atributo, el gestor de ZK creaun componente decada elemento de la colección especificada. Por ejemplo, en la página siguiente se declara una matriz de contactos, dentro de una etiqueta <zscript> en el que puede escribir código Java.

```
<window>
<zscript>
contacts = new String[] {"Monday", "Tuesday", "Wednesday"};
</zscript>
```

```
<listboxwidth="100px">
<listitemlabel="${each}"forEach="${contacts}"/>
</listbox>
</window>
```

Código 2.5: Ejemplo de atributo *forEach*

Fuente: Henri Cheng & Robie Cheng ZK Ajax without JavaScript Framework, 2008

Al evaluar un elemento con el atributo forEach, objetos de la colección se les asigna el valor de la variable (contacts en El Código 2.4) uno por uno. Por lo tanto, la página anterior es la misma en los siguientes términos:

```
<listboxwidth="100px">
<listitemlabel="Monday"/>
<listitemlabel="Tuesday"/>
<listitemlabel="Wednesday"/>
</listbox>
```

Código 2.6: Ejemplo de componente ListBox.

Fuente: Henri Cheng & Robie Cheng ZK Ajax without JavaScript Framework, 2008

2.4.5.4 El atributo use

La inserción de código en las páginas de forma incorrecta, puede causar un mal funcionamiento de las mismas. Es mejor tener código de control de componentes o lógica de componentes,independiente del código de la interfaz de usuario. Hay dos maneras de separar el código, de las vistas.

La primera es poner esta lógica de control en una clase Java separada, y registrar los eventosa los oyentes del componente llamando a los métodos adecuados. Por ejemplo, se podría invocar los métodos *onCreate ()*, *OnOK ()*, y *onCancel ()*que deben estar definidos en la clase MyClass como se muestra en El Código 2.7:

```
<windowid=”main”onCreate=”MyClass.init(main)”
onOK=”MyClass.save(main)”onCancel=”MyClass.cancel(main)”/>
```

Código 2.7: Asociar un componente a una clase para utilizar sus métodos

Fuente: Henri Cheng & Robie Cheng ZK Ajax without JavaScript Framework, 2008

Para que esto funcione, se debe tener una clase Java llamada MyClass como se muestra en El Código 2.8.

```
publicclass MyClass {
publicstaticvoid init(Window main) {
//does initialization
}
publicstaticvoid save(Window main) {
//saves the result
}
publicstaticvoid cancel(Window main) {
//cancel anychanges}}
```

Código 2.8:. MyClass.java

Fuente: Henri Cheng & Robie Cheng ZK Ajax without JavaScript Framework, 2008

La otra opción para la separación del código de la vista, es el de asignar el atributo *use* para especificar una clase y sustituir a la clase de componentes por defecto:<window use="MyWindow"/>

En este caso, se debe tener una clase Java llamada MyWindow, como se muestra en El Código 2.9.

```
publicclass MyWindow extends Window {
publicvoid onCreate() {
//does initialization
}
publicvoid onOK() {
//save the result
}
publicvoid onCancel() {
//cancel any changes
}}
```

Código 2.9: MyWindow.java

Fuente: Henri Cheng & Robie Cheng ZK Ajax without JavaScript Framework, 2008

2.4.5.5 Espacios ID

Como se mencionó anteriormente, ZK permite hacer referencia a cada componente por su atributo id.

Sin embargo, una representación visual suele ser descompuesta en varias páginas ZUML; por ejemplo, se podría tener una página de una orden de compra y una página de diálogo modal para las condiciones de pago. Si todos los componentes son los únicos en el mismo escritorio, se tiene quemantener la diferentes id para los componente creados para el mismode escritorio, esto resulta un problema de conflicto de identificadores. ZK ha introducido elconcepto de espacio de identificación para resolver dicho problema.

Un espacio de identificación se compone de un propietario de espacio * y sus componentes semejantes.

El espaciopropietario es como una carpeta de archivos, y los componentes son como archivos dentro de la carpeta. En un sistema de archivos,

aquellos con el mismo nombre pueden coexistir si se ponen en carpetas de archivos diferentes. De esta manera, se puede asignar el mismo identificador ZUML para los componentes que residen en diferentes espacios de ID.

En otras palabras, la singularidad de los identificadores de los componentes sólo se garantiza en el ámbito de un espacio de identificación único.

Además de resolver el problema del conflicto de identificación, los espacios de ID, hacen que sea fácilreferencia a un componente dentro o fuera de un espacio de identificación, simplemente invocando el método *getFellow ()*

2.4.6 Expresiones de Lenguaje (EL)

Al igual que en las páginas JSP, se pueden usar expresiones EL en cualquier parte de las páginas ZUML,excepto en los nombres de los atributos, o elementos e instrucciones de procesamiento. ZUML utiliza la misma sintaxis para las EL, que las páginas JSP, es decir, $ {expr}. Como se muestra en El Código 2.10:

```
<windowtitle="ELTest"width="200px">
<zscript>
String abc = "ABC";
</zscript>
<buttonlabel="${abc}"/>
</window>
```

Código 2.10: Ejemplo de EL.

Fuente: Henri Cheng & Robie Cheng ZK Ajax without JavaScript Framework, 2008

Cuando una expresión EL se utiliza como el valor de un atributo, puede devolver cualquier tipo de objeto, siempre que el componente acepte ese objeto como valor de entrada. Por ejemplo, la siguiente expresiónserá evaluada como un objeto Boolean: <windowif="${some> 10} ">

2.4.7 Eventos

Un evento es de la clase org.zkoss.zk.ui.event.Event y se utiliza para notificar a una aplicación de una acción en el navegador. Cada tipo de evento está representado por una clase distinta. Por ejemplo,

org.zkoss.zk.ui.event.MouseEvent denota una actividad del ratón, como hacer clic.

Para responder a un evento, una aplicación debe registrar uno o más detectores alevento. Hay tres formas de registrar un detector de eventos para un componente. Una de ellas es especificando el detector de eventos OnXxx como atributo del componente, donde XXX representa elnombre de la acción, como muestra El Código 2.11.

```
<windowtitle="Hello"border="normal">
<buttonlabel="Say Hello"onClick="alert(&quot;Hello World!&quot;)"/>
</window>
```

Código 2.11: Controlador de eventos con un atributo definido

Fuente: Henri Cheng & Robie Cheng ZK Ajax without JavaScript Framework, 2008

En El Código2.11, onClick es el evento que se activa con el objeto botón. El manejo de eventoscódigo para el evento onClick es *alert* ("Hola Mundo!").

La segunda manera de registrar un detector de evento es mediante la definición de la OnXxx () en laclase del componente asociado.

```
<windowtitle="Hello"border="normal">
<zscript>
classMyButton extends Button {
public void onClick(MouseEvent event) {
Messagebox.show("Hello World!");
}
}
</zscript>
<buttonlabel="Say Hello"use="MyButton"/>
</window>
```

Código 2.12: Controlador de eventos con un método definido

Fuente: Henri Cheng & Robie Cheng ZK Ajax without JavaScript Framework, 2008

Llamar al método *addEventListener* () para el componente que deseaescuchar, como muestra El Código2.13

```
<windowtitle="Hello"border="normal">
<buttonid="hellobtn"label="Say Hello"/>
<zscript>
hellobtn.addEventListener("onClick", new EventListener() {
public void onEvent(Event event) {
Messagebox.show("Hello World!");
}
});
</zscript>
</window>
```

Código 2.13: Controlador de eventos.

Fuente: Henri Cheng & Robie Cheng ZK Ajax without JavaScript Framework, 2008

Además de los eventos desencadenados por la actividad del usuario en el navegador, una aplicación puede dispararеventos mediante el uso del método *sendEvent ()*

2.4.8 Espacios de nombres en XML y ZUML (*namespaces*)

Hay que recordar que ZUML es un lenguaje basado en XML, utilizado para describir la representación visual de una página. ZUML divide de la dependencia del conjunto de componentes para su uso. En otras palabras,

diferentes tipos de componentes, tales como XUL y XHTML, pueden ser utilizados simultáneamente enla misma página ZUML, y otros lenguajes de marcado se pueden añadir de forma transparente. Sin embargo, sidos o más tipos de componentes se utilizan en la misma página, se tiene que utilizar espacios de nombres para distinguirlos. En El Código2.14 se muestra un ejemplo de la mezcla de dos tipos decomponentes.

```
<windowtitle="mix HTML demo"
xmlns:h="http://www.w3.org/1999/xhtml"
xmlns:x="http://www.zkoss.org/2005/zul"
xmlns:zk="http://www.zkoss.org/2005/zk">
<h:tableborder="1">
<h:tr>
<h:td>
column 1
</h:td>
<h:td>
<listboxid="list"mold="select">
<listitemlabel="AA"/>
<listitemlabel="BB"/>
</listbox>
</h:td>
<h:td>
<h:ulid="ul">
<h:li>The first item.</h:li>
<h:li>The second item.</h:li>
</h:ul>
<h:inputtype="button"value="ZHTML button Add Item"
zk:onClick="addItem()"/>
<x:buttonlabel="ZUL button Add Item"onClick="addItem()"/>
<zscript>
importorg.zkoss.zhtml.Li;
importorg.zkoss.zhtml.Text;
voidaddItem() {
Li li = new Li();
li.setParent(ul);
new Text("Item "+ul.getChildren().size()).setParent(li);
}
</zscript>
</h:td>
</h:tr>
</h:table>
</window>
```

Código 2.14: Mezcla de componentes ZUL y ZHTML

Fuente: Henri Cheng & Robie Cheng ZK Ajax without JavaScript Framework, 2008

2.4.9 Creación de cuadros de diálogo modal

ZK soporta diálogos modales, el procesamiento de la aplicación de eventos en hilo se suspende yespera una acción del usuario para continuar. Esta característica simplifica enormemente la programación weben ZK. Para simplificar aún más

algunas tareas. Ejemplos de cuadros de dialogo en el framework zk, son la creación de componentes *messagebox* y el componente *FileUpload.*

2.4.10 Dojo

La librería de JavaScript Dojo, es un elemento importante en las implementaciones de frameworks basados en Ajax. El Código 2.15 muestra una de las aplicaciones más famosas de Dojo, una lista fish-eye que muestra una lista de imágenes,con zoom en la imagen por la que se pase el mouse.

```
<zk>
<fisheyelistid="fi"style="position:absolute;margin:20px"?
attachEdge="top">
<fisheyeitemimage="/img/icon_browser.png"
label="Web Browser"onClick="alert(self.label)"/>
<fisheyeitemimage="/img/icon_calendar.png"
<fisheyeitemimage="/img/icon_users.png"
label="Users"onClick="alert(self.label)"/>
</fisheyelist>
<divheight="200px"width="100%"style="border:1px solid black;"/>
</zk>
```

Código 2.15: Un ejemplo del uso de Dojo para crear una lista de ojo de pescado

Fuente: Henri Cheng & Robie Cheng ZK Ajax without JavaScript Framework, 2008.

CAPÍTULO 3: APLICACIÓN DE LA METODOLOGÍA

3.1 Guía de desarrollo y programación, utilizando el framework ZK.

3.1.1 Introducción al desarrollo de aplicaciones con el framework ZK.

Una vez expuestos los conceptos más importantes del framework ZK en el Capítulo 2, es necesario ahora tener una guía referencial para desarrollar aplicaciones reales con dicho framework.

Es necesario tener en cuenta que el framework ZK, cubre las necesidades de desarrollo de la capa web en un sistema que adopte la especificación JEE, por lo tanto no se pretende explicar el desarrollo de los otros componentes de esta especificación

En esta guía se implementó desarrollando ejemplos de aplicaciones desarrolladas con el framework ZK, cada implementación tendrá funcionalidades específicas que ayudarán el aprendizaje del framework ZK.

Esta guía contiene también código, útil para el desarrollo de aplicaciones con el framework ZK, el cual puede ser copiado para el uso en diferentes aplicaciones que requieran de necesidades básicas en una aplicación como lo es las operaciones CRUD (crear, leer, actualizar, eliminar) sobre los datos de una base de datos.

3.1.1.1 Objetivo de la guía de desarrollo del framework ZK.

Exponer la suficiente información acerca del framework ZK, para que un programador no experto, pueda realizar una aplicación utilizando dicho framework,

basándose en los ejemplos y consejos del uso de tecnología expuestos en esta guía.

3.1.1.2 A quién está dirigida esta guía

A cualquier desarrollador que desee ampliar sus conocimientos en el diseño de aplicaciones web ágiles con el lenguaje Java.

Esta guía no se limita a personas que no tengan un amplio conocimiento de programación, por lo contrarió está dirigida a programadores no expertos, que desean explorar nuevas tecnologías y aprender de una manera rápida y dinámica un nuevo framework para el desarrollo de aplicaciones web dentro del lenguaje Java.

3.1.1.3 Perspectiva de desarrollo en el framework ZK.

Al ser el framework ZK un framework que corre del lado del servidor. El framework puede acceder a cualquier recurso que resida también en él.Se puede construir la interfaz de usuario con los componentes que brinda el framework, puede interactuar con la actividad del usuario y luego manipular los componentes para actualizar las páginas de acuerdo a las peticiones o actividades que el usuario haga en la página. Todo se lleva a cabo en el servidor. La sincronización de los estados de los elementos comprendidos entre el navegador y el servidor se realiza automáticamente por el framework y es transparente para la aplicación.

La interfaz de usuario está compuesta por componentes que se serializan como objetos POJO (Plain Old Java Object). Este es el enfoque más productivo para desarrollar una aplicación web moderna.

La arquitectura del framework ZK permite al desarrollador la optimización del diseño de la interfaz de usuario, introduciendo nuevos conceptos como lo son el ZSCRIPT, que es código embebido en las páginas para agilizar la funcionalidad de los componentes y el cual es muy útil para crear prototipos en muy corto tiempo. Además el desarrollador tendrá completa libertad de manipular los eventos del cliente, el efecto visual y la personalización en sí de la interfaz de usuario.

3.1.1.4 Perspectiva del desarrollo de componentes en el framework ZK.

Cada objeto de la interfaz de usuario de ZK consiste en un componente y un widget. Un componente, es un objeto Java que se ejecuta en el servidor y que representa a un objeto en la interfaz de usuario el mismo que puede ser manipulado por una aplicación Java. Un componente tiene todo el comportamiento de un objeto en la interfaz de usuario, excepto que no tiene parte visual. Un widget es un objeto de JavaScript. Se ejecuta en el cliente. Este objeto representa al objeto que interactúa con el usuario.

La relación entre un componente y un widget es uno a uno. Sin embargo, si un componente no está conectado a una página, no tendrá el widget correspondiente en el cliente. Esto quiere decir que la aplicación puede crear widgets directamente en el cliente, sin un componente correspondiente.La forma cómo se sincroniza los estados y se distribuye la carga,depende del motor del cliente ZK y del motor de actualizacion asíncrona ZK.

Estos dos componentes internos del framework, trabajan juntos para proporcionar un canal transparente y robusto que simplifica la implementación de las aplicaciones.

3.1.2 Directrices de tecnologías usadas en el framework ZK.

Estos son consejos en cuanto a la implementación de las tecnologías en el framework ZK, para ayudar a los desarrolladores a tomar decisiones que agilicen y optimicen la realización de sus aplicaciones.

3.1.2.1 MVC vs. Zscript

Estas dos tecnologías pueden ser adoptadas para desarrollar aplicaciones con el framework ZK, la idea principal de adoptar una de ellas, es guiarse en el desarrollo con un patrón de diseño que facilite la implementación de una aplicación. Ambas tecnologías tienen sus beneficios pero persiguen objetivos diferentes. Es común que se pueda confundir cuando se adopten una u otra. Ambas pueden coexistir en una aplicación que tenga el framework ZK, el éxito está en saber cuándo es mas conveniente usar una u otra.

3.1.2.1.1 Cuándo se debe usar MVC?

La arquitectura MVC (Modelo Vista Controlador) es un patrón arquitectónico que aísla el modelo de datos, las vistas en las que se presentan los mismos y la lógica con la que se manipulan los datos en las vistas. El patrón MVC es fácil de implementar y mantener por lo que es muy recomendado cuando se realiza aplicaciones grandes. El patrón MVC es la mejor práctica en el desarrollo de aplicaciones, especialmente para los sistemas que correrán en un ambiente de producción.

3.1.2.1.2 Cuándo se debe usar Zscript?

Zscript permite agregar código Java en las páginas ZULM, el mismo que se encarga de realizar la lógica en las páginas, es decir, ejecutan métodos que son llamados desde los componentes. Haciendo una analogía con el patrón MVC, es como si se uniera la vista y el controlador en un solo sitio. Esta tecnología acelera el ciclo de diseño, por lo que se recomienda usarla para la creación de prototipos y pruebas POC. Zscript también es bueno para la explotación de los componentes ZK y para detectar errores en el framework.

Sin embargo al tener que compilarse el código introducido en las páginas, el rendimiento no es bueno, y es propenso a errores. Por esta razón, no se sugiere utilizar Zscript en sistemas que van a salir a un ambiente de producción.

3.1.2.2 MVC Extractor.

ZK Studio proporciona una herramienta llamada MVC Extractor que se encarga de convertir el código ingresado mediante Zscript al patrón de diseño MVC automáticamente. Simplifica la transferencia del código en los prototipos hasta la salida de la aplicación a un ambiente de producción.

3.1.2.3 Data Binding.

3.1.2.3.1 Cuándo usar Data Binding?

Data Binding consiste en enlazar los datos del modelo de una aplicación y los componentes de la interfaz, automáticamente. Es muy recomendable utilizar el

enlace de datos siempre que sea aplicable, ya que puede ayudar a aumentar la productividad del programador y el código es fácil de leer y mantener.

Sin embargo, esta práctica requiere un conocimiento de EL (Expresiones de Lenguaje). Se debe tener en cuenta que hay una limitación: el enlace de datos no se aplica a los componentes que se crean después de que la página es inicializada y presentadas en el cliente, es decir no se hace enlazan los datos con los componentes que se crean dinámicamente en tiempo de ejecución.

3.1.2.4 ZULM vs. Richlet vs. JSP

3.1.2.4.1 Cuándo usar ZULM?

Para la codificación de las páginas web, el framework ZK presenta como principal opción su lenguaje de marcado ZULM, diseñado específicamente para agilizar la construcción de las páginas ya que está basado en XML. Este lenguaje no requiere un conocimiento muy avanzado de programación y funciona muy bien con el patrón de diseño MVC.

ZULM se recomienda usar como estándar para el diseño de las páginas web, a menos que el desarrollador tenga que hacer integración entre tecnologías, por ejemplo si se quisiera utilizar ZK con JSF y tenga más experiencia manejando otros lenguajes de marcado como XHTML o JSP.

Sin embargo, si la mayor parte de una página está escrita en HTML y JavaScript y el programador no está familiarizado con ZULM ni XML, se puede usar JSP para definir la página y después incluir el código ZULM en las partes que se requiera utilizar los componentes del framework ZK.

Hay que tener en cuenta que el uso de ZULM, no impide la creación dinámica de componentes en Java. De hecho es una práctica común el usar ZULM para la codificación de las páginas y usar Java puro para la manipular los componentes escritos en estas páginas.

3.1.2.4.2 Cuándo utilizar Richlet?

Un richlet es un pequeño programa de Java que componeuna interfaz de usuario para servir una solicitud. Se recomienda utilizar richlets si el programador tiene experiencia previa en programación gráfica Swing de Java.

3.1.2.4.3 Cuándo utilizar JSP?

Si el programador desea utilizar el framework ZK en páginas JSP, puede utilizar uno de los siguientes métodos:

1. Se puede utilizar <jsp:include> para incluir una página ZULM.
2. Otra alternativa es incluir los Tags[12]del framework agregando la dirección (http:/ / www. zkoss.org/product/zkjsp.dsp) en la página JSP directamente. Como se describió anteriormente, si toda la página está escrita en HTML y el diseñador no está familiarizado con el lenguaje ZULM, se puede usar JSP para diseñar la página y solo incluir ZULM para agregar los componentes que se necesiten.

[12] http://es.wikipedia.org/wiki/TAG

3.1.2.5 Bookmarks vs Multiple Pages

La mayoría de los frameworks para el desarrollo web en Java, establecen la navegación del sistema distribuyendo el mismo en páginas individuales. El framework ZK permite a los desarrolladores agrupar un conjunto de páginas según la funcionalidad de las mismas. Esto es lo que ZK llama Bookmarks o Desktop, páginas que están agrupadas para interactuar con el usuario de una manera más amigable. Se puede usar Bookmarks para cada módulo de un sistema.

La agrupación de páginas se la debe hacer de acuerdo a la funcionalidad de las mismas, a menos que sea una aplicación pequeña. Por ejemplo, no es una buena idea agrupar las páginas de registro del sistema o con las páginas de administración de datos o las que realizan procesos transaccionales en el sistema. A continuación se describirán algunos consejos para la agrupación de páginas en el framework ZK:

1. Si se tiene un conjunto de funcionalidades que trabajan sobre solo una unidad lógica, por ejemplo sobre solo una tabla en la base de datos, cada funcionalidad podría realizarse en una página individual y después este conjunto de páginas deberían estar agrupadas en una sola página.
2. Si se tiene que realizar una búsqueda en el sistema, es decir se debe indexar información por algún motor de búsqueda, es mejor dividir las páginas y activar la opción de rastrear.

No importa si en la interfaz de usuario se comparte la misma plantilla. La forma como se agrupan las páginas es utilizando templates o plantillas, el tag *include*, sirve

para incluir una página dentro de otra, es decir extender la página con código distribuido en otras páginas.

3.1.2.6 Namespaces nativos vs Componentes XHTML

En un documento ZUML, un espacio de nombres XML se utiliza para identificar cualquier funcionalidad en especial o un conjunto de componentes. Un namespace o espacio de nombres, es un contenedor abstracto creado para albergar una agrupación lógica de identificadores únicos. Dentro de una página ZULM al igual que en un documento XML, los namespaces se agregan de la misma forma, por lo que es muy sencillo agregar o quitar los namespaces de XHTML dentro de una página ZULM.

3.1.2.6.1 Cuándo usar namespaces nativos

Cuando se utiliza elementos XML, llamados también namespaces nativos, estos se pueden declarar en cualquier etiqueta ZUML, siempre y cuando sea una estructura HTML válida, es decir que la página este bien armada. Se sugiere utilizar esta tecnología si las etiquetas HTML son estáticas

3.1.2.6.2 Cuándo usar componentes XHTML?

El framework ZK proporciona también un conjunto de componentes para representar a cada etiqueta XHTML en el servidor. A diferencia de los namespaces nativos, estos componentes están construidos por el framework y alojados en el servidor. Se sugiere utilizarlos para mostrar información de forma dinámica, ya que se comportan

igual que los componentes ZK. Sin embargo, hay que tener consideración que al crearlos de forma dinámica, ya que estos se almacenan en la memoria del servidor, lo que puede afectar al rendimiento de las páginas.

3.1.2.7 Include, Macro, Composite y Templating

Las plantillas o *templates*, los componentes por composición, componentes macro y el tag *include*, son características que permiten al desarrollador optimizar la construcción de la interfaz de usuario, separándola a esta en módulos. Consiste en la reutilización de código estándar en las páginas, para hacerlas más escalables y fáciles de entender.

3.1.2.7.1 Cuándo usar include?

Es un atributo que permite incluir una página ZULM dentro de otra, así como también se puede incluir páginas estáticas, una página JSP o el resultado de un servlet.La limitación de usar el atributo *include*, es que no se puede encapsular su comportamiento en una clase Java.

3.1.2.7.2 Cuándo usar componentes macro?

Los componentes macro permiten a los desarrolladores definir un nuevo componente con una página ZUML. Se sugiere utilizar estos componentes, si se desea reutilizar una página ZUML en distintas páginas. Estas son algunas causas por las cuales se debería utilizar componentes macro:

1. Se puede encapsular el comportamiento del componentede los componentes en una clase Java.
2. No hay ninguna diferencia entre el uso de un componente macro y otros componentes del framework.

3.1.2.7.3 Cuándo usar componentes por composición?

Un componente por composición[13] es otra forma de definir un nuevo componente en el framework ZK, con la diferencia que puede ser personalizado por el desarrollador y se lo hace casi siempre para ser utilizado de forma genérica. Con este enfoque, el desarrollador puede extender un nuevo componente de otro ya existente. Sin embargo, se debe implementar una clase Java que represente al nuevo componente[14]. A diferencia de los componentes macro, se tiene que manejar la creación de los componentes por composición manualmente. Es aconsejable utilizar componentes por composición cuando se quiera determinar un estándar de recursos gráficos en la pantalla, estos recursos podrían ser componentes como botones, ventanas, cajas de texto, que serían componentes por composición definidos para extender su funcionalidad y personalizar sus estilos.

3.1.2.7.4 Cuándo usar Plantillas o Templating?

El concepto principal de una plantilla o template, es el permitir crear un modelo reutilizable de página. Esto permite al diseñador disminuir código y tiempo en el desarrollo. Es muy recomendable el uso de plantillas, incluso si no se tiene un

[13] Es un patrón particular que extiende uno o varios componentes existentes de una tecnología.
[14] Ejemplo de un componente por composición (http:/ / www. zkoss. org/ zkdemo/ composite/ composite_component)

amplio conocimiento de diseño de páginas web. Se considera una buena práctica la reutilización de código web, ya que ahorra tiempo, facilita el diseño, brinda un formato a las páginas, y hace más fácil la detección de errores y el mantenimiento del sistema.

3.1.3 Preparación del entorno de desarrollo

3.1.3.1 Instalación de Eclipse

Se ha elegido éste IDE de desarrollo, ya que al igual que STS (Springsource Tool Suite) tiene gran facilidad de integración con el framework ZK y a su vez con el conjunto de librerías de ZK para el desarrollo de aplicaciones llamado ZK Studio.

Se recomienda instalar Eclipse después de configurar JRE y el servidor Tomcat, o el servidor de aplicaciones que se elija. Este IDE tiene muchas herramientas útiles para el desarrollo de aplicaciones. Entre los entornos de desarrollo Java, Eclipse es el más popular en la actualidad. Es compatible con una gran cantidad de plug-ins, sobre todo para WTP (WebTools Platform). Está diseñado para el desarrollo de aplicaciones JEE e incluye un editor de código fuente para XML, HTML, JSP y otros proyectos JEE.

A continuación se enumeraran los pasos para la instalación del IDE Eclipse, es importante tomar en cuenta que la versión que se descargará, debe ser compatible con la versión actual de ZK Studio. Esto se puede verificar ingresando a la siguiente página (http://www.zkoss.org/download/zkstudio) y comprobando si la versión de Eclipse que se está descargando tiene soporte para las librerías de ZK Studio.

Instalación del IDE Eclipse:

1. Ingresar a la página de descargas que proporciona el IDE (http://www.eclipse.org/downloads/), como muestra la Figura 3.1.

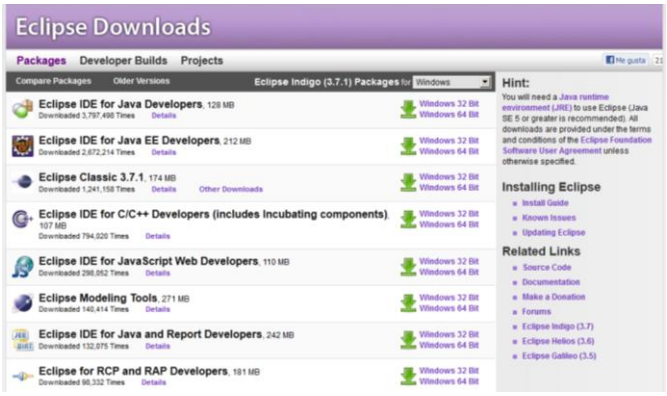


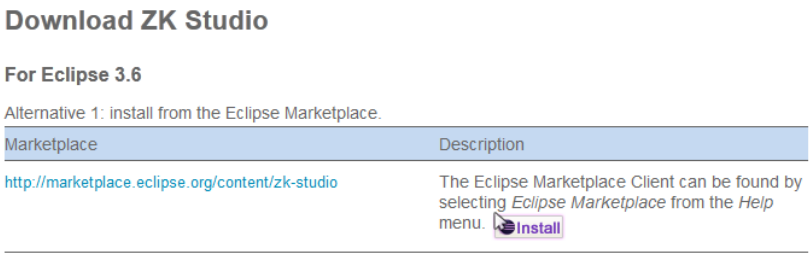
Figura 3.1: Opciones de descarga que proporciona la página de Eclipse.

Fuente: http://www.eclipse.org/downloads/.

2. Elegir la opción de Eclipse IDE for JEE Developers y descargar.
3. Una vez descargado, descomprir el archivo en un directorio fácil de acceder en la computadora.

3.1.3.2 Instalación de ZK Studio.

ZK Studio se puede instalar de varias formas tal como muestra la Figura 3.2:



Fuente: http://www.zkoss.org/download/zkstudio.

Figura 3.2: Opciones de instalación de las librerías de ZK Studio.

La primera opción es instalar desde Eclipse Marketplace que es como un centro de distribución de plug-ins para Eclipse. Si se elige esta opción, se deben seguir los siguientes pasos:

1. Ir a la barra de herramientas de Eclipse y elegir Help.
2. Elegir la última opción llamada Eclipse Marketplace.
3. Una vez ahí debe aparecer una ventana como muestra la Figura 3.3:

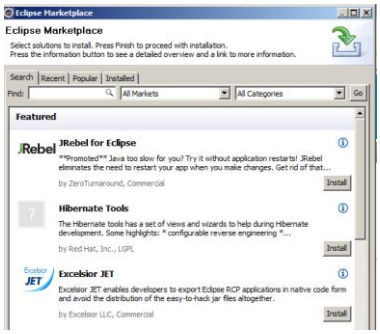


Figura 3.3: Ventana de Eclipse Marketplace.

Fuente: El autor.

4. En la ventana de Eclipse Marketplace buscar por el framework ZK y seleccionar la opción de instalar.
5. Completar la instalación aceptando los términos de la licencia.
6. Reiniciar Eclipse.

La segunda opción es instalar desde el sitio web de descarga que brinda el framework en la página (http://www.zkoss.org/download/zkstudio). Si se elige esta opción, se deben seguir los siguientes pasos:

1. En la barra de herramientas de Eclipse ingresar a Help, de ahí ingresar a la opción Install New Software. Deberá aparecer una ventana tal como muestra la Figura 3.4.

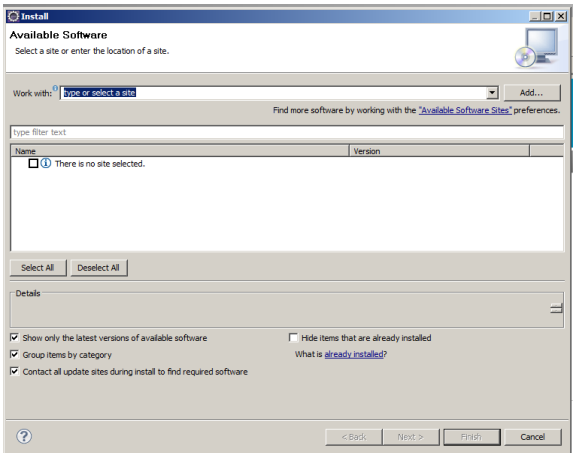


Figura 3.4: Ventana de descargas de Eclipse

Fuente: El autor.

2. Elegir Agregar. Esta opción brinda la posibilidad de agregar una nueva fuente de descarga donde el IDE buscará plug-ins disponibles para descargar e instalar.
3. Dar un nombre a la fuente que se a agrega y describe la dirección que se encuentra en la página del framework, para este caso es la dirección para Eclipse Helios o Eclipse 3.6 : (http://studioupdate.zkoss.org/studio/update/eclipse_3_6)
4. Una vez agregada la fuente, elegir todos las herramientas que se desea agregar e instalar, nos debe salir una pantalla tal como se muestra en la Figura 3.5:

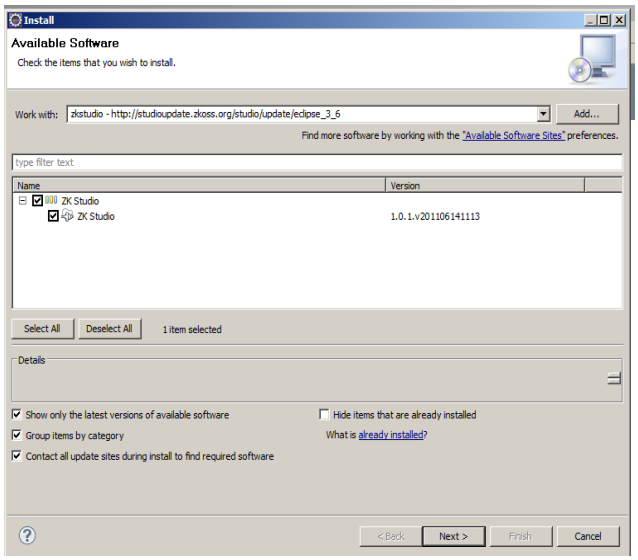


Figura 3.5: Ventana de instalación de descargas de Eclipse

Fuente: El autor.

5. Presionar Next las siguientes 2 veces y se completará la instalación. Una vez finalizada la descarga se debe reiniciar Eclipse y se podrá apreciar que la ventana de cambia por la ventana de bienvenida de ZK Studio, como se muestra en la Figura 3.6.

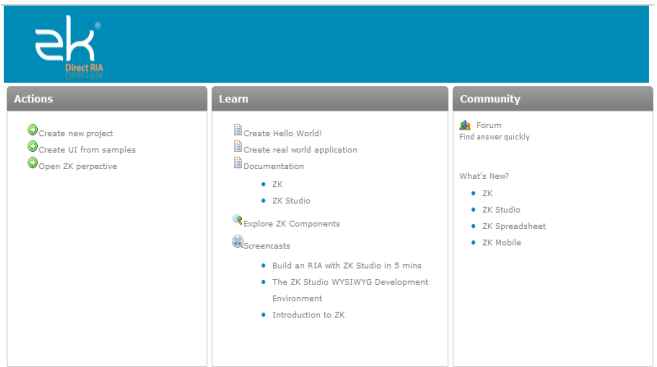


Figura 3.6: Ventana de bienvenida del framework ZK dentro de Eclipse

Fuente: El autor.

3.1.3.3 Creación de un proyecto ZK en Eclipse

Una vez ya instaladas todas las herramientas necesarias, se debe crear un nuevo proyecto bajo el framework ZK y comenzar a desarrollar una aplicación web. En las secciones siguientes, se explicará cómo desarrollar una aplicación web con el framework ZK, y cómo desplegar la aplicación en un servidor de aplicaciones para verla en un navegador web.

Un nuevo proyecto ZK se puede crear de dos formas en Eclipse. La primera creando directamente un proyecto tipo ZK. Esto se logra siguiendo los siguientes pasos:

1. En el menú de Eclipse, seleccionar File ➤ New ➤ Project. Ahí se elige el tipo de proyecto ZK en la carpeta que dice ZK, tal como muestra la Figura 3.7.

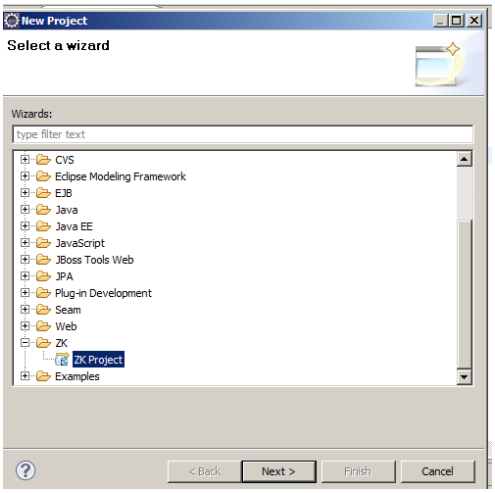


Figura 3.7: Ventana de selección de proyecto ZK.

Fuente: El autor.

2. Presionar siguiente y le dar un nombre al proyecto, en la ventana siguiente se puede ver que se muestra también la versión del framework con la que se trabajará y también la versión de servlet con la se va a construir el proyecto.
3. Elegir siguiente las siguientes dos veces y luego aplastar finalizar.

Crear un proyecto web dinámico en Eclipse. Para lograr esto se debe seguir los siguientes pasos:

1. Descargar las librerías del framework ZK en su distribución binaria (zk-x.y.z.zip) desde la siguiente dirección (http://www.zkoss.org/download/ (x.y.z es el numero de versión del framework)
2. En el menú de Eclipse, seleccionar File ➤ New ➤ Project. Ahí se podrá ver una nueva ventana de selección de tipos de proyectos.
3. En la ventana de selección de proyectos, elegir Web ➤Dynamic Web Project, y dar click en el botón Next, tal como se muestra en la Figura 3.8.

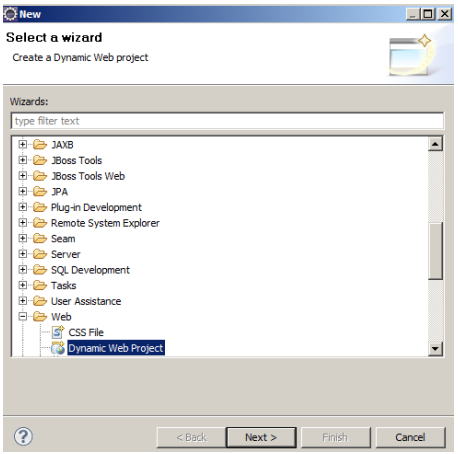


Figura 3.8: Ventana de selección de proyectos web.

Fuente: El autor.

4. Se debe poner un nombre al nuevo proyecto, es recomendar seguir un estándar para ponen los nombres a los proyectos.

5. Importar las librerías ZK y configurar los archivos web del proyecto, para esto es necesario dar click derecho sobre el proyecto creado, elegir la opción Java Build Path, a continuación se elige la opción Add External JARs, tal como se muestra en la Figura 3.9.

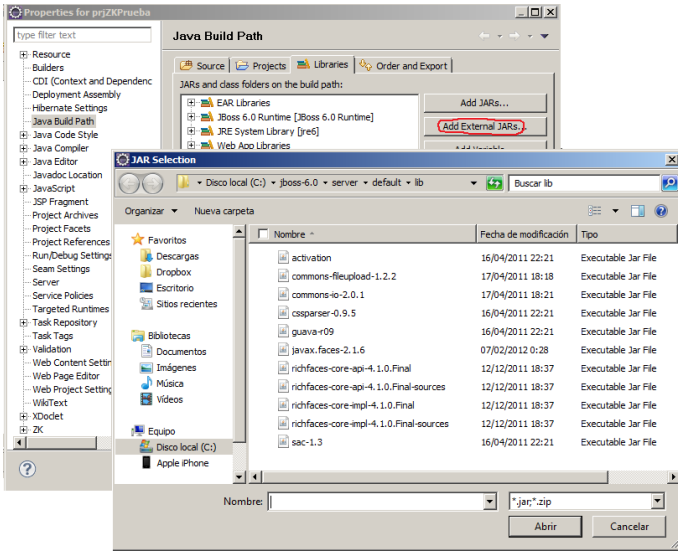


Figura 3.9: Ventana para agregar librerías externas a un proyecto web.

Fuente: El autor.

6. Agregar todas las librerías de la siguientes carpetas:

    - zk-x.y.z/dist/lib/*.jar
    - zk-x.y.z/dist/lib/ext/*.jar
    - zk-x.y.z/dist/lib/zkforge/*.jar

7. Copiar zk-x.y.z/demo/src/zkdemo/WebContent/WEB-INF/web.xml en la carpeta /WebContent/WEB-INF/ del proyecto y sobrescribir el archivo original web.xml.
8. Copiar zk-x.y.z/demo/src/zkdemo/WebContent/WEB-INF/zk.xml en la carpeta WebContent/WEB-INF/zk.xml.
9. Hacerclick derechoen la carpeta WebContent del proyecto, para mostrarel menú de contexto. Actualizar para que Eclipse asocie los archivos como se muestra en la Figura 3.10.

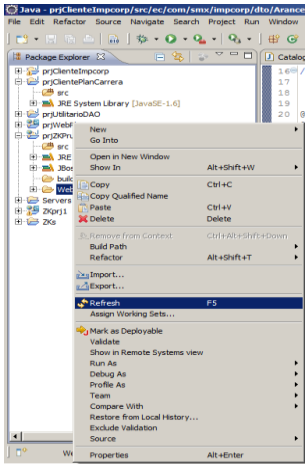


Figura 3.10: Ventana para actualizar un proyecto en Eclipse.

Fuente: El autor.

3.1.4 Programación bajo el framework ZK

La siguiente parte de la guía, es una referencia de cómo se pueden programar las páginas y los archivos Java utilizando el framework ZK, para lo cual se pondrán como ejemplo pequeños programas que ilustran una manera óptima y ágil en el desarrollo de aplicaciones.

3.1.4.1 Creación de páginas

Después de la implementación de las clases del modelo (que no está en el alcance de esta tesis) es momento de comenzar con el desarrollo de las páginas ZUL.

Es necesario aclarar, que la configuración del proyecto ZK depende mucho de como fue creado dicho proyecto. Si se creó utilizando un proyecto nuevo de ZK Studio como se muestra en la Figura 3.7, ya no es necesario configurar los archivos web.xml y zk.xml ya que el asistente los crea y configura automáticamente, además se crea una página ZUL llamada *index.zul* y *timeout.zul*.

Para crear páginas ZUL se debe dar click derecho sobre la carpeta en la que se desea incluir la página y después se elige: New ➤Other, desplegar la carpeta que dice ZK y una vez ahí escoger la opción ZUL, tal como se muestra en la Figura 3.11:

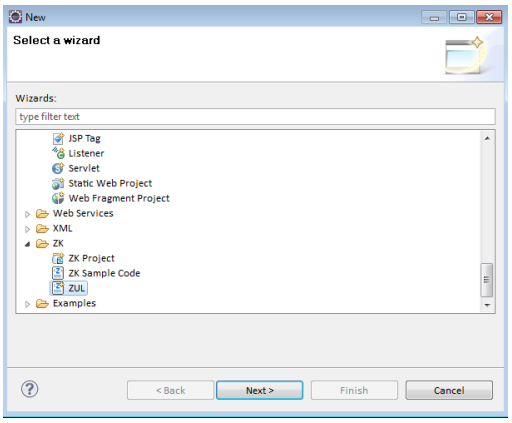


Figura 3.11: Ventana para agregar una nueva página ZUL.

Fuente: El autor.

La primera página que se debe crear es *index.zul.* Esta debe ser una página de navegación simplepara las páginas individuales. Una forma sencilla de ejecutar

las páginas individuales, es crear enlaces para cada página, en la página de inicio. En una página HTML simple, se podría usar una etiqueta *href*[15] . En una página ZUL la forma mas óptima para la navegación, es utilizarun *ToolBarButton*de la siguiente manera: <toolbarbutton label="Agregar una nueva página" href="crear.zul"/>

3.1.4.1.1 Árbol de componentes.

Como en unaestructura de un árbol, un componente padre puede tener varios componentes hijos. Por el contrario, un componente hijo, debe tener al menos un componente padre.

Algunos componentes aceptan sólociertos tipos de componentescomo hijos.Otros no permitentener componentes hijos.

Un componentesin ningún tipo de componentepadre se llamaun componentede la raíz.En una página está permitidotener varios componentes tipo raíz, a pesar de queesto no sucedamuy a menudo.

Se debe tener en cuenta quesi se está usandoZUML, existe una limitación deXML, lo que significa que sólo se permite unaraíz para dicho documento.Para especificarmúltiples raíces, se tiene que incluirloscomponentes de las raícescon la etiqueta.

La mayoría de lascoleccionesdevueltas poruncomponente, tal como*Component.getChildren ()*, son estructurasvivas. Esto significa que se

[15]http://www.htmlpoint.com/guida/html_10.htm

puedenagregar, quitaro borrar.Por ejemplo, paraseparartodos los componentes hijos, se podría hacerloen una declaración:*comp.getChildren().clear();*

3.1.4.1.2 Escritorio, Página y Componente

Una páginaesuna colección de componentes y se representa en una parte dela ventana del navegador.

Sólo los componentes deunapágina están disponiblesen el cliente y estos se eliminancuandose separan deunapágina.

Un escritorioes una colección depáginas.Representauna ventana del navegador(una ficha o un marcodel navegador).Puede serla imagen de unescritorioindependiente,lo que representa unpetición HTTP. Como muestra la Figura 3.12:

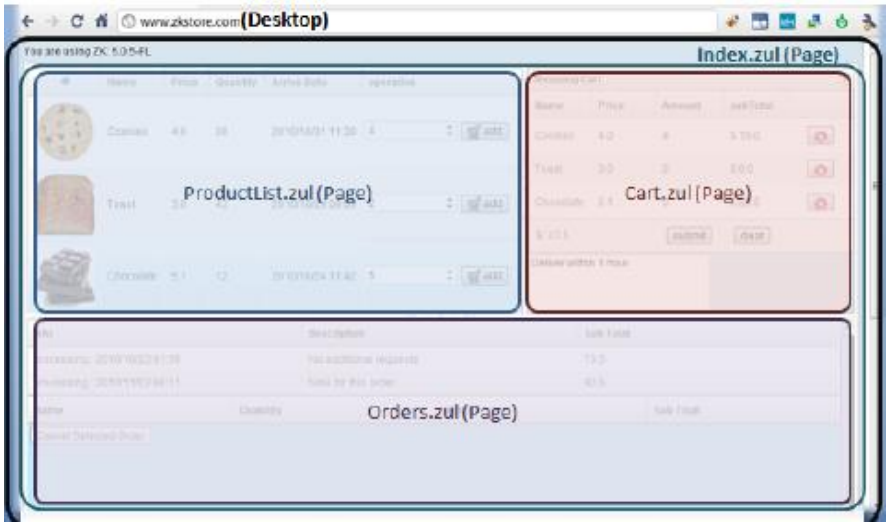


Figura 3.12: Representación de un Escritorio, Página y Componentes en un navegador web.

Fuente: ZK 5.08 Reference Guide.

Un escritorio estambién unámbito de aplicaciónlógica en la que una aplicación puedeacceder auna solicitud. Cada vez quese envía una peticióndel cliente, se asociacon el escritorio al quepertenece.

Esto también significa quela aplicación no puedeacceder a componentes enmúltiples escritoriosal mismo tiempo.

Tantoun escritorioy una páginase pueden crearde forma automática cuando el componente internoZKloadercarga una páginaZUMLo llama a unrichlet (Richlet.service (org.zkoss.zk.ui.Page).

Una segunda páginase crea cuandose agrega el componente *include.* Por ejemplo:

```
<window>
        <include src="another.zul" mode="defer"/><!—creación de otra página -->
</window>
```

Se debe tener en cuenta quesi el modono se especifica(es decir,el atributo *mode*), el componente *include* no serácapaz de crearuna nueva página. Por el contrario,va aanexar todoslos componentes creados por la página *another.*zul Por ejemplo:

```
<window>
        <include src="another.zul" mode="defer"/><!—creación de otra página -->
</window>
```

Esto es equivalente a lo siguiente:

```
<window>
<div>
        Component-based UI 15
        <zscript>
        execution.createComponents("another.zul", self, null);
        </zscript>
</div>
</window>
```

3.1.4.1.3 Formación de las páginas ZUL

Esta sección proporciona conceptos básicos deXMLpara trabajar conZK. Si está familiarizadocon XML, podríasaltarse esta sección.

XML esun lenguaje de marcadomuy similar aHTML, perocon una sintaxismás estricta y limpia.Todo el contenidoXML, no importa si está enun archivo ocomo una cadena de caracteres,se llama undocumento XML.

a. Elementos

Un elementoXML estodo a partir dela etiqueta inicial a la etiqueta final delelemento. Un elemento puedecontener otros elementos, ya sea texto simpleo una mezclade ambos.Los elementos también puedentener atributos. Por ejemplo:

```
<window title="abc">
        <button label="click me"/>
</window>
```

Ambos componentes;ventanay botónson elementos, mientras que el títuloesunatributo del elementoventana.Elelemento botónesta anidado en elelemento de la ventana. La ventana es elelemento principalde un botón, mientras que el botónes un elemento secundariode la ventana.

Laraíz del documentoesel elemento superior(sin ningún elemento de los padres).

Los elementos deben estar bien formados y cada elemento debe estar cerrado. Existen dos formas de cerrar un elemento. La primera es cerrarlos con un *tag* final como se muestra a continuación:

```
<window></window>
```

La segunda forma es cerrar los elementos sin un *tag* final, como se muestra a continuación:

```
<window/>
```

Los elementos deben estar correctamente estructurados. A continuación se muestra una manera correcta e incorrecta de estructurar los elementos.

Forma incorrecta:

```
<window>
        <groupbox>
                Hello World!
        </window>
</groupbox>
```

Forma correcta:

```
<window>
        <groupbox>
                Hello World!
        </groupbox>
</window>
```

XML trata a cada etiqueta como un nodo en un árbol. Un nodo sin un nodo principal es un componente de raíz, y es la raíz de un árbol. En cada archivo ZUL, solo se permite un árbol de componentes.

Alternativamente,se podría decir que XML nointerpretael textosi se usa CDATA como se muestra en el siguiente ejemplo:

```
<zscript>
<![CDATA[
        void myfunc(int a, int b) {
        if (a < 0 && b > 0) {
        //do something
        }
]]>
</zscript>
```

Se sugiereañadirsiempre la etiqueta <! [CDATA []]>dentro de su componente <zscript></zscript>. De esta manera todas las secuencias de escape y los caracteres especiales tales como"&","<" son ignorados por la estructura XML. Además, el código tambiénse hace mucho fácil de leer yde darle mantenimiento.

b. Comentarios

Un comentario se utiliza para dejar una nota o para deshabilitar temporalmente un bloque de código XML. Para agregar un comentario en XML, utilice <- *Comentario* -> para marcar el cuerpo comentario.

```
<window>
        <!—este es un comentario ignorado por ZK -->
</window>
```

c. Elementos especiales

Hay algunos elementos dedicados a la funcionalidad especial en lugar de un componente. Por ejemplo:

```
<zk>...</zk>
```

El elemento de ZK es un elemento especial que se utiliza para agregar otros componentes. A diferencia de un componente real (por ejemplo, o *hbox*o *div*), no es

parte del árbol componente que está siendo creado. En otras palabras, no representa ningún componente.

d. Atributos

Un atributo XML asigna un valor a la propiedad de un componente o al detector de eventos. Cada atributo, a excepción de los atributos especiales como *if* y *forEach*, representa un valor que se asigna a una propiedad de un componente después de su creación. El nombre del atributo es el nombre de la propiedad, mientras que el valor del atributo es el valor por asignar.

Al igual que en JSP, se puede usar *EL* para asignar el valor de los atributos. Por ejemplo:

```
<window title="${param.name}"/>
```

e. Eventos

Si el nombre del atributo se inicia *on*, y la tercera letra es mayúscula, un detector de eventos se le asigna. Por ejemplo, se puede registrar un detector de eventos para controlar el evento *onClick* de la siguiente manera:

```
<button onClick="do_something_in_Java())"/>
```

El valor del atributo debe ser un código válido de Java, y será interpretado cuando se reciba el evento. Se podría especificar lenguajes de programación diferentes al anteponerle el nombre del lenguaje Por ejemplo, se puede escribir el detector de eventos en *Groovy* como sigue.

```
<vlayoutonClick="groovy:self.appendChild(new Label('New'));">
        Click me!
</vlayout>
```

3.1.4.1.4 Programación del Lenguaje de Expresiones EL

Se utiliza el lenguaje de expresiones, para que en un documento ZUML, sea más fácil el acceder a los objetos disponibles en la aplicación, tales como datos de la aplicación y parámetros. Una expresión de lenguaje *EL,*es una expresión que acompaña $ y {}, es decir, la sintaxis es $ {expr}. Por ejemplo:

```
<element attr1="${bean.property}".../>
        ${map[entry]}
<another-element>${3+counter} is ${empty map}</another-element>
```

Cuando una expresión *EL* es usada como un valor de un atributo, podría devolver cualquier tipo de objeto, siempre y cuando el atributo lo permita. Por ejemplo, las expresiones siguientes se evaluarán como Boolean e Integer, respectivamente.

```
<window if="${some > 10}"><!-- boolean -->
<progressmetter value="${progress}"/><!-- integer -->
```

3.1.4.2 Scripts en ZULM

3.1.4.2.1 Insertar código Script del lado del servidor.

Para hacer más fácil la creación de una página web dinámica, el documento ZUML permite insertar código de script. Hay dos tipos de código *Script*: del lado del servidor y de l lado del cliente. En un documentoZUML, dependiendo de las necesidades, hay dosmaneras de integrar el código de Script del

lado del servido. La primera forma es utilizando el elemento *zscript*y la segunda forma es utilizando el controlador de eventos.

Se debe tener en cuenta que el rendimiento de *BeanShell*no es muy óptimo, como cualquier otro intérprete, se pueden encontrar errores sólo cuando es evaluado. Sin embargo, la incorporación decódigo Java en una página ZUML es una poderosa herramienta para el prototipado rápido.

3.1.4.2.2 Zscript

Se puede insertar el códigodentro del elemento zscript, de tal manera queserán evaluados cuando la página sea dibujada. Por ejemplo:

```
<zscript>
        //dentro de zscript
        //Se puede declar una funcòn , una variable o incluso una clase.
        void foo(String a) {
        //...
}
comp.addEventListener("onClick",
new EventListener() {
public void onEvent(Event event) {
//...
}
});
</zscript>
```

Se debe tener en cuenta que por defecto el código dentro del elemento zscript es Java, y puede se elegir en otros lenguajes, talescomo Groovy.

El código insertado se interpreta en tiempo de ejecución, por lo que los errores solo se encontrarán cuando se ejecute la aplicación. Además, se ejecuta en el servidor, por lo que se tiene acceso a las bibliotecas de Java. Se puede incluso definir variables, métodos o clases.

El Código 3.1 muestra un ejemplo claro de la utilización del elemento Zscript y *CDATA*, en el que se implementa una lógica para realizar el registro tipo login a una aplicación*:*

```
<zscript>
            <![CDATA[//@IMPORT
            importorg.zkoss.lang.Strings;
            ]]><![CDATA[//@DECLARATION
            publicvoiddoLogin(){
                  String user = usertb.getValue();
                  String pwd = pwdtb.getValue();
                  if(Strings.isBlank(user) || Strings.isEmpty(pwd)){
                        msg.setValue("*Se necesita un password!");
                        return;
                  }
                  if(!"1234".equals(pwd)){
                        msg.setValue("*Password incorrecto!");
                        return;
                  }
                  session.setAttribute("user",user);
                  loginDiv.setVisible(false);
                  userDiv.setVisible(true);
                  userName.setValue(user);
                  msg.setValue("");
            }

            ]]><![CDATA[
            String user = (String)session.getAttribute("user");
            ]]>

</zscript>
```

Código 3.1: Ejemplo de código embebido Zscript.

Fuente: El autor.

El código insertado en el elemento zscript, debe ser un texto XML válido. Para solventar este problema, se debe codificar todo el código con *XML CDATA* como se ilustró en el Código 3.1. A pesar de esto, el agregar código Java en una página ZUML es una poderosa herramienta para el prototipado rápido

.

3.1.4.3 Evaluación iterativa

3.1.4.3.1 El atributo forEach

De forma predeterminada, ZK crea instancias de un componente para cada elemento XML. Si se desea generar una colección de componentes, se puede especificar el atributo forEach.

Ejemplo:

```
<listbox>
        <listitem label="${each}" forEach="Apple, Orange, Strawberry"/>
</listbox>
```

Esto es equivalente a:

```
<listbox>
        <listitem label="Apple"/>
        <listitem label="Orange"/>
        <listitem label="Strawberry"/>
</listbox>
```

Cuando ZK loader recorre los elementos de la colección, se actualizarán

dos objetos implícitos: *each* y *forEachStatus*.

El objeto *each* representa el elemento que se reiteró, mientras que *forEachStatus* es una instancia de la clase ForEachStatus desde donde se puede recuperar el índice y el elemento anterior.

Si se tiene una variable que contiene una colección de objetos, entonces se puede especificar directamente en el atributo forEach. Por ejemplo:

```
grades = new String[] {"Best", "Better", "Good"};
```

De esta manera, se puede recorrer la colección gracias al uso del atributo. Se debe tener en cuenta que se debe utilizar lenguaje de expreciones para especificar la colección.

```
<listbox>
        <listitem label="${each}" forEach="${grades}"/>
</listitem>
```

La iteración depende del tipo del valor del atributo forEach:

• Si es java.util.Collection, se repite cada elemento de la colección.

• Si es java.util.Map, se repite cada Map.Entry del mapa.

• Si es java.util.Iterator, se repite cada elemento del iterador.

• Si es java.util.Enumeration, se repite cada elemento de la enumeración.

• Si es Object [], int [], corta [], byte [], char [], float [] o doble [], se repite cada elemento de la matriz.

• Si es nulo, no se genera (se omite).

• Si ninguno de los dos tipos anteriores es especificado, el elemento asociado será evaluadosolo una vez, como si una se tratara de una colección con un solo elemento.

3.1.4.3.2 El objeto Each

Durante la evaluación, un objeto llamado *each* se crea y se asigna al elemento de la colección especificada. Se debe tener en cuenta que el objeto *each* es accesible tanto en *EL* como en *zscript*. ZK preservará el valor de cada objeto, si se ha definido

antes, y se puede restaurarlos después de la evaluación del elemento asociado anteriormente.

3.1.4.4 Anotaciones

Una anotación es una forma especial de metadatos sintáctica, que pueden ser añadidas a los componentes. Definiciones de componentes, componentes y métodos, podrán ser anotados. Las anotaciones se pueden recuperar en tiempo de ejecución.El contenido y el significado de las anotacionesdependen totalmente de las herramientas o utilidades que el desarrollador utiliza.

Por ejemplo, undata-binding manager,podría examinar anotaciones al conocer la fuente de datos en la que el valor de un componente se almacena.

3.1.4.4.1 Anotaciones en ZULM

Las anotaciones pueden aplicarse a las declaraciones de los componentes y propiedades en las páginas ZUML.

3.1.4.4.2 Propiedades de las anotaciones

Para anotar una propiedad, se puede especificar una expresión de anotación en algún atributo del componente. En otras palabras, si elvalor del atributo es una expresión de anotación, se considera como una anotación del atributo, en lugar de un valor que se asigna al mismo.
El formato de una expresión es la anotación @{Annotación-nombre (attr-nombre1 = valor1 attr-, attr-nombre2 = attr-valor2)}.

Una anotación se compone de un nombre y un mapa de atributos que se adjunta con paréntesis. Cada atributo es un par de nombre-valor. Por ejemplo:

```
<listitem label="@{bind(datasource='author',value='selected')}"/>
```

En el ejemplo anterior, una anotación llamada *bind* está agregada en la propiedad de la etiqueta. La anotación *bind* tiene dos atributos: fuente de datos y valor.

Si el nombre de anotación no se especifica, se supone que es predeterminada. Por ejemplo, las siguientes dos declaraciones son equivalentes`<listitem label="@{bind(foo)}"/><listitem label="@{bind(value='foo')}"/>`

Aquí se muestra un Ejemplo más complejo:`<listitem label="@{selected.name}"/>`

3.1.4.5 Cómo re direccionar a otras páginas

Re direccionar a una página nueva, es muy útil y necesariopara dirigirnos a cierta página después deunaacción que se ejecuta. En ese caso, el framework ZKofrece la clase utilitaria *org.zkoss.zk.ui.Executions*.

Esta clase ofreceel método llamado *sendRedirect*. En una página deZUL, se puede acceder directamentea la instancia de ejecucionesy redirigira otra página(por ejemplo,*Exectutions.sendRedirect('index.zul')*). La redirección,envía unarespuesta al clientedándoleuna nueva dirección URL. El clientesolicita a lanueva URL y un re enviósucedeinternamente en elcontenedor de servlets. Elobjetivose le da laoportunidad de responder ala petición original.

3.1.4.6 Cómo programar Richlets

Un richlet es un pequeño programa Java, que componen una interfaz de usuario en Java para servir a la petición del usuario.

Cuando un usuario solicita el contenido de una URL, ser realizan las comprobaciones del componente interno ZK loader para saber si el recurso de la URL especificada es una página ZUML o un richlet.

Si se trata de una página ZUML, el ZK loader la creará automáticamente en función de los componentes de la página ZUML.

Si el recurso es un *richlet*, el ZK loader maneja el procesamiento del programa richlet. Cómo y cuando se puede crear componentes que están manejados por un Richlet. En otras palabras, es el trabajo del programador crear todos los componentes necesarios en respuesta a la solicitud.

La elección entre las páginas ZUML y richlets depende de preferencia y del grado de conocimiento y experiencia del programador.

3.1.4.6.1 Implementación de un Richlet

Es sencillo implementar un Richlet, primero se debe crear el código de la interfaz del Richlet antes de asociarlo con alguna URL.

Un richlet debe implementar la interfaz genérica Richlet. Sin embargo no es necesario codificar esto desde cero. Por el contrario, se podría extender el richlet de la clase *GenericRichlet*. Con *GenericRichlet*, lo único que se debe que hacer es

ejecutar el método Richlet.service (org.zkoss.zk.ui.Page). Este método es llamado cuando una URL asociada se solicita. A continuación el Código 3.2 describe una clase Java Richlet.

```
publicclassTestRichletextends GenericRichlet {
	// Richlet//
	publicvoid service(Page page) {
		page.setTitle("Richlet Test");
				final Window w = new Window("Richlet Test",
		"normal", false);
		new Label("Hello World!").setParent(w);
		final Label l = new Label();
		l.setParent(w);
		final Button b = new Button("Change");
		b.addEventListener(Events.ON_CLICK, new EventListener() {
			intcount;

			publicvoid onEvent(Event evt) {
				l.setValue("" + ++count);
			}
		});
		b.setParent(w);
		w.setPage(page);
	}
}
```

Código 3.2: Richlet implementado en una clase Java

Fuente: ZK Developer's Reference for ZK 5.0.8

Como se muestra en el Código 3.3, se tiene que invocar ala página*Component.setPage*(Page page)de forma explícitapara agregar un componente raíz a unapágina, y permitir que este disponible en el cliente.

Paratener un mejor control, se puede poner en práctica el método *Richlet.init(org.zkoss.zk.ui.RichletConfig)* y *Richlet.destroy*() parainicializary destruirtodos los recursosrequeridos por elrichletcuando se está cargado.

Además, se puede implementar el método *Richlet.getLanguageDefinition*()parautilizar un lenguaje de programación diferente,

como por defecto (por ejemplo,la implementación de unrichletpara dispositivos móviles). De forma predeterminada se asume el lenguaje ZUL.

Es importante tener en cuenta que al igual queun servlet, secreasolo una instancia derichlety se compartepara todos los usuariosdetodas las solicitudes dela direcciónasignada. Un richletdebe manejarpeticiones concurrentes, y se debe tener cuidado con la sincronización deacceso a los recursos.

Cuando una petición (no una petición Ajax sino una petición regular HTML) es hecha por un usuario, un escritorio o página es creado y después el método *Richlet.service(org.zkoss.zk.ui.Page)* es invocado para servir la petición. En conclusión, cada petición es manejada con un escritorio o página independiente.

3.1.4.7 Implementación de operaciones CRUD bajo el patrón MVC

Este ejemplo consiste en la creación de componentes que permitan ejecutar las operaciones básicas de un CRUD (Create – Read – Update -Delete), y la creación de la lógica de navegación de estos componentes.

Para ello se necesitará dos elementos esenciales en la construcción de aplicaciones bajo el framework ZK, los cueles son; una clase Java Composer y una página ZUL. Cada objeto de la interfaz de usuario se representa con un componente. Por lo tanto, la composición de la interfaz de usuario es la inclusión de un conjunto de componentes en las páginas. Para modificar la interfaz de usuario se tiene que modificar los estados y las relaciones de los componentes tal como muestra la Figura 3.13.

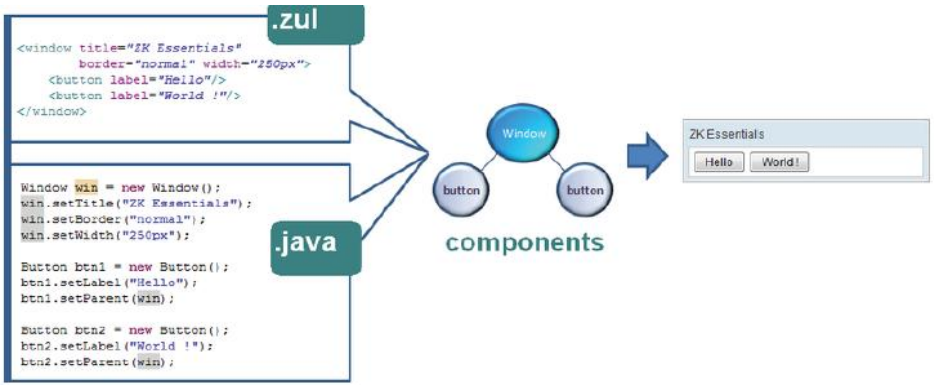


Figura 3.13: Opciones de creación de componentes en la interfaz de usuario.

Fuente: ZK 5.0.8 Developers Guide.

Para el ejemplo, a la clase java se la nombrará*CRUDComposer.java* y se encarga de alojar la programación de la lógica de navegación y ejecución de la página. Además será ahí donde se crean los objetos de los componentes de la página.

La página ZUL se llamará *crud.zul* y esta encargada de mostrar los componentes ZK que se utilizó para el ejemplo. El resultado final del ejemplo se muestra en la Figura 3.14.

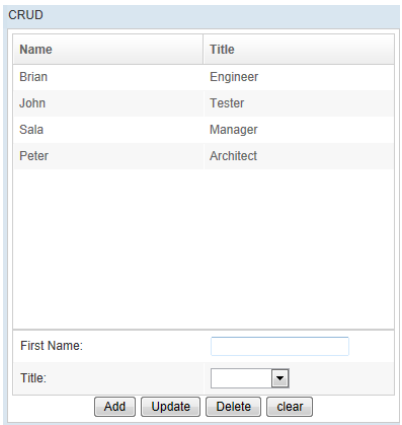


Figura 3.14: Página que permite realizar operaciones CRUD.

Fuente: El autor.

La forma como se unen estos dos elementos (Composer Java – Página ZUL), es mediante una declaración al inicio de la página en el componente *window*, indicando a que clase Composer esta ligada esta página, tal como muestra el Código 3.3:

```
<windowid="main"title="CRUD"apply="ec.com.kruger.plancarrera.web.composer.CRUDComposer"width="400px"border="normal">
```

Código 3.3: Vinculación de una clase Composer con una página ZUL.

Fuente: El autor

A continuación se ilustrará paso a paso como conseguir esta aplicación CRUD. Para ello se entiende que el proyecto ya esta configurado como un proyecto ZK.

3.1.4.7.1 Creación de la clase CRUDComposer.java

En el proyecto, se crea un nuevo paquete de clase y se agrega una nueva clase llamada CRUDComposer.java tal como se muestra en la Figura 3.15:

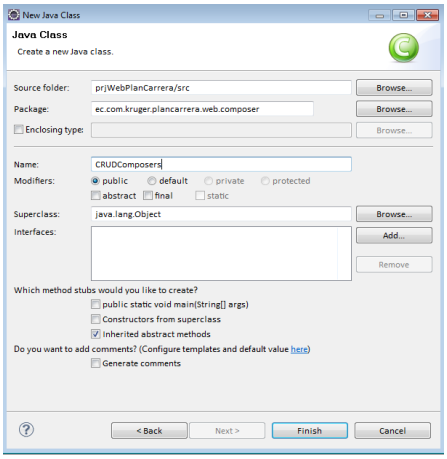


Figura 3.15: Ventana de creación de clases Java.

Fuente: El autor.

Una vez creada la clase,se importa todas las librerías del framework ZK necesarias para la ejecución de los componentes, tal como muestra la Figura 3.16:

```
import org.zkoss.lang.Strings;
import org.zkoss.zk.au.out.AuClearWrongValue;
import org.zkoss.zk.ui.Component;
import org.zkoss.zk.ui.Executions;
```

Figura 3.16: Librerías necesarias para la ejecución de la clase Composer.

Fuente: El autor.

Es necesario extender a nuestra clase CRUDComposer.java una nueva clase genérica llamada *GenericForwardComposer,* de esta manera se accedede a funciones básicas que deben implementar todas las clases Composer, tales como la función *doAfterCompose.*

La función *doAfterCompose* nos permite ejecutar todo lo que este dentro de la función una vez dibujado el árbol de componentes, de manera automática.

Extendiendo de *GenericForwardComposer* o *GenericAutowireComposer*, nos se tiene que asegurar que la totalidad de sus miembros o elementos son serializables (o transient), ya que la aplicación mantendrá una referencia al componente de aplicación.

Para el ejemplo, se utilizará una lista de objetos tipo *Person* que nos servirá para visualizar todas las personas que se agregó o editó y que se encuentran almacenadas en primera instancia.

En la función *doAfterCompose* se agregará datos a la lista de personas.

Se debe programar también las acciones que se ejecutaran en cada botón, para lo cual se utilizará las siguientes funciones:

1. onClick$add. Esta función se ejecuta al momento de aplastar el botón de Add, y tiene como objetivo es crear un nuevo objeto tipo *Person* y agregarlo a la lista de objetos existentes en ese momento, tal como se muestra a continuación:

```
publicvoid onClick$add() {
            Person person = new Person(getSelectedName(),
            getSelectedTitle());
      model.add(person);
      selected = person;
      binder.loadAll();
}
```

2. onClick$update. Esta función se ejecuta al momento de aplastar el botón Update. El objetivo de esta función es actualizar una fila seleccionada, para ello primero se condiciona si es que se ha seleccionado algún dato y después se procede a actualizar los datos del registro seleccionado, tal como se muestra a continuación:

```
publicvoidonClick$update() {
            if (selected == null) {
                  alert("Nothing selected");
                  return;
            }
            selected.setName(getSelectedName());
            selected.setTitle(getSelectedTitle());
            binder.loadAll();
      }
```

3. onClick$delete. Esta función se ejecuta al momento de aplastar el botón Delete y tiene como objetivo quitar de la lista de personas a la persona seleccionada en ese momento, para lo cual primero verifica que exista una persona seleccionada, tal como muestra a continuación:

```
publicvoidonClick$delete() {
```

```
        if (selected == null) {
              alert("Nothing selected");
              return;
        }
        model.remove(selected);
        selected = null;

        binder.loadAll();
    }
```

4. onSelect$titlelb. Esta función se ejecuta al momento de cerrar el cuadro de diálogo que se muestra cuando se intenta actualizar una persona, pero algún atributo (nombre o título) está vacío.

A continuación en el Código 3.4 se muestra toda la clase CRUDComposer.java:

```
publicclassCRUDComposerextends GenericForwardComposer {

    Listbox personslb;
    Textbox nametb;
    Listbox titlelb;

    AnnotateDataBinderbinder;

    List<Person>model = newArrayList<Person>();
    List<String>titleModel = newArrayList<String>();
    Person selected;

    publicCRUDComposer() {
    }

    publicvoid doAfterCompose(Component comp) throws Exception {
          super.doAfterCompose(comp);
          comp.setVariable(comp.getId() + "Ctrl", this, true);

          model.add(new Person("Brian", "Ingeniero"));
          model.add(new Person("John", "Tester"));
          model.add(new Person("Sala", "Manager"));
          model.add(new Person("Peter", "Arquitecto"));

          titleModel.add("");
          titleModel.add("Ingeniero");
          titleModel.add("Tester");
          titleModel.add("Manager");
          titleModel.add("Arquitecto");

          binder = newAnnotateDataBinder(comp);
          binder.loadAll();
    }

    public List getModel() {
          returnmodel;
    }
```

```
	public List getTitleModel() {
		returntitleModel;
	}

	public Person getSelected() {
		returnselected;
	}

	publicvoidsetSelected(Person selected) {
		this.selected = selected;
	}

	publicvoid onClick$add() {
		Person person = new Person(getSelectedName(),
getSelectedTitle());
		model.add(person);
		selected = person;
		binder.loadAll();
	}

	publicvoidonClick$update() {
		if (selected == null) {
			alert("Nothing selected");
			return;
		}
		selected.setName(getSelectedName());
		selected.setTitle(getSelectedTitle());
		binder.loadAll();
	}

	publicvoidonClick$delete() {
		if (selected == null) {
			alert("Nothing selected");
			return;
		}
		model.remove(selected);
		selected = null;

		binder.loadAll();
	}

	publicvoid onSelect$titlelb() {
		closeErrorBox(new Component[] {titlelb });
	}

	publicvoidonSelect$personslb() {
		closeErrorBox(new Component[] { nametb, titlelb });
	}


	privatevoidcloseErrorBox(Component[] comps){
		for(Component comp:comps){

	Executions.getCurrent().addAuResponse(null,newAuClearWrongValue(comp)
);
		}
	}
```

```
	private String getSelectedTitle() throwsWrongValueException {
		int index = titlelb.getSelectedIndex();
		if (index < 1) {
			thrownewWrongValueException(titlelb, "Debeseleccionar al
menosuno!");
		}

		returntitleModel.get(index);
	}

	private String getSelectedName() throwsWrongValueException {
		String name = nametb.getValue();
		if (Strings.isBlank(name)) {
			thrownewWrongValueException(nametb, "No debeestar en
blanco!");
		}
		return name;
	}


}
```

Código 3.4: Clase CRUDComposer.java.

Fuente: El autor.

3.1.4.7.2 Creación de la página crud.ZUL

Se crea una página llamada crud.ZUL para alojar los objetos que se ha creado en nuestra clase CRUDComposer.java, para lo cual se agrega una página tal como se muestra en la Figura 3.11 pero se asigna el nombre crud.ZUL. Una vez creada la página, se crea un componente tipo ventana que alojará todos los componentes que se utilicen de aquí en adelante. Se debe hacer el biding con el Composer tal como se muestra en el Código 3.1.

Después de esto se puede crear todos los componentes necesarios.

Lo primero que se mostrará en nuestra página es la lista de personas, para lo cual se utilizará un componente llamado *listbox*¸ el cual nos permite iterar una colección de objetos y mostrarlos en la pantalla. Para ello se utilizará el Código 3.5:

```
<listboxid="personslb"rows="10"model="@{mainCtrl.model}"
            selectedItem="@{mainCtrl.selected}">
            <listhead>
                  <listheaderlabel="Name"/>
                  <listheaderlabel="Title"/>
            </listhead>
            <listitemself="@{each=person}">
                  <listcelllabel="@{person.name}"/>
                  <listcelllabel="@{person.title}"/>
            </listitem>
      </listbox>
```

Código 3.5: Creación de un componente listbox.

Fuente: El autor.

El componente *listbox* a su vez encierra varios componentes los cuales son básicamente una cabecera representada por el componente *listhead* y los detalles, representado por el componente *listitem.* Cabe señalar que en este ultimo componente se utiliza lenguaje de expresión para acceder a un recurso del Composer, como la lista de personas. El componente *listitem* muestra los datos de las columnas en el componente *listcell.*

A continuación se creará el formulario de ingreso de datos y los botones de ejecución. Para ello se crea un componente *grid* que nos servirá para organizar de una mejor manera los inputs de el formulario. El Código 3.6 muestra como se puede crear un formulario en un componente *grid.*

```
      <grid>
            <rows>
                  <row>
```

```
                        Primer nombre:
                        <textboxid="nametb"value="@{mainCtrl.selected.name,
                        save-when=none}"/>
                  </row>
                  <row>
                        Título: <listboxid="titlelb"mold="select"
                  selectedItem="@{mainCtrl.selected.title,save-when=none}"
                              model="@{mainCtrl.titleModel}"

                              />
            </row>
      </rows></grid>
```

Código 3.6: Creación de un formulario dentro de un componente *grid*

Fuente: El autor.

Se puede notar que en los inputs para el nombre y el título se utiliza una vez más lenguaje de expresión para acceder a las objetos del Composer.

Por último se crea los botones de CRUD utilizando un componente llamado *hbox* como muestra el Código 3.7:

```
      <hboxpack="center"width="100%">
            <div>
                  <buttonid="add"label="Add"/>
                  <buttonid="update"label="Update"/>
                  <buttonid="delete"label="Delete"/>

                        <buttonlabel="clear"onClick="Clients.closeErrorBox(
                  new Component[]{nametb,titlelb})"/>
            </div>
      </hbox>
```

Código 3.7: Creación botones dentro de un componente *hbox*

Fuente: El autor.

A continuación se mostrará en El Código 3.8 la página crud.zul:

```
<windowid="main"title="CRUD"apply="ec.com.kruger.plancarrera.web.composer.C
RUDComposer"width="400px"border="normal">
      <listboxid="personslb"rows="10"model="@{mainCtrl.model}"
            selectedItem="@{mainCtrl.selected}">
            <listhead>
```

```
                  <listheaderlabel="Name"/>
                  <listheaderlabel="Title"/>
            </listhead>
            <listitemself="@{each=person}">
                  <listcelllabel="@{person.name}"/>
                  <listcelllabel="@{person.title}"/>
            </listitem>
      </listbox>
      <grid>
            <rows>
                  <row>
                        Primer nombre:
                  <textboxid="nametb"value="@{mainCtrl.selected.name,save-
                  when=none}"/>
                  </row>
                  <row>
                        Título: <listboxid="titlelb"mold="select"
                  selectedItem="@{mainCtrl.selected.title,save-when=none}"
                              model="@{mainCtrl.titleModel}"

                              />
                  </row>
            </rows>
      </grid>
</window>
```

Código 3.8: Página crud.zul

Fuente: El autor.

3.2 Guía de desarrollo de componentes gráficos.

3.2.1 Introducción a los componentes de ZK

Esta sección tiene como objetivo el brindar una vista general del desarrollo de componentes en framework ZK.

3.2.1.1 Objetivo de la guía de desarrollo de componentes gráficos

El objetivo de esta guía, es el brindar un conocimiento especifico acerca de cómo se deben programar los componentes del framework ZK, además de proporcionar

consejos y directrices de desarrollo de los componentes del framework, para sacar el máximo provecho a los mismos.

Se pretende que esta guía sirva como modelo de programación de componentes, para cualquier aplicación de mediano y corto alcance, realizada con el framework ZK.

3.2.1.2 A quién está dirigida esta guía

Esta guía esta dirigida a todo desarrollador, programador, diseñador interesado en la arquitectura de componentes del framework ZK. Esta guía no excluye el nivel de experticia ni la experiencia que tenga el lector, pero se recomienda que se tenga previo conocimiento de la arquitectura y desarrollo del framework ZK.

3.2.1.3 Qué es un Componente ZK

Todos los objetos de la interfaz de usuario en el framework ZK, se representa como un componente y un widget.

3.2.1.3.1 Componente

Un componente, es un objeto de Java que se ejecuta en el servidor y representa un objeto de la interfaz de usuario que puede ser manipulado por una aplicación Java

3.2.1.3.2 Widget

Un widget, es un objeto de JavaScript que se ejecuta en el cliente. Este representa el objeto Java en la interfaz de usuario. Por lo tanto, un widget por lo general tiene un aspecto visual y se ocupa de los eventos que se suceden en el cliente.

Una vez establecidas estas dos partes, se explicará a continuación cómo interactúan entre sí, para brindar una experiencia interactiva al usuario.

3.2.1.4 Cómo trabaja un Componente ZK

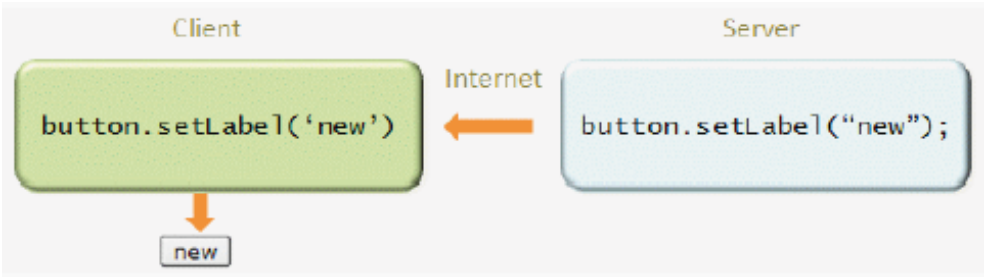
Un componente y un widget trabajan juntos para manejar todos los eventos que se realizan en la interfaz de usuario. El widget se encarga de captar la actividad del usuario y enviar las peticiones apropiadas al componente. El componente interactúa con el desarrollador de la aplicación, quién puede responder apropiadamente diciendo que widget debería actualizarse. Esta interacción, es demostrada en la Figura 3.9:

Código 3.9: Interacción entre el usuario, el componente ZK y la aplicación

Fuente: ZK 5.0.8 Component Development Essentials.

A continuación se presentará un ejemplo en el cual se invoca el método *setLabel* para cambiar la etiqueta de un componente botón. Este método se invocará desde el

cliente para cambiar la apariencia visual del componente, como se muestra en la Figura 3.10:



Código 3.10: Invocación de un método de un componente ZK.

Fuente: ZK 5.0.8 Component Development Essentials

Cuando el usuario hace click en el widget del botón, el evento onClick enviará una petición al servidor y notificará a la aplicación.

Manipular un componente en el servidor, es posible controlando un widget en el cliente. Por ejemplo, una aplicación debe ocultar o cambiar el orden de las columnas de un objeto *grid*, mientras la aplicación este corriendo en el servidor se puede manejar la actualizacion y recargando el contenido del *grid.* Esta técnica es llamada Server – Client fussion y se puede utilizar para mejorar la capacidad de respuesta y reducir el tráfico en la red.

3.2.2 Creación de un Componente ZK

3.2.2.1 JavaScript en los componentes ZK

Para que JavaScript represente a los widgets de una manera más fácil, ZK ha introducido un concepto de clase de JavaScript. En el Código 3.11se muestra una breve introducción sobre la definición de una clase en JavaScript.

El primer argumento es la clase base de la cual se extiende. En este caso, se extienden de la clase Object, que es raíz en la jerarquía de clases. El segundo argumento se compone de miembros (no estáticos) de la clase. En este caso, se definen dos miembros de datos (X y Y) y un método (distancia).

El tercer argumento es opcional y si es omitido quiere decir que la clase extendida no contienen miembros estáticos.

```
zk.$package('com.foo');
com.foo.Location= zk.$extends(zk.Object, {
x: 0,
y: 0,
distance: function (loc) {
return Math.sqrt(Math.pow(this.x- loc.x, 2) + Math.pow(this.y-
loc.y, 2));
}
},{
find: function (name) {
if (name == 'ZK')
return new com.foo.Location(10, 10);
throw'unknown: "+name;
}
})
```

Código 3.11: Ejemplo de creación de una clase en JavaScript.

Fuente: ZK 5.0.8 Component Development Essentials.

3.2.2.2 Implementación de un Componente

Cómo ya se ha dicho previamente, un componente ZK está formado por un componente en el lado del servidor generalmente programado en Java y un widget en el cliente escrito en JavaScript.

Este componente, es la instancia de una clase Java que extiende de la clase *AbstractComponent* o de una de sus clases derivadas. Existen muchas clases que

proporcionan diferentes niveles de funcionalidad. La Figura 3.17 se muestra la implementación de la clase base *HtmlBasedComponent*, la cuál es la clase base para los componentes basados en HTML.

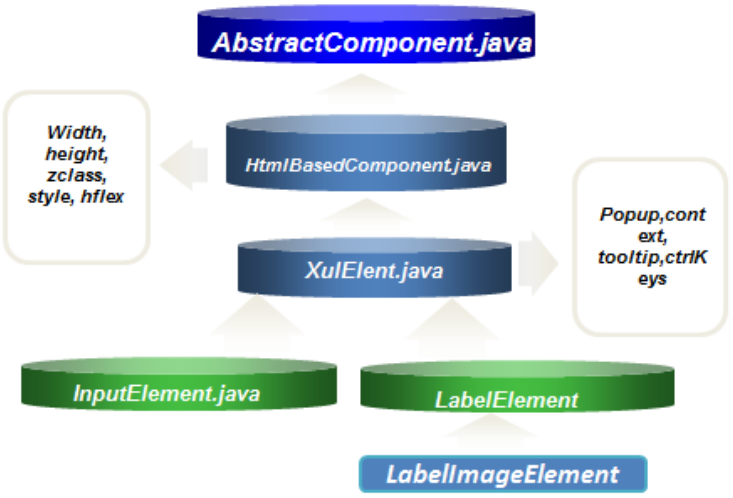


Figura 3.17: Implementación de una clase base como componente.

Fuente: ZK 5.0.8 Component Development Essentials.

Para implementar una clase como componente, se necesita decidir los siguientes aspectos:

a. Nombre de la clase. Por ejemplo: <mp>com.foo.SimpleLabel<mp>
b. Las propiedades del componente. En este caso se implementara una propiedad llamada *value* que contendrá el contenido visual del componente en el cliente.

Una propiedad usualmente debe tener los métodos *getter* y *setter.* El método *getter* tiene como funcionalidad principal el devolver el valor de la propiedad. El método *setter* al contrario, tienen como funcionalidad el asignar el valor a la propiedad del componente.

A continuación el Código 3.12 muestra la implementación de un componente del framework ZK.

```
package com.foo;
public class SimpleLabel extends org.zkoss.zk.ui.HtmlBasedComponent{

private String _value = "";
        public String getValue() {
return _value;
}
public void setValue(String value) {
I       f (!_value.equals(value)) {
        _value = value;
        smartUpdate("value", _value);
        }
}
protected void
        renderProperties(org.zkoss.zk.ui.sys.ContentRendererrenderer)
        throws java.io.IOException{
                super.renderProperties(renderer);
        render(renderer, "value", _value);
        }
}
```

Código 3.12: Implementación de un componente.

Fuente: ZK 5.0.8 Component Development Essentials.

3.2.2.3 Implementación de un Widget

Una vez explicada la implementación de un componente Java en framework ZK, es necesario implementar el widget que será la representación gráfica del componente.

A continuación la Figura 3.18 muestra la representación grafica de un widget y las jerarquías de sus clases

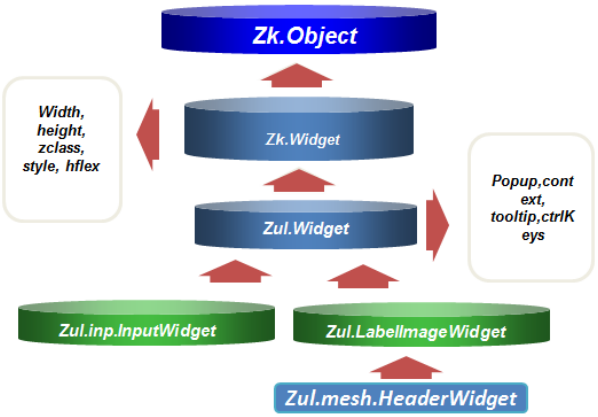


Figura 3.18: Implementación grafica de un widget.

Fuente: El autor.

La implementación de un widget debe ser representada e un archivo JavaScript independiente. El archivo debe ser introducido en el directorio, ***/web/js/package-path***, en la dirección donde se encuentra también la clase Java. Como el widget creado se llama SimpleLabel y el paquete se llama com.foo, el directorio completo del widget seria ***/web/js/com/foo/SimpleLabel.js.***

A continuación el Código 3.13 muestra la implementación del widget en el archivo JavaScript:

```
com.foo.SimpleLabel= zk.$extends(zk.Widget, {
        _value : '', // default value
        getValue: function() {return this._value;
},
        setValue: function(value) {
                if (this._value!= value) {this._value= value;
                if (this.desktop)
                this.$n().innerHTML= zUtl.encodeXML(value);
                }
        }
});
```

Código 3.13: Archivo JavaScript que representa la implementación de un widget

Fuente: ZK 5.0.8 Component Development Essentials.

3.2.3 Creación de archivos de configuración

El archivo lenguage-addon contiene una descripción de la relación que existe entre un componente y un widget. El directorio donde se debe incluir este archivo de configuración es el siguiente: /***metainfo/zk/lang-addon.xml.***El código 3.14 muestra el contenido del archivo lang-addon.xml.

```
<language-addon>
        <addon-name>simplelabel</addon-name>
        <language-name>xul/html</language-name>
        <component>
                <component-name>simplelabel</component-name>
                <component-class>com.foo.SimpleLabel</component-class>
                <widget-class>com.foo.SimpleLabel</widget-class>
                        <mold><mold-name>default</mold-name>
                                <mold-uri>mold/simple-label.js</mold-uri>
                                <css-uri>css/simple-label.css.dsp</css-uri>
                        </mold>
                </component>
</language-addon>
```

Código 3.14: Archivo lang-addon.xml

Fuente: ZK 5.0.8 Component Development Essentials.

3.2.4 Manejo de eventos

Cada componente dentro del framework ZK tiene un comportamiento distinto en el cliente. La forma de manipular este comportamiento es mediante el uso de eventos. Para ello se debedarmás funcionalidad al componente ya creado anteriormente llamado SimpleLabel. La nueva funcionalidad va a ser un nuevo click sobre el componente, el cual deberá limpiar el texto mostrado y llamar a un evento propio llamado <mp>onClear<mp>.

Esto significa que se debe cambiar el molde de la etiqueta para introducir un div que use el objeto. El Código 3.15 muestra como se debe codificar este cambio en el archivo de JavaScript en el cliente.

```
function (out) {
        out.push('<span', this.domAttrs_(), '><div id="value" style="float:left;">', this.getValue(),

        }
```

Código 3.15: Implementación de un evento en el componente propio SimpleLabel.

Fuente: ZK 5.0.8 Component Development Essentials.

Para implementar este evento se debe sobrescribir el método (zk.Desktop, zk.Skipper, _global_.Array) y registrar el métodoen un listener o escucha alojado en el servidor.

3.2.5 Comunicación Cliente-Servidor

La figura 3.19 muestra como es la comunicación entre las dos partes de un componente ZK; la clase Java alojada en el servidor y el widget JavaScript alojado en el cliente:

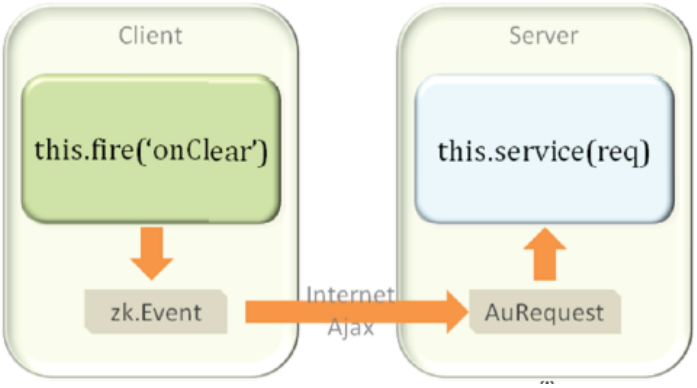


Figura 3.19: Comunicación entre un widget y un componente Java.

Fuente: ZK 5.0.8 Component Development Essentials.

Se dispara un evento en el cliente convirtiéndose en una petición asíncrona Ajax Esto sucede si se cumplen los siguientes puntos.

• El widget estaatado a un componente, es decir, se crea automáticamente para representar un componente. Tenga en cuenta queel servidor indica si un widget esta ligado o no a un componente.

• La propagación de los eventos no se detiene es decir, el evento fue escuchado por un oyente del lado del servidor

La imagen anterior demuestra cómo el evento onClear se envía al servidor y es procesado. El Código 3.16 muestra como se localiza el evento en la clase del componente SimpleLabel.java.

```
public void service(org.zkoss.zk.au.AuRequest request, boolean
        everError) {
                final String cmd= request.getCommand();
                        if (cmd.equals(ClearEvent.NAME)) {
                                ClearEventevt= ClearEvent.getClearEvent(request);
                                cleared = evt.getCleared();
                        Events.postEvent(evt);
        } else
        super.service(request, everError);
}
```

Código 3.16: Comunicación Cliente-Servidor en un componente.

Fuente: ZK 5.0.8 Component Development Essentials.

3.2.5.1 Escuchas de lado del Servidor

Los escuchas o *listeners* son los encargados de recibir ejecutar las acciones de los eventos que ocurren en el cliente. Los *listeners* se encuentran encapsulados en los archivos que componen el framework, y residen en el servidor de aplicaciones donde

se despliega el programa. Si se quiere registrar un detector de eventos propio es decir implementar el propio *listener,*se debe declarar los eventos que el *listener* manejará. Se puede hacer esto usando la función llamada *addClientEvent.*

El Código 3.17 muestra la declaración de un evento en servidor.

```
static {
        addClientEvent (SimpleLabel.class, ClearEvent.NAME, 0);
}
```

Código 3.17 : Agregar un evento mediante la función *addClientEvent*

Fuente: Fuente: ZK 5.0.8 Component Development Essentials.

Se debe tener en consideración que si el evento no es registrado en el cliente, este nunca será enviado al desde el cliente. Sin embargo, para el ejemplo que se ha realizado existe un problema si el evento no es re direccionado al cliente.

El Código 3.18, se mostrará a continuación cómo se debe utilizar el método que se ha creado en el servidor.

```
public void service(org.zkoss.zk.au.AuRequest request, boolean
                everError) {
                final String cmd= request.getCommand();
                        if (cmd.equals(ClearEvent.NAME)) {
                        ClearEventevt= ClearEvent.getClearEvent(request);
                        _cleared = evt.getCleared();
                Events.postEvent(evt);
        } else
super.service(request, everError);
}
```

Código 3.18 : Utilización de un evento propio.

Fuente: Fuente: ZK 5.0.8 Component Development Essentials.

El tercer argumento del método <mp>*addClientEvent*</ pf> se utiliza para declarar un evento como importante. El argumento es una combinación de indicadores enteros que se encargan de manejar el evento. La bandera CE_IMPORTANT se utiliza para indicarque este evento es importante y debe ser enviado al servidor, incluso si no hay un *listener* del lado del servidor registrado para este evento.

3.2.6 Encapsular en un archivo Jar

Empaquetar un componente, es generar un archivo Jar, que tendrá componentes desarrollados y es portable reutilizable para cualquier otro proyecto.

Por lo general se recomienda empaquetar componentes genéricos, que no dependan de otros archivos y que se puede implementar en cualquier sistema. Es muy aconsejable empaquetar componentes, ya que evita la duplicación de código y reduce el tiempo de desarrollo.

La Figura 3.20 muestra cual es la estructura genérica en el framework ZK, de un archivo que será empaquetado.

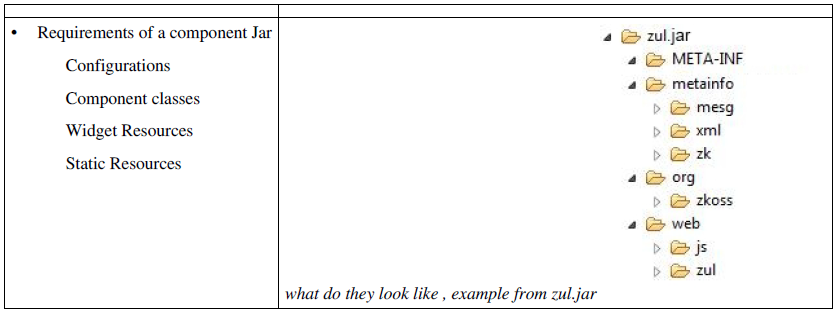


Figura 3.20 : Estructura básica de un proyecto ZK, empaquetado en un archivo Jar

Fuente: Fuente: ZK 5.0.8 Component Development Essentials.

3.3 Guía de estilos en las páginas.

La forma más común de personalizar los componentes, es mediante el uso de hojas de estilo en cascada. Esta técnica es conocida desde HTML, por lo que el framework ZK se adapta perfectamente a este recurso.

3.3.1 Introducción a los estilos en las páginas del framework ZK.

La presente guía a continuación describe los conceptos deestilos en los componentes ZK.

3.3.1.1 Objetivo de la guía

Brindar el conocimiento suficiente para implementar un estilo en un componente del framework ZK y manejar el diseño grafico de los mismos.

3.3.1.2 A quien esta dirigida

Esta guía esta enfocada a todos los diseñadores web que manejan el framework ZK y se encargan de darle la apariencia a las páginas, desarrollando estilos CSS propios para lograr una presentación agradable y elegante en sus aplicaciones web.

3.3.2 Conceptos de estilos en el framework ZK.

3.3.2.1 Sclass

La clase sclass, es una forma de agregar un estilo CSS a un elemento. Por defecto, no se le asigna nada a la clase sclass. Una vez que se le asigna un valor no vacío, se añade al elemento raíz como clase adicional CSS.

Por ejemplo, la siguiente declaración le asigna la clase CSS al componente zk-botón: <button sclass="cabecera"/>

Sclass no va a cambiar la denominación de las clases predeterminadas CSS, por lo que todas las reglas CSS por defecto se aplicarán de forma hereditaria. Esta clase es útil para afinar cualquier componente.

El Código 3.21 muestra la utilización de la clase sclass en un componente.

```
<zk>
<style dynamic="true">
        .mydb {
                border: 1px solid blue;
}
</style>
<textbox sclass="mydb" value="having Sclass"/>
<textbox value="Without Sclass"/>
</zk>
```

Código 3.19: Utilización de la clase sclass.

Fuente: Fuente: Fuente: ZK 5.0.8 Component Development Essentials.

3.3.2.2 Zclass

Es un patrón para nombrar las clases que extienden de las librerías raíz del framework. El nombre asignado a *zclass* se utilizará para nombrar a todas las clases CSS asociadas con la estructura DOM[16] (Document Object Model) del componente. A cada tipo de componente se le asigna una clase única *zclass.*

Las clases ZK también se utilizan para nombrar el estilo CSS que representa las diferentes actividades del usuario, como el click, el enfoque, o pasar el ratón por encima, cuando un usuario mueve su ratón sobre un componente, una clase

[16] http://es.wikipedia.org/wiki/Document_Object_Model

adicional de CSS será añadida automáticamente al elemento raíz del DOM del componente.

El nombre asignado a zclassse utiliza para nombrar las clases CSS asociadas con la estructura de un componente, incluyendo la raíz y los componentes hijos. Además, cada tipo de componente, se asigna con una clase zclass una única y se aplican todas las reglas CSS. La Figura 3.22, muestra un componente personalizado bajo los atributos sclass y zclass.

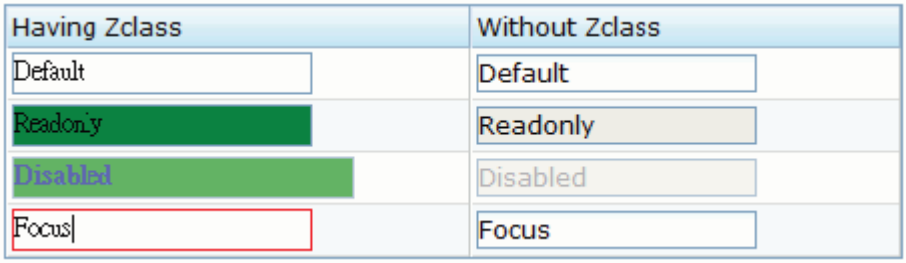


Figura 3.21: Componente ZK personalizado su aspecto CSS

Fuente: Fuente: Fuente: ZK 5.0.8 Component Development Essentials.

3.3.2.3 Personalización

En resumen, hay tres formas diferentes para cambiar el estilo de los componentes de ZK:

1. Cambiar la clase sclass. Si la clase zclass no se cambia, el aspecto original (es decir, las reglas CSS) se aplicarán también. Es útil para afinar un componente particular.
2. Cambiar la clase zclass.Sicambia la clase zclass, todos los elementos secundarios del DOM se asociaran con una clase diferente. Por lo tanto, no

se heredan cualquiera de los aspectos originales, pero es útil si se desea diseñar un aspecto completamente diferente

3. Sobrescribir las reglas CSS. No es necesario cambiar las clases sclass o zclass si se quiere cambiar el aspecto de cada instancia del componente.

La relación de entre sclass y zclass se muestra a continuación en la Figura 3.21.

La clase sclass en el cuadro rojo,se añadirá a la clase CSS del elemento raíz del DOM del componente, mientras que la zclass en el cuadro verde, será utilizada como un prefijo para el nombrar a las clases CSS de los elementos secundarios del componente.

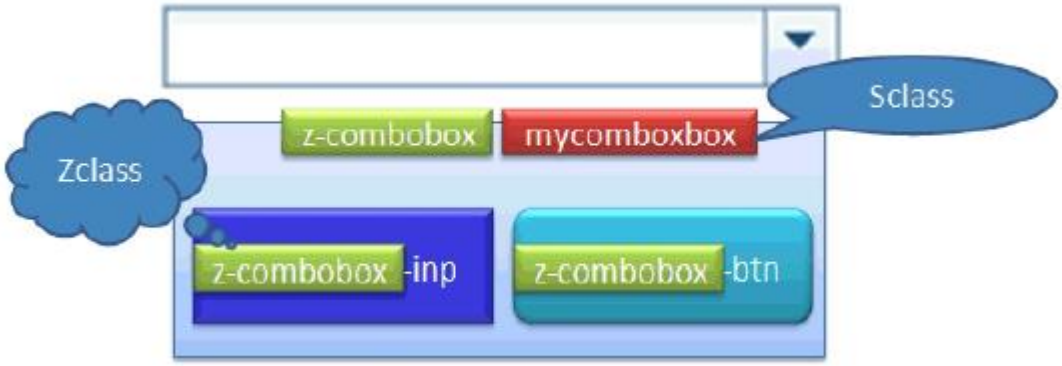


Figura 3.22: Relación de las clases de estilo de un componente ZK.

Fuente: ZK 5.0.8 Style Guide.

3.3.2.4 Sobrescribir estilos

En sustitución a los estilos originales de los componentes, se puede modificar las reglas CSS de este componente. El Código 3.20 muestra a continuación un ejemplo de una caja de texto que tiene sobrescrito su clase de estilos.

```
<zk>
        <style dynamic="true">
                .z-textbox {
                background: black;
                color: white;
        border: 3px outset blue;
}
</style>
        <textbox value="Default"/>
</zk>
```

Código 3.20: Sobrescribir clases CSS que contienen los componentes

Fuente: ZK 5.0.8 Style Guide.

3.4 Comparación entre el framework ZK y el framework JSF 2.0.

3.4.1 Introducción

La siguiente es una comparación de las principales características que tienen en común los frameworks ZK y JSF. No se pretende decidir cual es mejor entre los dos, ya que ambos poseen excelentes características tanto en la arquitectura como en la implementación de aplicaciones dinámicas web.

En la actualidad la especificación JEE6, aconseja utilizar como estándar para el desarrollo de la capa web, el framework JSF en su versión actual 2.0. Este framework, está construido para acoplarse fácilmente a cualquier aplicación JEE, es decir no limita al programador, un modelo de arquitectura de aplicaciones estándar. El gran problema a destacar del framework JSF 2.0, son sus excesivas implementaciones, que tratan de parchar y brindar más funcionalidad al mismo. Estas implementaciones, son frameworks aliados que funcionan con la arquitectura de JSF. Entre los más destacados, están las librerías de Richfaces y Primefaces, que se agregan a la aplicación web.

Esta característica puede ser utilizada y aprovechada al cien por ciento, cuando se tiene más experiencia en el desarrollo de aplicaciones con el framework JSF, ya que en sí, estas implementaciones tratan de enriquecer la funcionalidad y la presentación de los componentes del mismo,pero traen consigo muchos problemas de incompatibilidad y bugs inherentes en a la funcionalidad y arquitectura de sus componentes.

Este problema, hace que la programación no sea tan ágil y que el programador se vea confundido al momento de elegir cual componente de estas implementaciones es el mas adecuado, o simplemente utilizar un componente nativo del framework JSF. Es por ello que se requiere de más experiencia y conocimiento de la arquitectura web y tecnologías como JavaScript y Jquery que nos ayuden a corregir estos errores al momento de utilizar el framework JSF.

Por el contrario, el framework ZK, no tiene implementaciones aliadas, y hace que el funcionamiento de sus componentes y la implementación de tecnologías como Ajax Y JavaScript sea totalmente transparente para el desarrollador web. Esta característica del framework es lo que lo hace más ágil en su programación e implementación en aplicaciones dinámicas enriquecidas.

3.4.2 Arquitecturas

La arquitectura de los frameworks web, juegan un papel muy importante para el rendimiento y la utilización de los mismos, ya que depende de cómo estos han sido construidos, para que después el programador pueda implementar dicho framework en una aplicación de la manera más fácil y rápida.

La arquitectura de un framework, no debería ser un impedimento ni un problema con el cuál el programador tenga que lidiar al momento de desarrollar con cualquier framework web. Por lo contrario, debe brindar al programador las suficientes herramientas tecnológicas para que el desarrollo sea ágil pero robusto a la vez.

3.4.2.1 Arquitectura JSF

El framework JSF, esta construido bajo el patrón de diseño MVC, que en la actualidad es el paradigma de programación más usado por varios frameworks web.

El framework se compone por librerías que residen en el lado del servidor y contienen todos los elementos del framework, incluyendo el conjunto de componentes que se despliegan en las paginas XHTML.

Cómo el framework implementa el patrón MVC, separa la lógica de navegación de las páginas, en clases Java llamadas Controllers o Controladores, los cuales en la última versión del framework ya puedenser programarlos utilizando solo anotaciones.

Tiene como estándar de presentación el lenguaje de marcado XHTML, que es una extensión de la tecnología XML y HTML.

La tecnología Ajax actualmente viene inherente en la arquitectura del framework JSF, esto quiere decir que el desarrollador no debe programar nada extra para que esta tecnología se aplique, lo cual se asemeja mucho a ZK. La actualizacion de secciones en las páginas, esta controlado por el atributo *render* que contienen la

mayoría de los componentes del framework. Este atributo indica que sección debe ser actualizada después de una petición al servidor.

A continuación la Figura 3.23, muestra el ciclo de vida de una petición JSF.

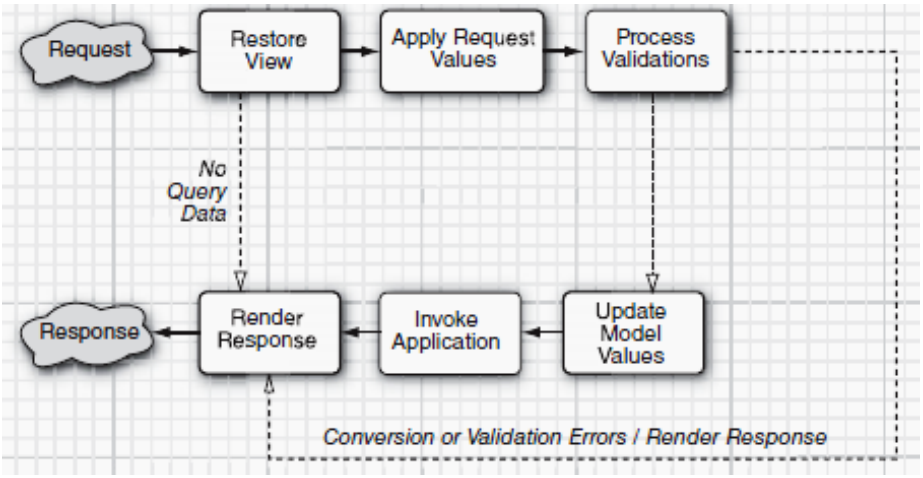


Figura 3.23: Ciclo de vida de una petición JSF

Fuente: David Geary, Cay Horstmann Core Java Server Faces Third Edition

La especificación de JSFdefine seisfases:

1. Restaurar la vista
2. Aplicarlos valores de respuesta
3. Validaciónde procesos
4. Actualización delos valores del modelo
5. Invocarla aplicación
6. Mostrarla respuesta

La fasede restauración,recuperael árbol de componentesdela página solicitada si este ya fue mostrado, caso contrario construyeun árbolnuevo de componentes en el caso que se muestre por primera vez.

Sino hay valoresde petición,la implementaciónJSF se salta todas las fases siguientes y muestra la respuesta, es decir se salta de la fase dos a la fase seis. Esto ocurre cuandouna páginase muestrapor primera vez. Caso contrario,la siguiente fasees lafase en la cual seaplicanlos valores requeridos. Enesta fase la aplicaciónrecorre todos los objetos de la página, buscando si alguno tiene el atributo *required* el cual valida que ese componente no puede tener un valor nulo.En el caso que exista un componente que sea requerido y que tenga un valor nulo, el framework realiza una validación nativa, saltándose a la fase seis una vez más y mostrando un mensaje de error.

La siguiente fase llamada validación de procesos, es aplicar todos los convertidores o validadores que se haya implementado a los componentes en las páginas, por ejemplo un convertidor de String a Integer.

En la siguiente fase se actualizaran los valores a los cuales el componente de JSF está haciendo referencia. Para unir el valor del componente con el valor del modelo, el framework JSF al igual que el framework ZK utilizan Lenguaje de Expresiones.

La siguiente fase es donde se realiza la lógica escrita en los controladores, es decir se ejecutan los métodos que controlan la navegación y funcionalidad de las páginas.

La sexta y última fase, se encarga de cargar el árbol de componentes actualizado, que reflejará los cambios en los valores de los componentes, si ese fuera el caso. Si no hubiera cambios en los valores, o el comportamiento de los componentes, la página se mostraría exactamente igual a antes de la petición.

3.4.2.2 Arquitectura ZK

El framework ZK está estructurado básicamente por tres componentes internos, dos residen en el lado del servidor (ZK AU, ZK loader) y uno en el lado del cliente (ZK Client Engine) como se muestra en la Figura 2.2. A esto se resume la arquitectura del framework ZK ya que todas las demás tecnologías que implementa el framework están abstraídas de forma transparente en estos tres componentes.

Una petición en el framework ZK, está formada por las siguientes fases:

1. Se ejecuta el evento JavaScript.
2. El motor del cliente de ZK capta este evento.
3. El motor de cliente de ZK, envia este comando al servidor a travez de un *XMLHttpRequest.*
4. En el lado del servidor, el motor de actualización asíncrona recibe el comando enviado desde el cliente, y a su vez envía el evento al escucha o *listener* de eventos registrado para dicho componente.
5. El listener o escucha ejecuta la lógica del evento.
6. El listener del evento notifica al motor de actualización asíncrona que la ejecución del evento ha sido terminada.
7. El motor de actualizacion asíncrona prepara una respuesta Ajax y la envía al navegador por medio de unXMLHttpResponse.
8. El motor del cliente ZK, recibe la respuesta y actualiza el arbol de componentes DOM de la página.

3.4.3 Características comunes

3.4.3.1 Modelo Vista Controlador

JSFconectala vistay el modelo de datos utilizando Lenguaje de Expresiones tal como lo hace el framework ZK. Un componentese puede conectar auna propiedad del beande unmodelo de objetos, tales como:*<h:inputTextvalue="#{user.name}"/>.* Las aplicacionesJSFreaccionan ante eventos que se producen en el cliente y que maneja el controlador, dejando la lógica centralizada en el servidor.

El framework ZK, implementa de manera similar el patrón MVC, llamando a sus controladores *Composer*. Además agrega una herramienta muy útil llamada Zscript, que permite fusionar los controladores Java con las páginas ZUL, es decir agregar lógica de navegación y comportamiento en las páginas, lo que agiliza el desarrollo y la ayuda a las pruebas de construcción de componentes. Lo interesante de esta tecnología, es que el código con el que se ingrese en las páginas, no necesariamente debe ser Java ya que se compila con un BeanShell que acepta cualquier lenguaje derivado de Java, como Groovy.

3.4.3.2 Validación ycontrol de errores

JSFvalida los campos de formularios en las pantallas mediante validadores propios del framework, sin embargo también se puede construir validadores personalizados para un caso específico. Cuandolos usuarios introduzcandatos no válidos,se muestran los mensajes de error que son las respuestas a las validaciones que el framework hace en cliente. Sin embargo una validación puede ser omitida si se especifica el atributo *immediate* en los componentes JSF. Este atributo obliga a

saltarse las fases tres, cuatro y cinco del ciclo de vida JSF, actualizando automáticamente el árbol de componentes.

Por el contrario el framework ZK maneja las validaciones en el servidor y también en el cliente gracias al motor del cliente ZK. También se puede utilizar lógica en las páginas para determinar la validación de un campo o una acción. El programador tiene la libertad de elegir donde quiere poner el mensaje de error, y en que momento quiere mostrarlo utilizando eventos programados en las clases Composer.

3.4.3.3 Internacionalización

En JSF se agregan archivos con extensión *.properties*, que sirven para la codificación de caracteresespeciales y la selección depaquetes de recursos. Estos archivos son declarados en el archivo de configuración *faces-config.xml*, en donde el programador deberá agregar la ruta del archivo *.properties*, que se utilizará en la aplicación web. A continuación el Código 3.21 muestra como se agrega un archivo *properties* en el archivo de configuración *faces-config.xml*

```
<resource-bundle>
<base-name>ec.com.smx.impcorp.common.recurso.impcorp</base-name>
<var>msg_impmod</var>
</resource-bundle>
```

Código 3.21: Configuración de un properties en el archivo faces-config.xml

Fuente: El autor.

El framework ZK soporta todos los textos multilenguaje compatibles.El usuario ve los textosen los componentes ZK, definidos previamente en la configuración del explorador, es decir adopta el idioma preferido de cada explorador. Esto se consigue

con lafunción "l" (idioma) agregada en el *taglib*.Las etiquetasse definen al igual que en el framework JSF en un archivo .*properties* en la direcciónWEB-INF/i3-labeldentro del proyecto web. Debe especificarla siguiente líneaen el archivo deZULpara recuperar el texto: *<?taglib uri="http://www.zkoss.org/dsp/web/core" prefix="c" ?>*.

3.4.3.4 Componentespersonalizados

En el framework JSF, los desarrolladores pueden crear componentes personalizados de dos formas. La primera es utilizando una implementación de la arquitectura del framework, que permite extender la funcionalidad de los componentes ya existentes dentro del framework, de acuerdo a las nuevas necesidades que quieran agregar. Estos componentes son llamados: Componentes por composición.

La segunda forma, es desarrollar un componente nativo JSF, es decir crear clases propias que extiendan de la clase base *UIBaseComponet*, e implementar su comportamiento y funcionalidad dentro de una página. Este proceso de desarrollo, requiere mucho más esfuerzo y tiempo que el anterior.

Por ejemplo,si secrea uncomponente por composición *calendario*con todas las característicashabituales, más la utilidades extras que el desarrollador quiera implementar, la utilización de este componente se logra implementando la siguiente línea de código ejemplo*:<personalizados: calendariovalue="#{evento.semana}"startOfWeek="Sat"/>*.

En este ejemplo, la palabra: *personalizados* representa al namespace que se le asignó a la ruta donde están los componentes personalizados dentro de la

aplicación. Cabe señalar que el framework JSF, permite empaquetar todos estos recursos, tales como componentes y estilos, en archivos Java o archivos .*Jar.*

En el caso del framework ZK, uncomponente por composición,esuna colección de componentesZK trabajandoen conjuntopara brindaralgunas funcionalidades extra. Pro ejemplo, si se quisiera construir un componente por composición llamado *imagenEtiqueta*, elcomponente mostraría una imageny una etiquetaconun encabezado.Este ejemplo serepresentaría a partir depocasdeclaracionesde componentesen el archivo ZUL al cual se le quiera agregar el componente.

Uncomponente compuestoextiende de un componenteZKexistente, al igual que en el framework JSF. En el constructor del componente*imagenEtiqueta*, los elementos de interfaz de usuarioasociados con estecomponente secrean enbase en la plantilla*imagenEtiqueta.zul*. Sin necesidad de crear una clase *Composer*, que maneje el comportamiento de el componente creado.

3.4.3.5 Uso de Ajax

El uso de la tecnología AjaxenJSFproporciona un estándaro canal de comunicaciónque seinvocade forma transparenteen las accionesdel lado del servidory las actualizacionesdel lado del cliente.

Se podría de decir que en este punto los dos frameworks convergen de manera similar, ya que no es necesario codificar las peticiones ni las respuestas al momento de utilizar la tecnología Ajax.

3.4.4 Componentes

A continuación se realizó un listado de componentes para cada framework. Cabe señalar que no se pretende hacer una descripción detallada de cada componente perteneciente a los dos frameworks, si no más bien conocer cuales son los componentes que nos brinda cada framework para mejorar la visualización y funcionalidad de las páginas.

3.4.4.1 Componentes JSF

Para listar los componentes que conforman el framework JSF, se ha escogido la implementación Richfaces en su versión 4, ya que brinda una mayor cantidad de componentes y funcionalidades de las que el framework JSF trae de forma nativa.

A continuación la Tabla 3.1 agrupa los componentes que están en la implementación Richfaces 4:

Acciones Ajax	a4j:ajax
	a4j:commandButton
	a4j:commandLink
	a4j:jsFunction
	a4j:poll
	a4j:param
	a4j:param
Colas Ajax	a4j:queue
	a4j:attachQueue
Ajax Output/Contenedores	a4j:outputPanel
	a4j:status
	a4j:region

	a4j:mediaOutput
	a4j:log
Validaciones	Client SideValidation
	rich:graphValidator
	rich:message
	rich:messages
Iteración de datos	a4j:repeat
	rich:dataTable
	rich:extendedDataTable
	rich:collapsibleSubTable
	rich:dataScroller
	rich:list
	rich:dataGrid
Árboles	rich:tree
	TreeAdaptors
Output/Paneles	rich:panel
	rich:togglePanel
	rich:tabPanel
	rich:collapsiblePanel
	rich:accordion
	rich:popupPanel
	rich:progressBar
	rich:tooltip
Menús	rich:panelMenu
	rich:toolbar
	rich:dropDownMenu
Inputs	rich:autocomplete
	rich:calendar
	rich:inputNumberSlider
	rich:inputNumberSpinner
	rich:inplaceInput
	rich:inplaceSelect
	rich:select

	rich:fileUpload
Drag and Drop	Drag and Drop

Tabla 3.1: Listado de componentes de Richfaces 4.

Fuente: El autor.

3.4.4.2 Componentes ZK

A continuación la Tabla 3.2 muestra una agrupación de los components ZK:

Funcionalidad	Context menu, tooltip y popup
	Hflex y Vflex
	Mouseless data entry y navigation
	Drag and Drop
Paneles&Ventanas	Windows (modal, pop-up, overlapped, embedded)
	Panel (embedded, overlapped, minimized, maximized, draggable)
	Macro y composite components
	Hbox, Vbox, Hlayout, Vlayout, Splitter
	Tabbox (accordion, horizontal/vertical layout, lightweight)
	Groupbox (3D, legend) Borderlayout
	Columnlayout
	Portallayout
	Tablelayout
Grid &Árboles	Grid, Listbox, Tree Paging (nested, one-to-many)
	Column menu, sorting, frozen column, checkmark
	Live model-driving Grid, Tree
	Grouping
	Master-detailToolbar&MenuMenu (image, label, file upload, y arbitrary HTML content)
	Hyperlink y redirect
	Fisheyemenu
Formularios e Inputs	Combobox, Bybox
	Number input (BigDecimal, double, int)

	Datebox, Timebox, Calendar
	Slider, Spinner, Progressbar WYSIWYG Editor
	Captcha
	Color picker
Datos & Reportes	Flash Chart Integrate JFreeChart (2D/3D, bar, pie, stock, dial, gantt)
	Integrate JasperReports (PDF/Excel/ODT/XML/HTML)
	MIT Timeline
	MIT Timeplot
	Google Maps XML generator
	Excel-likeSpreadsheet
	Model-drivenPivottable
	Scheduling calendar
Animaciones	Slide, fade, drop, y todas la animaciones Jquery
	Drag-and-drop, drag-and-size Bookmarking/history management (inter-iframe supported)

Tabla 3.2: Listado de componentes ZK.

Fuente: El autor.

3.4.5 Pruebas de rendimiento

En la siguiente comparación,un localizador deGoogle Mapsserá ejecutado por el framework ZK y por la implementación de JSF 2 llamada Icefaces,con el fin dedemostrar la diferencia de rendimiento entrelos frameworks.

Suponiendo quese está codificandounlocalizador deGoogle Mapspara un cliente,los requisitosson:

1. Unainterfaz de usuariocon un campode entrada de texto, un botón y un mapa de Google.

2. Cuando el usuariohace clic enel botón, eltexto enel cuadro de textoseutilizacomo una palabra clavepara encontrarla latitud, longitudy la descripción deese lugar.
3. Visualización de laubicación en elmapa de Googley mostrarla descripcióncomo unaventana de información.

Siya se ha implementadouna clase de Javaque se llama *localizador.java*. Esta clase tomará una cadena de caracteres como entraday devolverla información requerida. Sólo seestá llevando a caboel trabajode lainterfaz de usuario ypartesde la recuperación de datos. El Código 3.21 muestra la implementación en el framework ZK.

ZK (1 Archivo, 36 líneas de código):

```
<?xmlversion="1.0"encoding="utf-8"?>
<zk>
<scriptsrc="http://maps.google.com/maps?file=api&v=2&key=KEY"
type="text/javascript">
</script>
<zscript>
 import com.macroselfian.gps.Locator;
 public void locate(){
 String location = tb.getValue();
 double[] pos = Locator.locate(location);
mymap.panTo(pos[0], pos[1]);
mymap.setZoom(16);
ginfo.setOpen(true);
ginfo.setLat(pos[0]);
ginfo.setLng(pos[1]);
ginfo.setContent(Locator.getInfo(location));
mymap.openInfo(ginfo);
 }
</zscript>
<window>
<divalign="center">
<separatorspacing="50px"/>
<vbox>
<labelvalue="Macroselfian Google Map Locator"/>
<hboxwidth="600px">
<textboxwidth="550px"id="tb"/>
<buttonwidth="50px"onClick="locate();"label="Search"/>
</hbox>
<gmapsid="mymap"zoom="16"...>
```

```
<ginfoid="ginfo"/>
</gmaps>
</vbox>
</div>
</window>
</zk>
```

Código 3.22: Gmap.zul.

Fuente: http://ria.dzone.com/articles/zk-vs-ajax-jsf-components-simp-0.

Adaptarla arquitectura DirectRIA, ZK esmodelo cien por ciento orientado a eventos, que separala interfaz de usuariode la lógica denegocio. Con el Código 3.21 se pueden observar las siguientes ventajas dela arquitectura DirectRIA*:*

1. Los desarrolladores acceden directamente adatos sinclases o *beans*de respaldoadicionales, lo que ofrece un enfoquemás intuitivoque un *managed bean* necesario en la arquitectura de JSF.
2. Sin necesidad de configuraciónadicional paracada componentey de un controlador o Composer que maneje los eventos de la página, seofrece mayor flexibilidadque la tradicional.

A continuación el Código 3.22 muestra la codificación del mismo requerimiento hecho en Icefaces.

Icefaces (3 archivos, 223 líneas de código)

```
<f:viewxmlns:h="http://java.sun.com/jsf/html"
xmlns:f="http://java.sun.com/jsf/core"
xmlns:ice="http://www.icesoft.com/icefaces/component">
<ice:outputDeclarationdoctypeRoot="HTML"
doctypePublic="-//W3C//DTD HTML 4.01 Transitional//EN"
doctypeSystem="http://www.w3.org/TR/html4/loose.dtd"/>
<html>
<head>
```

```
<metahttp-equiv="Content-Type"content="text/html; /><link
href="xmlhttp/css/xp/xp.css"type="text/css"rel="stylesheet">
<style>
 .iceGmp {

 }
 .iceGmpMapTd {
 vertical-align: top;
 }

 .iceGmpMapTddiv.gmap {
 width: 800px;
 height: 600px;
 }
</style>
</link>
</head>
<body>
<ice:form>
<ice:outputConnectionStatus/>
<ice:inputTextvalue="#{locator.location}"width="200px"
partialSubmit="true"/>
<ice:commandButtontype="submit"value="Submit"/>
<ice:gMaplatitude="#{locator.lat}"longitude="#{locator.lng}"
zoomLevel="16">
<ice:gMapControlid="largectrl"name="GLargeMapControl"/>
<ice:gMapControlid="scalectrl"name="GScaleControl"/>
<ice:gMapControlid="typectrl"name="GMapTypeControl"/>
<ice:gMapMarkerid="gmarker">
<ice:gMapLatLngid="glatlng"latitude="#{locator.latMark}"
longitude="#{locator.lngMark}"/>
</ice:gMapMarker>
</ice:gMap>
</ice:form>
</body>
</html>
</f:view>
```

Código 3.23: Gmapx.jspx

Fuente: http://ria.dzone.com/articles/zk-vs-ajax-jsf-components-simp-0.

El Código 3.23 a implementa la clase LocatorBean.java, necesaria para hacer la lógica de negocios y la obtención de datos del ejemplo, utilizando las librerías de Icefaces.

```
packagecom.macroselfian.gps;

publicclassLocatorBean {
Locator _locator;
```

```
String _lng ="0.0";
String _lat ="0.0";
String _lngMark ="0.0";
String _latMark ="0.0";
String _title ="";
String _info ="";

public String getLocation() {
return_locator.getKey();
}

publicvoidsetLocation(String location) {
_locator = new Locator(location);
_lng = _locator.getLng()+"";
_lat = _locator.getLat()+"";
_lngMark = _locator.getLng()+"";
_latMark = _locator.getLat()+"";
_title = _locator.getTitle();
_info = _locator.getInfo();
}

publicLocatorBean() {
setLocation("");
}

public String getLatMark(){
return_latMark;
}

publicvoidsetLatMark(String lat){
_latMark = lat;
}

public String getLngMark(){
return_lngMark;
}

publicvoidsetLngMark(String lng){
_lngMark = lng;
}

public String getLat(){
System.out.println("getLat"+_lat);
return_lat;
}

publicvoidsetLat(String lat){
System.out.println("setLat");
_lat = lat;
}

public String getLng(){
return_lng;
}

publicvoidsetLng(String lng){
_lng = lng;
}
public String getTitle() {
```

```
return _title;
}

publicvoidsetTitle(String title){
_title = title;
}

public String getInfo() {
return _info;
}

publicvoidsetInfo(String info){
_info = info;
}}
```

Código 3.24: LocatorBean.java

Fuente: http://ria.dzone.com/articles/zk-vs-ajax-jsf-components-simp-0.

Por último el Código 3.24 representa la configuración en el archivo faces-config.xml

```
<!DOCTYPEfaces-configPUBLIC
"-//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.0//EN"
"http://java.sun.com/dtd/web-facesconfig_1_0.dtd">
<faces-config>
 ...
<managed-bean>
<managed-bean-name>locator</managed-bean-name>
<managed-bean-class>com.macroselfian.gps.LocatorBean</managed-bean-class>
<managed-bean-scope>session</managed-bean-scope>
</managed-bean>
<!---->
<application>
<locale-config>
<default-locale>en</default-locale>
</locale-config>
</application>
...
</faces-config>
```

Código 3.25: Configuración del archivo faces-config.xml para agregar un bean.

Fuente: http://ria.dzone.com/articles/zk-vs-ajax-jsf-components-simp-0.

Cabe recalcar que esta última configuración se puede omitir utilizando las anotaciones @ManagedBean y @SessionScoped en la clase *LocatorBean.java.*

3.4.6 Tabla de resumen

A continuación se mostrará en la Tabla 3.3, un cuadro de resumen de las características de ambos frameworks.

	JSF	ZK
Lenguaje	Java	Java, Groovy, ZULM
Ajax	Sí	Jquery
MVC	Sí	Sí
MVC push-pull	pull	Push-pull
Internacionalización	Sí	Sí
ORM	Sí, agregando extensiones	Cualquier ORM
Extensiones para pruebas	JUnit	JUnit, ZTL
Extensiones para migración de Bases de Datos		HibernetUtil,SpringUtil
Extensiones de seguridad	Sí	Spring Security
Templates	Facelets, JSP	Macro componentes, composition.
Caching	Sí	Sí
Validaciones	Validaciones nativas	Cliente, Servidor

Tabla 3.3: Tabla comparativa entre el framework ZK y el framework JSF

Fuente: El autor.

3.4.7 Conclusiones de la comparación

Cómo se puede ver en la descripción de ambas arquitecturas, los dos frameworks han sido construidos para simplificar la implementación de la tecnología Ajax.

Se puede apreciar claramente, que el framework ZK maneja de una mejor manera las peticiones que se realizan al servidor, ya que tiene un componente que reside en el cliente y es un conjunto de código JavaScript y Jquery a diferencia del framework JSF que centraliza sus escuchas y componentes internos en el servidor.

El rendimiento del framework ZK en las peticiones Ajax es mucho mejor al del framework JSF, ya que tiene el componente interno llamado ZK AU, que esta reside en el servidor y esta en constante comunicación con el componente de ZK que reside en el cliente. Esto hace que haya una comunicación más directa entre el cliente y el servidor, mejorando el rendimiento de las peticiones Ajax.

En comparación conZK, las implementaciones de JSFson un conjunto de componentes construidos bajo la arquitectura de JSF. Esto indicaque Ajaxu otra tecnología o implementación JSFcomoIcefaces o Richfacesno se excluyendelciclo de vida deJSF y su configuración.

CAPÍTULO 4: PROTOTIPO DESARROLLADO CON EL FRAMEWORK ZK

4.1 Descripción técnica del prototipo

El prototipo implementa un Sistema de Administración de Carreras Profesionales para la empresa Kruger Corporation. Las carreras profesionales, están dadas en base al perfil profesional de cada empleado. El perfil profesional se asigna al empleado al momento que el mismo ingresa a la empresa, este puede variar de acuerdo al departamento al que pertenece.

El usuario tiene la facilidad de entrar al portal de la empresa y dirigirse fácilmente al sistema de administración de carreras llamado Profesional K. Una vez ahí se encontrará con una pantalla de Log In que lo direcciona al sistema de administración de carreras Profesional K.

El prototipo fue desarrollado bajo la plataforma Java JEE. Se utilizó la base de datos MySQL 5.5 dónde se construyó el modelo de datos diseñado previamente.

Para la lógica de negocio se utilizó el framework Spring 3.1, que permitió un desarrolló ágil y con una integración para todas las capas del prototipo. Este se enfocó más en la capa web del sistema, con el propósito de explotar al máximo la capacidad del framework y aprovechar todos los componentes de una manera fácil y óptima. Se incluyó también, librerías extras del framework que contienen componentes fabricados, como un calendario y un lector de noticias RSS que fueron incluidos con el propósito de enriquecer la aplicación web.

El sistema está dividido en 3 proyectos descritos a continuación:

4.1.1 Proyecto de modelo

Llamado *prjModeloPlan*, se encarga de almacenar toda la lógica de negocio del sistema, es decir es aquí donde residen todos los servicios y la implementación de los mismos.Este proyecto se genera como un archivo Java, es decir tiene una extensión tipo Jar y debe residir en las librerías compartidas del servidor. Es aquí donde se implementa el framework Spring y se crean todos los beans necesarios para que exista una integración y administración ágil de los servicios creados. El proyecto está distribuido como lo muestra la Figura 4.1:

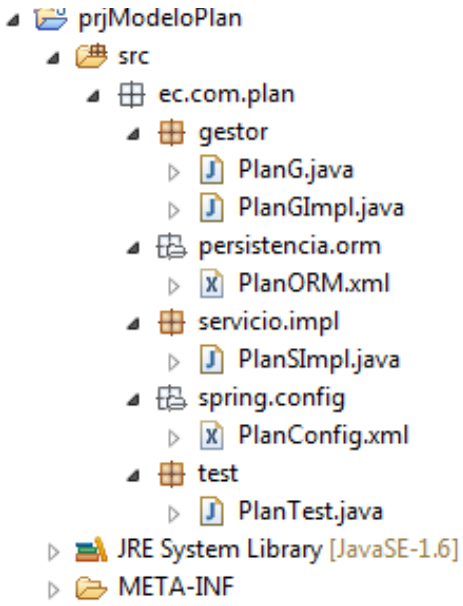


Figura 4.1: Distribución del proyecto *prjModeloPlan.*

Fuente: El autor.

4.1.2 Proyecto cliente

Está definido como *prjClientePlan*, se encarga de implementar todas las clases y servicios que van a ser consumidos por otros clientes o aplicaciones, es decir es aquí donde se pone todos los recursos compartidos se expondrán del sistema. Este proyecto tiene una dependencia con el proyecto modelo *prjModeloPlan* ya que define las interfaces de los servicios que serán implementados posteriormente por el proyecto modelo. Cabe señalar que al igual que el proyecto modelo, este proyecto se genera como un Jar y debe residir en las librerías compartidas del servidor. A continuación la Figura 4.2 muestra como está distribuido el proyecto cliente.

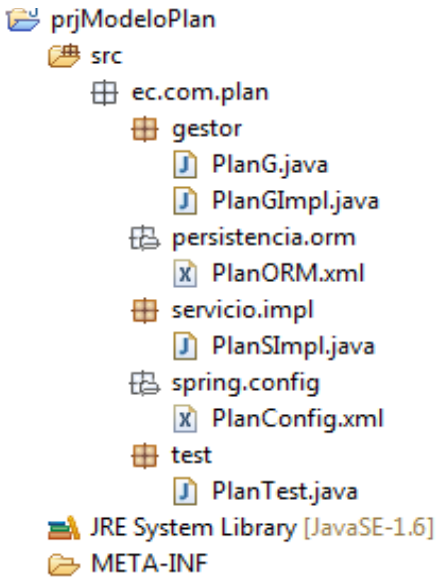


Figura 4.2: Distribución del proyecto *prjClientePlan.*

Fuente: El autor.

4.1.3 Proyecto web

Es un proyecto web dinámico llamado*prjWebProject* y se encarga de almacenar todas las páginas y controladores de las mismas donde reside la lógica de navegación y la programación de los eventos que se ejecutan en las vistas. Este proyecto implementa el framework ZK y es desplegado al igual que los dos proyectos anteriores en el servidor. Este proyecto debe ser generado como un archivo web Java es decir debe tener una extensión War y debe residir en la carpeta donde se desplieguen los proyectos del servidor. A continuación la Figura 4.3 muestra la distribución del proyecto web.

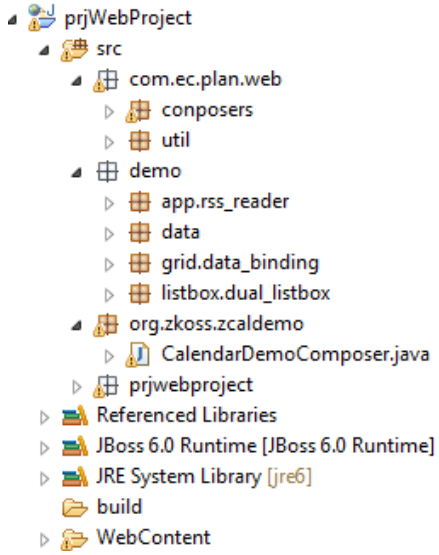


Figura 4.3: Distribución del proyecto *prjWebProject*

Fuente: El autor.

4.2 Manual de usuario

4.2.1 Introducción

A continuación este manual le permitirá a un usuario del sistema como utilizar el prototipo construido con el framework ZK llamado Profesional K, encargado de administrar el plan profesional de cada empleado dentro de la empresa Kruger Corporation.

4.2.2 Ingreso al sistema

El ingreso al sistema Profesional K, se lo debe realizar por medio del portal de la empresa en mención, ingresando a www.kruger.com.ec y dirigiéndose al link que dice Profesional K, tal como lo muestra la Figura 4.4:

Figura 4.4: Link de ingreso al sistema desde la página de Kruger Corporation.

Fuente: www.kruger.com.ec

Una vez ingresado en el link correspondiente, se accede a la pantalla de ingreso del sistema, en donde se ingresael nombre de usuario y la contraseña tal como lo muestra la Figura 4.5:

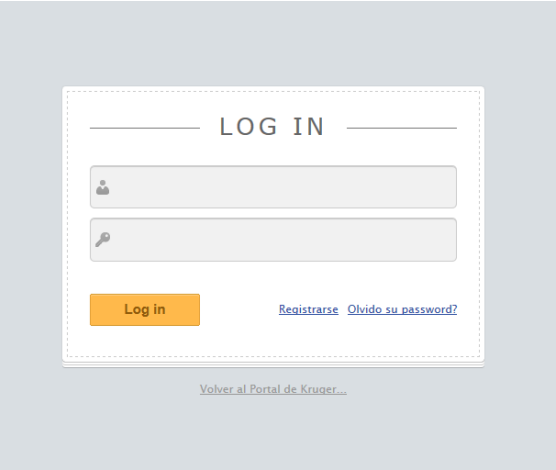


Figura 4.5: Pantalla de ingreso al sistema.

Fuente: El autor

Una vez en esta pantalla,ingresar los datos correspondientes por defecto serán *admin* para el nombre de usuario y *adminadmin* para la contraseña. Después de ingresar estos datos, se presiona el botón de Log In, el mismo que nos re direccionará al menú principal del sistema, tal como lo muestra la Figura 4.6:

Figura 4.6: Menú principal del prototipo desarrollado con el framework ZK.

Fuente: El autor.

En esta pantalla se puedeescoger las opciones del sistema. En la parte superior de la pantalla se tiene cinco links, los cuales nos dirigen al portal de Kruger, a la página de clientes de Kruger, a la página del framework ZK y a la página de Log In del sistema, respectivamente.

A continuación se tiene una compilación de imágenes de interés, construidas con la integración de código HTML 5 en las páginas .ZUL propias del framework ZK.

Debajo de esta colección de imágenes se tiene las tres secciones principales del sistema; empleados, planes de carrera y la sección de noticias de interés, respectivamente.

4.2.3 Administración de empleados

Para ingresar a esta sección, se debepresionar el botón de empleados desde el menú principal del sistema, tal como lo muestra la Figura 4.7:

Figura 4.7: Botón de ingreso a la administración de empleados.

Fuente: El autor.

En esta pantalla se puede realizar las siguientes acciones:

1. Buscar un empleado de acuerdo a los criterios de búsqueda establecidos, por ejemplo la Figura 4.8 muestra cómo el usuario puede buscar un empleado por

el departamento al que pertenece el empleado.

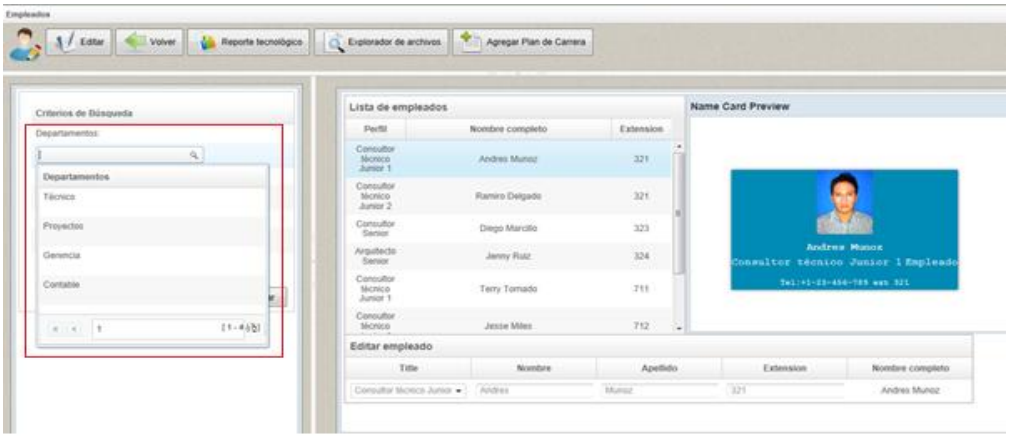


Figura 4.8: Filtro de búsqueda de empleado por departamento

Fuente: El autor.

Cabe señalar que también se puede filtrar a los empleados por sus perfiles y sus nombres, el filtro de perfiles, muestra una imagen en la lista desplegada del componente. La Figura 4.9 muestra el filtro de búsqueda por perfil.

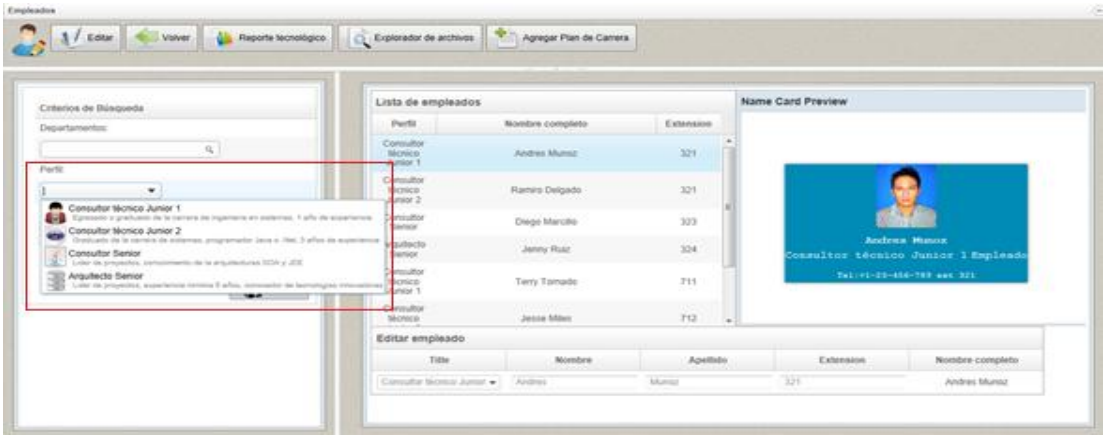


Figura 4.9: Filtro de búsqueda de empleado por perfil.

Fuente: El autor.

2. Una vez filtrado el o los empleados que se buscan, se puede editar los datos principales del mismo, esto se logra cambiando los datos que se despliegan en las cajas de texto que se encuentran en la parte inferior de la pantalla, tal como muestra la Figura 4.9.

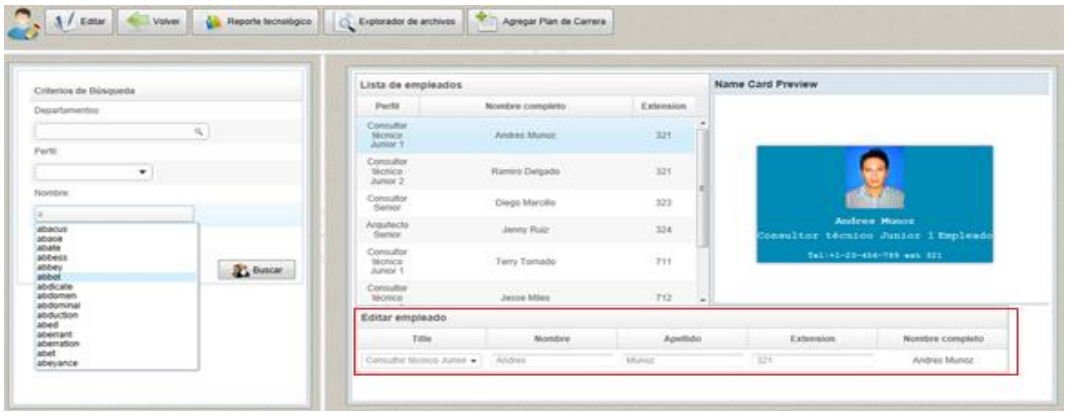


Figura 4.10: Filtros de edición de un empleado.

Fuente: El autor.

Se puede notar que inmediatamente que el usuario cambie los datos, se actualizará el contenido de la tarjeta de presentación en la derecha y los datos de la lista del empleado seleccionado.

3. Se puede obtener una figura estadística tipo pastel que nos indica en porcentajes, cuales son las tecnologías que los empleados conocen. A continuación la Figura 4.10 muestra como se despliega esta utilidad.

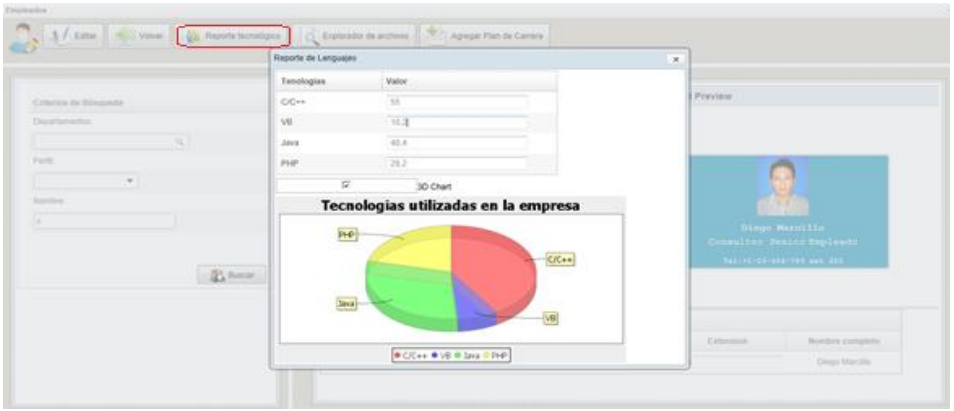


Figura 4.11: Figura pastel de tecnologías utilizadas por los empleados.

Fuente: El autor.

4. Por último se pude visualizar los planes de carrera agregados a cualquier empleado seleccionando al empleado desde la lista desplegada previamente y aplastando el botón: *agregar plan de carrera* en la parte superior de la pantalla. Una vez hecho esto nos aparecerá una ventana emergente tal como muestra la Figura 4.11

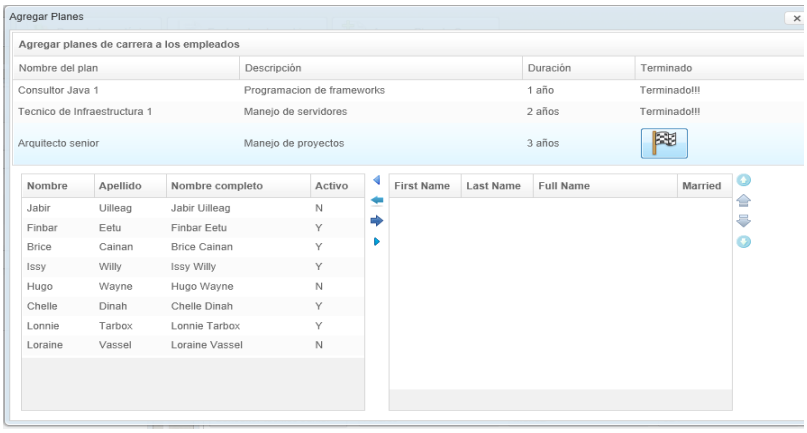


Figura 4.12: Pantalla emergente para agregar un plan de carrera a un empleado.

Fuente: El autor.

4.2.4 Administración de planes de carrera y eventos.

Esta sección se encarga de administrar y visualizar los planes de carrera existentes para cada empleado dentro de la empresa. Se accede a esta pantalla dando un click en el botón del menú principal de la pantalla situado como lo muestra la Figura 4.13.

Figura 4.13: Botón de ingreso a la pantalla de administración de carreara.

Fuente: El autor.

Existe un menú en la cabecera con tres botones de navegación y una tabla en centro de la pantalla que indica el listado de empleados vs las carrearas que se han asignado a los mismos. En la parte superior de la tabla se cambiar el modo de visualización de los datos de una manera dinámica y automática. La Figura 4.14 muestra la pantalla de administración de carrera profesional.

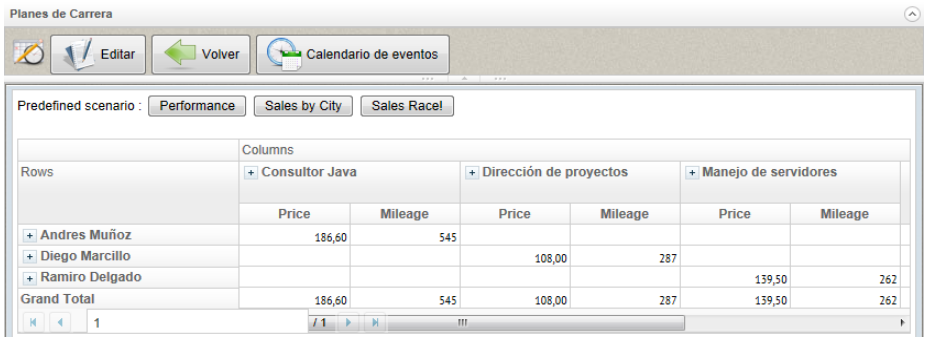


Figura 4.14: Pantalla de administración de carreras profesionales.

Fuente: El autor.

En la pantalla también existe la posibilidad de navegar en un calendario de eventos al cualse accedeal dar click en el botón de *calendario de eventos* en la parte superior de la pantalla.Inmediatamente aparecerá un calendario con todos los eventos creados hasta la fecha actual, donde el usuario tiene la facilidad de establecer un evento asignándole una fecha de inicio y una fecha de finalización, tal como muestra la Figura 4.15.

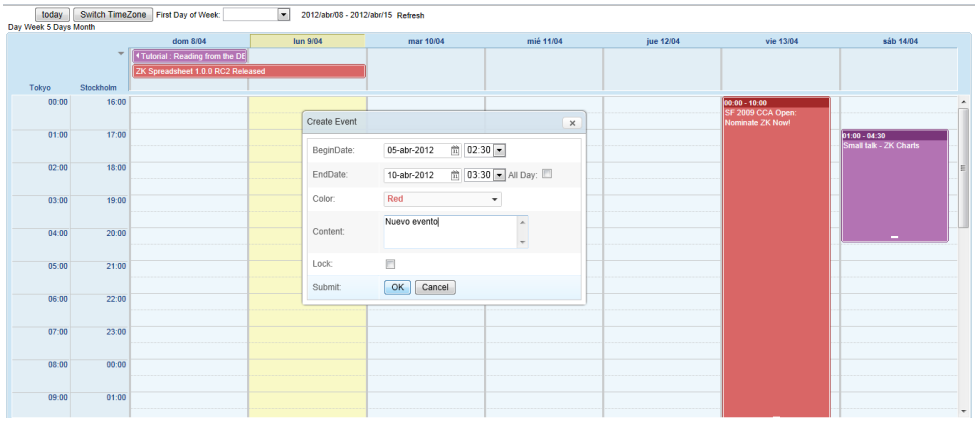


Figura 4.15: Calendario de eventos.

Fuente: El autor.

4.2.5 Lector de noticias

El lector de noticias RSS es una funcionalidad muy necesaria hoy en día en cualquier portal tecnológico o de administración, ya que provee de noticias de interés a cualquier usuario del sistema sin necesidad de navegar a otras paginas relacionadas. Se ingresa a esta sección dando click en el botón derecho del menú principal del sistema, tal como lo muestra la Figura 4.16.

Figura 4.16: Botón de ingreso a la pantalla de noticias

Fuente: El autor.

La pantalla está dividida en dos secciones principales tal como muestra la Figura 4.16. En la sección de la izquierda están todos los titulares de las noticias que recibió el lector desde el internet al momento de acceder a la pantalla y en la parte derecha están los artículos completos de cada noticia.

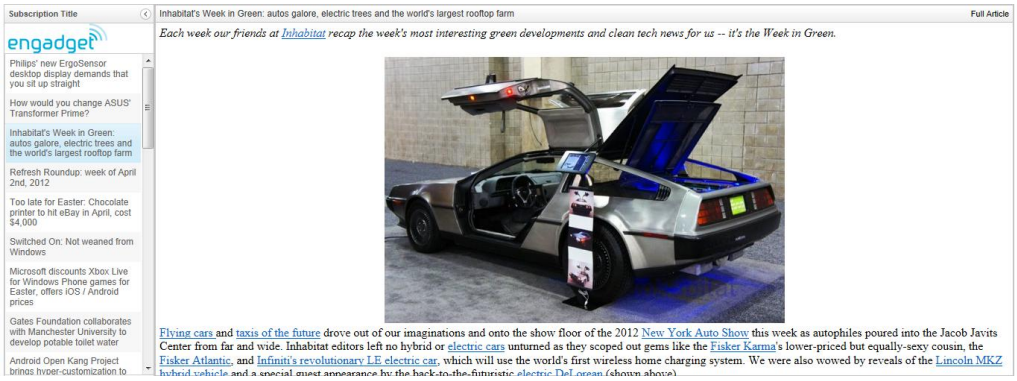


Figura 4.17: Pantalla de noticias utilizando el framework ZK.

Fuente: El autor.

CAPÍTULO 5: CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

1. El framework ZK es ideal para realizar prototipos de aplicaciones web escritas en Java, las mismas que pueden ser adaptadas e integradas con cualquier framework adicional como Spring o utilizar tecnologías de soporte en diferentes capas de la plataforma JEE como Hibernate o JPA.
2. El framework ZK, reduce el tiempo de desarrollo de las aplicaciones web ya que abstrae la implementación de tecnologías importantes para la construcción de aplicaciones webs dinámicas como Ajax y JavaScript.
3. La inclusión de código Zscript en las páginas web, agiliza la navegación de las mismas, y nos permite modelar de manera más óptima una aplicación.
4. A diferencia deotros frameworks dónde aun es necesario programan detalles delas llamadasAjax al servidor, ZKprotege deestas complejidades ypermite al desarrollador centrarse enla lógica de negocio.
5. Los componentes del framework ZK se comportan de igual forma en cualquier navegador web, lo que no sucede con componentes del framework JSF que sufre de incompatibilidades al momento de desplegarse en algunos navegadores.
6. ZKproporciona las herramientaspara suministrarunainterfaz de usuarioenriquecida,con una complejidadmínima. La capacidad de utilizarZULpara crear prototipos deuna pantalla conretroalimentación de usuarios,es muy útil cuando se implementa aplicaciones web dinámicas. El

prototipopuedese puede convertir enla interfazactual,reduciendo el tiempo dedesarrollo inicial.

7. Con la lectura de una guía básica del framework ZK como la expuesta en el presente trabajo, es totalmente posible que un programador inexperto desarrolle una aplicación web con el framework ZK, ya que no necesita tener conocimientos previos de tecnologías embebidas como JavaScript y Ajax, más si tener una noción básica de programación Java y HTML.
8. En relación al framework JSF, la comunidad que brinda soporte al framework ZK no estásegmentada en grupos que hacen implementaciones propias tales como Richfaces o Primefaces en el caso de JSF. Esto potencia al framework al existir una sola implementación del mismo. El framework se enriquece cada día por una comunidad que expone y resuelve desde el problema más sencillo expuesto por el usuario inexperto hasta el caso de uso más complejo.
9. En el prototipo construido en el presente trabajo, se explotó al máximo los componentes del framework ZK, combinando los mismos con tecnologías como HTML5 incluidas en las páginas y recibiendo datos de frameworks como Spring.

5.2 Recomendaciones

1. Usarel framework ZK si se necesita tener un desarrollo ágil y se quiere ver resultados en pantalla en muy poco tiempo.
2. Estudiar la arquitectura del framework, ya que esto hace más fácil aun la programación del mismo al tener una noción clara de como trabajan en conjunto los componentes internos del framework.
3. Incluir código Java en las páginas solo al momento de realizar prototipado o pruebas a los componentes, más no al momento de realizar páginas que saldrán a un servidor en producción. En este caso es recomendable utilizar el patrón de diseño MVC.
4. Visitar la página oficial del framework y descargar gratuitamente los manuales y guías de programación que proporciona la comunidad de desarrolladores, así como también probar el funcionamiento de los componentes en la página, antes de implementarlos el sistema que se esté desarrollando.
5. Construir las aplicaciones empresariales utilizando un proyecto web que consuma recursos del servidor y que se solo se dedique a la navegación y lógica de las páginas y no a la persistencia de datos o lógica de negocio.

BIBLIOGRAFÍA

ZKAjax Without JavaScript™ Framework.(2008) Henri Chenand Robbie Cheng

ZK Developer's Guide (2008). Markus Stauble and Hans ‘Jürgen Schumacher

ZK Essentials (2010). ZK Community

ZK 5.0.8 Developer's Reference (2011). . ZK Community

ZK 5.0.8 Configuration. Reference (2011). ZK Community

ZK 5.0.8 Style Reference (2011). . ZK Community

ZK 5.0.8 Components. Reference (2011). ZK Community

HOJA DE LEGALIZACIÓN DE FIRMAS

ELABORADA POR

_____

Andrés Esteban Muñoz Onofa

DIRECTOR DE LA CARRERA

_____

Ing. Mauricio Campaña

Lugar y Fecha: Sangolquí, 24 de Abril del 2012