UNIVERSIDAD DEL AZUAY

FACULTAD DE CIENCIAS DE LA ADMINISTACIÓN

ESCUELA DE INGENIERÍA DE SISTEMAS

“Evaluación de frameworks realizados en java para aplicaciones on-line”

Tesis previa a la obtención del Título de:

Ingeniero de sistemas

Autor:

Manuel Gerardo Orellana Cordero

Director:

Ing. Pablo Esquivel

Cuenca, Ecuador

2013

DEDICATORIA

Esta tesis la dedico a mis padres, quienes con su paciencia, entrega e incondicional cariño han sabido guiarme en el cumplimiento de esta importante meta en mi vida.

A mis hermanos y hermana que con su gran ejemplo han sido guías en el camino profesional y humano.

A verónica que ha sido un apoyo incondicional en los buenos y malos momentos.

AGRADECIMIENTOS

Agradezco a mis padres que han apoyado incondicionalmente para el logro de cada una de las metas de mi vida,

A mis hermanos por sus muestras de preocupación en cada una de las etapas de esta tesis.

A la Universidad del Azuay y a sus catedráticos por brindar sus valiosos conocimientos, especialmente al Ing. Pablo Esquivel por el apoyo brindado en esta tesis.

A los amigos que han estado pendientes del trabajo brindando ánimos y apoyo en lo que pudieran.

INDICE DE CONTENIDOS

RESUMEN

La construcción ágil y eficiente de aplicaciones web empresariales es hoy en día una necesidad para los desarrolladores de software; debido a esto, el uso de un framework de aplicaciones web es necesario.

El objetivo que se persigue es encontrar la mejor herramienta para el desarrollo de aplicaciones web empresariales desarrollada en java, para cumplir dicho objetivo se plantea primeramente identificar y detallar los tres frameworks web java más populares, luego obtener conocimiento empírica a partir del desarrollo de una aplicación sencilla en cada uno, y por último utilizar un modelo de calidad de software para determinar el mejor.

ABSTRACT

Nowadays the agile and efficient construction of enterprise web applications is a necessity for software developers, which is why the use of a framework to develop web applications is essential.

The goal is to find the best tool for the development of enterprise web applications in java. In order to fulfill this goal, first we propose to identify and detail the three most popular java web frameworks. Next, we need to obtain the empirical knowledge through the development of a simple application in each framework. Finally, we need to employ a quality software model in order to determine which the best one is.

UNIVERSIDAD DEL
AZUAY
DPTO. IDIOMAS

Translated by,
Diana Lee Rodas

INTRODUCCIÓN

En la actualidad existen una gran cantidad de herramientas y lenguajes de programación para poder desarrollar aplicaciones web con calidad, seguridad y con una interfaz gráfica amigable y atractiva para el usuario. Muchas de estas herramientas son Open Source o de código abierto y otras son de pago.

Dentro del medio del desarrollo de software se puede observar que muchos programadores de aplicaciones web no tienen la costumbre de utilizar un patrón de diseño o un estándar para codificar e implementar un software, debido a esta falta de estándares y de herramientas adecuadas el código resulta ilegible y la implementación de mejoras y pruebas resultan muy costosas. Además, otro de los problemas resultantes es que se enfocan en realizar partes de la aplicación que ya están implementadas y probadas por librerías. Este conjunto de librerías y herramientas para el desarrollo en nuestro ámbito de aprendizaje se lo llama *framework*.

Existen diferentes frameworks con propósitos distintos, algunos orientados al desarrollo de aplicaciones web, otros para desarrollar aplicaciones multiplataforma, etc. La llegada de los frameworks web al mundo de la programación significó un gran avance para el desarrollo de aplicaciones y por ende a la evolución del internet. Es así, que gran parte de los programadores web empezaron a usar estos marcos de desarrollo. Debido al incremento en la utilización de estas nuevas herramientas de trabajo se produjo una proliferación de los mismos.

La cantidad de frameworks web actuales es muy numerosa; por lo que, el programador de aplicaciones web al elegir su marco de trabajo observa una amplia gama de opciones. Dichas opciones de trabajo son muy diversas y están desarrolladas en distintos lenguajes, existen frameworks en php, java, python entre otros.

El presente trabajo de investigación se enfocará en la variedad de frameworks realizados en java para aplicaciones web. Los frameworks web java son caracterizados por ser Open Source, y también por la existencia de una gran variedad de los mismos, para dar distintas funcionalidades, herramientas y arquitecturas de desarrollo al programador de acuerdo al tipo de aplicación que este construirá.

Capítulo 1

Frameworks Web

Este capítulo abordará los frameworks web con una visión general, primero desde su definición, componentes, características, tipos, etc. Posteriormente se estudiará los frameworks web java, sus conceptos y un análisis estadístico mediante búsquedas de google de los estos frameworks. Por último, se realizará una breve descripción de los 10 frameworks más populares según los criterios establecidos.

1.1. Definición de framework web

Se define un framework como un conjunto de librerías y componentes junto con una documentación y metodología de uso, que permite diseñar, construir e implementar aplicaciones con mayor calidad y agilidad de programación. Se puntualiza que un framework provee de:

- Librerías: Un conjunto de subprogramas utilizados para el desarrollo de aplicaciones, estos contienen código y datos que proporcionan servicios a programas independientes.
- Componentes: Un componente de software es un elemento de un sistema que ofrece un servicio predefinido, y es capaz de comunicarse con otros componentes, de una manera más simple; un componente es un objeto escrito de acuerdo a unas especificaciones.
- Documentación: Un framework debe ofrecer una cantidad de documentos, foros y tutoriales para el aprendizaje y elaboración de software sobre el mismo.
- Metodología de uso: Cada framework tienen una metodología de uso; es decir, una arquitectura a la cual apegarse para el desarrollo de aplicaciones. Esta arquitectura muestra la separación entre la lógica del negocio y la de la capa de presentación.

Un framework web cumple con las características ya mencionadas y la diferencia sustancial con los frameworks para aplicaciones desktop se centra simplemente en el ambiente para al cual están dirigidos.

1.2. Características de los frameworks web

Existen gran cantidad de frameworks para aplicaciones web cada uno con características que los hacen únicos, Johnson determina las siguientes como las particulares principales de un framework (1).

> *Controladores: Utilizan un solo servlet que tiene función de controlador, para toda aplicación o gran parte de ella.*

Configuración: Una configuración, generalmente escrita en un archivo XML en donde se indicará al servlet controlador a través de propiedades a quien delegar la responsabilidad de atender las peticiones entrantes.

Vistas: Las cuales pueden tener nombres claves sin necesidad que exista una relación con el nombre del archivo de la vista que se tiene que cargar para que sea desplegada. La implementación de una vista con un nombre en particular puede cambiar sin afectar código del controlador.

Además de estas características comunes descritas por Johnson se considera importante acotar las siguientes:

- Abstracción de sesiones: No es necesario manipular directamente las sesiones, el framework debe implementar mecanismos para hacerlo automáticamente.
- Abstracción de URLs: No debe ser indispensable manipular directamente las URLs ya que el framework debería poseer mecanismos que faciliten estas tareas.
- Acceso a datos: Herramientas e interfaces necesarias para integrarse con elementos de acceso de datos tales como JDBC u ORM.
- Autentificación y control de acceso: Mecanismos de identificación de usuarios mediante login y password.
- Integración: Herramientas que faciliten la integración de aplicaciones que usen la misma arquitectura.
- Unificación: Se establece una estructura común para toda la aplicación de desarrollo.

1.3. Tipos de frameworks

Los frameworks pueden clasificarse de distintas formas; es decir, de acuerdo al enfoque que estos ofrecen, de esta manera se tiene las siguientes.

1.3.1 Clasificación de acuerdo a su funcionalidad

- Orientados a la aplicación: Los frameworks de aplicación son aquellos que están basados en los request HTTP; es decir, utilizan el API Servlet. Estos se enfocan en el desarrollo de aplicaciones completas.
- Orientados a la interfaz del usuario: Este tipo de clasificación se refiere a los frameworks que tienen como objetivo el desarrollo de la interfaz del usuario; generalmente, tienen bien separado la lógica de la interfaz, con la del negocio y los datos. La mayoría de estos frameworks web utilizan un modelo MVC (Modelo vista controlador) para cumplir este objetivo. Un ejemplo de este tipo es Java Server Faces.

- Orientados a aplicaciones de publicación de documentos: Los framework orientados a la publicación de documentos se centran en ofrecer un entorno para la publicación y transformación de documentos XML. Como ejemplo, el framework Cocoon que es un framework basado en java que ofrece estas características.
- Orientados al control de eventos: Un framework orientado al control de eventos debe posibilitar las herramientas al desarrollador para controlar cualquier suceso ocurrido durante la ejecución del sistema, pudiendo ser eventos internos o creados por el usuario.
- Orientados a varios elementos: Este tipo de frameworks combinan dos o varias de las tipologías citadas anteriormente.

1.3.2 Clasificación de acuerdo a los componentes que ofrece

- Full-Stack: El objetivo principal de los frameworks full-stack es proveer al desarrollador un conjunto de componentes que abarque todo lo que se necesita para construir una aplicación web, centrándose en la correcta interacción de estos componentes.
- Glue: El objetivo principal de estos frameworks es tener un conjunto de adaptadores e interfaces de código que puedan manejar varios componentes; en otras palabras, que estos componentes funcionen bien al ser utilizados junto a otros frameworks.

1.4 Como elegir un framework

El desarrollador de aplicaciones web al momento de escoger un marco de trabajo para el software a realizar, debe tener en cuenta muchas de las distintas capacidades que el framework puede ofrecerle. La elección es de vital importancia; ya que, el framework se transformará en la columna vertebral del software y el futuro del negocio en el que será implementado depende de que el framework cumpla con todas las características necesarias. Con esto en mente se detalla a continuación algunos aspectos notables que pueden ayudar a la selección del framework.

- Licencia: Las limitaciones legales que ofrece un software es un factor determinante para la elección; ya que, la licencia nos indica lo que se puede y no se puede hacer con él.
- Complejidad: Un framework debe ofrecer suficientes características para su uso efectivo y al mismo tiempo debe ser simple para que pueda ser aprendido con facilidad.
- Patrones de diseño: Un framework debe proveer de una arquitectura de desarrollo, no es simplemente un conjunto de librerías y clases que puedan ser aplicadas. Conocer el diseño del framework es esencial para construir aplicaciones sobre él.
- Disponibilidad de ejemplos: Los desarrolladores del framework deben facilitar un conjunto de aplicaciones relevantes que muestren las funcionalidades del framework.

- Documentación: Un conjunto de documentos con las características y posibilidades del framework.
- Soporte: Es importante saber si el framework tiene una comunidad de soporte para sus usuarios y cuál es el costo.

1.5. Frameworks web java

Los frameworks web realizados en java nacen históricamente cuando la arquitectura J2EE ya no era lo suficientemente productiva, en el sentido que era muy trabajoso el desarrollo de aplicaciones web con el estándar JSP (Java Server Pages) de la época. Si bien el estándar J2EE se adoptó como figura central del desarrollo web, esta presentaba problemas en el control de flujo y en el mantenimiento de páginas web con demasiado código java. Debido a esta necesidad de producir aplicaciones más grandes y robustas nace el concepto de framework en java.

El concepto de frameworks en java fue creado a mediados del 2001 con el primero denominado Struts en su versión 1.0, seguido por WebWork y Tapestry en el 2002, Java server Faces en el 2004 y desde entonces nacen una gran cantidad de frameworks con distintas funcionalidades y de diferentes tipos mencionados anteriormente, teniendo en la actualidad más de 50 frameworks realizados en java para aplicaciones web.

1.6. Frameworks web java existentes y selección de los más usados

Para comenzar en detalle con el estudio de los frameworks web realizados en java; es necesario identificar cuales existen en el mercado. Según la fuente de java-source se encuentran 55 frameworks web java.

En la siguiente figura se detallan los frameworks existentes, ordenados de acuerdo a los más populares con un criterio de su número de coincidencias en google.

Frameworks web registrados en java-source

Framework	Búsqueda en Google	Número de Búsquedas
Spring	Spring + Framework	22.300.000
Richfaces	Richfaces + Framework	11.100.000
Seam	Seam + Framework	10.610.000
Struts	Struts + Framework	10.530.000
JavaServerFaces	JavaServerFaces + Framework	10.100.000

Google Web Toolkit	Google Web Toolkit + Framework	8.800.000
Stripes	Stripes + Framework	7.680.000
Tapestry	Tapestry + Framework	5.960.000
Wicket	Wicket + Framework	5.420.000
RIFE	RIFE + Framework	4.830.000
Oracle Adf	Adf + Framework	4.640.000
JAT	JAT + Framework	4.500.000
Verge	Verge + Framework	4.480.000
Hamlets	Hamlets + Framework	3.730.000
Maverick	Maverick + Framework	3.230.000
MyFaces	MyFaces + Framework	3.070.000
Macaw	Macaw + Framework	2.870.000
wingS	Wings + Framework	2.820.000
Brill	Brill + Framework	2.000.000
DWR	DWR + Framework	1.930.000
Millstone	Millstone + Framework	1.870.000
Cocoon	Cocoon + Framework	1.840.000
SOFIA	SOFIA + Framework	1.700.000
Strecks	Strecks + Framework	916.000
Roma Meta Framework	Roma + Meta + Framework	833.000
Helma	Helma + Framework	651.000
Caramba	Caramba + Framework	619.000
Barracuda	Barracuda + Framework	617.000
WebWork	Webwork + Framework	495.000
Makumba	Makumba + Framework	371.000
IceFaces	IceFaces + Framework	316.000
Vaadin	Vaadin + Framework	312.000
PrimeFaces	PrimeFaces + Framework	271.000
OpenXava	OpenXava + Framework	232.000
RSF	RSF + Framework	209.000
Restlet	Restlet + Framework	207.000
Sombrero	Sombrero + Framework	180.000

Vroom	Vroom + Framework	165.000
Chrysalis	Chrysalis + Framework	149.000
Mentawai	Mentawai + Framework	109.000
JVx	JVx + Framework	97.500
Anvil	Anvil + Framework	92.200
ThinWire	ThinWire + Framework	56.200
WebOnSwing	WebOnSwing + Framework	47.200
Aranea Web Framework	Arena + Web + Framework	40.700
Calyxo	Calyxo + Framework	35.100
AribaWeb	AribaWeb + Framework	29.800
JOSSO	JOSSO + Framework	29.600
JPublish	Jpublish + Framework	27.000
SwingWeb	SwingWeb + Framework	19.000
Pustefix	Pustefix + Framework	17.200
SerfJ	SerfJ + Framework	15.900
jZeno	JZeno + Framework	12.800
Swinglets	Swinglets + Framework	8.510

Tabla 1.1: Frameworks Web ordenados mediante coincidencias de Google.

Para el presente trabajo de investigación se tomaran los 10 frameworks web java más populares y se realizará una breve descripción de los aspectos más relevantes de cada uno, con sus ventajas y desventajas. En el capítulo dos se profundizará el estudio a tres de estos frameworks, exponiendo sus características, funcionalidad y su arquitectura.

1.7. Descripción básica de los diez frameworks más populares

Los diez frameworks web java más populares para el desarrollo de aplicaciones on-line se exponen en el siguiente cuadro:

Frameworks web más populares

Framework	Búsqueda en Google	Número de Coincidencias
Spring	Spring + Framework	20.300.000
Richfaces	Richfaces + Framework	11.100.000

Seam	Seam + Framework	10.610.000
Struts	Struts + Framework	10.400.000
JavaServerFaces	JavaServerFaces + Framework	10.100.000
Google Web Toolkit	Google Web Toolkit + Framework	8.800.000
Stripes	Stripes + Framework	7.680.000
Tapestry	Tapestry + Framework	5.960.000
Wicket	Wicket + Framework	5.420.000
RIFE	RIFE + Framework	4.830.000

Tabla 1.2: Los diez Framework con más coincidencias.

A continuación se detalla las características más importantes de cada uno de los frameworks expuestos en la tabla 1.2:

1.7.1. Spring

Spring es un framework de código abierto, creado con el objetivo de facilitar el desarrollo de aplicaciones empresariales. Fue desarrollado gracias a la colaboración de grandes programadores, quienes lograron combinar las siguientes herramientas javas J2EE, EJB, Servlets y JSP, en un solo framework de aplicación al cual llamaron Spring. Se lo considera un framework liviano de gran funcionalidad; ya que, no requiere grandes recursos para su ejecución y aporta una gran funcionalidad al desarrollador.

Spring ofrece un modelo MVC para el desarrollo de aplicaciones web, con lo que da solución a varias de las capas de la arquitectura Web como: presentación, lógica del negocio e integración de datos. De igual manera, ofrece soporte AOP (Programación orientada a Aspectos) y abstracciones para manejo de datos JDBC.

En la parte central de Spring se incorporan los conceptos de IoC (Inversión de control) y DI (Inyección de dependencia). Estos se centran en el principio de que en lugar que el código de la aplicación llame a una clase de una librería, el framework llama al código y de esta acción nace el nombre de la tecnología ya que se invierte la acción de la llamada.

Características

- Proporciona el patrón MVC para la construcción de aplicaciones.

- Utiliza inyección de dependencias para independizar el código del negocio con el del framework.
- Ofrece AOP para centrar el código en las preocupaciones y aspectos importantes del negocio.
- Es un framework liviano y no invasivo.
- Usa objetos POJO en la implementación de la lógica del negocio.
- Permite la integración con otros frameworks para el mapeo de objetos de la base de datos.

Ventajas

- La inyección de dependencia aporta soporte para otros frameworks.
- Soporte JDBC.
- Mucha documentación disponible.
- Herramientas adicionales como Spring IDE.
- Es un framework completo para todas las capas.

Desventajas

- Su configuración es costosa, se requiere mucho código XML.
- Curva de aprendizaje media-alta.
- Los fallos solo pueden detectarse en tiempo de ejecución debido a la inversión de control.
- Es demasiado flexible, tanto que no existe un controlador padre.

1.7.2. Richfaces

Richfaces es un framework web open-source que extiende el framework JSF, añadiéndole capacidades Ajax sin la necesidad de usar JavaScript. Se caracteriza por una gran facilidad de integración de Ajax en la vista y que sus componentes están listos para su uso, con lo cual el desarrollador puede ahorrar tiempo de inmediato aprovechando las características de estos componentes. Además, ya que es una extensión de JSF implementa también el patrón de diseño MVC.

Richfaces se crea a partir del framework Ajax4jsf y en la actualidad es mantenido por JBoss y Red Hat. En un comienzo Richfaces era una librería de componentes comercial, luego JBoss y Red Hat convinieron que sea un proyecto open source.

El framework funciona mediante sus propias etiquetas, las cuales generan eventos que envían peticiones al contenedor Ajax. Estos eventos son generados según acciones del usuario, disparando la funcionalidad antes descrita. De acuerdo a la forma de funcionamiento se concluye que el desarrollador no tiene que preocuparse de crear el código JavaScript.

Características

- Extiende la funcionalidad de JSF.
- Incorpora Ajax en sus componentes.
- Se centra en la creación de páginas web ricas.
- Orientado a la interfaz de usuario.

Ventajas

- Se integra perfectamente en el ciclo de vida de JSF.
- Incluye funcionalidades Ajax.
- Contiene elementos visuales para el desarrollo de una aplicación web rica.
- Soporte de CSS themes o skins.
- Tiene una comunidad activa.
- Mejor aspecto y más componentes que JSF

Desventajas

- Alto consumo de memoria del navegador
- Requiere de conocimientos JSF

1.7.3. Seam

Seam es un framework de integración de tecnologías que tiene como objetivo facilitar el desarrollo de aplicaciones JEE. Es un proyecto open-source creado por JBOSS que lo provee de una comunidad de respaldo. El aspecto más relevante de Seam está en su arquitectura; ya que, integra el uso de varias tecnologías existentes para la creación de aplicaciones web. Por lo que facilita la implementación del modelo MVC de forma intuitiva y rápida para la programación, pero sin perder la potencia y características de JEE.

Características

- Define un modelo de componentes uniforme para la implementación de la lógica del negocio.
- Integración de JSF con EJB3, Seam unifica el framework de presentación JSF (Java Server Faces) y de lógica del negocio con EJB3 (Enterprice Java Bean en su versión 3)
- Implementación AJAX, permite la utilización de tecnologías de dos frameworks JSF basados en AJAX como lo son Richfaces y IceFaces.
- Facilita la opción de manejo de procesos de negocio transparentes con jBPM.
- Poca dependencia de XML con la implementación EJB3 para los archivos de configuraciones exceptuando JSF que sigue siendo muy dependiente de XML.

Ventajas

- Integra varias tecnologías estándar.
- Permite al desarrollador usar una arquitectura uniforme en sus aplicaciones con el modelo MVC
- Integra Ajax a sus aplicaciones sin la necesidad de escribir código JavaScript.
- Poca dependencia de XML para la configuración.
- Aplicaciones más robustas pues se centra en el desarrollo y no simplemente en servir paginas HTML.

Desventajas

- Curva de aprendizaje media-alta.
- Demasiado contenido lo que puede hacer que el desarrollador no aproveche todas sus funcionalidades.

1.7.4. Struts

Struts fue el primer frameworks J2EE que salió al mercado. Es un framework que implementa el patrón de diseño MVC y básicamente está construido sobre las tecnologías de Servlet, JSP, JavaBeans y XML. Puede operar bajo cualquier contenedor de Servlets. Aunque ofrece un patrón de diseño MVC, no proporciona características propias para el desarrollo de la capa del modelo; es decir, a la hora de afrontar un proyecto será flexible a utilizar distintos objetos de negocio como lo son JavaBeans, Java Data Objects, u objetos de acceso de datos

Características

- Struts proporciona un controlador el cual es el corazón del framework.
- Proporciona acceso rápido a las clases java desde la vista.
- Simplifica mucho el proceso de desarrollo de la capa de presentación, permitiendo dedicar más tiempo a codificar la lógica del negocio.
- Las interrelaciones entre Acciones y páginas u otras acciones se especifican por tablas XML.
- Librerías de entidades para facilitar las operaciones que realizan las páginas JSP.
- Contiene herramientas para validación de campos de plantillas bajo varios esquemas que van desde validaciones locales en la página, hasta las validaciones de fondo hechas a nivel de acciones.
- Soporte para Ajax.

Ventajas

- El objetivo del proceso de desarrollo es la calidad del producto.
- Muy popular, con mucho material disponible.
- Curva de aprendizaje media.
- Apropiado tanto para desarrolladores independientes como equipos grandes de desarrollo.
- Permite crear aplicaciones Web de manera rápida y efectiva.
- Admite plug-ins para poder incrementar su funcionalidad.
- Se adapta a la incorporación de diferentes bibliotecas de tags.

Desventajas

- No está diseñado para facilitar la creación de componentes propios.
- Las vistas quedan atadas al dispositivo en el que se renderizan.
- No define de manera adecuada el nivel de lógica de negocio.
- Muy Orientado a la capa de presentación.

1.7.5 JavaServerFaces

Java Server Faces (JSF) es el framework de aplicaciones web creado por Sun Microsystems, basado en el patrón MVC, fue liberado en marzo del 2004. Este marco de trabajo está destinado a facilitar el desarrollo de interfaces para entornos web. JSF facilita el desarrollo de interfaces gráficas de usuario y proporciona una muy extensa colección de herramientas pre-desarrolladas para

la creación. Así mismo, realiza una separación entre comportamiento y presentación al implementar el modelo MVC.

Características

- Orientado a la interfaz gráfica de usuario.
- Ofrece una separación bastante clara entre la lógica de presentación y la del negocio.
- Flexible, posibilita al desarrollador crear sus propios componentes o modificar los existentes.
- Proporciona una arquitectura rica para manejar el estado de los componentes, procesar datos, validar la entrada del usuario y manejar eventos.
- Introduce los términos Manage Bean y Backing Bean al mundo java.
- Implanta en la aplicación web un controlador para la gestión de formularios y navegación entre páginas.

Ventajas

- Es un estándar de desarrollo ya que fue creado Sun Microsystems.
- Distintos fabricantes implementan JSF como: MyFaces, IceFaces, PrimeFaces entre otros.
- Ha aprendido de otros frameworks.
- Mejora el concepto de componentes UI.

Desventajas

- Utilizar AJAX no es sencillo.
- No hay suficiente documentación.
- Su éxito se basa en el número de componentes.
- No se desempeña bien en la seguridad de la aplicación.

1.7.6. Google Web Toolkit (GWT)

GWT es un framework de desarrollo web de google para facilitar la creación de aplicaciones AJAX y mejorar radicalmente la experiencia de los usuarios con la web, permite a los desarrolladores utilizar el lenguaje de programación java para construir aplicaciones independientes del navegador.

Características

- Componentes de la interfaz de usuario dinámicos.

- Procedimientos de llamadas remotas (RPC): permite el paso de objetos java entre cliente y servidor sobre HTTP.
- Depuración en tiempo real: GWT traduce el código java a JavaScript al momento de la ejecución en el browser. Para el desarrollo de la aplicación esta corre sobre una java virtual machine, lo que permite la posibilidad de depurar la aplicación durante su construcción.
- Interoperabilidad: Permite mezclar código JavaScript cuando las librerías de GWT no sean las suficientes para las necesidades de la aplicación.
- Compatibilidad: Usa un compilador de Java a JavaScript lo que lo hace compatible con la mayoría de navegadores más utilizados.

Ventajas

- Interfaces finales similares a las de aplicaciones desktop, alto dinamismo entre pantallas.
- No es esencial tener conocimientos en JavaScript.
- Multiplataforma y Multinavegador.
- Reduce la carga en el servidor.
- Permite gran seguridad.

Desventajas

- Curva de aprendizaje alta.
- Despliegue de la aplicación costoso para el navegador.
- Consumo de memoria del navegador.

1.7.7. Stripes

Stripes es una versión simplificada Struts, conserva parte de la estructura y funcionalidad pero facilita un gran número de tareas pequeñas que en su versión padre no estaban bien definidas. Aporta un conjunto de nuevas funcionalidades como transparencia en la subida de ficheros, manejo de errores y excepciones, validaciones, pruebas unitarias, manejo de estados y la posibilidad de integrar Ajax.

Características

- Asocia cada formulario con un ActionBean; es decir, proporciona un modelo MVC.
- Brinda un nivel de seguridad llamado Stripes-security.
- No existen archivos de configuración son reemplazados con anotaciones java.

- Compatibilidad con Ajax pero no proporciona soporte nativo.
- Proporciona un conjunto de plantillas para crear vistas de manera simple.
- Manejo de URLs amigables.
- Provee soporte para integración con Spring framework.
- Compatible con Google App Engine

Ventajas

- Curva corta de aprendizaje.
- Para su puesta a punto solo es necesario configurar dos archivos (Stripes filter y Stripes Dispatcher).
- Provee una arquitectura de desarrollo MVC.
- Emite archivos de configuración.
- Su sencillez lo hace estable.

Desventajas

- Poca documentación.
- Hereda los problemas de Stripes.
- Problemas de validación.
- No proporciona soporte nativo para Ajax.

1.7.8. Tapestry

Tapestry en un framework de aplicaciones orientado a componentes, se caracteriza por ser una plataforma de creación de páginas web dinámicas, interactivas, robustas y altamente escalables. Construye aplicaciones complejas desde componentes simples y reusables, los cuales son la base de su funcionamiento. Entrega una arquitectura basada en componentes, permitiéndole asumir responsabilidades que facilitan el desarrollo como lo son la construcción y envió de URL, almacenamiento de estado persistente en el cliente o en el servidor y validación de entrada de usuario.

Características

- Las páginas son un tipo especial de componente que no pueden tener parámetros.
- Utiliza plantillas, estas son ficheros en formato HTML plano donde se define la apariencia visual de la página.
- Cada página se implementa con una clase.

- Para la resolución de expresiones usa una librería OGNL (Object graph navigation lenguaje).
- El desarrollo de componentes es similar al de las páginas, con la diferencia que los componentes si pueden recibir parámetros.

Ventajas

- Simplicidad en el desarrollo de interfaces web.
- Permite aplicaciones ricas e interactivas con la construcción de componentes.
- Diseñados los componentes necesarios, construir la aplicación no es más que unirlos facilitando así la reusabilidad y mantenimiento.
- Las aplicaciones son fácilmente escalables, con la construcción de nuevos componentes.
- Aporta modos de diagnóstico para errores.
- Forma parte del proyecto Apache.

Desventajas

- Su documentación es escasa.
- Poca bibliografía.
- Desarrolladores java no ven aceptable que no use JSP.
- Curva de aprendizaje media-alta pues cambia el paradigma de los Servlet típicos de java.

1.7.9. Wicket

Es un framework ligero para aplicaciones web basado en componentes creado por Apache Foundation. Separa la lógica del negocio con la lógica de presentación y estas se integran simplemente con una Wicket id proporcionada por el framework. Debido a esta forma de manejo del desarrollo hace que la programación de aplicaciones web sea simple.

Características

- Las páginas y componentes son objetos java que soportan encapsulación, herencia y eventos.
- Las instrucciones de HTML y Java son paralelas y están asociados solo por una id del framework por lo que facilitan su programación.
- Las URLs no exponen información sensible de las sesiones ni de los componentes.
- Elimina la necesidad de manejar HttpSession a mano.

- Tiene soporte para todas las características básicas de HTML.

Ventajas

- Con el uso de este framework se pueden desarrollar aplicaciones fáciles de mantener y de probar.
- Ideal para desarrolladores Java pues mantiene los principios del lenguaje.
- Se puede usar otras herramientas para el desarrollo de la interfaz ya que usa solamente código HTML en la capa de presentación.
- Maneja las sesiones automáticamente.

Desventajas

- Necesidad de un conocimiento avanzado en programación orientada a objetos.
- Las plantillas HTML coexisten junto con el código java.
- Todo debe hacerse en java.

1.7.10. RIFE

Rife es un framework con una completa guía de herramientas y APIs para la implementación de las características comunes de la web. Cada una de las herramientas es usable por si sola pero en conjunto pueden ofrecer poderosas características de integración que ayudan a incrementar la productividad; por lo tanto, se puede decir que el framework fue diseñado para una perfecta separación de tareas durante el ciclo de desarrollo. Con este marco de trabajo cada declaración y definición es manejado en una sola parte del código.

Características

- El motor web de RIFE ofrece alta prioridad al mantenimiento de aplicaciones pero sin comprometer la productividad de las mismas.
- Los estados de las transacciones deben declararse en una estructura del framework lo que hace que todos los desarrolladores conozcan la lógica y el flujo de datos de la aplicación.
- Agrega inyección de control a sus páginas.
- Proporciona un marco de gestión de contenidos que se orienta a los procesos de almacenamiento, carga, validación, transformación y transmisión de diferentes tipos de contenido.
- Brinda su propio motor de autentificación y control de sesiones.

- Las definiciones de tareas pueden opcionalmente almacenarse en una base de datos, de esta manera están presentes luego de reiniciar la aplicación.

Ventajas

- Perfecto para el desarrollo en grupos, ya que cada desarrollador se dedica a una tarea específica para luego integrarlas.
- Las aplicaciones web pueden ser probadas fuera de un contenedor web.
- Brinda mecanismos de autentificación y control de sesiones.
- Todo el personal de desarrollo conoce la lógica del sistema.

Desventajas

- Se corre riesgos con la integración de las distintas tareas.
- Se pueden obtener efectos no deseados al ensamblar el sistema.

1.8. Selección de los frameworks para investigación

El cuadro de búsquedas en google del punto 1.6 muestra los frameworks java existentes y reconocidos en el mercado, para el posterior estudio y desarrollo se toma como referencia los tres primeros frameworks; ya que, estos de acuerdo a las búsquedas muestran ser los más populares de la web. A continuación los frameworks que serán usados para la aplicación práctica y posteriormente para la evaluación.

Frameworks web más populares

Framework	Búsqueda en google	Número de Coincidencias
Spring	Spring + Framework	20.300.000
Richfaces	Richfaces + Framework	11.100.000
Seam	Seam + Framework	10.610.000

Tabla 1.3: Frameworks Web seleccionados para investigación.

1.9. Conclusiones

Los desarrolladores de aplicaciones web cuando buscan integrar un framework a su trabajo se encuentran con una abrumadora cantidad de opciones lo cual hace muy difícil la elección. Si bien los frameworks nacieron con el objetivo de facilitar, simplificar y construir aplicaciones robustas, la proliferación de estos complica mucho la tarea de elección. La tabla 1.1 ilustra esta situación; ya que, se encuentra en la fuente de java-source 55 framework para el desarrollo.

La proliferación de los framework web trae consigo distintos tipos de desarrollo y arquitecturas que dependiendo de la necesidad del desarrollador se deben tomar en cuenta para la elección; es así que las últimas tendencias toman como referencia al patrón de desarrollo MVC o la orientación a componentes. Estas dos arquitecturas tratan de separar de distintas maneras la lógica del negocio, la de presentación y la de datos y cada framework lo implementa de manera distinta.

Una de las principales características de los actuales framework es su integración con Ajax, unos lo manejan de manera nativa mientras que otros necesitan acomodarse a esta funcionalidad.

No hay ningún framework que exista que solucione todos los problemas del desarrollo; ya que, cada uno ha sido desarrollado con objetivos diferentes, por lo tanto al elegir un framework se debe centrar en cuáles son las necesidades y objetivos del proyecto a realizar; por lo que, se debe escoger de acuerdo a esto el mejor framework evaluando sus ventajas y desventajas.

Capítulo 2

Características y evaluación teórica frameworks web java

Este capítulo abordará los tres frameworks web realizados en java más populares de acuerdo al cuadro de búsqueda y análisis previamente realizado. Para esto, se detallará las características esenciales de cada uno de los siguientes:

Frameworks web más populares

Framework	Búsqueda en Google	Número de Coincidencias
Spring	Spring + Framework	22.300.000
Richfaces	Richfaces + Framework	11.100.000
Seam	Seam + Framework	10.610.000

Tabla 2.1: Framework Web seleccionados para investigación.

Al terminar este capítulo se tendrá una concepción clara de las características, ventajas y desventajas de cada uno de los frameworks descritos en la tabla 2.1; así también, se realizará una evaluación mediante el modelo QSOS.

Una de las principales directrices para el desarrollo de aplicaciones web en cualquiera de estos frameworks es el patrón de diseño Modelo-Vista-Controlador, este patrón es usado en la gran mayoría de los frameworks web java existentes; por lo que, se considera de gran importancia definirlo antes de profundizar con el estudio de Spring, Richfaces y Seam.

2.1 Modelo-Vista-Controlador (MVC)

MVC es un patrón de arquitectura para el desarrollo de software, su funcionalidad reside en la separación de una aplicación en tres componentes distintos. La separación se centra en la interfaz de usuario, la lógica del negocio y los datos de la aplicación.

Características de la arquitectura:

- Modelo: Encapsula los datos y las funcionalidades. Es la representación específica de la información con la cual el sistema opera.
- Vista: Muestra la información al usuario, es la interfaz de la aplicación.

- Controlador: Recibe las entradas del usuario, este traduce los eventos a solicitudes de servicio para el modelo o la vista.

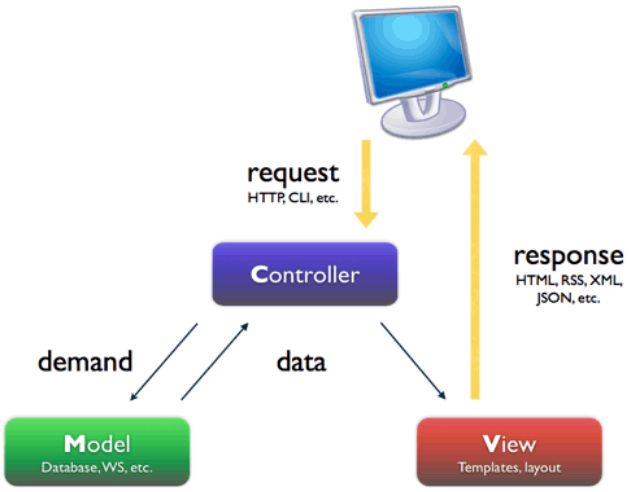

Figura 2.1: Modelo vista controlador. (2)

Al detallar lo expuesto en la figura 2.1 se puede resumir el flujo de control de la arquitectura de la siguiente manera:

1 El usuario interactúa con la interfaz de usuario de alguna forma (Vista).
2 El controlador recibe la notificación de la vista y gestiona el evento.
3 El controlador accede al modelo, actualizándolo o extrayendo datos de acuerdo a la acción del usuario.
4 El controlador delega a los objetos de la vista, la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario.
5 La interfaz del usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

2.2 Spring Framework

Spring es un framework open source, creado por Rod Johnson. Fue desarrollado para manejar toda la complejidad de aplicaciones empresariales en un mismo entorno. Es un framework liviano y no intrusivo; es decir, generalmente los objetos que el desarrollador programa no tienen dependencias en clases específicas de Spring.

2.2.1Tecnologías utilizadas

Spring ha sido realmente una fuente de innovación, muchas de las tecnologías existentes fueron creadas para este framework y luego se vieron popularizados en muchos otros. Las tecnologías que usa principalmente son: Inyección de dependencias y Programación Orientada a Aspectos. (1)

2.2.1.1 Inyección de dependencias (IoC)

La tecnología con la que Spring más se identifica es la inyección de dependencias. Esta tecnología se puede explicar desde el principio de “no me llames, yo te llamo”, en una aplicación java tradicional el código es el responsable del flujo de control, llamando a las librerías del framework según este lo necesite. Con el principio mencionado el código del framework es el que se encarga de invocar al código de la aplicación; es decir, en vez de que la aplicación llame al código del framework, sucede lo contrario.

IoC es un concepto mucho más amplio pues incluye EJB, el modelo del Servlet y la forma en la que Spring usa los llamados a las interfaces para permitir la adquisición y liberación de recursos como conexiones JDBC. Es así como el creador del framework define la inyección de control de la siguiente manera:

> “La inyección de dependencia está basada en los constructos del lenguaje java, más que en el uso de una interfaz específica para un framework. En vez que el código de una aplicación use el API de un framework para resolver dependencias tales como configuración y objetos colaboradores, las clases de la aplicación exponen sus dependencias a través de sus métodos o constructores, así el framework puede llamar al cualquier valor apropiado en tiempo de ejecución, basado en su configuración” (1)

La inyección de dependencias tiene un número de ventajas sobre la configuración tradicional, como un ejemplo las siguientes:

- Las clases de la aplicación se auto documentan.
- La documentación de dependencias está siempre a la fecha.
- Existe poco o nada XML para el manejo de la configuración, todo está hecho a través de lenguaje java.
- El código de las clases de la aplicación se centra solamente en la lógica del negocio. No existe la necesidad de ocupar tiempo en escribir el código de configuración en las clases.

2.2.1.2 Programación orientada a Aspectos (AOP)

La programación orientada a aspectos provee una forma diferente de pensar acerca de la estructura del código en comparación a la programación orientada a objetos (OOP). En la OOP se modelan objetos del mundo real o conceptos y se los organiza con herencias. AOP al contrario nos posibilita pensar acerca de preocupaciones o aspectos del sistema en si, como el manejo de transacciones que siempre es aspecto fundamental en una web con alta transaccionalidad.

Al programar varios objetos en OOP se ve la necesidad de escribir muchas veces código repetitivo en cuanto al servicio de transacciones; ya que, cada objeto tendrá alguna parte de código que difiera del otro, la OOP por sí sola no ayuda a modularizar esto, mientras que AOP provee un paradigma que resuelve aspectos como estos.

AOP es usada por el framework para una variedad de propósitos, algunos de ellos son transparentes para el desarrollador pero son el resultado de AOP. (1)

Manejo de transacciones declarativas: este es el servicio más importante que proporciona Spring. Es análogo al valor que provee java con EJB pero con las siguientes ventajas:

- Puede ser aplicado a cualquier POJO; es decir, a cualquier clase simple que no dependa del framework.
- Trabaja con una variedad de APIs para manejo de transaccionalidad.
- Soporta semántica adicional para minimizar la necesidad de depender de un API para manejo de transacciones propietarias.
- Soporte a objetos dinámicos: El uso de un proxy AOP puede permitir objetos dinámicos tales como objetos provenientes de scripts en un lenguaje como groovy. Pone en práctica interfaces adicionales que permiten un estado para ser consultados y manipulados.
- Seguridad: Acegi Security para Spring es un framework asociado que usa Spring AOP para enviar un sofisticado modelo de seguridad.

Existe un valor agregado en el uso de AOP en el código de la aplicación. Es usado frecuentemente para manejar aspectos como:

- Auditorias: Aplica una consistente política de auditoría.
- Manejo de excepciones: Emplea un consistente manejo de excepciones como mandar un e-mail al administrador en el evento de una excepción lanzada por un objeto.

- Recargas: Intenta realizar recuperaciones transparentes, por ejemplo en el caso de un fallo en una invocación remota.

2.2.2 Módulos de Spring

Spring está diseñado con algunos módulos bien definidos. Tomados como un todo, estos módulos proveen todo lo necesario para desarrollar aplicaciones empresariales, no es esencial usar todos los módulos de Spring para desarrollar una aplicación; ya que, se pueden usar solo los necesarios para la aplicación a desarrollar o también se pueden tomar módulos de otros frameworks, pues Spring tiene la capacidad de integración con algunos frameworks del mercado. (3)

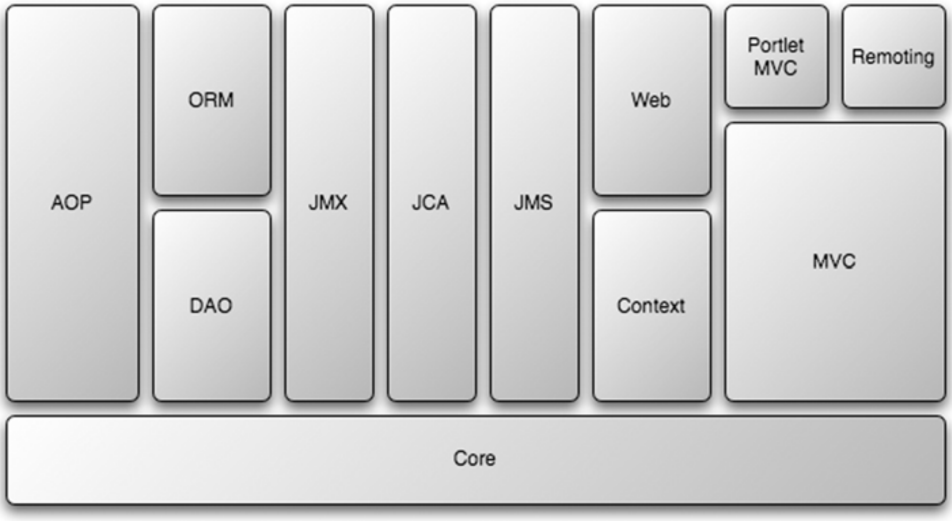

Figura 2.2: Módulos de Spring Framework. (3)

Módulo Core Spring: Provee la funcionalidad fundamental del framework. Este módulo contiene el BeanFactory y este es imprescindible para la comunicación con el patrón de diseño MVC, aparte de esto realiza el manejo de sesiones. El Core contiene todos los parámetros para la comunicación e integración de todos los distintos módulos del framework.

Módulo Context: El application Context de Spring se construye en el Core del mismo, el Core es lo que hace de Spring un contenedor pero el módulo Context es lo que lo hace un framework. Este módulo extiende el concepto de BeanFactory añadiendo soporte para eventos del ciclo de vida de la aplicación, validaciones y mensajes

Módulo AOP: El módulo AOP implementa la tecnología descrita en el punto 2.2.1.2

JDBC y DAO: Cuando se trabaja con objetos JDBC muchas veces resulta en código repetitivo, estos módulos de Spring simplifican las cosas para mantener el código de la base de datos limpio y simple. Este módulo también entrega un conjunto de excepciones dados por varios errores en la base de datos. JDBC y DAO usan el módulo AOP para proveer manejo de servicios transaccionales para los objetos en la aplicación.

Módulo ORM (Mapeo objeto-relacional): Para aquellas personas que prefieran no usar JDBC y usar un objeto para mapeo-relacional, Spring proporciona soporte DAO para algunos frameworks ORM, incluyendo soporte para Hibernate, Java Persistence API, Java Data Objects.

Módulo JMX (Administración de extensiones java): Módulo que agrega la funcionalidad de monitorear y reconfigurar una aplicación cuando está ejecutándose.

Módulo JCA (Conector API Java EE): Provee una forma estándar de integrar aplicaciones java con una variedad de sistemas de información empresarial, incluyendo mainframe y bases de datos.

Módulo MVC: El paradigma modelo-vista-controlador es un enfoque común para la construcción de aplicaciones web, Spring usa este paradigma en la construcción de sus aplicaciones.

Módulo Portlet MVC: Muchas aplicaciones están basadas en el hecho de que un request a una página web resulta en una nueva página que presenta cierta información o muestra cierta forma al usuario. En contraste las aplicaciones basadas en Portlet agregan varios bits de funcionalidad a una sola página web. Esto provee una vista a varias aplicaciones al mismo tiempo.

Módulo web: El módulo web de Spring da un especial soporte de clases para los módulos MVC y Portlet MVC, también incluye soporte para varias tareas orientadas a la web como subida de archivos e integración con Apache Struts y Java Server Faces.

Módulo Remoting: Una aplicación no siempre puede realizar una tarea por sí misma, a veces necesita de otra aplicación para hacer el trabajo, Spring Remoting habilita la función de tener objetos java como objetos remotos.

Módulo JMS (Java Message Service): La comunicación orientada a mensajes es un poco más confiable que la tradicional; ya que, garantiza el envío de mensajes incluso si la red no es confiable. Este módulo ayuda a crear mensajes dirigidos a POJOs que son capaces de consumir mensajes asíncronos.

2.2.3 Principios de Spring

Los módulos de Spring framework fueron creados con el fin de dar al desarrollador cierto valor adicional al framework. Con una visión bastante clara de lo que pretendían hacer con el software, sus autores basan su éxito en los siguientes principios. (1)

- Spring es un framework no invasivo: Si bien algunos framework fuerzan al desarrollador a crear su código dependiendo del framework, forzándolo a extender e implementar distintas clases del mismo, Spring procura minimizar la código-dependencia hacia las clases del framework, en Spring pueden ser usadas clases que se desarrollaron incluso sin tener todavía un conocimiento de Spring.
- Spring provee un modelo consistente de programación, usable en cualquier ambiente: Muchas aplicaciones web simplemente no necesitan ejecutarse en costosos servidores, simplemente pueden correr en un contenedor web tales como Tomcat o Jetty. Spring proporciona un modelo de programación en el cual aísla el código de la aplicación de los detalles del ambiente de ejecución, haciendo al código menos dependiente del contexto de ejecución.
- Spring tiene como objetivo facilitar el diseño orientado a objetos en aplicaciones J2EE, Spring intenta remover algunos de los impedimentos de la orientación a objetos tradicionales, para así obtener código más coherente sobre todo reusable.
- Spring promueve las buenas prácticas en la programación tales como programación hacia interfaces más que a clases. Al usar la inyección de dependencias en Spring reduce radicalmente la complejidad al codificar la interfaz. Así las llamadas usan los objetos a través de la interfaz.
- Spring está diseñado para que las aplicaciones sean lo más fáciles de probar: Spring usa inyección de dependencia y por lo tanto resulta en que sus objetos son POJOs.
- Spring es consistente: En diferentes partes del framework y en diferentes ambientes de ejecución el enfoque que utiliza Spring es consistente. Una vez aprendida una parte del framework el conocimiento será útil para distintas partes del mismo.
- Spring promueve la opción de arquitectura: Si bien Spring aporta una columna vertebral para el desarrollo, también tiene como objetivo facilitar el intercambio de arquitecturas, por ejemplo el cambiar de un framework objeto-relacional a otro dentro de Spring deberá tener un mínimo impacto sobre el código de la lógica del negocio.
- Spring no reinventa la rueda: Como en el punto anterior Spring nos deja opciones abiertas para la elección de uno u otro framework para el manejo de datos, interfaces, etc. No se

reinventa la rueda, no pretender desarrollar su propio framework objeto-relacional o framework de interfaces, permite usar los ya existentes, simplemente facilita la forma de usarlos.

Todas estas características, módulos y valores hacen de Spring framework sea uno de los más usados en el mundo, debido a esto la comunidad de desarrollo y soporte de Spring es una de las más grandes haciéndolo a la vez muy atractivo para estudiantes y profesionales del área informática.

2.3 Richfaces Framework

JBoss Richfaces es un framework open-source que añade capacidades de Ajax a aplicaciones JSF sin la necesidad de escribir código en JavaScript. (4)

Richfaces aprovecha las capacidades del framework JSF incluyendo su ciclo de vida, facilidades de conversión, validación y sus componentes principales. Al estar entrelazado con JSF es necesaria una corta descripción de JSF en sus componentes usados por Richfaces para entender las funcionalidades y lo que Richfaces hereda de JSF. A continuación una descripción del framework JSF.

2.3.1 JavaServer Faces y Richfaces

JSF es un framework para el desarrollo de aplicaciones web que simplifica el desarrollo de interfaces de usuario para aplicaciones empresariales java. Provee a las aplicaciones web de un administrador de ciclo de vida a través de un servlet controlador. Fue creado con el fin de entregar una arquitectura bien definida para el desarrollo y crear un estándar de controles para aplicaciones.

2.3.1.1 Componentes de JSF importantes

JSF introduce grandes avances en la programación de tal forma que dichos avances se han extendido y hoy en día se usan bajo los términos Managed Bean y Backing. (5)

Managed Bean: Es un objeto identificado para el ambiente de la aplicación para el cual se describe:

- Una identificación.
- Un ámbito (scope) que puede ser request, sesión, application, etc.
- Propiedades.

Backing Bean: Es usualmente un Bean común de java que sirve de soporte para un objeto manejado dentro de la aplicación. Su ventaja es que pueden ser compartidos por un mismo

Managed Bean, de manera que para diferentes páginas se pueden agrupar comportamientos comunes en un mismo Bean que comparte con ambos.

2.3.1.2 Ciclo de vida JSF utilizado por Richfaces

El ciclo de vida completo de una aplicación JSF es:

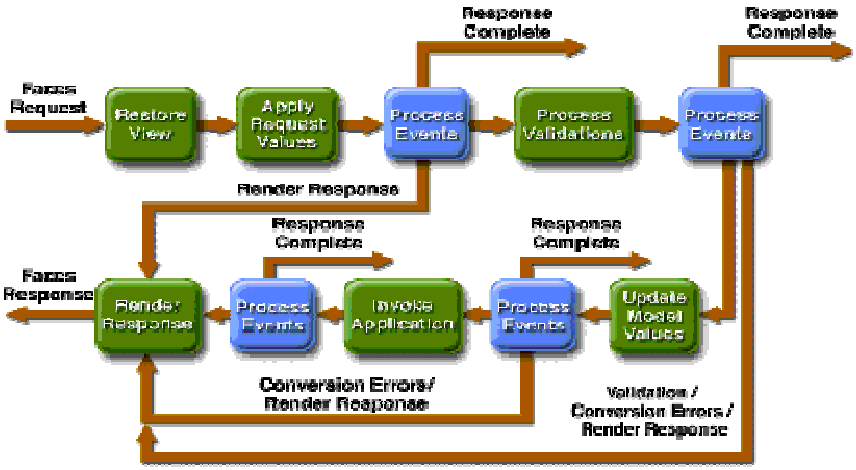

Figura 2.3: Ciclo de vida JSF. (6)

1. Restaurar Vista (Restore View): El servlet localiza la vista correspondiente a la URL de la petición recibida.

   Si es la primera visita: genera el árbol de componentes.

   Si la vista ya fue generada recupera la vista correspondiente.

2. Actualizar valores de la petición (Apply request values): A partir de los valores de entrada que llegan en la petición HTTP, se actualizan los valores vinculados a los nodos del árbol de componentes que representa la interfaz del usuario.

   Se realizan las conversiones String-objeto necesarias.

   Si existen errores se salta a la fase de respuesta e incluye mensajes de error.

3. Procesar Validaciones (Process Validations): JSF recorre recursivamente el árbol de componentes y revisa que los valores recibidos por cada uno sean aceptables.
   - Actúan los validadores asociados a cada componente.
   - Si existen errores se salta a la fase de respuesta e incluye los correspondientes mensajes de error.

4. Actualizar modelo (Update Model Values): Se actualiza los valores de los atributos de los Managed Bean vinculados a los componentes JSF.

5. Invocar aplicación (Invoke Application): Se invocan los métodos de los Managed Beans (tendrán sus atributos sincronizados con las entradas del usuario).
   Se invocan los manejadores de eventos registrados en los componentes JSF.
   Se invocan los métodos de acción y se toman las decisiones de navegación.
6. Producir Respuesta (Render Response): Una vez decidida cual es la vista a enviar al cliente, se genera la representación de sus componentes.

Existen tres escenarios posibles y cada uno pasa por distintas fases del ciclo de vida

Petición JSF genera respuesta JSF:
   Ciclo de vida completo.
Petición no-JSF genera respuesta JSF
   Fase 1 y 6 del ciclo de vida.
Petición JSF genera respuesta no-JSF
   Fase de la 1 a la 5.
   Desvió a productor no JSF.

2.3.1.3 Convertidores JSF

Transforman de forma conveniente los valores de los objetos Java vinculados a los componentes JSF. Todos los componentes que heredan de UIOutput pueden tener un convertidor asociado. Sirven para transformar de objeto a cadena y viceversa e implementan la interfaz Converter.

2.3.1.4 Validadores JSF

Los componentes JSF tienen un atributo de entrada llamado required, este define si se debe asignar un validador que obligue a rellenar el campo u otros validadores estándares de JSF como validadores de rango, de fechas, etc. Los validadores son responsables de comprobar antes de actualizar los atributos de Managed Beans y que los dichos valores recibidos en los componentes JSF cumplan las restricciones especificadas. Cada componente JSF que reciba datos de entrada puede tener asociados uno o más validadores.

2.3.2 Beneficios de Richfaces

Al tener una referencia teórica de lo que JSF puede ofrecer, en su arquitectura y funcionalidad se puede observar cuales son las características que ofrece Richfaces; ya que, siendo una extensión de JSF añade cierta funcionalidad y nuevos beneficios para el desarrollador. Un

desarrollador al trabajar con Richfaces estará en capacidad de agregar una funcionalidad rica a sus páginas web en cuanto a la interfaz, Richfaces le permitirá agregar beneficios como los siguientes:

- Intensificar los beneficios de JSF agregando Ajax a sus funcionalidades, le permitirá invocar validaciones y conversiones en el servidor durante el ciclo request-response de Ajax
- Agregar capacidades Ajax a proyectos JSF existentes. El framework proporciona dos librerías de componentes (Core Ajax y UI). La librería Core agrega funcionalidad Ajax en páginas existentes sin la necesidad de escribir código JavaScript o reemplazar elementos existentes de Ajax con nuevos.
- Crear vistas rápidamente basadas en elementos ya fabricados. La librería UI de Richfaces trae consigo una variedad de componentes para agregarlos al de JSF.
- Desarrollar componentes propios, la fábrica de componentes de Richfaces permite crear los propios para una interfaz rica con funcionalidad Ajax.
- Introduce un paquete de recursos con aplicación para clases java, provee un avanzado soporte para manejo de recursos como imágenes, JavaScript y hojas de estilo CSS.
- Crear modernas interfaces ricas con tecnología basada en skins, Richfaces provee unas características de skins que permite fácilmente definir y administrar diferentes esquemas de colores y otros parámetros de la interfaz de usuario.
- Probar y crear componentes, acciones, listeners y páginas al mismo tiempo. Esta facilidad creara casos de prueba de los componentes en el momento del desarrollo de los mimos.

Los componentes de las dos librerías de Richfaces vienen listos para usar, por lo que los desarrolladores inmediatamente ganan tiempo y las ventajas antes mencionadas para la creación de aplicaciones web.

2.3.3 Arquitectura de Richfaces

La siguiente figura muestra los elementos más importantes de la arquitectura que Richfaces provee desde su librería de componentes.

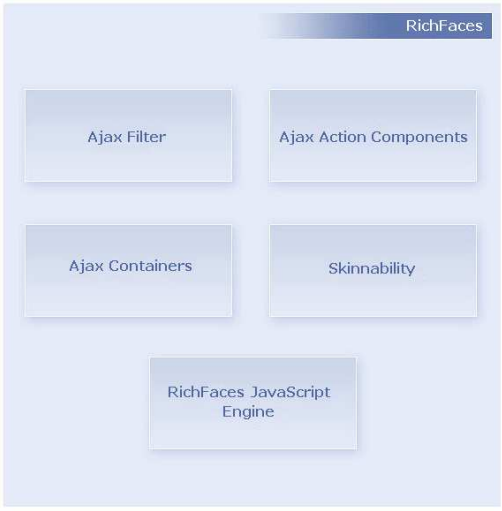

Figura 2.4: Arquitectura Richfaces. (7)

Ajax Filter: Para obtener todos los beneficios del framework es necesario manejar el Ajax Filter, esto es debido a que el filtro reconoce los distintos tipos de request que puede llegar desde la UI. La siguiente figura muestra la diferencia entre procesar una petición JSF y una de Richfaces.

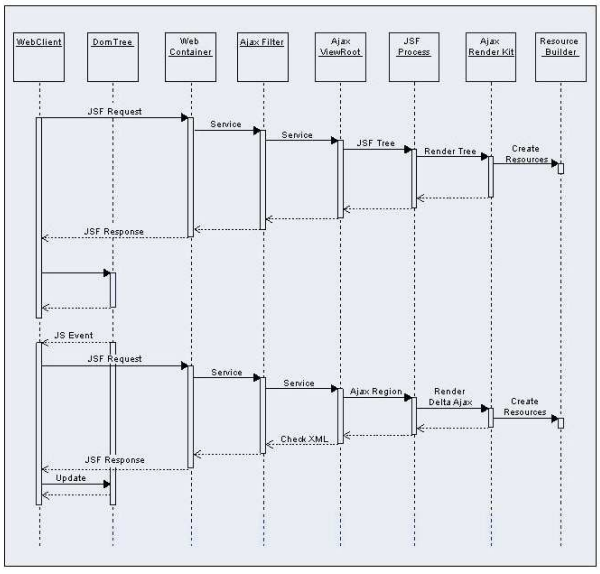

Figura 2.5: Ciclo de petición JSF y Richfaces. (7)

Como se puede observar en la figura 2.5 una petición JSF esta codificada de tal manera que refresca toda la página en cualquier petición, en el segundo caso en la parte inferior de la figura se observa como la petición depende del tamaño de la región Ajax, el filtro analiza el contenido de una respuesta Ajax antes de enviarlo al cliente.

Ajax Action Components: Dentro de la librería de componentes de Richfaces existen los action components para enviar peticiones Ajax desde el lado del cliente, como ejemplos de estos componentes se pueden citar los siguientes: <a4j:commandLink>, <a4j:poll>, <a4j:support> y <a4j:commandButton> entre otros que en el lado del cliente enviaran peticiones de tipo Ajax para que se refresque solamente el contenido requerido.

Ajax Containers: Son dentro de la interfaz del usuario los que describen el área dentro de la página que deberá ser decodificada durante la petición Ajax. AjaxViewRoot y AjaxRegion son implementaciones de contenedores.

Skinnability: Está diseñado para realizar una mezcla de parámetros entre;

- Parámetro skin definido en el framework
- Hojas de Estilos de usuario para las clases
- Hojas de estilo CSS predefinidas para componentes.

Esta característica del framework está diseñada para unificar la apariencia y la sensación de los elementos HTML estándares y los componentes de Richfaces. Los Skins pueden ser aplicados a todos los controles sobre una página basados en los atributos de cada elemento. También provee un conjunto de estilos CSS que pueden ser aplicados.

JavaScript Engine: Este motor de JavaScript se ejecuta en el lado del cliente. Se caracteriza por "saber" cómo actualizar las diferentes áreas de la página con la información de la respuesta Ajax. Está disponible automáticamente por lo que facilita la tarea del desarrollo.

2.4 Seam Framework

Seam es un framework open-source creado por JBoss con el objetivo de integrar los frameworks más importantes de Java EE en uno solo. Seam satisface sus labores de framework de dos formas fundamentales (8):

- Seam Simplifica Java EE: Seam aporta un número de atajos y simplificaciones al estándar Java EE, haciendo más fácil usar efectivamente los componentes de negocios y de web provistos por Java EE.
- Seam extiende java EE: Seam integra un número de nuevos conceptos y herramientas a la plataforma Java EE. Estas extensiones brindan nueva funcionalidad y características a las aplicaciones web.

2.4.1 Principios de Seam

Seam siendo uno de los frameworks más completos del mercado para aplicaciones empresariales java está inspirado en los siguientes. (Ten Good Reasons to use Seam)

- Solo un tipo de "cosas": Seam define un modelo uniforme de componentes para la lógica del negocio en la aplicación. No existe distinción entre los componentes del nivel de presentación y los componentes de lógica de negocio, es decir se puede establecer las capas del negocio de acuerdo a la arquitectura que se desee.
- Integra JSF con EJB 3.0: JSF y EJB 3.0 son dos de las mejores características de Java EE, JSF es sobresaliente con la capa de presentación mientras que EJB 3.0 hace un trabajo excelente en la capa de negocios. Desafortunadamente Java EE no tiene un modelo que integre de manera satisfactoria estas dos tecnologías. Seam resuelve este problema eliminando todo el código para forzar una comunicación entre estos dos y haciendo que estos se integren de manera natural, así el desarrollador se puede centrar en pensar en los procesos de negocio.
- Ajax Integrado: Seam soporta las dos mejores soluciones para el manejo de Ajax con JSF: Richfaces y IceFaces. Estas soluciones en especial Richfaces enunciado en el punto 2.3 de este capítulo, agrega capacidades Ajax a la interfaz del usuario sin la necesidad de escribir código JavaScript.
- El proceso de negocio como primer constructo: Como una opción Seam aporta con un proceso transparente de negocio con jBPM. Con esto resultará muy fácil implementar flujos de trabajo complejos y administrar tareas.
- Manejo de estados declarativos: En las primeras etapas de EJB se introdujeron los conceptos de "administración de transacciones declarativa" y "seguridad declarativa", estos son ejemplos de un problema aún más amplio que está asociado con el manejo de estados en un contexto particular de la aplicación. Seam toma el concepto de administración de estados declarativos y lo lleva mucho más lejos, lo transporta a los estados de la aplicación. Con este acercamiento el problema del manejo de sesiones con su seguridad y transacciones se ve solucionado.
- Biyección: La arquitectura de inyección de dependencias y de inversión de control se ve en muchos frameworks en la actualidad. En vez de estos criterios de arquitectura Seam introduce la Biyección, este concepto difiere con los anteriores al ser dinámico, contextual y bidireccional. La Biyección permite a los componentes manipular de manera segura los valores de las variables de contexto.

- Administración del espacio de trabajo y manejo de múltiples ventanas: Una aplicación Seam le permite al usuario navegar libremente entre múltiples pestañas del navegador, cada una asociada con un diferente y seguramente aislado espacio de trabajo.
- Preferencia de anotaciones sobre XML: Tradicionalmente en otros frameworks los archivos de configuración se realizan mediante XML causando confusión en muchos desarrolladores. Con las anotaciones Seam se elimina la molestosa configuración a través de archivos XML e introduce simples anotaciones a los objetos que lo necesiten.
- Las pruebas del sistema son fáciles: Las pruebas de integración han sido tradicionalmente difíciles para las aplicaciones web java, pero Seam soluciona esto aportando un sistema de pruebas como una de las funcionalidades más importantes del framework. Una aplicación Seam se puede probar fácilmente con JUnit o TestNG.
- Hay más en una aplicación web que servir páginas HTML: Los frameworks web de la actualidad se quedan muy cortos, se centran simplemente en tomar la información que un usuario ingresa en una forma y la inyecta en los objetos java pertinentes. Un framework web completo debería manejar problemas como: persistencia, concurrencia, peticiones asíncronas, manejo de seguridad, email, mensajería, PDF, servicios web y aún más características. Con esto en mente la cantidad de características y problemas que maneja Seam es sorprendente.

2.4.2 Los modelos de componentes y de servicios de Seam

JBoss Seam provee un modelo de componentes y de servicios en el framework, los cuales se encargan de las dos formas fundamentales de realizar sus labores, simplificar Java EE y extender Java EE respectivamente.

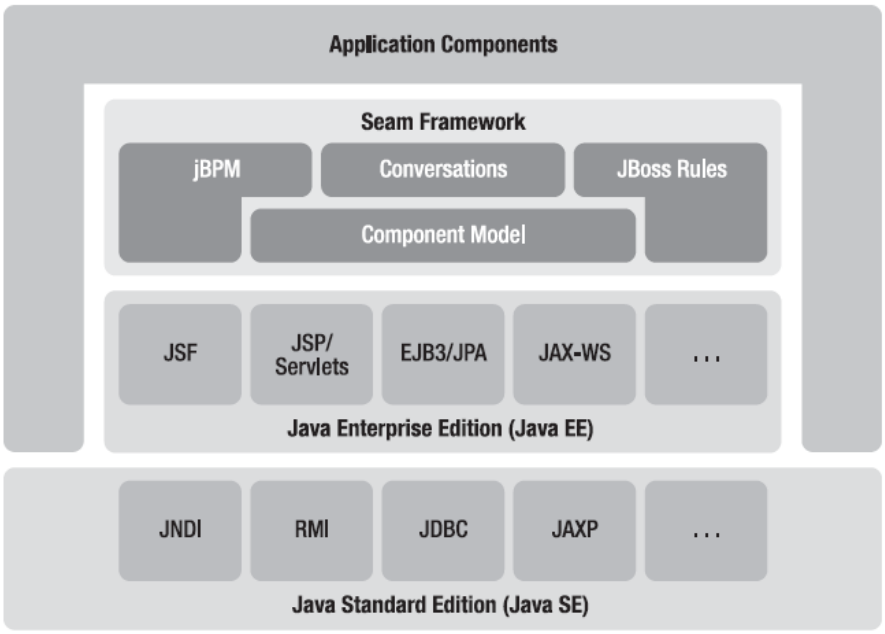

Figura 2.6: Componentes de Seam Framework. (8)

2.4.2.1 El modelo de componentes de Seam

Los atajos y simplificaciones que Seam aporta sobre el modelo tradicional de Java EE están en su mayoría basados en su modelo de componentes. Este modelo de componentes puede ser considerado en esencia una extensión del modelo de componentes JSF Managed Beans (observado en el punto 2.3.1.1 de este capítulo), pero tiene más usabilidad que una simple capa de componentes web.

Un beneficio clave que provee el modelo de componentes de Seam es el uso directo de componentes EJB como Backing Bean para páginas JSF. El modelo de JSF permite llamar a los Java Bean para ser usados como EJB que están declarados en un archivo de configuración JSF. Los EJB pueden ser llamados desde los Managed Beans, sirviendo de fachada para los componentes EJB. Seam provee un puente directo entre el modelo de componentes JSF y el modelo de componentes EJB, permitiendo usar un EJB directamente como un JSF Managed Bean.

El modelo de componentes de Seam también da suporte a la biyección de la cual ya se habló en el punto 2.4.1 como uno de los principios de Seam, en los siguientes apartados se detallará el soporte adicional que se extiende de la inyección de dependencias:

- Propagación de dependencias en dos sentidos: Un componente puede tener inyectado una referencia por el contenedor, pero también funciona en viceversa; es decir, un componente también puede inyectar una referencia al contexto que lo envuelve.
- Actualizaciones dinámicas: Tradicionalmente en otros framework la inyección de referencias se realiza una sola vez al invocar la página, la biyección es realizada cada vez que se invoca un componente. Esta es la clave en el modelo de componentes de Sea; ya que, los componentes pueden ser de estado y por lo tanto los componentes y sus Beans dependientes pueden evolucionar a través de invocaciones
- Múltiples contextos: Las dependencias entrantes y salientes pueden ser establecidas a través de múltiples contextos de Seam, en vez de ser forzadas a existir en un único contexto.

2.4.2.2 Componentes de Servicios Seam

Los componentes principales de servicios de Seam se pueden resumir en tres: componente de contextos Seam, Diálogos Seam y eventos Seam.

2.4.2.2.1 Componente de contextos Seam

Los componentes de Seam proveen varios contextos o alcances en tiempo de ejecución, soporta los contextos típicos usados en componentes web (request, page, sesión, y alcances de la aplicación). Pero este servicio de Seam agrega algunos contextos adicionales que pueden ser de utilidad en una aplicación empresarial.

Una adición importante de Seam, es que los componentes también tienen un contexto de ámbito de conversación y procesos de negocio.

Los contextos de proceso de negocio representan un estado para una posible ejecución larga de un proceso de negocio. Desde que los procesos de negocio pueden tener una ejecución más larga que el tiempo de vida de un request, una sesión o incluso que la misma aplicación, el contexto de proceso de negocio usa servicios persistentes para su estado de datos. Los procesos de negocio también pueden involucrar varios usuarios potenciales; por lo que, el contexto soporta acceso concurrente.

2.4.2.2.2 Diálogos Seam

Los diálogos Seam es un muy interesante y potencialmente poderoso concepto que agrega el framework. Una manera de describir el concepto de una conversación Seam es definirlo como

una verdadera transacción orientada al usuario. Seam provee otra dimensión de transacciones, definida por lo que el usuario quiere o necesita hacer dentro de la aplicación.

Otra manera de describir un diálogo menos formal pero más aceptable, es que los diálogos proveen otra capa entre el ámbito de un request y el de una sesión en la aplicación web. Un diálogo puede agrupar datos a través de múltiples request y se puede rastrear múltiples agrupaciones de este tipo dentro de una única sesión de usuario.

2.4.2.2.3 Eventos Seam

JSF provee un modelo de eventos en su modelo de componentes, pero Seam introduce dos nuevos tipo de eventos que pueden ser usados dentro de las aplicaciones Seam, a continuación una descripción de cada uno:

- Seam page actions: Son eventos que se disparan cuando un usuario realiza un request, pero antes el componente o página que realizó el request es refrescada. Se puede especificar qué acciones serán invocadas en el request de una página. Las acciones de la página son implementadas por operaciones de componentes.
- Seam component-driven event: Proporciona un sistema de notificación de eventos en general para los componentes Seam. Cualquier componente puede disparar un evento en cualquier lugar del código de negocio y cualquier operación del componente en la lista de notificaciones serán invocados.

2.5 Comparación teórica

Para la realizar una comparación teórica de los frameworks estudiados en este capítulo se tomará como referencia algunas directrices teóricas comunes a todos estos.

Las características que se tomaran en cuenta para la evaluación teórica de los frameworks estudiados son las siguientes:

1. Soporte Ajax
2. ORM (Mapeo de objetos relacionales)
3. Frameworks para pruebas unitarias
4. Seguridad del framework
5. Componentes Disponibles
6. Tecnologías Utilizadas
7. Beneficios

8. Ventajas y Desventajas.

En las siguientes tablas se detallará como cada uno de los frameworks estudiados cumple con las características especificadas.

Framework	Soporte Ajax
Spring	Ajax implementado a través de código JavaScript.
JBoss Richfaces	Componentes de Richfaces listos para usar con Ajax, agrega componentes Ajax para usar en código JSF.
JBoss Seam	Integración con frameworks que soportan Ajax, Richfaces y Icefaces son opciones para la incorporación de Ajax en el proyecto.

Tabla 2.2: Soporte Ajax.

Framework	ORM (object-relational-mapping)
Spring	Integra módulo ORM para las personas que prefieran usar un framework ORM en vez de JDBC.
JBoss Richfaces	Al ser un framework perteneciente a JBoss permite integración con Hibernate de forma natural.
JBoss Seam	Integra de manera intuitiva soluciones ORM, Seam integra la capa ORM con la capa de lógica de negocio y la de presentación de forma transparente al usuario.

Tabla 2.3: Mapeo de objetos relacionales.

Framework	Frameworks de prueba unitarias
Spring	Se integra perfectamente con JUnit, las pruebas se realizan fácilmente ya que implementa POJOs en la mayoría de las clases pertenecientes a la lógica del negocio.
JBoss Richfaces	No integra pruebas unitarias en JUnit hasta la version3.3
JBoss Seam	Aporta un sistema de pruebas como una de sus funcionalidades más importantes y sobresalientes, se integra fácilmente con JUnit o TestNG

Tabla 2.4 Adaptación a frameworks de pruebas unitarias.

Framework	Seguridad
Spring	Acegi Security para Spring es un framework asociado que usa Spring AOP para enviar un sofisticado modelo de seguridad.

JBoss Richfaces	No proporciona un modelo de seguridad de datos, ya que es un framework enfocado a la capa de presentación.
JBoss Seam	Proporciona una herramienta llamada gkadmin para el manejo de seguridad en la red.

Tabla 2.5: Seguridad en los frameworks.

Framework	Componente Disponibles
Spring	Los componentes de Spring son los citados en el punto 2.2.2, entre ellos los más citados para el desarrollo son Spring DAO, Spring ORM, Spring AOP, Spring Web y el Spring Core
JBoss Richfaces	Usa los componentes de JSF Backing Bean y Managed Bean y a estos les agrega una librería de componentes Richfaces y otra de componentes Ajax.
JBoss Seam	Integra un modelo de componentes en el que maneja una extensión de EJB, una extensión de JSF Managed Bean, componentes para el manejo de anotaciones, también agrega componentes de servicios como componentes de contextos, diálogos y eventos.

Tabla 2.6: Componentes más destacados.

Framework	Tecnologías Utilizadas
Spring	Las tecnologías que más resaltan de Spring son: la programación orientada a aspectos AOP y la inyección de dependencias IoC
JBoss Richfaces	Usa tecnologías JSF para el manejo de Beans e integra tecnología Ajax a sus componentes.
JBoss Seam	Soporta integración automática de tecnologías para ORM y para la capa de presentación. Agrega Biyección que modifica la inyección de dependencias, integra las tecnologías EJB 3.0 con JSF de manera automática sin código de pegamento.

Tabla 2.7: Principales tecnologías.

Framework	Beneficios
Spring	Framework muy popular con una gran comunidad que lo respalda. Buena integración con frameworks para la capa de presentación y de datos. Arquitectura muy flexible basada en interfaces.

	Gran seguridad de datos. Introduce AOP.
JBoss Richfaces	Excelente para la capa de presentación. Se obtienen aplicaciones ricas fácilmente. Gran cantidad de componentes. Fácil creación de componentes nuevos. Los Skins hacen que la creación de interfaces sea estupenda.
JBoss Seam	Cada vez gana más popularidad en el mercado, creciendo cada día la comunidad que lo soporta. Excelente Framework de integración. Gran cantidad de herramientas para el desarrollo. Es perfecto para aplicaciones empresariales de mayor importancia.

Tabla 2.8: Beneficios Importantes.

Framework	**Ventajas**	**Desventajas**
Spring	Código de aplicaciones más limpio. Configuración de la aplicación simplificada. Buena integración con productos open-source populares. Buen soporte para AOP. Seguridad a escala empresarial. MVC flexible. Facilidad de pruebas.	Configuración costosa, se requiere de mucho código XML.
JBoss Richfaces	Variedad de componentes listos para usar. Integra Ajax a sus componentes.	El crecimiento de los componentes Richfaces no es muy grande. Richfaces no admite JSF 2.0,

	Facilidad para integración de Ajax en cualquier zona JSF. Soporta el uso de temas o Skins lo que lo hace más atractivo. Usa el ciclo de vida de una aplicación JSF	solo se puede usar con la versión de JSF 1.2
JBoss Seam	Une estándares JEE (EJB 3 y JSF). Estable por su diseño. Integra Ajax (Richfaces y IceFaces). Integración con procesos de negocio (jBPM). Manejo de espacios de trabajo. Gran nivel de abstracción, se encarga de detalles de la programación.	Debido a su nivel de abstracción esconde detalles que se pueden convertir en un gran problema. Demasiadas herramientas pueden hacer que el desarrollador no las aproveche todas.

Tabla 2.9: Ventajas y Desventajas de los Frameworks analizados.

2.6 Evaluación según un modelo teórico

La evaluación teórica de los frameworks es una directriz importante para la selección de los mismos, este tipo de métodos provee un marco de referencia para determinar aspectos no funcionales que son de gran importancia, se puede citar como ejemplo características como el equipo de desarrollo que está detrás del framework, la estrategia que mantiene la evolución, madurez, entre otros. Como se puede constatar estos aspectos son de gran importancia para aventurarse en el desarrollo con algún framework, no se puede tomar como marco de referencia para el desarrollo un framework inmaduro, sin un equipo de desarrollo serio y que no tenga ninguna estrategia para la evolución; ya que, al entrar en el desarrollo pueden suscitarse problemas con el framework y si no se tienen los aspectos antes citados en cuenta, los proyectos que se estén realizando tienen grandes posibilidades de fracaso.

Un modelo para la evaluación teórica de frameworks es el modelo QSOS, este tiene grandes capacidades para evaluar características no funcionales del software y en especial para evaluar software de tipo ope-source que es lo que más interesa para la presente obra.

2.7 Evaluación según modelo QSOS

El QSOS es un modelo creado con el objetivo de facilitar a las empresas o a cualquier entidad que vea necesario realizar una evaluación de software open-source, debido a este objetivo deriva su nombre QSOS (Method for Qualification and Selection of Open Source software); es decir, método para la calificación y selección de software de código abierto. Si bien este método fue creado para software open-source, sus principios también son aplicables a software comercial.

La aplicación del modelo QSOS tiene más utilidad al evaluar aspectos no funcionales del software de código abierto, los cuales determinan aspectos como el futuro del software, el soporte que tienen, la comunidad que desarrolla el software y unos pocos aspectos técnicos como la industrialización en el desarrollo, disponibilidad de plataformas y otros pocos aspectos técnicos de utilidad para conocer temas de servicios y modularidad.

El modelo ha pasado por un largo proceso de desarrollo y refinamiento, en su versión actual se encuentra la versión 1.6, en la cual se han añadido ciertos criterios para la evaluación y calificación como lo son la disponibilidad de plataformas y la adaptabilidad técnica.

2.7.1 Proceso de Evaluación

El proceso de evaluación consta de 4 pasos bien definidos, en la siguiente figura se muestra un diagrama de los mismos:

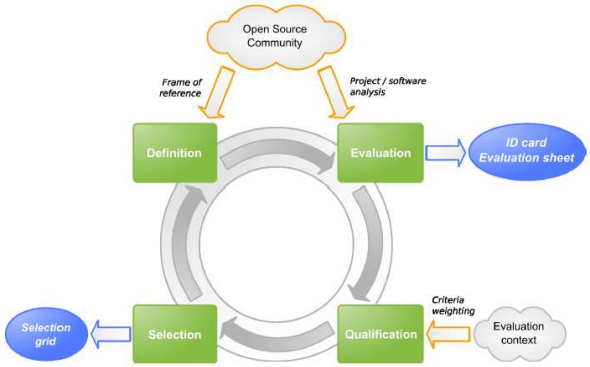

Figura 2.7: Los cuatro pasos de QSOS. (9)

Como muestra la figura 2.7 los cuatro pasos para la ejecución del modelo son: Definición, Evaluación, Calificación y Selección. A continuación se definirá cada uno de los pasos y los correspondientes resultados a la evaluación de los tres frameworks web java que se están estudiando.

Para este trabajo de investigación serán descritos pero no serán tomados en cuenta los criterios de clasificación y selección, debido a que estos se basan en la experiencia y necesidades del usuario. Solamente se establecerá un análisis de la evaluación del software.

2.7.1.1 Definición

La definición se la toma como un marco de referencia para los siguientes tres pasos, este punto tiene como objetivo definir varios elementos que serán usados por el resto del proceso.

Los marcos de referencia son:

- Familias de software: Para el presente caso la familia de software a usar se define como: "Frameworks web realizados en Java"
- Tipos de licencia: Existen muchos tipos de licencias dentro del software libre, entre las características que puede tener una licencia se puntualizan las siguientes:
    - Licencia Propietaria: Define si la licencia puede llegar a ser de propietarios o debe permanecer libre.
    - Viralidad de la licencia: El código fuente del producto puede estar afectado por otros módulos, con la viralidad estos módulos permanecerán bajo la misma licencia.
    - Herencia de la licencia: El código derivado hereda inevitablemente la licencia o se le pueden aplicar restricciones adicionales.

Framework	Licencia	Propietaria	Viralidad	Herencia
Spring	Apache 2.0	Si	No	No
Richfaces	LGPL	No	Parcial	Si
Seam	LGPL	No	Parcial	Si

Tabla 2.10: Licenciamiento Frameworks.

Tipos de comunidades: Los tipos de comunidades identificadas son:

- Desarrollador solitario: El software es desarrollado por una sola persona.
- Grupo de desarrolladores: Muchas personas colaborando de forma informal.
- Organización de desarrolladores: Un grupo de desarrolladores colaborando de una forma formal.
- Entidad Legal: Una entidad legal, en la mayoría de casos no lucrativa que maneja el desarrollo y tiene las licencias del producto.

- Entidad Comercial: Una organización comercial con empleados desarrollando el software, quienes son remunerados por la venta de servicios comerciales o por versiones comerciales del software.

El software estudiado en este capítulo está en el último tipo de comunidad; ya que, Richfaces y Seam son desarrollados por JBoss que es una entidad comercial y en el caso de Spring este es desarrollado por Spring Foundation, también una entidad comercial.

2.7.1.2 Evaluación

El objetivo de este punto es llevar a cabo la evaluación del software, este punto consiste en recolectar información de la comunidad open-source para los siguientes dos puntos:

Construir la carta de identidad del software.
Construir la hoja de evaluación del software.

La carta de identidad establece información general acerca del software evaluado, los parámetros dentro de la carta no son puntuados, pero establece una base para la hoja de evaluación que debe realizarse.

Información General
Referencia, datos de creación
Autor
Tipo de Software
Descripción
Licencia
URL
Sistemas operativos compatibles
Origen
Servicios Existentes
Documentación
Cursos de formación
Ofertas de consultoría
Aspectos Funcionales y Técnicos
Tecnologías de implementación
Requisitos Técnicos

Tabla 2.11: Carta de identidad del Software. (9)

La hoja de evaluación: La hoja de evaluación comprende una serie de parámetros que califican al software de acuerdo a un puntaje. El puntaje puede ser 0, 1 o 2 de acuerdo al cumplimiento del criterio.

Ver Anexo 1

La hoja de evaluación tiene cinco criterios fundamentales y existen otros criterios que los creadores del método aumentan según sea el tipo de software.

Las cinco principales familias son:

- Durabilidad intrínseca: Hace referencia a la edad y estabilidad del software.
- Industrialización del desarrollo: Indica que tan popular es el software y cuál es el tamaño de la comunidad de desarrollo y uso.
- Disponibilidad de plataformas: Indica para qué sistemas operativos el software tiene soporte.
- Adaptabilidad técnica: Señala si el software es extensible por el usuario y que tan fácil es hacerlo.
- Estrategia: Muestra la estrategia de la entidad desarrollador; es decir, el tipo de licencia y si el software tiene un marco formal para su desarrollo.

2.7.1.3 Calificación

La calificación de este método se basa en establecer filtros de acuerdo a lo que el usuario considera más importante; es decir, el objetivo de este paso es el de establecer un contexto específico la selección del framework de acuerdo a las necesidades y limitaciones que el usuario o empresa especifiquen.

2.7.1.4 Selección

Este criterio es usado para la selección del software para su posterior desarrollo, existen dos modos para la selección, una selección estricta y una libre.

- Selección estricta: Se elimina el software incompatible con la tarjeta de identidad, se elimina el software que no está de acuerdo a los criterios establecidos en la calificación.
- Selección libre: Este método es menos estricto que el anterior ya que en lugar de eliminar el software que no cumple los criterios, lo clasifica según el cumplimiento de los filtros.

2.7.2 Resultados Evaluación y Análisis

Luego de realizar la evaluación según los criterios mencionados en el anexo 2 se obtiene la siguiente tabla de resultados.

Durabilidad Intrínseca		**Framework**		
		Spring	Richfaces	Seam
Madurez	Edad	2	2	2
	Estabilidad	1	2	2
	Historial de problemas	2	1	2
	Probabilidad de bifurcación	2	0	1
Adopción	Popularidad	2	2	2
	Referencias	2	1	2
	Calidad de la comunidad	2	2	2
	Libros	2	1	1
Estilo de liderazgo	Tamaño del equipo de desarrollo	2	2	2
	Estilo de la Administración del equipo	1	2	2
Actividad	Numero de Desarrolladores	2	2	2
	Actividad en Errores	2	2	2
	Actividad en funcionalidad	2	2	2
	Actividad en liberaciones de nuevas versiones	1	2	2
	Totales	1.79	1.64	1.86
Solución Industrializada		Framework		
		Spring	Richfaces	Seam

Servicios	Formación	2	1	2
	Soporte	2	1	2
	Consultoría	1	1	1
Documentación	Documentación	2	2	2
Método de calidad	Aseguramiento de calidad	1	1	1
	Herramientas	1	1	2
	Totales	1.5	1.17	1.67
Disponibilidad de plataformas		Framework		
		Spring	Richfaces	Seam
Paquetes	Windows	2	1	1
	Debian/Ubuntu	2	2	2
	Red Hat/Fedora	2	2	2
	Suse	2	2	2
	Mandriva	2	2	2
	Mac OS X	2	2	2
	Solaris	2	2	2
	Totales	2	1.86	1.86
Adaptabilidad Técnica		Framework		
		Spring	Richfaces	Seam
Modularidad	Modularidad	2	2	2
	Facilidad de extensión de código	1	1	1
	Totales	1.5	1.5	1.5
Estrategia		Framework		
		Spring	Richfaces	Seam
Licencia	Permisividad	2	1	1

	Protección contra extensiones del framework	0	1	1
Copyright	Dueños de Copyright	2	2	2
Modificación del código fuente	Modificación del código fuente	1	1	1
RoadMap	RoadMap (mapa de versiones y liberaciones)	1	2	2
Patrocinador	Patrocinador	2	2	2
Independencia estratégica	Independencia estratégica	2	2	2
	Totales	1.43	1.57	1.57
Industrialización del Desarrollo		Framework		
		Spring	Richfaces	Seam
Herramientas	IDE	2	1	1
	Herramientas de Construcción	2	2	2
	Herramientas de Pruebas	1	1	1
	Herramienta gráfica para el interfaces	2	2	2
	Totales	1.75	1.5	1.5

Tabla 2.12: Resultados de análisis mediante QSOS.

Tabulando resultados se obtiene la siguiente tabla y gráfico.

	Framework		
	Spring	**Richfaces**	**Seam**
Durabilidad Intrínseca	1.79	1.64	1.86
Solución Industrializada	1.5	1.17	1.67
Disponibilidad de plataformas	2	1.86	1.86
Adaptabilidad Técnica	1.5	1.5	1.5
Estrategia	1.43	1.57	1.57
Industrialización del Desarrollo	1.75	1.5	1.5

Tabla 2.13: Resumen de resultados QSOS.

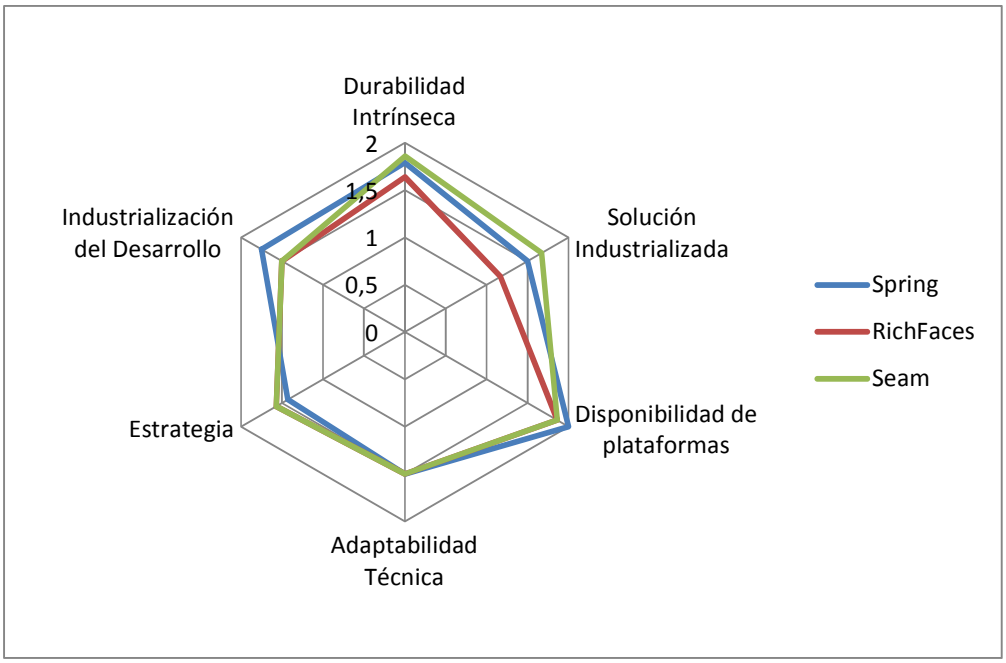

Figura 2.8: Gráfico de Resultados Evaluación.

En la figura 2.8 Se observa como los diferentes frameworks en cada uno de los criterios de evaluación tienen distintos valores, haciéndolos más factibles en unos y menos en otros, con esa información se realiza el siguiente análisis:

- Durabilidad intrínseca: Se observa un empate entre Spring y Seam quedando atrás Richfaces lo que nos indica que los dos primeros tienen más posibilidades de permanecer en el mercado.

- Industrialización del Desarrollo: Spring al tener su IDE propio y adaptabilidad con los IDEs más populares deja atrás a Richfaces y Seam.
- Estrategia: Se observa que Seam y Richfaces tienen una Estrategia similar por pertenecer a la misma entidad, Spring por otro lado está un poco relegado ya que no tiene muy claro un roadmap con liberaciones de nuevas versiones y actualizaciones.
- Solución Industrializada: Seam está en primer lugar ya que a pesar de ser un framework joven en comparación a Spring posee gran documentación, soporte, libros, cursos de formación y consultoría, Richfaces es el framework que se queda atrás en esta área ya que existen pocos libros y cursos de formación que lo respalden.
- Disponibilidad de plataformas: Los tres frameworks al estar realizados en java están disponibles para cualquier plataforma.
- Adaptabilidad técnica: Se observa igualdad ya que los frameworks están compuestos de módulos bien definidos y la extensión de código en la aplicación se la hace con cierta dificultad.

2.8 Conclusiones

La arquitectura modelo-vista-controlador es un patrón constante en la mayoría frameworks web del mercado, en esta arquitectura el modelo, la vista y el controlador interactúan de tal manera que el controlador interactúa con el modelo y con la vista sirviendo de puente entre estos dos, con el objetivo de la protección de datos y de minimizar la incongruencia de los mismos.

La característica principal de Spring Framework son las tecnologías que utiliza: la inyección de dependencias (IoC) y la programación orientada a aspectos (AOP). Con la IoC Spring posibilita la creación de clases no dependientes, es decir clases que envuelven su funcionalidad sin necesidad de extender o implementar clases del framework. Con la AOP Spring posibilita al desarrollador centrarse en aspectos relevantes del negocio antes que en modelar objetos de la vida real como la OOP, evitando así el código repetitivo en la aplicación.

Richfaces es un framework que extiende a JSF, es decir usa varias de sus características por lo que es necesario tener un conocimiento de JSF para el desarrollo en Richfaces. Las características que añade este framework son: añadir capacidades Ajax a un proyecto sin escribir código JavaScript y trae consigo una librería de componentes listos para usar. Esto hace que las aplicaciones en Richfaces tengan componentes más atractivos y con una funcionalidad mejorada.

Seam es un framework que ofrece una gran variedad de facilidades y tecnologías, integra una nueva tecnología llamada biyección para el manejo de dependencias, viene integrado de "fabrica" con Hibernate un framework para manejo de objetos relacionales y el más usado en java, para la utilización de Ajax ofrece la posibilidad de integración con Richfaces y IceFaces que son los frameworks más exitosos en manejo de esta funcionalidad.

Seam Framework ofrece un modelo de componentes y servicios que simplifican y extienden las capacidades de Java EE, con estos componentes ofrece la biyección, y con sus servicios facilitan el manejo de contextos, realizan "conversaciones" entre los request del usuario y las sesiones del mismo, introducen nuevos eventos de los que incluía JSF.

Seam y Richfaces brindan soporte Ajax dentro de sus funcionalidades mientras que para agregar esta funcionalidad a Spring es necesario utilizar código JavaScript.

Spring ofrece dos módulos para el manejo de datos, uno para JDBC y otro para ORM, Seam viene integrado con Hibernate en su módulo ORM con un manejo intuitivo del mismo y Richfaces ofrece la funcionalidad de integración pero mediante código XML.

Spring y Seam tienen ambos un módulo para la seguridad dentro de una aplicación mientras que Richfaces al ser un framework de la capa de presentación no tiene ningún módulo de seguridad.

Spring, Seam y Richfaces ofrecen integración con JUnit como framework de pruebas unitarias, con la diferencia que Seam también ofrece integración con TestNG y que Richfaces solo ofrece integración JUnit en sus últimas versiones.

Spring, Richfaces y Seam ofrecen diferentes beneficios como: Richfaces es un framework excelente para la capa de presentación y ofrece funcionalidades Ajax fácilmente, Spring es un framework completo que introduce el concepto de AOP permitiéndole al desarrollador centrarse en puntos importantes para el negocio, Seam debido a su diseño es estable, ofrece grandes capacidades de integración y posee herramientas para los procesos del negocio.

Cada uno de los frameworks posee distintos tipos de herramientas que pueden hacerlo atractivo al desarrollador, pero en el momento de la selección es necesario determinar nuestras necesidades y observar como cada uno de los frameworks va a responder ante estas. Cada framework tiene sus ventajas y desventajas enfocándose en ciertos aspectos y dejando de lado otros, por esta razón determinar cuál es el que mejor se ajusta a nuestro proyecto es un aspecto vital para el desarrollador.

La herramienta de evaluación QSOS es muy útil para decisiones de selección de software libre, ya que ofrece al usuario o empresa un conjunto de criterios fácilmente medibles y estos dan una pauta de hacia dónde va el desarrollo del software, de su disponibilidad para ciertas plataformas y de cómo el desarrollo de aplicaciones esta industrializado dentro de la comunidad.

Capítulo 3

Modelos de Evaluación de Calidad de software

En la actualidad el software y en específico el software web juega un papel muy importante en el desarrollo de las organizaciones, no solo se ve involucrado en los procesos de negocios, producción y administración, sino también forma una parte integral de las estrategias corporativas para la generación de ventajas competitivas. De esta manera las empresas al realizar una compra de software privativo o elegir un software libre para sus negocios, se ve en la necesidad de escoger un software de calidad, que cumpla con eficiencia y eficacia los procesos necesarios para las necesidades del negocio.

La cantidad de software disponible en la actualidad es muy grande por lo que la calidad del mismo se vuelve un factor notable para la elección. Cuando se elige un software buscando la mejor calidad es imprescindible usar un modelo de evaluación de calidad de software, ya que un modelo de este tipo ofrece una guía con las mejores prácticas y un conjunto de criterios de evaluación que permita medir la calidad.

3.1. Definición de calidad

Un producto de software tiene una gran cantidad de diferencias a otro tipo de productos, por lo que definir su calidad es un poco particular, como aspecto introductorio para definir la calidad de software, es necesario definir este conjunto de características que lo identifican de otros productos o servicios. Las características que hacen al software un producto único son las siguientes:

- El software se desarrolla o construye, el proceso de manufactura es muy distinto al de un producto clásico.
- El software no se desgasta, el software es inmune a los males ambientales, por lo que debería tener la forma de curva ideal.
- El software es intangible, es un producto mental.
- El software con errores no se rechaza, ya que los errores en algún punto son inevitables.
- El mantenimiento de software es una actividad mucho más compleja que el mantenimiento de hardware.

La calidad y en específico la calidad de software tiene un gran número de definiciones, como ejemplo las siguientes como más importantes:

- Según Deming, calidad es “conformidad con los requisitos y confianza en el funcionamiento”
- La calidad es la suma de todos aquellos aspectos o características de un producto o servicio que influyen en su capacidad para satisfacer las necesidades, expresadas o implícitas (ISO 8402)
- Grado con el cual el cliente o usuario percibe que el software satisface sus expectativas. (IEEE 729-83)

Al observar las definiciones de calidad antes descritas se puede notar que no existe una definición universal, por lo que se vuelve un concepto subjetivo; para el presente trabajo de investigación se resume todas estas definiciones en la siguiente:

> La calidad de software es el grado en el que un sistema software en todas sus partes, satisface las necesidades y expectativas del cliente.

Es importante destacar que el software para su desarrollo tiene un conjunto de etapas (especificación de requerimiento, análisis, diseño, codificación, etc.) y cada una de estas etapas debe tener calidad, no basta con tener en cuenta la calidad cuando el producto esté terminado, dado que en esta última etapa los problemas pueden no tener una solución evidente o la solución puede ser muy costosa.

3.2 Tipos de calidad de software

La calidad de software al ser un concepto subjetivo, que depende del contexto y que significa diferentes cosas para diferentes personas, existe en tres tipos de dimensiones. (10)

- Calidad programada: La calidad programada, es aquella que se espera al inicio del proyecto, lo que se ha especificado hacer para el proyecto.
- Calidad realizada: La calidad realizada es el resultante de la finalización del proceso de desarrollo del software.
- Calidad necesaria: La calidad necesaria es aquella que el usuario espera, se la puede definir como lo que la gente necesita, para el óptimo desarrollo de sus actividades diarias.

Otra aproximación a la calidad está definida de acuerdo a los elementos y niveles en donde la calidad puede estar presente, la calidad puede estar a nivel organizacional, al nivel de proceso de software, al nivel del producto de software y por último en la calidad de los datos. Pese a que el presente trabajo de investigación está enfocado a la calidad a nivel de producto de software, es

conveniente para una comprensión global del tema, dejar sentados el resto de niveles en donde puede estar presente la calidad:

Calidad a nivel organizacional: la calidad a nivel organizacional consiste en la creación de una infraestructura organizativa apropiada para fomentar el trabajo de calidad de todas las personas y departamentos de la empresa. De esta forma se definen estándares para el trabajo a nivel de proyectos, en el caso de la empresa de software la infraestructura organizativa prevé las distintas actividades para el desarrollo y mantenimiento.

Calidad a nivel de proceso de software: Para alcanzar la calidad en un software, es necesario que esta intervenga en todos los procesos de construcción del mismo; así como, debe haber satisfacción por parte de todos los elementos que intervienen en el proceso, se pueden definir los siguientes: satisfacción de la alta dirección, satisfacción del personal involucrado en el desarrollo del sistema y por último la satisfacción del usuario final. En cada una de las etapas del proceso de desarrollo de software hay que tener en cuenta controles de calidad, ya que la calidad final del software siempre se construye de forma conjunta por todas las partes del proceso, no como una instancia final del mismo. Una metodología trascendente para asegurar la calidad a nivel de proceso de software se la observa en CMMI, ya que toma en cuenta todas las áreas de proceso importantes para un proyecto de software y definen una metodología para asegurar la calidad de las mismas.

Calidad a nivel de software: Es determinada como la calidad final que tiene el software para su uso; es decir, contempla aspectos como funcionalidad, portabilidad, fiabilidad, etc. Según sea el modelo utilizado para medir la calidad. Los modelos y estándares de calidad de software ayudan a poner en práctica el concepto general de calidad, estos modelos serán estudiados en detalle más adelante en este capítulo, explorando su estructura y composición, hasta los distintos tipos existentes en el mercado.

Calidad a nivel de datos: Dado que los datos es un componente esencial en los sistemas informáticos, la calidad de los mismos pese a tener una gran importancia para los sistemas, es difícil de definir. La calidad de datos se vuelve un concepto multidimensional, ya que puede ser tratado desde distintos puntos de vista según las necesidades de los consumidores de datos o de los diseñadores de sistemas. Algunas de las dimensiones que se pueden definir al tratar con la calidad de datos son: facilidad de acceso, cantidad apropiada de datos, facilidad de comprensión, facilidad de interpretación, facilidad de manipulación, etc.

3.3. Introducción a los modelos de calidad de software

Al hablar de calidad de software una de las principales inquietudes es la siguiente: ¿Es posible encontrar un conjunto de propiedades en un software que entreguen un indicio de cumplimiento de calidad? Para dar respuesta a esta interrogante nacieron los Modelos de Calidad. Dichos modelos definen la calidad de una forma jerárquica y tienen como objetivo resolver la complejidad del software mediante la descomposición; es decir, mediante un conjunto de factores y criterios que un producto software debe cumplir para ser de calidad.

El primer paso para implantar un modelo de calidad es determinar el modelo que se va a usar, esta es una actividad crucial; ya que, un modelo de calidad implica un cambio de mentalidad y una formación en el personal desarrollador. Un modelo de calidad de software se puede implementar en dos instancias:

Al introducir el concepto de calidad en una empresa, es decir cuando una empresa desarrolladora desea crear sus productos con calidad. En esta instancia la empresa tiene un conjunto de factores y criterios que definen la calidad para el desarrollo.

Cuando una empresa desea adquirir un software y se ve rodeada de opciones, entonces el modelo se utiliza para evaluar la calidad de un software o componente, para definir cuál es el más conveniente. Este aspecto tiene como premisa que la calidad de un producto de software es tan buena como los requisitos que describen el problema, el diseño que modela la solución, el código que conduce a un programa ejecutable y las pruebas que ejercitan el software para detectar errores.

El presente trabajo de investigación se centrara en los modelos de calidad de software a nivel de producto. Para realizar esta tarea primero se debe conocer los modelos de calidad que existen y que serán descritos en el siguiente punto.

3.4. Modelos de calidad de software

Los modelos de calidad nos permiten poner en práctica el concepto general de calidad, para esto los se definen de una forma jerárquica. Generalmente los modelos se descomponen en factores de calidad que se derivan en un conjunto de criterios de calidad y estos en un conjunto de métricas. (11)

- Factores de Calidad: Representan la calidad desde un punto de vista de usuario y son las características que componen la calidad.

- Criterios de Calidad: Son atributos que cuando están presentes, contribuyen a la calidad del factor asociado a estos.
- Métricas: Son medidas cuantitativas de ciertas características del producto, que cuando están presentes, dan una indicación del grado en que dicho producto posee un determinado criterio de calidad.

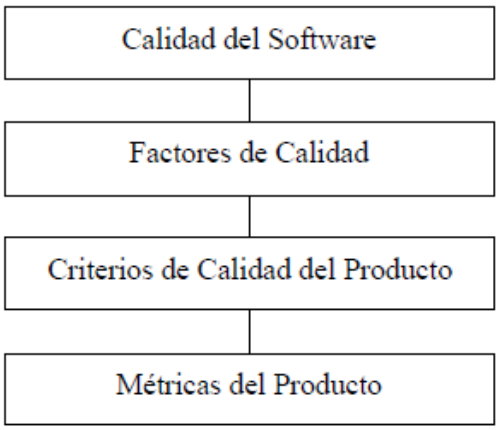

Figura 3.1: Estructura de un modelo de calidad.

Existen tres aspectos que abarca la calidad de software

Calidad Interna: Medible a partir de las características intrínsecas del software, tales como el código fuente, documentación, etc. La totalidad de características del producto son observadas desde una perspectiva interna.

Calidad Externa: Medible a partir del comportamiento del producto, es decir que el producto haga lo que se supone debe hacer. La calidad del producto se observa cuando este se ejecuta y se mide durante pruebas en un entorno simulado. La totalidad de características del producto son observadas desde una perspectiva externa.

Calidad de uso: Medible a partir de la utilización efectiva por parte del usuario. Se la puede definir como la calidad que percibe el usuario del producto cuando lo usa en un entorno y contexto específico.

3.4.1 Tipos de modelos de calidad

Para el estudio de los modelos de calidad primero es necesario definir los tipos de modelos existentes. (10)

- Modelos Fijos: Ofrecen un catálogo para la evaluación de componentes de software, estos modelos son rígidos, reutilizables y fáciles de utilizar.
- Modelos a la medida: Los modelos de este tipo se realizan para evaluar un componente de software en concreto, por esta razón son desechables pero muy flexibles ya que se ajustaran a lo que se desea medir.
- Modelos mixtos: Ofrece una combinación de los dos anteriores, brinda un conjunto de factores rígidos y otros flexibles para la medición.

En los siguientes puntos se describirán algunos de los diversos modelos de calidad de software, que se han desarrollado durante el pasar de los años para la evaluación de software.

3.4.2 Modelo de calidad de McCall

McCall fue el primero en implementar un conjunto de características o factores para evaluar el software, su modelo de calidad fue presentado en el año 1977, y se creó motivado por USA Air Force y DoD. El modelo es fijo y organiza los factores en tres ejes desde los cuales se puede contemplar la calidad del producto, estos son:

- Operación del producto: Define cinco factores para determinar las características operativas del producto.
- Revisión del producto: Define tres factores que definen como el producto puede soportar cambios.
- Transición del producto: Define tres factores que definen la adaptabilidad del producto hacia nuevos entorno.

Este modelo se organiza 11 factores en los tres ejes nombrados anteriormente, desde los cuales se puede observar la calidad de un producto. Cada factor tiene asociado sus respectivos criterios o características que a su vez determinan la calidad de cada uno de los factores. La siguiente Tabla ilustra el modelo de calidad propuesto por McCall:

Visión del usuario	**Factores de Calidad según McCall**
Operación del producto	Facilidad de uso
	Integridad
	Corrección
	Confiabilidad

	Eficiencia
Revisión del producto	Facilidad de mantenimiento
	Facilidad de prueba
	Flexibilidad
Transición del producto	Reusabilidad
	Interoperabilidad
	Portabilidad

Tabla 3.1: Factores de calidad de McCall.

Cada uno de los factores que describen cada uno de los ejes tiene asociado un criterio de calidad como lo describe la siguiente tabla:

Eje	Factor	Criterio
Operación del producto	Facilidad de uso	Facilidad de operación: Atributos del software que determinan la facilidad de operación del software
		Facilidad de comunicación: Atributos del software que proporcionan entradas y salidas fácilmente asimilables
		Facilidad de aprendizaje: Atributos del software que facilitan la familiarización inicial del usuario con el software y la transición del modo actual de operación
	Integridad	Control de acceso: Atributos del software que proporcionan control de acceso al software y los datos que maneja
		Facilidad de auditoría: Atributos del software que facilitan la auditoria de los accesos al software
		Seguridad: La disponibilidad de mecanismos que controlen o protejan los programas o los datos
	Corrección	Completitud: Atributos del software que proporcionan la implementación completa de todas las funciones requeridas.
		Consistencia: Atributos del software que proporcionan uniformidad en las técnicas y notaciones de diseño

		e implementación
		Trazabilidad: Atributos del software que proporcionan una traza desde los requisitos a la implementación con respecto a un entorno operativo concreto
	Fiabilidad	Precisión: Atributos del software que proporcionan el grado de precisión requerido en los cálculos y los resultados
		Consistencia
		Tolerancia a fallos: Atributos del software que posibilitan la continuidad del funcionamiento bajo condiciones no usuales.
		Modularidad: Atributos del software que proporcionan una estructura de módulos altamente independientes.
	Eficiencia	Eficiencia en Ejecución: Atributos del software que minimizan el tiempo de procesamiento.
		Eficiencia en almacenamiento: Atributos del software que minimizan el espacio de almacenamiento necesario.
Revisión del producto	Facilidad de mantenimiento	Modularidad
		Simplicidad
		Consistencia
		Concisión: Atributos del software que posibilitan la implementación de una función con la menor cantidad de códigos posible.
		Auto-descripción: Atributos del software que proporcionan explicaciones sobre la implementación de las funciones.

	Facilidad de prueba	Modularidad
		Simplicidad
		Auto-descripción
		Instrumentación: Atributos del software que posibilitan la observación del comportamiento del software durante su ejecución para facilitar las mediciones del uso o la identificación de errores.
	Flexibilidad	Auto-descripción
		Capacidad de expansión: Atributos del software que posibilitan la expansión del software en cuanto a capacidades funcionales y datos.
		Generalidad: Atributos del software que proporcionan amplitud a las funciones implementadas.
		Modularidad
Transición del producto	Facilidad de reutilización	Auto-descripción
		Generalidad
		Modularidad
		Independencia entre Sistema y software: Atributos del software que determinan su dependencia del entorno operativo.
		Independencia del hardware: Atributos del software que determinan su dependencia del hardware.
	Interoperabilida	Modularidad
		Compatibilidad de comunicaciones: Atributos del software que posibilitan el uso de protocolos de comunicación e interfaces estándar.
		Compatibilidad de datos: Atributos del software que posibilitan el uso representaciones de datos estándar.
		Estandarización de datos: El uso de estructuras de datos y de tipos estándar a lo largo de todo el programa.
	Portabilidad	Auto-descripción

		Modularidad
		Independencia entre Sistema y software
		Independencia del hardware

Tabla 3.2: Modelo de calidad de McCall.

3.4.3 Modelo de calidad de BOEHM

Es uno de los modelos de calidad más conocidos en el mercado, fue presentado por Barry Boehm en el año de 1978; esta propuesta de modelo se basa en que el software debe:

- Hacer lo que el usuario quiere que haga.
- Utilizar los recursos computacionales correcta y eficientemente.
- Ser fácil de usar y aprender para los usuarios.
- Estar bien diseñado, bien codificado y sus pruebas y mantenibilidad no deben ser complicadas.

Este modelo agrega algunas características a las existentes en el modelo de MacCall y es representado mediante características de alto nivel, de nivel intermedio y características primitivas. Las características de alto nivel de este modelo representan requerimientos generales de uso y son definidas de la siguiente manera:

- Utilidad per-se: Característica que representa cuan usable, confiable y eficiente es el producto en sí mismo.
- Mantenibilidad: Característica que representa la facilidad para modificar, entender y rastrear el sistema en cada una de las etapas del ciclo de vida.
- Utilidad general: Característica que indica si el sistema puede seguir usándose aun si cambia el ambiente en el que se desarrolla.

Así como el modelo de calidad de McCall, este modelo también se divide de una forma jerárquica definida de la siguiente manera:

Característica de Alto Nivel	**Características Intermedias**	**Características primitivas**
Utilidad per-se	Confiabilidad	Auto-contención
		Exactitud
		Complejitud
		Consistencia

		Robustez
	Eficiencia	Accesibilidad
		Eficiencia de uso de dispositivos
	Usabilidad	Robustez
		Accesibilidad
		Comunicación
Mantenibilidad	Testeabilidad	Comunicación
		Auto-contención
		Estructuración
	Facilidad de entendimiento	Consistencia
		Estructuración
		Concisidad
		Legibilidad
	Modificabilidad	Estructuración
		Aumentabilidad
Utilidad general	Portabilidad	independencia de dispositivos
		auto-contención

Tabla 3.2: Características de calidad de Bohem.

3.4.4 Modelo de calidad de Gilb

Modelo propuesto por Tom Gilb en 1988. Lo interesante de este modelo es que presenta como aspecto fundamental la definición de atributos de calidad que realmente le interesan al usuario y el nivel de calidad que debe tener cada uno de ellos. Se le considera un modelo mixto ya que propone ciertas características de calidad pero las subcaracterísticas y atributos dependerán y serán definidas de acuerdo a cada proyecto. Gilb propone el siguiente conjunto de características que se establecen como base para proporcionar una guía a los equipos que evalúen la calidad de un producto de software. (12)

- Corrección: Se establece que un producto es correcto o no de acuerdo al grado con el que este cumple la función requerida por el usuario. Si un programa no opera correctamente, no dará valor agregado a sus usuarios
- Facilidad de Mantenimiento: Grado en el que el sistema ofrece la posibilidad de corregir errores de un programa, adaptarlo a nuevos entornos o mejorarlo si es el caso o el cliente lo requiere.

- Integridad: Habilidad de un sistema para resistir ataques contra su seguridad, tanto accidentales como intencionados, Gilb sugiere la utilización de los siguientes atributos como base:
    - Seguridad: Probabilidad de que se pueda repeler un determinado ataque contra el sistema
    - Amenazas: Probabilidad de que un ataque de cualquier tipo ocurra en un tiempo determinado.
- Facilidad de uso: Es un intento por cuantificar el grado de facilidad del sistema para el usuario, es decir que tan amigable e intuitivo es el sistema para el usuario.

Las características definidas por Gilb se pueden medir mediante varias subcaracterísticas o métricas detalladas. Para cada una de ellas, se deben especificar los siguientes conceptos:

- Nombre y definición de la característica.
- Escala o unidades de medición.
- Recopilación de datos o prueba.
- Valor previsto.
- Valor óptimo.
- Valor en el sistema actual.
- Comentarios.

3.4.5 Modelo de calidad GQM (Goal-Question-Metric)

Modelo desarrollado en el año de 1998 por Basili y Rombach; esta es una propuesta basada como su nombre lo indica en objetivos/metas para la definición de modelos de calidad; es decir, si recurrimos a los tipos de modelos de calidad definidos en el punto 3.4.1 se puede distinguir que este es un modelo a la medida. Plantea un enfoque de definición basándose en la identificación de objetivos a lograr. Con este paradigma se logra desarrollar y mantener un conjunto significativo de métricas que ayudan a:

- Alinear las métricas con las metas y objetivos del negocio.
- Mejorar el proceso del software.
- Gerenciar el riesgo.
- Mejorar la calidad del producto.

Este modelo proporciona la estructura para obtener los objetivos cruciales del proyecto y consta de las siguientes 3 etapas (11):

- Listar los objetivos principales del desarrollo y mantenimiento del proyecto.
- Para cada objetivo, se deben establecer preguntas que deben contestarse para saber si se están cumpliendo los objetivos.
- Decidir que medir para poder contestar las preguntas de manera adecuada; es decir, desarrollar un conjunto de métricas que ayuden a responder la pregunta.

El modelo GQM está orientado a las metas y por este hecho las medidas resultantes obtenidas en la evaluación deben ser relacionadas para tener un enfoque a un contexto global de lo que se desear evaluar; es por esto que la interpretación de resultados en este modelo es de gran importancia. El modelo se descompone en tres niveles que se ejemplifican en la siguiente figura:

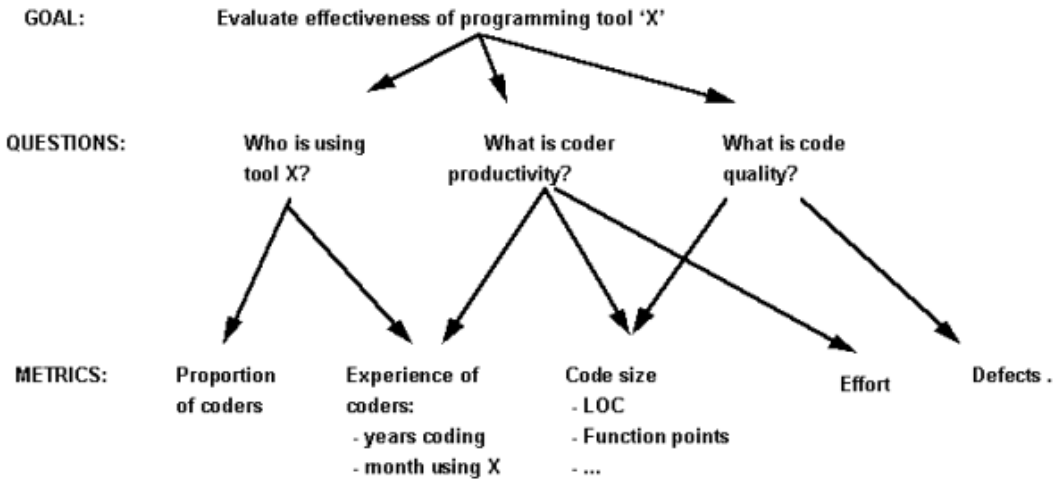

Figura 3.2: Ejemplo de métricas derivadas de objetivos y preguntas. (11)

Los tres niveles de este modelo se definen a continuación:

- Nivel de objetivo o nivel conceptual: El objetivo o meta es definido en cuanto a los requerimientos de la organización, se debe tener en cuenta toda la variedad de razones que implica el ambiente y estado actual de la empresa. Este representa el nivel máximo de característica de calidad.
- Nivel de preguntas o nivel operacional: Se la define como un conjunto de preguntas para representar de mejor manera la meta especifica del nivel 1. Cada característica de nivel 1 se la descompone en un conjunto de preguntas o subcaracterísticas.

- Métricas o nivel cuantitativo: Cada métrica está asociada con una pregunta o característica de nivel 2, las métricas proporcionan un valor cuantitativo para determinar si una pregunta o meta se ha cumplido y en qué grado lo ha hecho.

3.4.6 Modelo de calidad de FURPS

Modelo desarrollado por Hewlett-Packard en 1987 en el que se describen una serie de factores de calidad. Modelo que puede ser sintetizado en dos pasos: asignación de prioridades sobre las características del modelo y definición de atributos y métricas de calidad.

El modelo de FURPS incluye cinco categorías que se describen en la siguiente tabla:

Sigla	Tipo de Requerimientos		Descripción
F	Funtional	Funcional	Características
			Capacidades
			aspectos de seguridad
U	Usability	Facilidad de Uso	Factores humanos (interacción)
			Ayudas
			Documentación
R	Reliability	Fiabilidad	Frecuencia de fallos
			Capacidad de recuperación de un fallo
			Grado de previsión
P	Performance	Rendimiento	Tiempos de respuesta
			Productividad
			Precisión
			Disponibilidad
			Uso de recursos
S	Supportability	Soporte	Adaptabilidad
			Facilidad de configuración

Tabla 3.3: Modelo FURPS.

Como se puede observar en la tabla 3.3 este modelo puede ser separado en dos partes, correspondientes a aspectos funcionales y no funcionales:

- Aspectos Funcionales (F): Definen las capacidades y funciones que debe tener el sistema, sin tener en cuenta restricciones de tipo físico.
- Aspectos no Funcionales (URPS): Estos aspectos describen atributos que el sistema debe tener y que son requeridos.

3.4.7 Modelo de calidad de ISO 9126-1

Estándar internacional para la evaluación de software, fue propuesto debido a que durante muchos años se buscaba un patrón común para poder comparar productos, naciendo así como una variante al modelo de McCall y Boehm. El modelo de calidad de la ISO clasifica la calidad de software en un conjunto estructurado de características y subcaracterísticas.

En este modelo se definen los conceptos de calidad externa, calidad interna y calidad en uso, nombrados anteriormente en este capítulo. Este modelo usa estas distintas perspectivas de calidad en las distintas etapas del ciclo de vida del software:

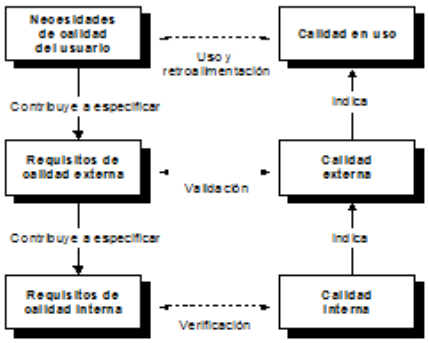

Figura 3.3: Perspectivas de calidad en el ciclo de vida del software. (13)

El modelo de calidad de ISO consta de seis características que definen la calidad interna y externa del producto software a evaluar, estas se derivan en subcaracterísticas que pueden ser medidas mediante métricas. El modelo de calidad ISO 9126-1 se define bajo la siguiente estructura jerárquica:

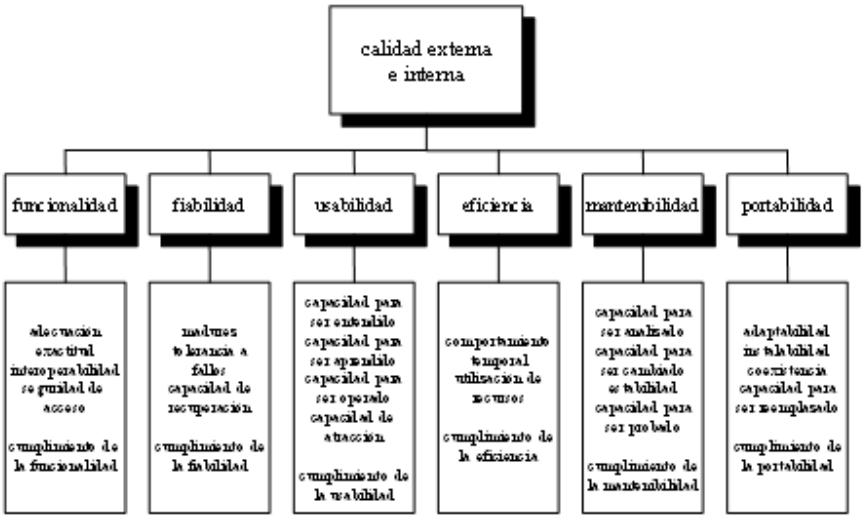

Figura 3.4: Estructura modelo de calidad para calidad interna y externa. (13)

Funcionalidad: la funcionalidad es descrita por el modelo como la capacidad que tiene el producto software para proporcionar funciones que satisfacen necesidades declaradas, al ser usadas bajo condiciones específicas. Se puede entender a la funcionalidad como la extensión que tiene el producto refiriendo a la cantidad de posibilidades que un sistema provea. Un buen sistema de software tiene en cuenta que la excesiva cantidad de funciones puede perjudicar a la facilidad de uso del mismo.

Fiabilidad: La fiabilidad esta descrita como la capacidad del software de mantener un nivel especificado de prestaciones usado en condiciones específicas, en otras palabras se puede decir que es la probabilidad de que un sistema software no falle durante un determinado periodo de tiempo en un entorno concreto. La fiabilidad o confiabilidad es un factor de calidad importante, ya que si el sistema falla frecuentemente, no importará mucho si el resto de factores tienen un buen nivel de calidad.

Usabilidad: La usabilidad es también llamada en otros modelos como facilidad de uso, ISO lo define como la capacidad del producto para ser entendido, aprendido, usado y ser atractivo para el usuario. La usabilidad incluye factores como la facilidad de instalación, operación y monitoreo así como también toma en cuenta los diferentes entornos de usuarios a los que puede afectar el software, incluyendo los distintos niveles de experiencia de los mismos. El desarrollo de un sistema sin contar la usabilidad del mismo puede crear gran cantidad de riesgos importantes para una organización, ya que con esto disminuirá la satisfacción del usuario respecto al sistema con una probabilidad de aumento de errores por la falta de conocimiento del usuario.

Eficiencia: Es la habilidad o capacidad que tiene el software para mantener prestaciones apropiadas relativas al uso sobre los recursos de hardware, es decir el software es eficiente cuando cumple con sus funciones usando lo mínimo en recursos de hardware como CPU, memoria, red, etc. El software es eficiente si realiza un uso racional de los recursos que le son dispuestos.

Mantenibilidad: El mantenimiento de software es una de las actividades más costosas de la ingeniería de Software, por lo que la facilidad de mantenimiento es un punto crucial en la adquisición o producción de software. La mantenibilidad se define como la capacidad de un producto para ser modificado, las modificaciones pueden incluir correcciones (mantenimiento correctivo), mejoras (mantenimiento perfectivo) o adaptación del software a nuevos entornos (mantenimiento adaptativo).

Portabilidad: La portabilidad se define como la capacidad de un producto software para ser transferido desde un entorno a otro, estos pueden ser entornos de hardware o software.

3.4.8 Modelo de calidad de Dromey

Modelo de calidad que fue creado por Robert Dromey en 1996, este sugiere una técnica genérica para construir un modelo de calidad. Resalta que la calidad de un producto software está determinada en gran medida por los componentes del mismo, en la dimensión de sus propiedades tangibles y las propiedades de su composición.

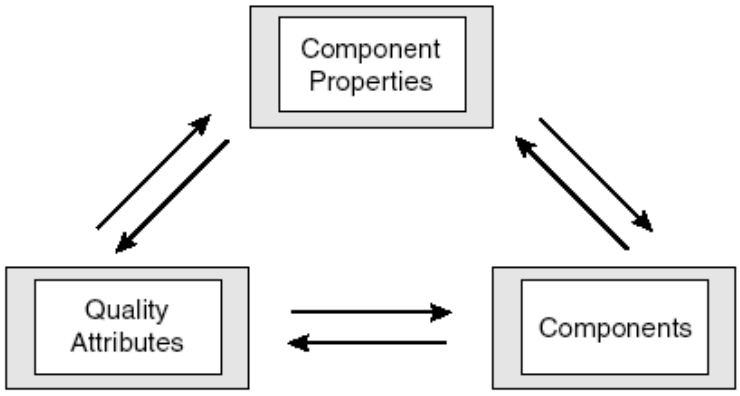

Figura 3.5: Aspectos básicos del modelo de DROMEY. (14)

El modelo tiene el propósito de trabajar con una estructura que permite construir y utilizar un modelo de calidad práctico para evaluar las etapas de Determinación de los requerimientos, Diseño e Implementación. La idea que propone Dromey es que el reconocimiento de que la

evaluación de calidad es diferente para cada producto y que de aquí surge la necesidad de una forma más dinámica para modelar el proceso necesario.

El modelo determina una serie de propiedades para analizar la calidad y las categoriza en cuatro ramas básicas con propiedades asociadas a cada una de ellas. Las siguientes son las ramas básicas:

- Correctitud: Rama encargada de la medición de que tan correcto es el software en torno a la funcionalidad que debe tener el mismo y a su confiabilidad.
- Internas: Rama encargada de la medición de que tan bien un componente ha sido entregado de acuerdo a su objetivo, implementación, o que tan bien ha sido compuesto.
- Contextuales: Determinan que tan bien los componentes son compuestos y las influencias que ejercen sobre la calidad del producto,
- Descriptivas: Un software útil es aquel en el cual es fácil de entenderlo y utilizarlo de acuerdo a su propósito.

La siguiente tabla ilustra las ramas básicas del modelo con sus propiedades asociadas.

Modelo de Dromey	
Propiedades del producto	Atributos de Calidad
Correctitud	Funcionalidad
	Confiabilidad
Internas	Mantenibilidad
	Eficiencia
	Confiabilidad
Contextuales	Mantenibilidad
	Reusabilidad
	Portabilidad
	Confiabilidad
Descriptivas	Mantenibilidad
	Reusabilidad
	Portabilidad
	Usabilidad

Tabla 3.4: Modelo de Dromey.

El modelo de calidad de Dromey se estructura en torno a cinco pasos:

1. Seleccionar el conjunto de atributos que se necesitan evaluar, estos pueden ser aquellos de alto nivel por ejemplo usabilidad, portabilidad, etc.
2. Realizar una lista de todos los componentes o módulos del sistema, en este paso se determinan los distintos componentes del producto con un apropiado nivel de detalle
3. Identificar las propiedades de calidad de cada componente; es decir, en cada uno determinar y categorizar las implicaciones de calidad.
4. Determinar cómo afecta cada propiedad en los atributos de calidad
5. Evaluar el modelo de calidad.

3.5 Selección de un modelo de calidad

Se han enumerado algunos de los modelos de calidad más conocidos ya sea por hechos históricos como el de McCall por ser el primer modelo de calidad conocido o por ser un estándar internacional como el ISO 9126-1, pero al analizar la gran cantidad de modelos que existen actualmente, tomando en cuenta que en este capítulo solo se detallaron algunos de ellos se encuentra la problemática de cuál es el modelo más adecuado para las futuras evaluaciones.

Para la selección de un modelo de calidad se considera adecuado definir ciertos criterios para poder comparar los distintos modelos, aspectos como la estructura de cada modelo, la organización que mantiene al mismo, la edad del modelo en el mercado, la comunidad que está en contacto con el modelo, los detalles de características y subcaracterísticas son aspectos importantes para la selección. Si bien la definición de estos criterios es de mucha importancia para la selección del modelo de calidad, es también importante definir cuáles son los aspectos más importantes para el dominio especifico en el cual se desarrollará el modelo, para el caso actual el tema para el cual se tiene que seleccionar el modelo son los frameworks web java. Se tomaran los siguientes criterios para la selección:

- Tipo de modelo: Como ya se definió en el punto 3.4.1 del presente capitulo existen tres tipo de modelos de calidad de software en referencia a su construcción, al ser los frameworks un tema muy específico se tomará en cuenta como más apropiados los modelos que sean mixtos, ya que proveen características pre definidas y también permiten extenderse un poco en lo referente al tema tratado.
- Relación entre elementos y estructura: las relaciones y estructura de cada modelo son también un aspecto importante, ya que un modelo con una estructura bien definida y una

relación coherente entre sus elementos, ayudará a una mejor aplicación del mismo, y de esta manera se podrá obtener un resultado más preciso en la evaluación.

- Características y subcaracterísticas: Son un aspecto vital para la selección de un modelo de calidad ya que estas actúan con la estructura del modelo como eje sobre el cual se evaluará los frameworks, para que el modelo sea el adecuado debe tener las características y subcaracterísticas que sustenten de forma general los aspectos que puede tener un framework.
- Edad del modelo, documentación y comunidad: Estos aspectos pese a no ser parte de la funcionalidad que se tiene de un modelo, se deben tomar en cuenta en la selección, debido a que se debe tener suficiente documentación para conocer en detalle al modelo y también para tener una idea de la metodología de aplicación del mismo. Por último una comunidad que usa activamente el modelo indica la acogida que tiene el modelo a nivel internacional.

Todos estos criterios detallados son importantes para la selección de un modelo, es por esto que se debe realizar una comparativa de cada uno de los modelos que han sido estudiados en este capítulo, para así determinar el que será usado para la posterior evaluación de calidad. El siguiente cuadro ilustra algunos de los criterios descritos:

Modelo	Año de publicación	Tipo de Modelo	Documentación por búsquedas en Google (Modelos + Nombre)	Desarrollador del modelo
McCall	1977	Fijo	235000	Autor aislado
Boehm	1978	Fijo	325000	Autor aislado
GILB	1988	Mixto	21700	Autor aislado
GQM	1998	Medida	43100	Autores aislados
FURPS	1987	Fijo	5240	Empresa multinacional HP
ISO 9126-1	2001	Mixto	647000	Organización Internacional de estandarización
Dromey	1996	Medida	3270	Autor aislado

Tabla 3.5: Características importantes para selección de un modelo de calidad.

En la tabla 3.5 se observan algunos de los criterios importantes para la selección de un modelo, al analizar se puede observar que el modelo que por ahora es el más conveniente para el tema tratado, es el modelo de la ISO ya que es el más actual debido a su año de publicación, es un tipo de modelo mixto que como ya se definió anteriormente es el tipo de modelo más conveniente

para el propósito de la evaluación de frameworks, fue desarrollado por la organización internacional para la estandarización lo cual garantiza que estará siempre en evolución y que fue desarrollado a partir de un estudio extenso de cuáles son las características de calidad que debería cumplir el software. En la siguiente tabla se detallaran aspectos igual de importantes para la selección como lo son detalles de la estructura de cada uno de los modelos, así como las características que en cada uno está definido por defecto:

Modelo	Estructura	Elementos Estructurales	Definiciones
McCall	Jerárquica	3 Ejes de calidad, 11 Factores, 23 Criterios	Elementos definidos, métricas a definir por el evaluador
Boehm	Jerárquica	3 Características de alto nivel, 7 características intermedias, 15 características primitivas	Características definidas, métricas para definir por el evaluador
GILB	Jerárquica	3 Características, subcaracterísticas y métricas a definir	Características definidas, subcaracterísticas y métricas a definir por el evaluador
GQM	Jerárquica	Modelo basado en objetivos, preguntas y métricas a definirse, no proporciona ninguna pre-establecida	Modelo definido, objetivos, preguntas y métricas a definir por el evaluador
FURPS	Jerárquica	5 Tipo de requerimientos, un funcional (F) y 4 no funcionales (URPS), 16 característica	Requerimientos definido, características definidas, métricas a definir por el evaluador
ISO 9126-1	Jerárquica	6 Características, 27 subcaracterísticas definidas asociadas a las características	Características definidas, subcaracterísticas definidas, métricas a definir por el evaluador
Dromey	Basado en tres características de componentes	3 características que deben ser tomadas en cuenta, 4 ramas básicas que se deben asociar a cada propiedad	Componentes, Propiedades de los componentes, atributos de calidad a ser definidos por el evaluador, 4 ramas básicas definidas de asociación

Tabla 3.6: Criterios estructurales para selección de un modelo de calidad.

La estructura es un aspecto relevante en la selección de un modelo de calidad, como se puede observar en la tabla 3.6, casi todos los modelo estudiados están caracterizados por tener una estructura bien definida en forma jerárquica, por lo que el siguiente aspecto a analizar trata los elementos estructurales de cada uno de los modelos, siendo el modelo de Dromey y el GQM

modelos a la medida, proporcionan solo una referencia estructural de cómo debería ser el modelo por lo que se no se los considera convenientes para el tipo de evaluación que se requiere realizar, el resto de modelos proporciona una estructura con elementos útiles en una evaluación. Luego de un análisis de todos los factores, elementos, características y subcaracterísticas que ofrecen cada uno de los modelos, se constató que el modelo que más se acerca a las necesidades de evaluación para un framework es el modelo de la ISO, ya que ofrece un conjunto de características requeridas para la evaluación de software no solo útiles en el dominio que se busca en el presente trabajo de graduación, sino también útiles para cualquier dominio. Las subcaracterísticas que este modelo ofrece en torno a la funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad son las adecuadas para cualquier software y pueden ser usadas en cualquier proyecto.

Tanto los aspectos como la actualidad del modelo, la cantidad de documentación que existe en la web, el tipo de modelo y los aspectos estructurales definidos, hacen que el modelo ISO 9126-1 sea el más adecuado para los fines de evaluación de frameworks web java.

3.6 Conclusiones

Los modelos de calidad de software son herramientas muy valiosas a la hora de contemplar la posibilidad de adquirir software COST o de seleccionar un software para el desarrollo como es el caso contemplado de los frameworks web. Los modelos contemplan una variedad de factores sobre el software que posibilitará al evaluador tomar decisiones sobre lo más conveniente para la empresa o proyecto en el cual se desarrolle, lo cual mejorará la productividad, y hará que el desarrollo de software se base en una herramienta que refleje de mejor manera las necesidades de la organización.

Al momento de estudiar los modelos para la evaluación de calidad de un framework, se observan una gran cantidad de opciones las cuales pueden confundir al evaluador, es por esto que se debe implantar una metodología que tome en cuenta ciertos criterios para la selección del modelo. En el caso de los frameworks web se tomaron como criterios aspectos como el tipo de modelo, su estructura, la completitud del mismo, edad, documentación disponible y la organización que respalda al mismo; todos estos, son criterios llevaron hacia la conclusión de que el modelo más conveniente para evaluar un framework web es el modelo ISO 9126-1. El modelo 9126-1 tiene gran cantidad de características que apoyan la evaluación de la mejor manera posible, ofreciendo al evaluador un catálogo completo características y subcaracterísticas para evaluar cualquier tipo de software, teniendo además otros factores que lo hacen más atractivo, como el hecho de ser un estándar internacional o ser uno de los modelos más actuales y que exploran de mejor manera la problemática actual de la calidad del software.

Capítulo 4

Aplicación Práctica

La aplicación de un framework de forma práctica es de gran importancia, ya que desde un aspecto teórico se pueden describir las tecnologías que cada uno de los frameworks utiliza para su desarrollo, pero es desde el punto de vista empírico que se observa la utilidad que tienen en la vida real. En la aplicación práctica se desarrollará un sistema con cada uno de los frameworks estudiados en el capítulo 2. La aplicación práctica tratará un sistema de facturación y tendrá un enfoque de desarrollo en cascada, siguiendo cada uno de los pasos determinados por este, buscando bajo esta premisa el éxito en el desarrollo del sistema.

4.1 Enfoque de desarrollo en cascada

Los enfoques de desarrollo son también llamados modelos, estos modelos proveen una estructura funcional para el desarrollo de software, es decir proporcionan un conjunto de pasos y una manera de fabricar el software, para el sistema de facturación desarrollado en este capítulo se usará un modelo de desarrollo en cascada, ya que la simplicidad del mismo facilita el desarrollo de aplicaciones ejemplo. Un modelo de desarrollo en cascada destaca por ser un proceso secuencial en el cual se ordena rigurosamente las etapas del proceso para la fabricación de software, de tal manera que cada una de las etapas para comenzar debe esperar que finalice la etapa anterior a la misma. El propósito del modelo en cascada es garantizar que se cumplan cada una de las etapas.

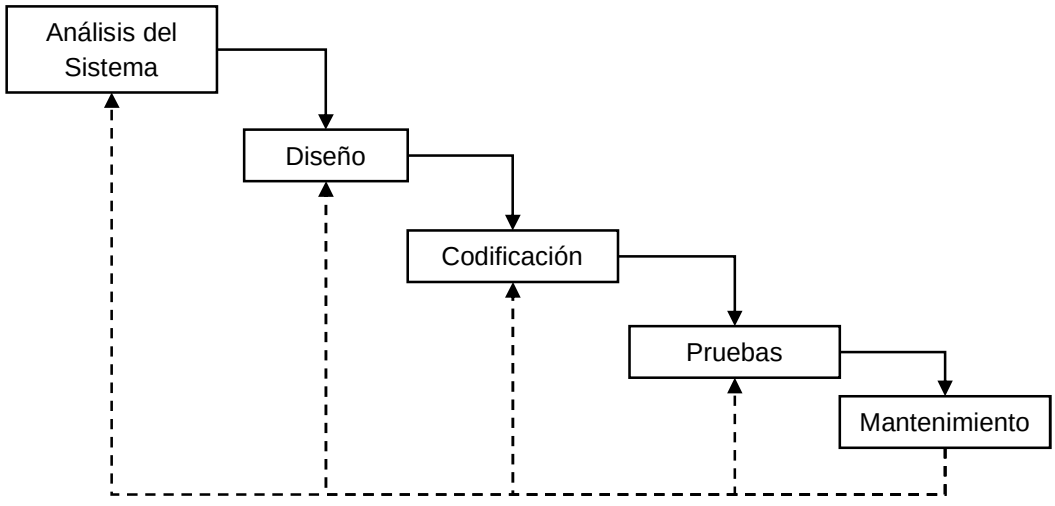

Figura 4.1: Etapas del modelo en cascada. (15)

Los procesos comunes en la construcción de un software están ejemplificados en el enfoque, siendo todos importantes para tener los resultados deseados, como una breve introducción a los mismos lo siguiente:

- Análisis del sistema: el proceso de análisis está basando en la recopilación de los requisitos el cual se centra e intensifica especialmente en el software. El ingeniero de software debe comprender el ámbito de la información, así como la función, el rendimiento e interfaces requeridas.
- Diseño: el diseño del software se enfoca en cuatro atributos distintos del programa: la estructura de los datos, la arquitectura del software, el detalle procedimental y la caracterización de la interfaz. El proceso de diseño traduce los requisitos en una representación del software.
- Codificación: el diseño debe traducirse en una forma legible para la máquina. El paso de codificación realiza esta tarea.
- Pruebas: una vez que se ha generado el código comienza la prueba del programa. Las pruebas se centran en la lógica interna del software, y en las funciones externas, realizando pruebas que aseguren que la entrada definida produce los resultados que realmente se requieren.
- Mantenimiento: el software sufrirá cambios después de que se entrega al cliente. Los cambios ocurrirán debido a la aparición de errores a lo largo del tiempo o a que el software deba adaptarse a cambios en su entorno o debido a que el cliente requiera ampliaciones funcionales o del rendimiento.

Como se puede observar el modelo en cascada es muy simple en comparación con los otros modelos existentes en el mercado, debido a su simplicidad es ideal para el desarrollo de las aplicaciones en cada uno de los frameworks, ya que el presente capitulo tiene la intención de desarrollar conocimientos de los frameworks en base a su aplicación. Para finalizar este punto y concretar las ventajas que tiene este modelo así como sus desventajas.

Ventajas:

- La planificación del producto es sencilla, solo se deben seguir en orden los pasos propuestos por el modelo.
- La calidad del producto resultante es alta.
- Permite trabajar con personal poco cualificado.

Desventajas

- Necesidad de tener todos los requisitos al principio. Lo normal es que el cliente no tenga perfectamente definidas las especificaciones del sistema, o puede ser que surjan necesidades imprevistas.
- Si se han cometido errores en una fase es difícil volver atrás
- El cliente no verá los resultados hasta el final, con lo que puede impacientarse.
- No se tienen indicadores fiables del progreso del trabajo (síndrome del 90%).

4.2 Análisis del sistema

El análisis del sistema es un punto central en el desarrollo de sistemas, ya que desde el análisis se estudia y comprende el problema que el sistema va a solucionar, se busca tener claro cuáles son las necesidades del cliente para así tener lo más claro posible cual va a ser el dominio en el cual el sistema se desarrollará. Como se entiende ya los requisitos son un aspecto central del sistema y desde estos se desarrollará el mismo, por tanto se debe buscar una metodología para recolectarlos, priorizarlos y eliminar la ambigüedad que puede existir en los mismos.

4.2.1 Especificación de requisitos

El sistema que se pretende construir es un sistema desarrollado en la web que permita al usuario la gestión de clientes, artículos y facturas, por lo que se comenzará definiendo el ámbito en el cual se desarrollará el sistema el cual tendrá en consideración los siguientes puntos:

- Gestión de clientes.
- Gestión de productos.
- Generación e ingreso de facturas de compra de productos.
- Facturas de venta.
- Anulación de facturas.
- Autentificación de usuarios.

El sistema tendrá solamente un actor el cual será el usuario del mismo, encargado de realizar todos los mantenimientos y de realizar las facturas, en el siguiente gráfico se ilustra los casos de uso que contemplará el sistema:

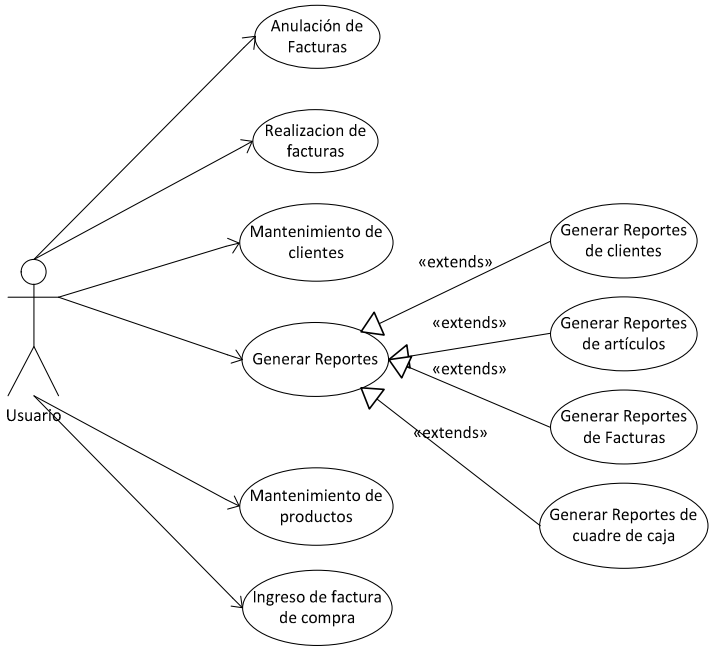

Figura 4.2: Diagramas de casos de uso.

Cada uno de los casos de uso descritos en el gráfico debe que ser descritos y priorizados, para de esta manera establecer cuáles son las prioridades y detallar los requisitos asociados a cada uno.

Ver anexo 2

4.2.2 Diagrama de flujo de datos

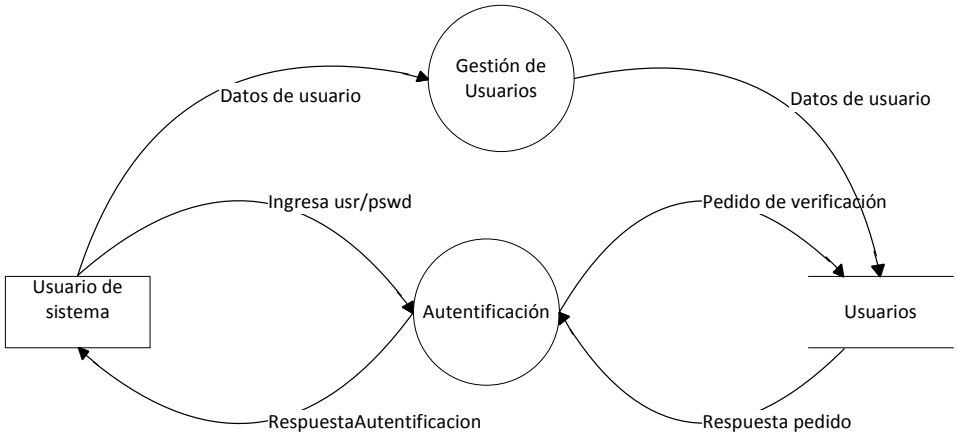

Figura 4.3: Diagrama de autentificación y gestión de usuarios.

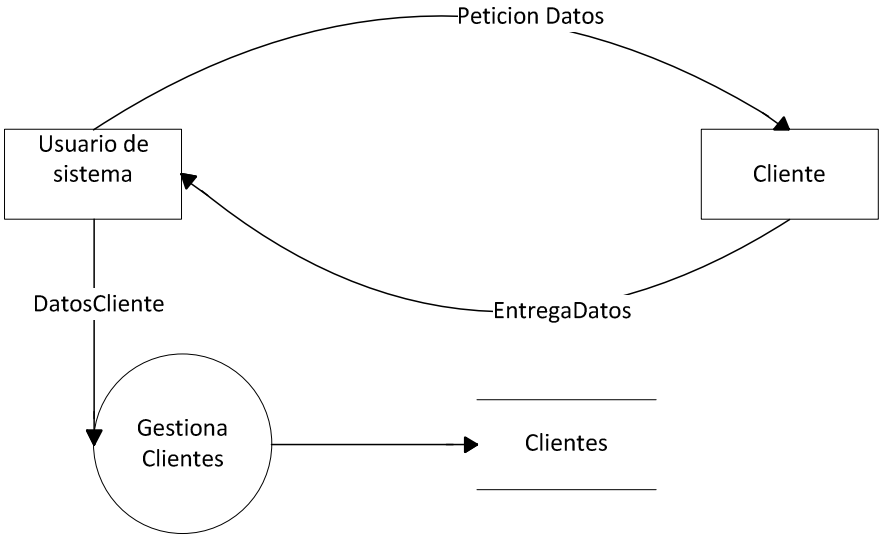

Figura 4.4: Diagrama de Gestión de clientes.

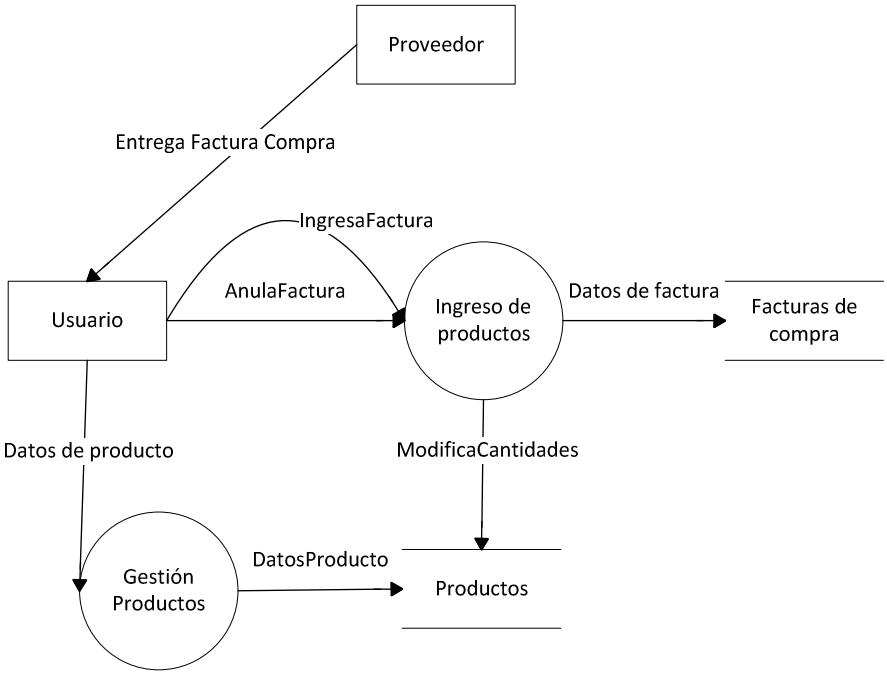

Figura 4.5: Diagrama de gestión e ingreso de productos.

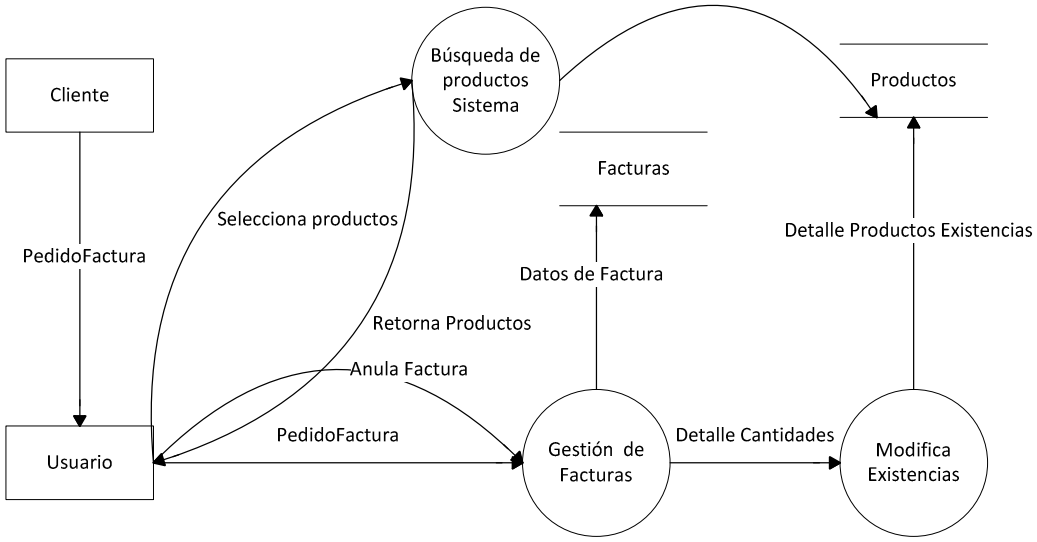

Figura 4.6: Diagrama de facturación.

4.2.3 Modelo de contenido e interacción

Al realizar una aplicación web se debe considerar siempre los conceptos de contenido e interacción, ya que en estos puntos se detallan los elementos estructurales que son necesarios para satisfacer los requisitos y se detalla también la interacción que tendrá el usuario final con el aplicativo. Estos aspectos son previos al diseño del sistema y algunos autores los consideran como una parte que provee contenido sustancioso al análisis de sistemas orientados a la web.

4.2.3.1 Modelo de contenido

Modelar los elementos estructurales de la aplicación web es una tarea importante y para esto nos ayuda esta etapa del análisis. Existen varias formas de presentar un modelo de contenido, mediante el modelo CRC (Clase-Responsabilidad-Colaborador) y otra forma de representarlo que es la que será usada en el presente trabajo de graduación es mediante diagramas de clase más diagramas de colaboración.

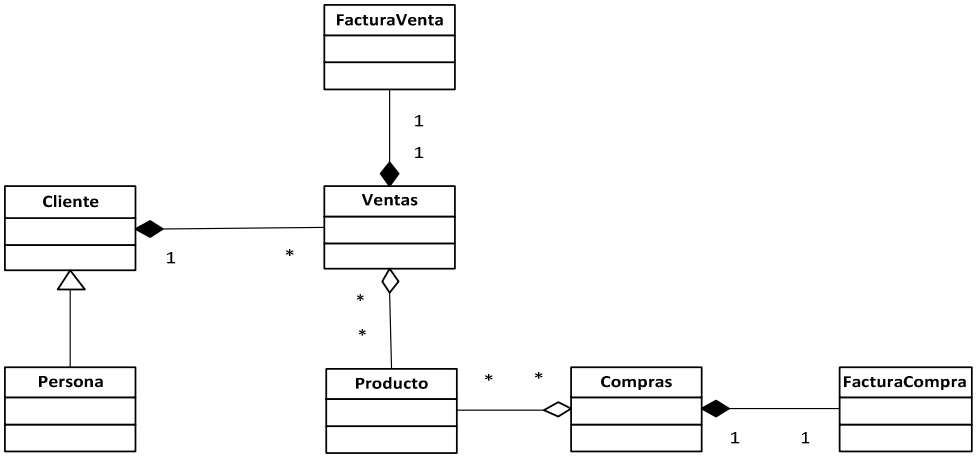

Figura 4.7: Diagrama de clases UML.

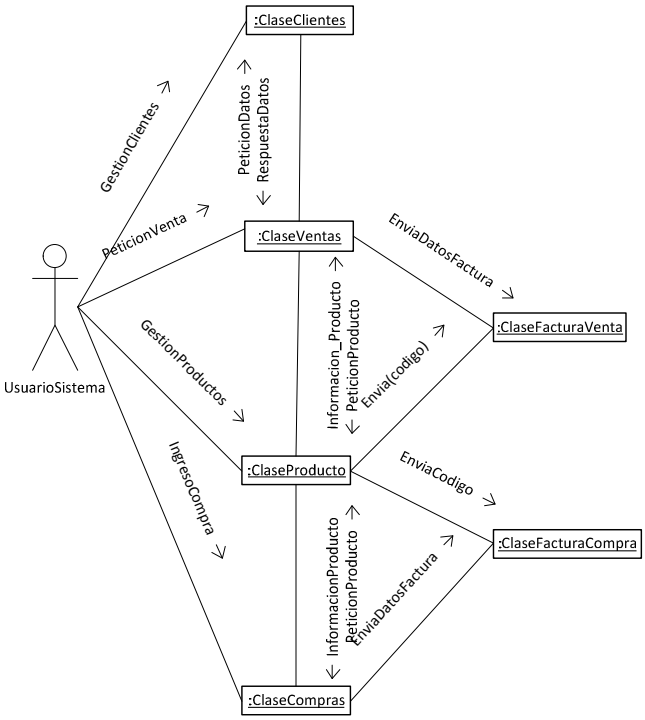

Figura 4.8: Diagrama de colaboración UML.

4.2.3.2 Modelo de interacción

El modelado de interacción es la parte del análisis de sistemas web que permite establecer los parámetros de comunicación del usuario final con la aplicación web, o dicho en otras palabras la conversación que existirá entre estas dos partes. Para la determinación de la interacción se puede utilizar cuatro técnicas:

- Casos de uso: proporcionan una visión simple de la interacción, utilizado también en la especificación de requisitos, véase punto 4.3.1
- Diagramas de secuencia: Utilizado posteriormente en este punto, “muestra como interactúa el usuario con las clases del contenido y análisis”. (16)
- Diagramas de estado: Es otra de las formas de representar la interacción, añade un aspecto interesante al agregar información sobre los patrones de navegación. (16)
- Prototipo de interfaz de usuario: actividad del diseño que se recomienda anticiparla en el análisis, para el caso actual se realiza en un punto posterior véase 4.4.2 diseño de interfaz.

Diagrama de secuencia:

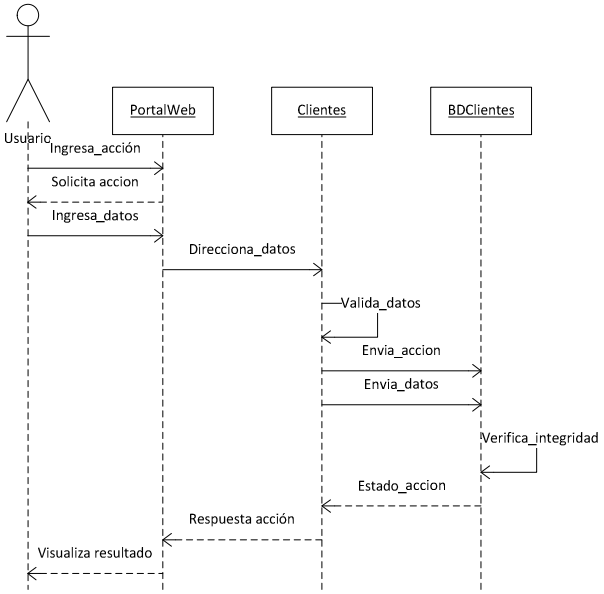

Figura 4.9: Diagrama de clientes.

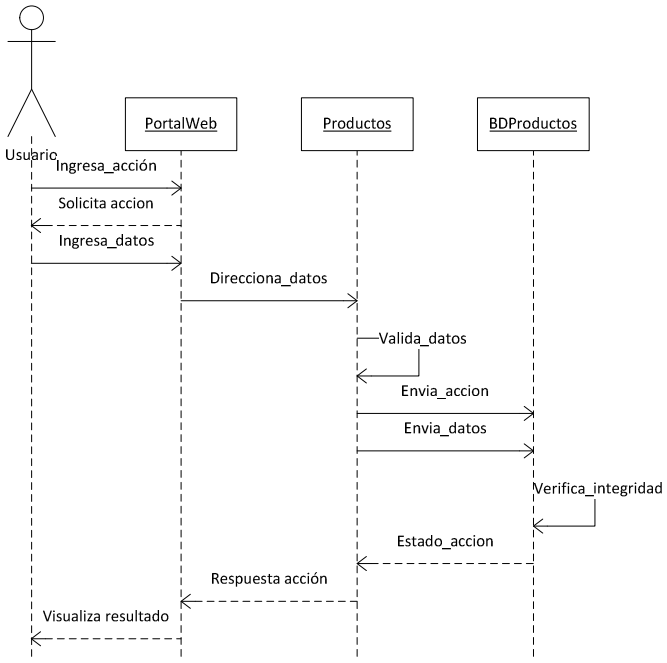

Figura 4.10: Diagrama de productos.

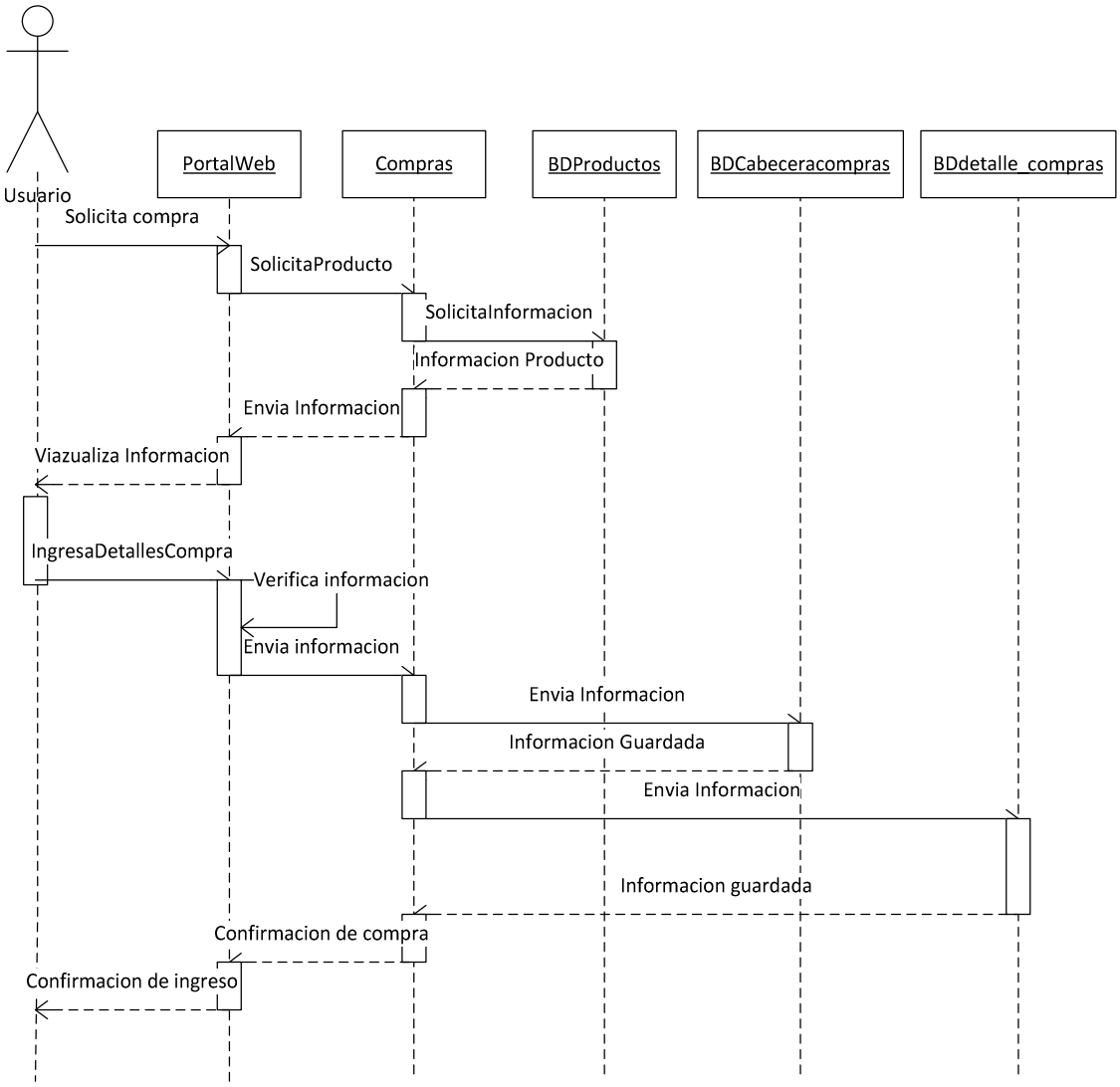

Figura 4.11: Diagrama de compras.

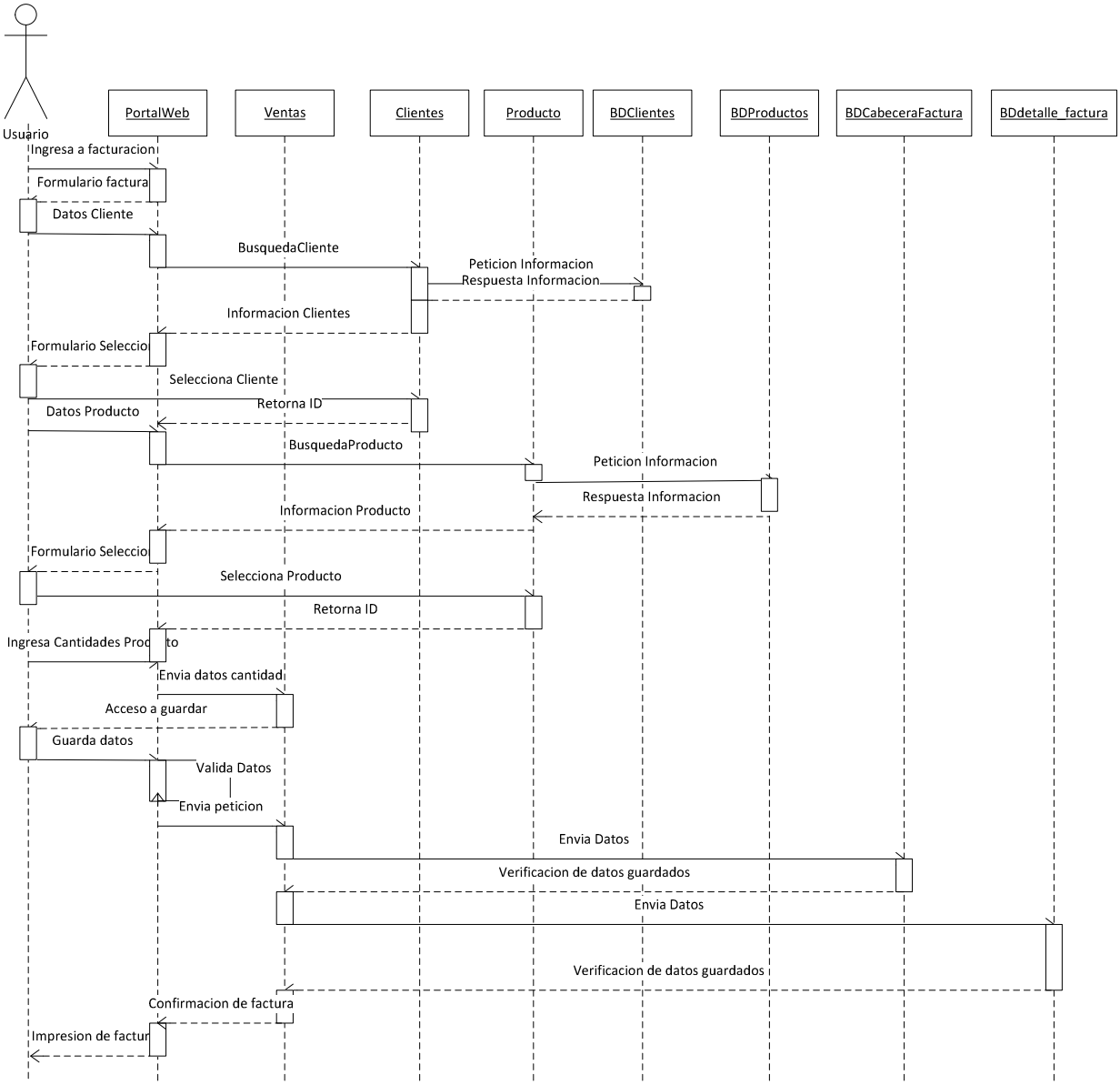

Figura 4.12: Diagrama de ventas.

4.3 Diseño del sistema

El diseño del sistema es otra de las actividades cruciales para el éxito del proyecto, en esta etapa se centra el diseño de aspectos como los datos que manejará el sistema, la arquitectura del mismo, los procesos que se implementarán y la interfaz que será definida para el mismo. Tener un diseño de sistema solido ayudará a la implementación del mismo.

4.3.1 Diseño de la base de datos

Uno de los puntos centrales de cualquier aplicación en el manejo de los datos, por lo que la actividad de diseño de la base de datos es un aspecto de gran importancia. Para el caso de la facturación se define el siguiente diseño de datos tomando los aspectos mencionados antes en el análisis como fundamento para su construcción.

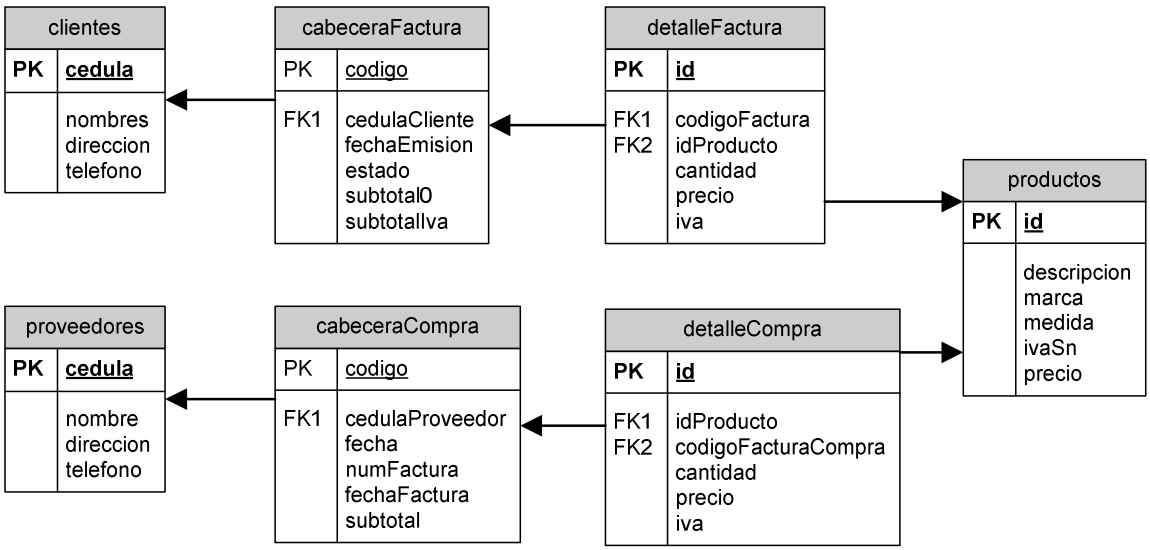

Figura 4.13: Diagrama Entidad Relación.

4.3.2 Diseño de la interfaz

La ingeniería web se caracteriza por poner gran énfasis en los temas de interfaz de usuario, ya que la manera de comunicación entre el usuario y el sistema es el eje central de este tema. Es por esto que una interfaz web debe ser fácil de usar, aprender, navegar y funcional; todas estas características son dependientes de la habilidad del diseñador web y de la herramienta usada para crear la navegación, lo que en el caso estudiado es el framework. Dentro del diseño de la interfaz muchas veces se incluye también el diseño estético, lo cual lleva a la creación de una plantilla para la aplicación web, la plantilla a continuación es el resultado de este proceso de diseño.

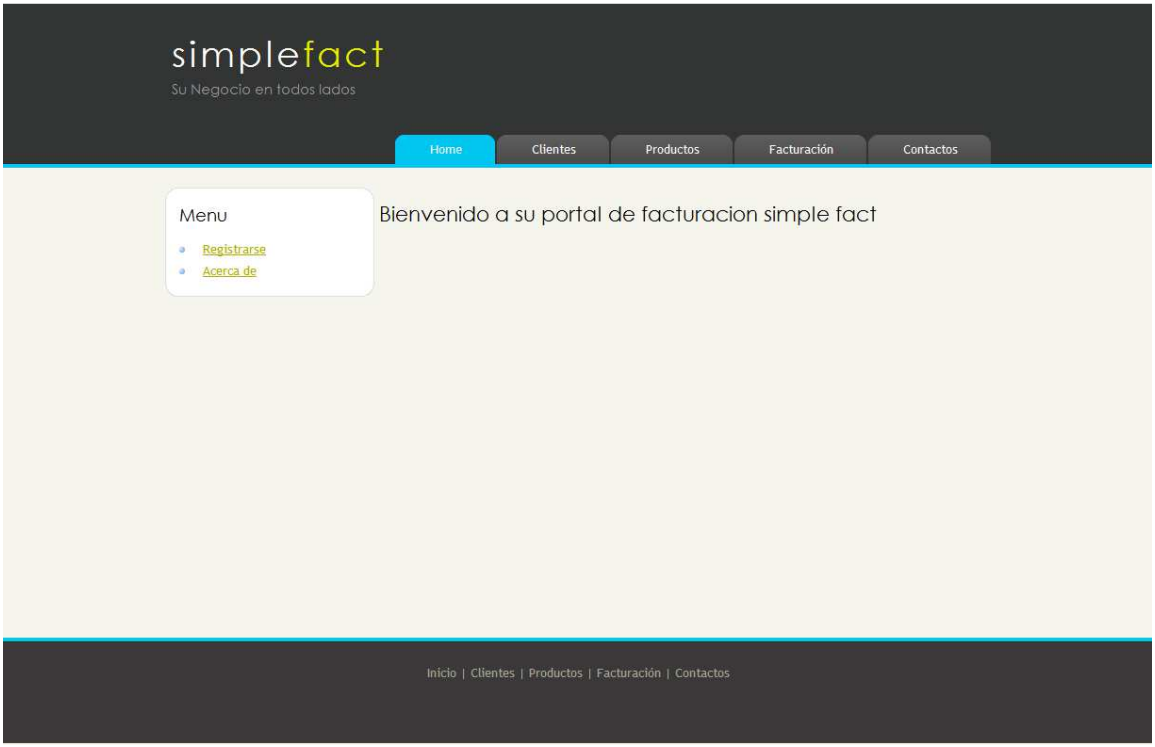

Figura 4.14: Plantilla de Diseño de Interfaz.

4.3.3 Diseño de contenido

Un diseño de contenido es una manera de representar los objetos y la relaciones que existen en una aplicación web, el detalle de este conjunto de estructuras ayuda al ingeniero web a definir la idea de lo que se desea tenga la página web que se realizará, lo que posteriormente facilitará el desarrollo. Para el ingeniero informático centrado en la web el diseño de contenido provee una estructura de los objetos que insertará en una página web, conforme las necesidades del usuario y los requerimientos de la página.

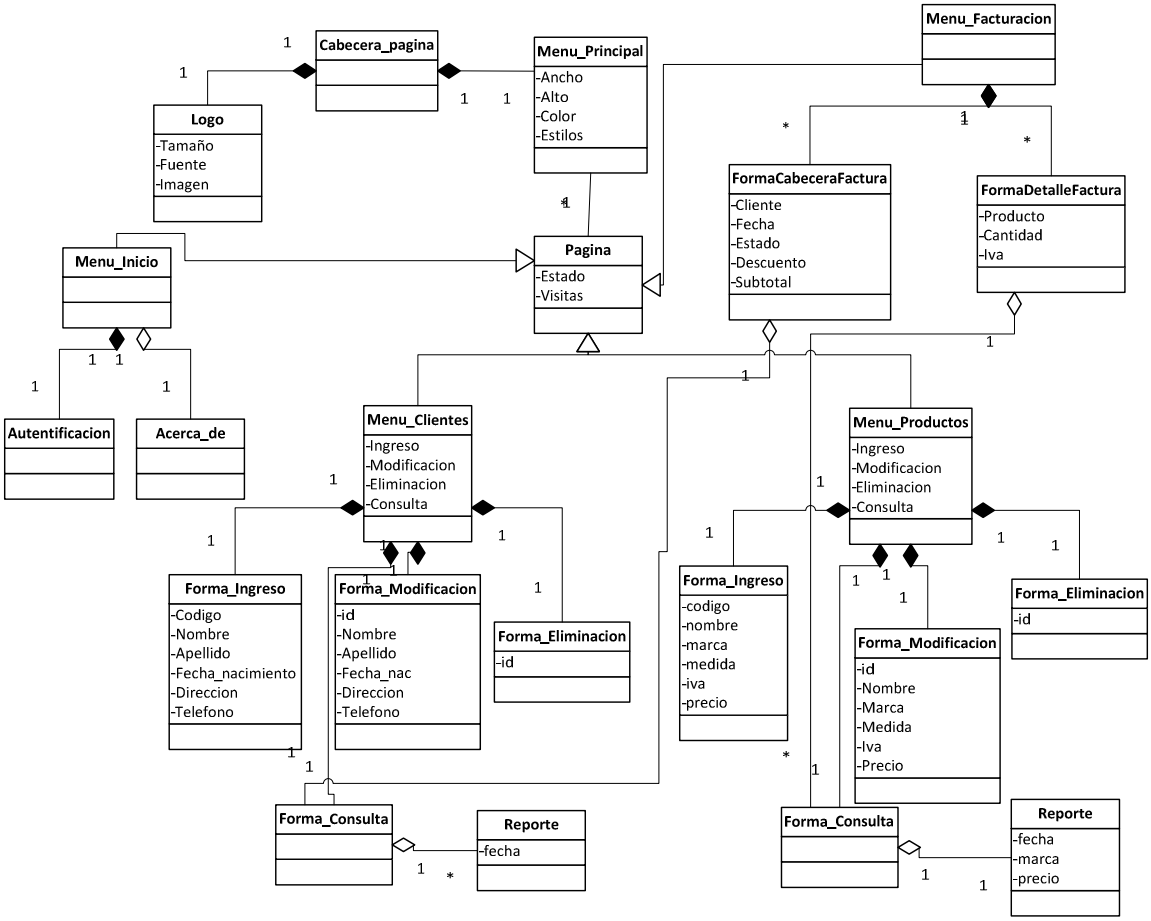

Figura 4.15: Diagrama UML de Contenido.

4.3.4 Arquitectura del sistema

Un framework web java utiliza una arquitectura modelo-vista-controlador como se indicó en el punto 2.1, con esto el framework web logra desacoplar la interfaz del usuario o vista, del modelo del negocio y por ultimo del controlador. Este desacoplamiento se observa mejor en el siguiente gráfico:

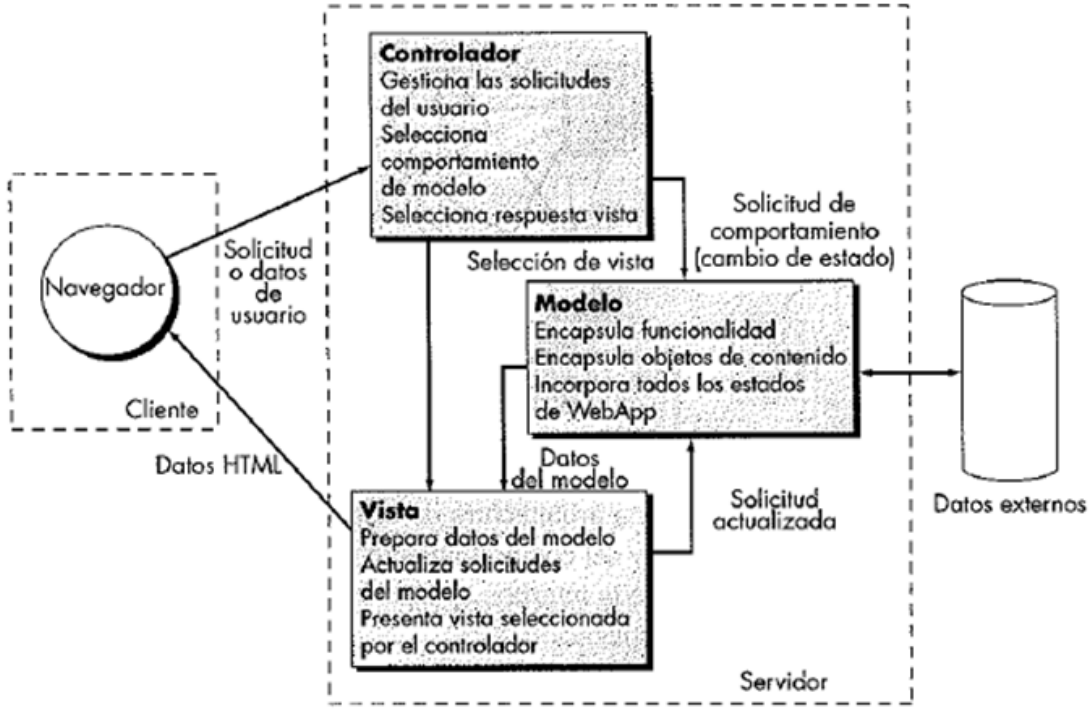

Figura 4.16: Arquitectura MVC para el desarrollo de aplicaciones. (16)

4.4 Codificación de los aplicativos con cada uno de los framework

En la ingeniería de software y en el enfoque de desarrollo en cascada, la codificación del software suele ser el proceso más laborioso, ya que en este punto es donde se materializará el análisis y diseño previamente realizado. La codificación del software es el acto de plasmar en líneas de código una solución a un problema determinado, bajo los lineamientos del análisis y diseño.

La codificación del software o también llamada implementación requiere el conocimiento del lenguaje de programación en el que se realizará la aplicación, la presente obra como ya se ha mencionado en los capítulos anteriores se centra en el lenguaje java, específicamente en los frameworks web del mismo.

Dos de los aspectos más importantes y poderosos de las aplicaciones web java existentes son la aplicación de un framework dedicados al mapeo de objetos relacionales ORM y los mecanismos de reportería; el primero encargado del acceso a datos y el último encargado de mostrar la información al usuario, es por esto que se considera que antes de la codificación se señalen las herramientas que serán utilizadas para estos aspectos mencionados. Como ORM Hibernate es la herramienta ideal y como herramienta de reportería JasperReports

4.4.1 Hibernate

Herramienta de mapeo objeto relacional (ORM) usado en aplicaciones desarrolladas en java. El objetivo de esta herramienta de persistencias es realizar el mapeado de una base de datos relacional en objetos, los cuales pueden ser usados por la aplicación. Se busca solucionar las diferencias entre los distintos esquemas para así tener un solo modelo orientado a objetos, con el cual se optimicen los procesos de desarrollo. Hibernate proporciona un dialecto para la comunicación con la mayoría de bases de datos relacionales existentes como Postgres, MySql, Oracle, DB2, etc.

Hibernate proporciona un conjunto de instrucciones para realizar las tareas pertinentes en una base de datos; es decir, la manipulación de datos y la consulta de los mismos. La consulta de datos es realizada no con SQL típico como es la costumbre, el framework de persistencias proporciona su propio lenguaje de consultas llamado HQL (Hibernate Query Languaje), que de igual manera es orientado a objetos; es así, que se facilita de gran manera la fabricación de consultas. (17)

4.4.2 JasperReports

JasperReports es una herramienta open-source realizada en java para el desarrollo de informes o reportes, está compuesto por una herramienta llamada iReport para la realización diseños y prototipos de reportes y por un conjunto de librerías para visualización de las mismas en un entorno java. La herramienta ha alcanzado una gran popularidad en el desarrollo de aplicaciones java; debido a que, ofrece una gran facilidad para el desarrollo de reportes a través de plantillas y por la variedad de formatos en el que el reporte puede ser exportado. Los formatos admisibles al desplegar un reporte de este tipo son los siguientes: CVS, XML, TXT, HTML, RTF y Jasper viewer; conforme se liberan nuevas versiones de la herramienta los desarrolladores de la misma ofrecen incluso más formatos. Un ejemplo de un reporte creado en esta herramienta se observa a continuación:

Figura 4.17: Ejemplo de Reporte desarrollado con Jasper Reports.

La simplicidad y facilidades que ofrece la herramienta son muchas, ya que permite una creación sencilla y rápida de reportes complejos, dejando a la opción del desarrollador la creación de algunos grupos dentro del reporte; así como, la creación de reportes tabulares, agrupados como ya se mencionó y matriciales. La siguiente es una pantalla que refleja la herramienta, específicamente en el asistente de creación de reportes (18):

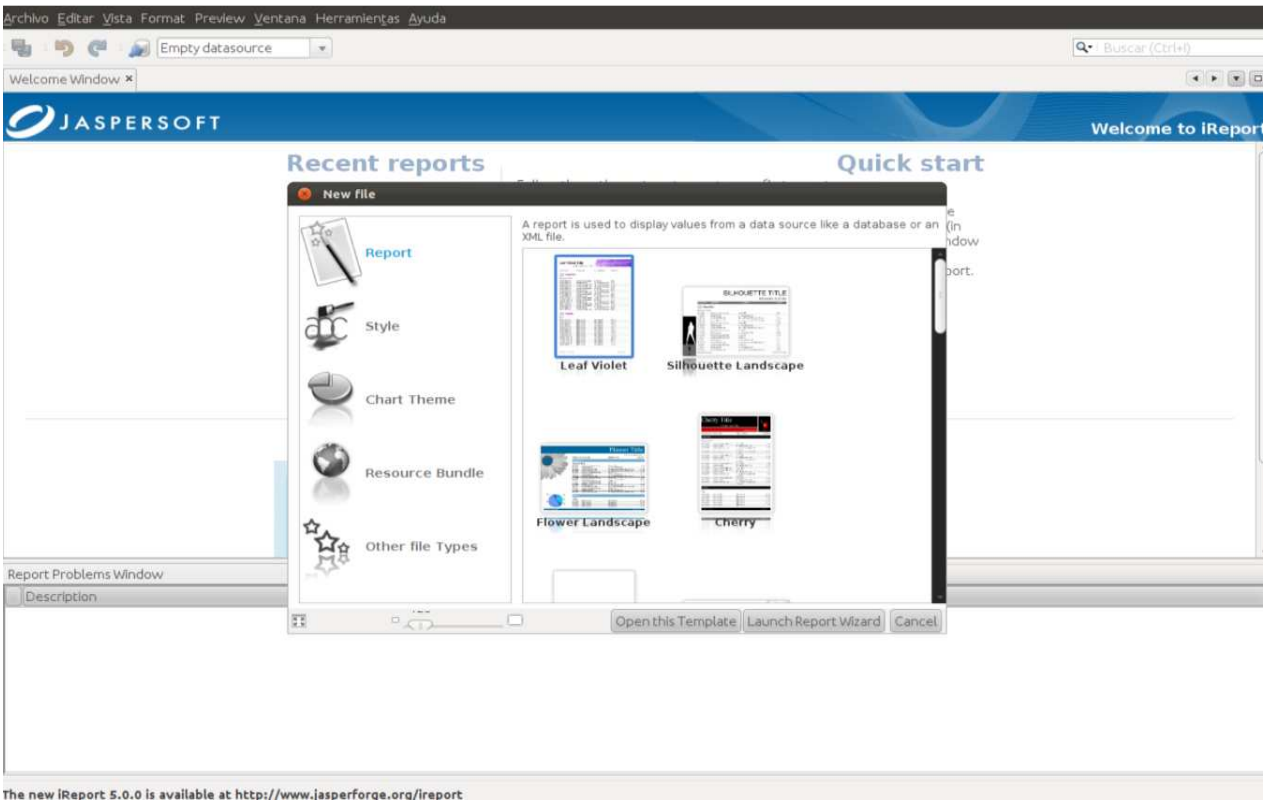

Figura 4.18: Herramienta iReport para desarrollo de Reportes.

4.5 Implementación con Spring Framework

Spring es un framework que por su gran cantidad de módulos provee una gran funcionalidad al mismo, el objetivo de los desarrolladores de este framework fue el de facilitar la programación dando prioridad a las anotaciones sobre archivos XML de configuración. Pese a esto la configuración sigue siendo bastante tediosa y consume una gran cantidad de tiempo, ya que los errores de configuración son difíciles de detectar y no hay una guía clara dentro de los manuales del framework de cómo implementar de la mejor manera dichas configuraciones. Por otra parte la programación de las vistas, lógica del negocio, y controlador es bastante comprensible ya que se guía por principios básicos de repositorios, servicios y controladores, lo cual facilita el desarrollo del paradigma MVC.

4.5.1 Aspectos Técnicos

Para el desarrollo de una aplicación web en Spring framework se deben tener en cuenta los siguientes requerimientos técnicos:

- Java 1.5 o superior para las versiones 3.0.x y 3.1.x para versiones inferiores se puede utilizar java 1.4.
- Maven 2.6 o superior, de preferencia maven 3.
- Eclipse 3.6 o superior.
- Dependencias de Spring Framework descargadas por maven.
- Servidor de aplicaciones web.

Algunos de los Servidores de aplicaciones soportados por Richfaces son:

- Apache Tomcat 6 o superior
- WAS 5.1 o superior
- Appserver
- WebLogic.
- Jboss 4.2.x – 5
- Websphere 7.0 y superiores

4.5.2 Conexión hacia la base de datos

Dentro del aspecto de las conexiones a la base de datos Spring framework ha puesto especial empeño, ya que dentro de sus módulos integra algunas soluciones para abarcar prácticamente las alternativas más usadas dentro de este punto. Integra Spring jdbc, Spring ORM y

Spring Data en sus versiones más recientes, con esto se puede realizar conexiones sencillas con jdbc o integrar frameworks para el manejo de persistencias y mapeado de las mismas. En la parte práctica la conexión a la base de datos se realiza mediante archivos de configuración, teniendo en cuenta que se deben tener actualizadas las dependencias de maven de Spring y de los frameworks de persistencias. La configuración se la realiza en el archivo del application-context.xml, de la siguiente manera:

Configuración de JPA y Hibernate

```
<bean id="dataSource"
class="org.Springframework.jdbc.datasource.DriverManagerDataSource">
<property name="driverClassName" value="${jdbc.driverClassName}"/>
<property name="url" value="${jdbc.url}"/>
<property name="username"  value="${jdbc.username}"/>
<property name="password" value="${jdbc.password}"/>
</bean>

<bean id="entityManagerFactory"
class="org.Springframework.orm.jpa.LocalContainerEntityManagerFactoryBean"
p:dataSource-ref="dataSource"
p:jpaVendorAdapter-ref="jpaAdapter">
<property name="loadTimeWeaver">

<bean id="jpaAdapter"
class="org.Springframework.orm.jpa.vendor.HibernateJpaVendorAdapter"
p:database="${jpa.database}"
p:showSql="${jpa.showSql}"/>
<bean id="transactionManager" class="org.Springframework.orm.jpa.JpaTransactionManager"
p:entityManagerFactory-ref="entityManagerFactory"/>
```

4.5.3 Capa de la Vista

La vista en Spring es uno de los aspectos más básicos ya que en su mayoría usa JSP y JSTL, aunque integra algunos componentes propios a la vista para el mejor manejo y comunicación con el controlador. El desarrollo de la parte funcional de la vista es la parte más sencilla ya que no integra ninguna tecnología que facilite integración Ajax o componentes más vistosos, etc.

4.5.4 Capa del Modelo

La capa de la lógica del negocio en Spring considera las buenas prácticas de programación en java como principio fundamental, ya que el desarrollo se lo realiza mediante interfaces e implementaciones de las mismas, otro aspecto que cabe destacar es que Spring también maneja Beans, pero lo hace de una forma un tanto distinta, ya que estos no son declarados por configuración, sino que son declarados por anotaciones, en las que se especifican el tipo de Bean,

para el modelo un Bean puede ser de tipo @Repository, @Service, y @Component que engloba la funcionalidad de los dos primeros, es decir en Spring se puede separar la capa de datos de la de servicios, con las anotaciones correspondientes.

Capa del modelo de negocio en Spring

Repositorio

```
@Repository(value = "clienteDao")
public class JPAClienteDao implements ClienteDao {

private EntityManager em = null;

@PersistenceContext
public void setEntityManager(EntityManager em) {
        this.em = em;
}
@SuppressWarnings("unchecked")
public List<Cliente> getClienteList() {
        return em.createQuery("select p from Cliente p order by
p.nombre").getResultList();
        }

}
```

Servicio

```
@Service
public class ImpClienteService implements ClienteService {

        private static final long serialVersionUID = 1L;

        //private List<Product> products;
   @Autowired
   private ClienteDao clienteDao;

   public void setClienteDao(ClienteDao clienteDao) {
      this.clienteDao = clienteDao;
   }

   @Transactional
        public List<Cliente> getClientes() {
      //return products;
                return clienteDao.getClienteList();
        }
```

Dentro de las funcionalidades de Spring, por el cual es bastante conocido es el uso de la inyección de dependencias y la programación orientada a aspectos, la inyección de dependencias se la puede realizar mediante las anotaciones @Inject y @Autowire como se observa en el código de

servicio, mientras que la programación orientada a aspectos mediante la etiqueta @Transactional. Si bien la programación orientada a aspectos va mucho más haya que lo mencionado, es importante recalcar que el uso que se le dé a la misma depende mucho del contexto en el que se quiera desarrollarla.

4.5.5 Capa del Controlador

El controlador en Spring es también un Bean como lo son también el servicio y repositorio, este tipo de Bean es muy específico ya que desde este se gestionan todos los request de una página y es aquí donde se forman los mapeos de la navegación de la aplicación, para definir un controlador se lo hace mediante la anotación @Controller. Toda petición a la página web sea esta un vínculo hacia otra página o una acción que genere la página web debe estar mapeada en el controlador. Desde este se pueden hacer tareas tan sencillas como redireccionar una página, hasta pasar objetos de una página a otra. Desde el punto de vista de la programación, es el controlador en donde se efectúa el mayor costo en tiempo y dificultad de aprendizaje.

Controlador Spring

```
@Controller
public class FacturaVentaController {

        @Autowired
    private ProductoService productoService;
        @Autowired
    private ClienteService clienteService;
        @Autowired
    private FacturaVentaService facturaService;
        @Autowired
        FacturaVentaService facService;
        private cabeceraFacturaEstados cabfac= new cabeceraFacturaEstados();

@RequestMapping(value = "/facturaVenta", method = RequestMethod.GET)
public ModelAndView get(Model model) {
        detalleFacturaEstadosForm productoForm = new detalleFacturaEstadosForm();
        productoForm.setProductosAtributos(productosFacturas);
        model.addAttribute("productoForm", productoForm);
        model.addAttribute("cliente", clienteAttribute);
        model.addAttribute("cabFactura", cabfac);
        return new ModelAndView("facturaVenta");
}

@RequestMapping(value = "/agregarProducto", method = RequestMethod.POST)
public ModelAndView save(@ModelAttribute("productoForm") detalleFacturaEstadosForm
contactForm) {
        List<detalleFacturaEstados> contacts = contactForm.getProductosAtributos();
        if(null != contacts && contacts.size() > 0) {
```

```
                FacturaVentaController.productosFacturas = contacts;
        }
        return new ModelAndView("redirect:/seleccionarProductos");
}
```

El controlador utiliza anotaciones como @RequetMapping para el manejo de las peticiones de direccionamiento de páginas web y para la ejecución de métodos de las mismas. Otra característica importante es el manejo de formularios y esto se lo realiza mediante atributos agregados para llenar formularios con información de consultas por ejemplo y atributos de entrada para obtener la información que un usuario ha ingresado en un formulario.

4.5.6 Implementación de Seguridad

Spring tiene dentro de su funcionalidad un módulo dedicado exclusivamente a la seguridad, incluyendo aspectos como la autentificación de usuarios, el manejo de permisos mediante roles y a la encriptación de contraseñas en la base de datos. Dentro del contexto de una aplicación de negocio esta herramienta es realmente útil, ya que la protección de los datos es un aspecto de gran preocupación dentro de cualquier organización. La seguridad de Spring principalmente se implementa con un archivo de configuración y un Bean de Servicio que implementa una clase propia de Spring.

Seguridad Spring

Archivo de Configuración

```
<global-method-security secured-annotations="enabled" />
<http auto-config="true"  use-expressions="true">
        <intercept-url pattern="/login" access="permitAll"/>
        <intercept-url pattern="/denied" access="hasRole('ROLE_USER')"/>
        <intercept-url pattern="/" access="hasRole('ROLE_USER')"/>
        <intercept-url pattern="/Springapp/mantenimientoClientes.htm"
access="hasRole('ROLE_ADMIN')"/>
        <form-login login-page="/login"
                authentication-failure-url="/login/failure"
                default-target-url="/home"/>
        <access-denied-handler error-page="/denied"/>
        <logout invalidate-session="true"
                logout-success-url="/logout/success"
                logout-url="/logout"/>
</http>

<authentication-manager>
      <authentication-provider user-service-ref="customUserDetailsService">
      </authentication-provider>
</authentication-manager>
```

Bean de Servicio

```
public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException
{
        try {
                UsuarioSpring domainUser = userRepository.findByUsuario(username);
                98lients enabled = true;
                98lients accountNonExpired = true;
                98lients credentialsNonExpired = true;
                98lients accountNonLocked = true;
                return new User(
                                domainUser.getUsuario(),
                                domainUser.getContrasenia(),
                                enabled,
                                accountNonExpired,
                                credentialsNonExpired,
                                accountNonLocked,
        getAuthorities(domainUser.getRolesSprings().get(0).getRol()));

                } catch (Exception e) {
                        throw new RuntimeException€;
                }
        }
public List<String> getRoles(Integer role) {
        List<String> roles = new ArrayList<String>();
        if (role.intValue() == 1) {
                roles.add("ROLE_USER");
                roles.add("ROLE_ADMIN");
        } else if (role.intValue() == 2) {
                roles.add("ROLE_USER");
        }
        return roles;
}
```

Los roles son manejados de manera jerárquica y son asignados dentro de la base de datos un rol por usuario, es decir se restringe a una relación uno a uno, un usuario puede tener un solo rol. Dentro del programa se realiza la conversión, la base de datos básica para la autentificación de usuarios en Spring consta de dos tablas, una de usuarios y otra de roles.

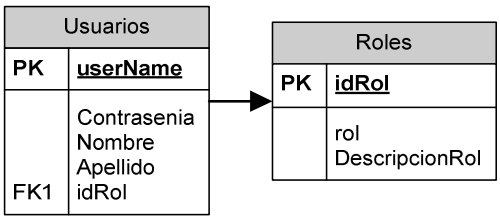

Figura 4.19: Diagrama Entidad Relación para autentificación en Spring.

4.5.7 Generación de Reportes

Spring no implementa reportes propios, por lo cual se integra con Jasper Reports que es una de las soluciones java más conocidas del mercado para este fin. La integración depende de un archivo de configuración en donde se declaran cada uno de los reportes como un Bean que extienden de una clase Spring, aunque la integración con Jasper Reports es una gran ventaja, tiene una restricción que para muchos desarrolladores puede convertirse en un problema, ya que la información generada para el reporte debe enviarse desde el programa como una lista de registros, lo cual no es muy conveniente cuando se tienen consultas muy complejas.

Reportes en Spring

Bean de reporte

```
<bean id="pdfReportProductos"
class="org.Springframework.web.servlet.view.jasperreports.JasperReportsPdfView"
        p:url="/resources/reports/report_articulos.jrxml"
        p:reportDataKey="datasource" />

</beans>
```

Llamada a reporte desde controlador

```
@RequestMapping("/reporteArticulos")
public ModelAndView generatePdfReportProdcutos(ModelAndView modelAndView){;
      Map<String,Object> parameterMap = new HashMap<String,Object>();
      List<Producto> list=procService.getProductos();
      JRDataSource JrdataSource = new JRBeanCollectionDataSource(list);
      parameterMap.put("datasource", JrdataSource);
      return new ModelAndView("pdfReportProductos", parameterMap);
        }
```

4.5.8 Aplicación resultante Spring

Luego de la codificación de todos los aspectos nombrados en los puntos anteriores se tiene como resultado la siguiente aplicación:

Figura 4.20: Página de registro desarrollada en Spring.

Figura 4.21: Vista de Página Inicial desarrollada en Spring.

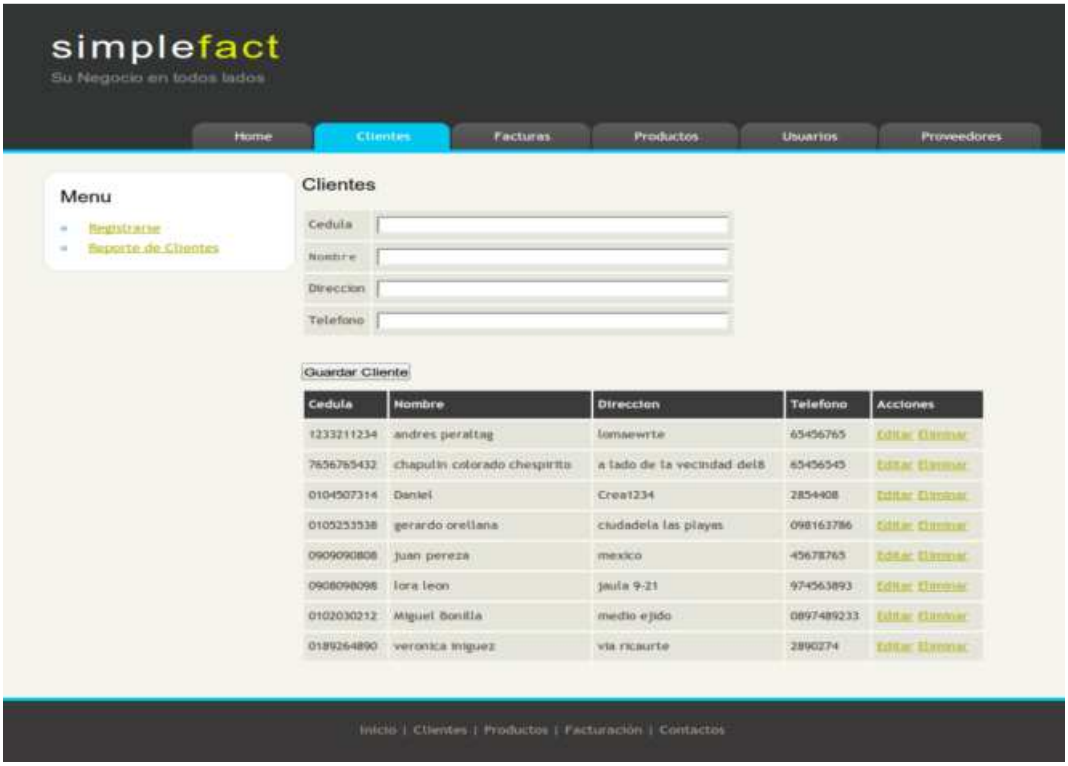

Figura 4.22: Vista de página de mantenimiento desarrollada en Spring.

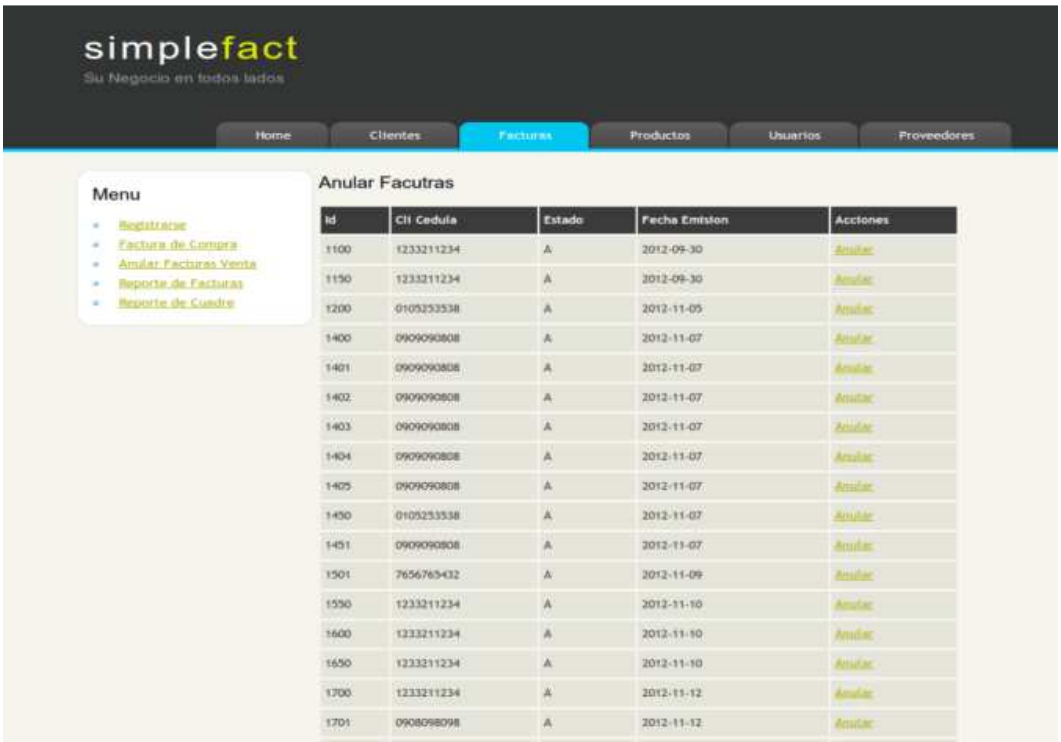

Figura 4.23: Vista de página de anulación facturas desarrollada con Spring.

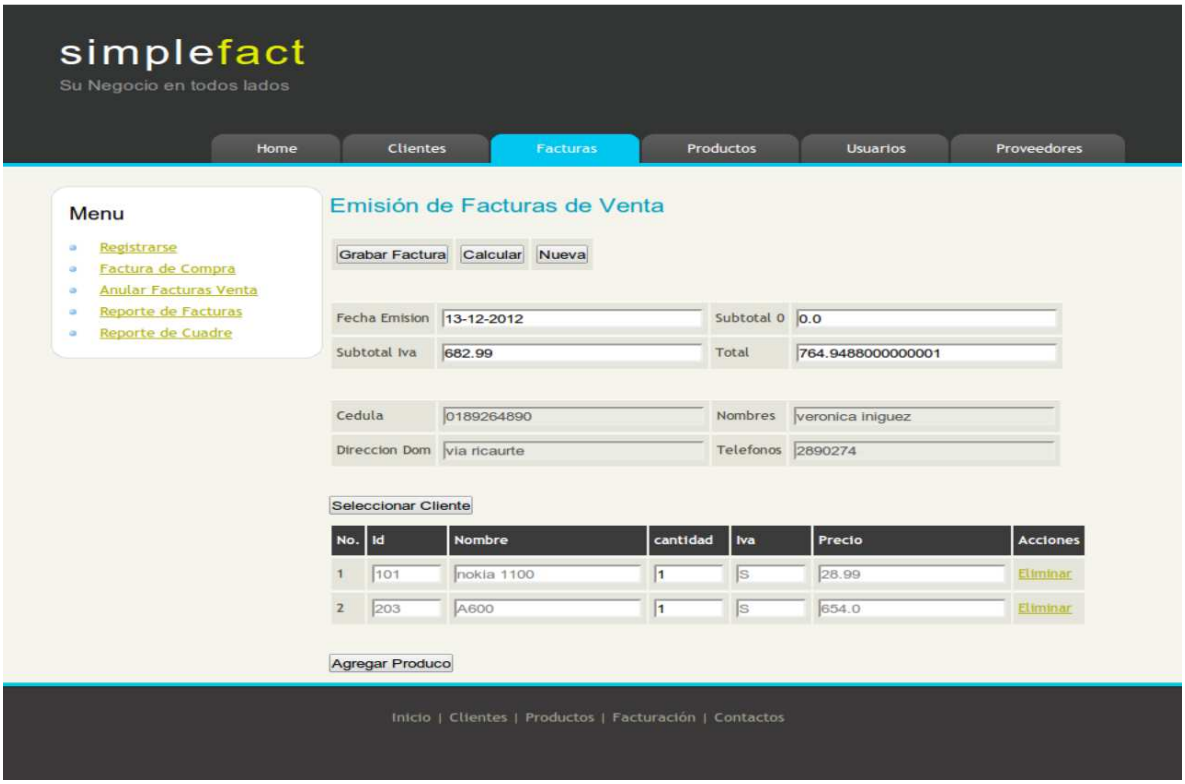

Figura 4.24: Vista de factura desarrollada con Spring Framework.

4.6 Implementación con Richfaces Framework

Richfaces es un framework que provee un modelo sencillo de implementación, ya que se basa en un estándar java que es JSF, gracias a esto la implementación de la lógica del negocio y el controlador resulta bastante sencillo. La mayor cantidad de posibilidades para el desarrollador se encuentra en la capa de presentación, es decir en la vista, ya que es aquí en donde el framework entrega una gran cantidad de componentes listos para usar, los cuales tienen una riqueza visual y funcional envidiable. Pese a la gran riqueza funcional del framework ya nombrada, este tiene grandes falencias en áreas como la seguridad o la reportería, lo cual hace que la codificación en solitario de este framework no sea una buena opción a la hora de desarrollar una aplicación empresarial.

4.6.1 Aspectos Técnicos

El desarrollador debe tener en cuenta los siguientes requisitos Técnicos para utilizar Richfaces Framework:

- Java 1.5 o superior.
- Java Server Faces JSF versión 1.2 para Richfaces 3.3 o inferior, versión 2 para Richfaces 4.
- Eclipse 3.3 o superiores.
- Navegador web en el lado del cliente.
- Richfaces Framework.
- Servidor de aplicaciones web.

Algunos de los Servidores de aplicaciones soportados por Richfaces son:

- Apache Tomcat 5.5 y superior
- WebLogic 9.1 y superior
- Jetty 6.1.x
- Sun Application Server 9 y superior
- Glassfish V2, V3
- Jboss 4.2.x – 5 para Richfaces 3.x y superiores para Richfaces 4.
- Websphere 7.0 y superiores

4.6.2 Conexión hacia la base de datos

Richfaces es un framework enfocado a la capa de presentación, debido a este enfoque la conexión hacia la base de datos puede realizarse de varias maneras, indistintamente del framework. Las posibilidades de conexión son bastante amplias, ya que se la puede realizar simplemente con conectores jdbc o con frameworks especializados para el manejo de persistencias. Para el ejemplo se utiliza JPA (Java Persistence Api) como manejador de persistencias y Hibernate como proveedor de las mismas. La integración con estos frameworks es automática, y se puede realizar de dos maneras, en la primera solamente se necesitan importar las librerías necesarias y crear los archivos de configuración propios de los frameworks de persistencias, y la otra forma más ampliamente usada de integrar estas tecnologías es mediante la integración de Richfaces a un proyecto ya creado de JSF, JPA y Hibernate, ya que esto implica solo importar las librerías necesarias de Richfaces.

4.6.3 Capa de la Vista

La implementación de la vista es uno de los aspectos más trabajosos en el desarrollo de un sistema con este framework, ya que es aquí donde se explota la funcionalidad del mismo. Dentro de la vista se puede usar gran variedad de componentes con funcionalidad Ajax. También es posible agregar aspectos presentes en JSF como validaciones, facelets y componentes del mismo. Al usar

este framework se debe tener presente el encabezado de cada una de las vistas, ya que cada una debe tener las etiquetas para cada uno de los componentes disponibles.

Layout principal

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:f="http://java.sun.com/jsf/core"
        xmlns:h="http://java.sun.com/jsf/html"
    xmlns:ui="http://java.sun.com/jsf/facelets"
        xmlns:a="http://Richfaces.org/a4j"
        xmlns:rich="http://Richfaces.org/rich">
```

Vistas

```
<f:view xmlns="http://www.w3.org/1999/xhtml"
    xmlns:f="http://java.sun.com/jsf/core"
        xmlns:h="http://java.sun.com/jsf/html"
    xmlns:ui="http://java.sun.com/jsf/facelets"
        xmlns:a="http://Richfaces.org/a4j"
        xmlns:rich="http://Richfaces.org/rich">
```

En la tabla se muestra como cada uno de los tipos de componentes que se pueden utilizar tienen asignados etiquetas para referirse a los mismos, para el caso del ejemplo los componentes Richfaces, tienen asignado la etiqueta rich. Las etiquetas son escogidas libremente por el desarrollador.

4.6.3.1 Facelets

Es uno de los aspectos más poderosos dentro de la perspectiva de la vista, ya que el desarrollador puede crear una sola plantilla en donde especificará el contenido que puede ser completado por otra vista, es decir el código HTML y CSS de la página web está presente solo en una página web la cual heredara sus características a las que el desarrollador desee. La forma de trabajar con facelets es creando un componente facelets en el layout o página principal.

Componente facelets página principal

```
<ui:insert name="content"/>
```

En las vistas simplemente se hace referencia al facelet declarado en la página principal.

Facelets en las vistas

```
<f:view xmlns="http://www.w3.org/1999/xhtml"
    xmlns:f="http://java.sun.com/jsf/core"
```

```
        xmlns:h=”http://java.sun.com/jsf/html”
    xmlns:ui=”http://java.sun.com/jsf/facelets”
        xmlns:a=”http://Richfaces.org/a4j”
        xmlns:rich=”http://Richfaces.org/rich”>
        <ui:composition template=”/layout_principal_home.xhtml”>
            <ui:define name=”content”>
             Contenido del layout
              </ui:define>
       </ui:composition>
</f:view>
```

4.6.3.2 Componentes Richfaces

Richfaces posee una gran cantidad de componentes que implementan skins y capacidades Ajax, en la declaración de la vista se puede observar existen dos componentes del framework, los cuales se identifican con dos etiquetas distintas, una propia para los componentes Ajax y otra para componentes Richfaces. Un aspecto a tomar en cuenta desde la interfaz es que los componentes están referenciados a las clases del modelo, es decir a los beans, por lo que los componentes que ejecutan acciones o retornan resultados deben estar asociados a un método o atributo de una clase java.

Componente llamando a atributo
<h:inputText id=”nombre” value=”#{clientesBean.actual.nombre}> </h:inputText>
Componente llamando a método
<a:commandButton value=”Ingresar” action=”#{clientesBean.ingresar}” reRender=”listado “ execute=”@form”/>

4.6.3.3 Skins

Una característica muy llamativa del framework es el soporte de skins, con esto el desarrollador tiene algunas posibilidades para cambiar el aspecto de sus páginas de forma rápida, los skins afectan manejados por Richfaces afectan a los controles desarrollados en JSF, Ajax y rich. Como ya se mencionó la implementación es bastante sencilla, ya que solo hay que añadir algunas líneas de código al archivo de configuración web.xml.

Skins Richfaces
<context-param> <param-name>org.Richfaces.skin</param-name> <param-value>deepMarine</param-value> </context-param>

4.6.4 Capa del Modelo

Richfaces al ser un heredero de JSF, la codificación del modelo se la realiza de la misma forma que su predecesor, manejando la lógica del negocio mediante Beans, actuando estos como encapsuladores del código para evitar manejar una mayor cantidad de clases más simples. Dentro de los Beans se plasma la lógica del negocio, y es aquí donde los procesos requeridos por el sistema se ven plasmados.

BeanClientes

```
@ManagedBean(name = "clientesBean")
@SessionScoped
public class ClientesBean implements Serializable{
	private Cliente actual = new Cliente();

	public ClientesBean(){

	}
	public Cliente getActual() {
		if (actual == null)
			actual = new Cliente();
		return actual;
	}

	public void setActual(Cliente actual) {
		this.actual = actual;
	}
```

Para que una clase se convierta en un bean, existen dos formas de realizarlo y eso va a depender del tipo de framework de persistencias que se esté utilizando, por ejemplo en el caso de un ORM como Hibernate, los beans deben estar declarados en un archivo de configuración XML siendo este el caso más común ya que aplica también con conectores simples como JDBC, pero en el caso actual al utilizar JPA con Hibernate, para declarar un bean simplemente se hace anotaciones sobre el mismo. En el ejemplo existen dos anotaciones @ManagedBean(name = "clientesBean") y @SessionScoped, la primera anotación declara el bean y la segunda determina el alcance del mismo.

4.6.5 Capa del Controlador

El controlador está definido por los archivos de configuración web.xml y el archivo de faces_config, en estos dos archivos se puede configurar todas las opciones tanto de navegación como configuraciones propias del framework, una de las configuraciones más típicas es la declaración de los beans en el archivo faces_config.

Declaración de Beans

```
<managed-bean>
    <description>UsernName Bean</description>
    <managed-bean-name>productoBean</managed-bean-name>
    <managed-bean-class>ProductoBean</managed-bean-class>
    <managed-bean-scope>request</managed-bean-scope>
    <managed-property>
      <property-name>name</property-name>
      <property-class>java.lang.String</property-class>
      <value/>
    </managed-property>
  </managed-bean>
```

El flujo de navegación de la aplicación también puede estar definido en este archivo, práctica que no se considera muy aconsejable si el desarrollador no necesita enviar parámetros o variables entre páginas, ya que las reglas de navegación deben registrarse para cada una de las posibilidades y esto puede resultar muy costoso por el tiempo y además bastante confuso. En el archivo web.xml se definen algunas de las reglas para la ejecución de la aplicación, tales como los sufijos por defecto de las páginas web, el tiempo de actualización, el método de guardado, el estado del proyecto, y muchas otras particularidades que pueden ser configurables.

Archivo web.xml

```
<display-name>simpleFact</display-name>
  <context-param>
 <param-name>javax.faces.DEFAULT_SUFFIX</param-name>
 <param-value>.xhtml</param-value>
 </context-param>
 <context-param>
  <param-name>javax.faces.PROJECT_STAGE</param-name>
  <param-value>Development</param-value>
 </context-param>
 <context-param>
  <param-name>javax.faces.FACELETS_REFRESH_PERIOD</param-name>
  <param-value>2</param-value>
 </context-param>
<context-param>
  <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
  <param-value>server</param-value>
 </context-param>

<servlet>
  <servlet-name>Faces Servlet</servlet-name>
  <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
 </servlet>
```

```
 <servlet-mapping>
  <servlet-name>Faces Servlet</servlet-name>
  <url-pattern>*.jsf</url-pattern>
 </servlet-mapping>
</web-app>
```

Como se puede observar los archivos de configuración en una aplicación que utiliza Richfaces son bastante sencillos, ya que JSF se encarga de realizar los mapeos de las páginas de forma transparente para el desarrollador, una de las ventajas principales de una aplicación de este tipo es que los redireccionamientos a las páginas web son automáticos y en si el controlador esta implementado de forma implícita, aunque por la programación parezca que en la vista se accede directamente a los métodos y atributos de las clases Bean, esto no es cierto ya que el controlador siempre está gestionando estos accesos, respetando el paradigma MVC.

4.6.6 Implementación de Seguridad

Uno de los puntos débiles de los frameworks orientados a la capa de presentación es la seguridad, ya que estos no proporcionan ningún mecanismo para realizar tareas vitales como lo son la autentificación de usuarios y la asignación de permisos mediante roles. Richfaces no es la excepción a este caso, ya que si se desea integrar seguridad, esta deberá ser programada mediante código Java Script o mediante la integración de algún otro framework.

4.6.7 Generación de Reportes

Richfaces no trae consigo herramientas para la generación de reportes y tampoco para la integración de otras herramientas al mismo, es por esto que la reportería se convierte un problema considerable al usar este framework, ya que si bien se puede usar reportes de Jasper Reports y correrlos desde la aplicación, acarrea el problema que se corta el hilo de ejecución dejando el código que esta después de la llamada al reporte inutilizable.

Reportes en Richfaces

```
public void reporteClientesHttp() throws JRException, SQLException, IOException {
try {
        Connection connection;
        ObtenerConexion con=new ObtenerConexion();
        connection=con.getConnection();
        FacesContext context = FacesContext.getCurrentInstance();
        HttpServletResponse response = (HttpServletResponse)
context.getExternalContext().getResponse();
        InputStream reportStream =
context.getExternalContext().getResourceAsStream("/reports/report_clientes_todos.jasper");
        ServletOutputStream servletOutputStream =response.getOutputStream();
        JasperRunManager.runReportToPdfStream(reportStream,servletOutputStream, new
```

```
HashMap(), connection);
        connection.close();
        response.setContentType("application/pdf");
        servletOutputStream.flush();
        servletOutputStream.close();
} catch (Exception e) {
        System.out.println("Mensaje de Error1:"+e.getMessage());
        e.printStackTrace();
}
        }
```

4.6.8 Aplicación resultante Richfaces

Luego de la codificación de todos los aspectos nombrados en los puntos anteriores con Richfaces Framework se tiene como resultado la siguiente aplicación:

Figura 4.25: Página de registro desarrollada en RichFaces.

Figura 4.26: Página de inicio desarrollada en RichFaces.

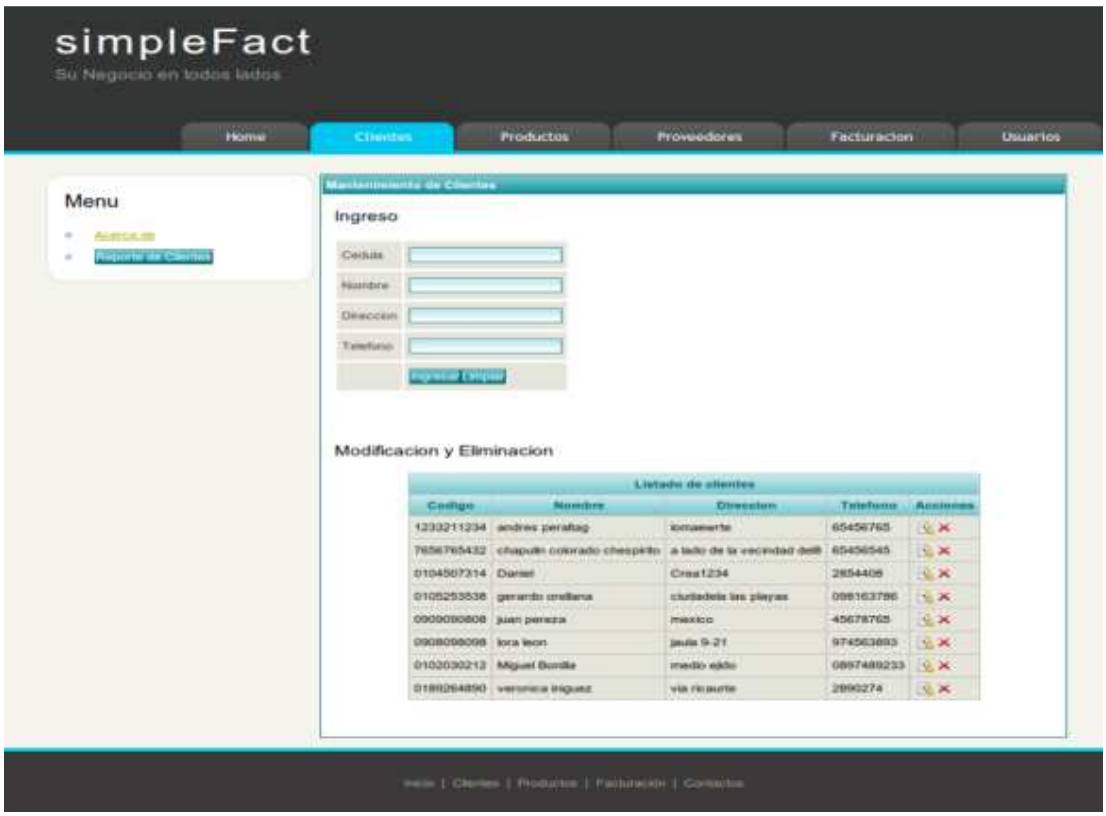

Figura 4.27: Página de mantenimiento desarrollada en RichFaces.

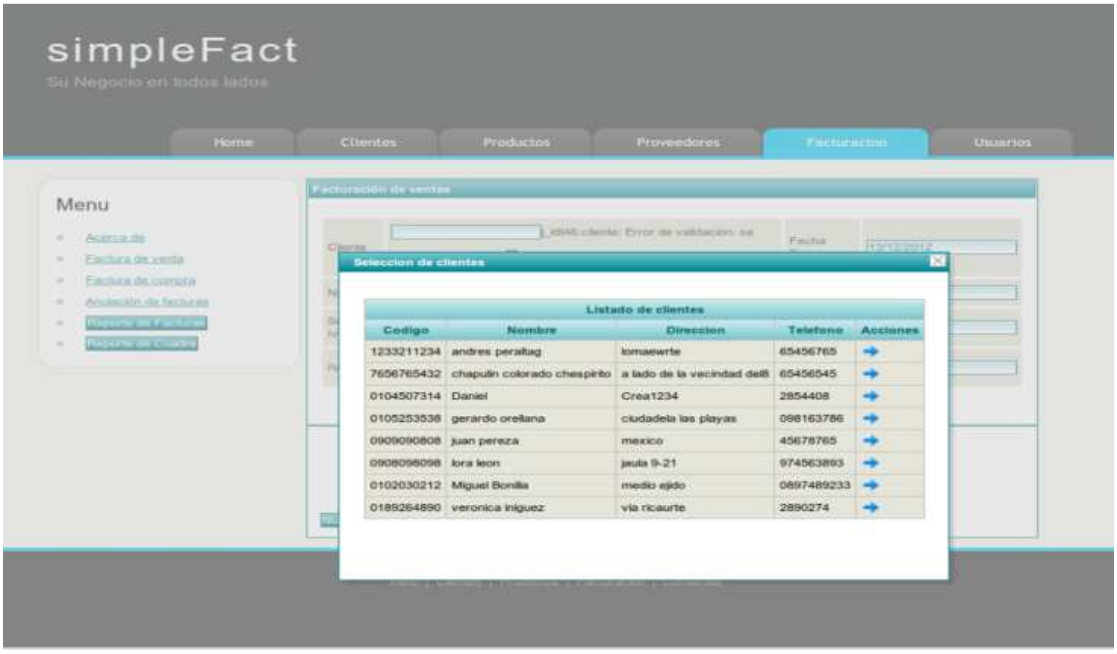

Figura 4.28: Modal de selección desarrollada en RichFaces.

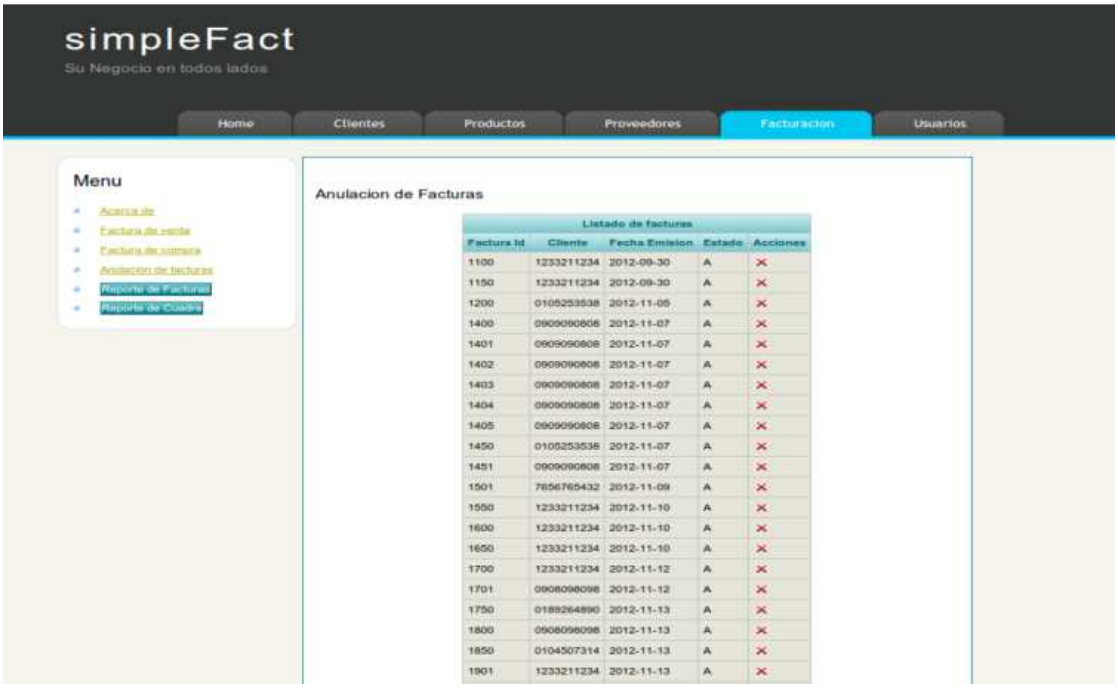

Figura 4.29: Página de anulación de facturas desarrollada en RichFaces.

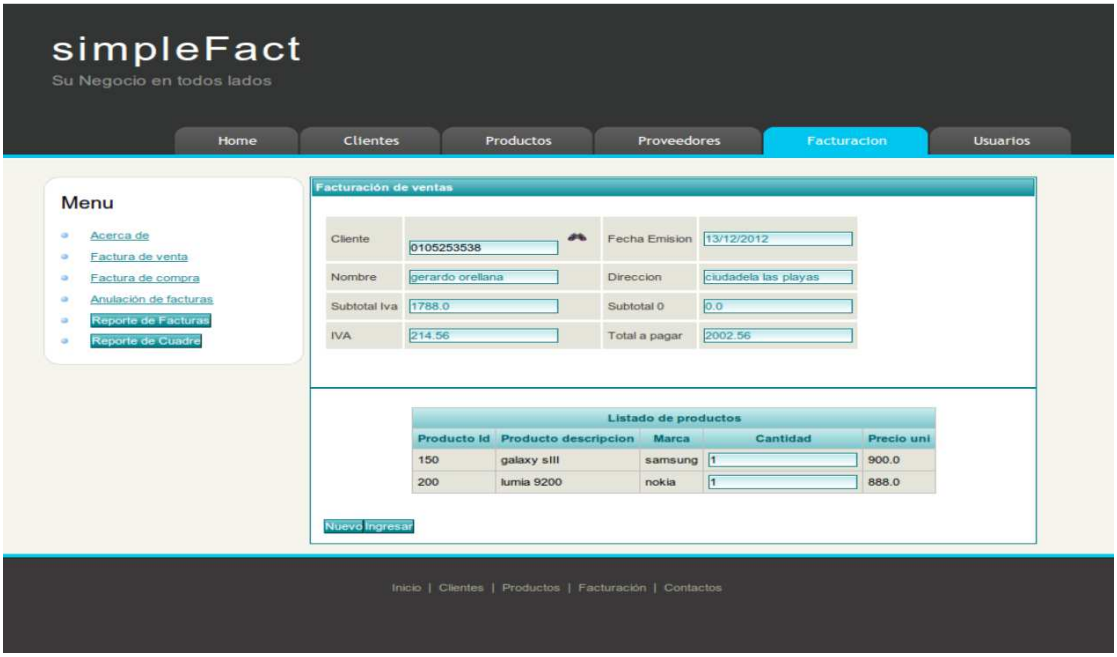

Figura 4.30: Página de Factura desarrollada en RichFaces.

4.7 Implementación con Seam framework

Seam es un framework muy completo que ofrece herramientas y funcionalidades para cada uno de los componentes del paradigma MVC, ya que en la vista tiene una integración automática con Richfaces y Icefaces, en el modelo con JPA y Hibernate y por último en el controlador ofrece gran cantidad de posibilidades para el manejo de seguridad y navegación entre páginas. Las aplicaciones web con Seam pueden ser construidas rápidamente ya que proporciona herramientas para la generación automática de proyectos tomando como fundamento una base de datos, es decir el framework ofrece la posibilidad de generar los mantenimientos de cada una de las tablas de forma automática, generando por si solo las vistas y las persistencias con las tecnologías antes mencionadas.

4.7.1 Aspectos Técnicos

El desarrollador debe tener en cuenta los siguientes requisitos Técnicos para utilizar Jboss Seam Framework:

- Java 1.5 o superior.
- Jboss Seam 2.3.
- Jboss Tools 3.3.
- Apache Ant.
- JDBC de la base de datos que se desea usar

- Eclipse 3.7 o superiores.
- Navegador web en el lado del cliente.
- Servidor de aplicaciones web.

Algunos de los Servidores de aplicaciones soportados por Richfaces son:

- Apache Tomcat 6 o superiores
- Glassfish V3
- Jboss AS 5 o superiores

4.7.2 Conexión hacia la base de datos

La conexión hacia la base de datos en Seam está ya implícita al crear un proyecto, durante la creación del proyecto se especifica toda la información necesaria de la base de datos para que Seam simplemente realice todas las configuraciones necesarias. El framework crea una conexión con Hibernate como proveedor ORM y JPA como manejador de las persistencias, enfoque que es muy útil para proporcionar seguridad a los datos y proporcionar facilidades al momento de la programación.

Configuraciones de creación (Base de Datos)
Ruta driver JDBC: /opt/jboss-as-7.1.1.Final/standalone/lib/postgresql-8.3-603.jdbc3.jar Dialecto Hibernate: org.hibernate.dialect.PostgreSQLDialect JDBC Class Driver: org.postgresql.Driver JDBC DataSource Class: org.postgresql.jdbc3.Jdbc3ConnectionPool JDBC URL: jdbc:postgresql://localhost:5432/nombre_base Usuario: postgres Clave: clave_postgres Esquema BD: public Catalogo BD: nombre_base Utilizar tablas desde BD: y

Luego de introducir las configuraciones es Seam el que se encarga de crear los archivos de configuración para las persistencias y las clases persistentes de cada una de las tablas de la base de datos.

4.7.3 Capa de la Vista

La vista de Seam funciona de la misma manera que JSF y Richfaces, todos los componentes de estos dos grandes frameworks funcionan en Seam de forma natural, agregando funcionalidad Seam también proporciona soporte para Icefaces, otro framework de la capa de presentación que presenta funcionalidades Ajax. Todos estos componentes e incluyendo los controles propios que adiciona Seam, hacen de este framework impecable en la capa de presentación.

Los controles de Seam para esta capa son utilizados para aumentar la funcionalidad del framework proporcionando soporte específico para algunos de los conceptos propios como el de las conversaciones entre páginas o el manejo de contextos y también para realizar conversiones y validaciones aún más específicas que los controles de Richfaces y JSF.

Controles Seam más comunes

s:link → Incluye posibilidad de enviar parámetros

```
<s:link view="/CuentasUsuarioList.xhtml"
        value="Cuentas de usuario"
        id="CuentasUsuarioId"
        includePageParams="false"
        propagation="none"/>
```

s:validate y s:decorate → Valida los campos y decora un campo luego de su validación

```
<s:validateAll>
  <s:decorate id="fecha" template="layout/display.xhtml">
          <ui:define name="label">Fecha</ui:define>
          <h:outputText value="#{cabeceraCompraHome.instance.fecha}">
             <f:convertDateTime type="date" dateStyle="short"/>
          </h:outputText>
       </s:decorate>
</s:validateAll>
```

s:button → Permite realizar evaluaciones en tiempo real para distintos redireccionamientos o valores

```
<s:button view="/#{empty cabeceraCompraFrom ? 'CabeceraCompraList' :
cabeceraCompraFrom}.xhtml"
  id="done"
  value="#{cabeceraCompraHome.instance.proveedores ¡= null ¿ 'Change' : 'Select'}/>
```

s:div → Permite que el contenido del div sea ocultado o mostrado dinámicamente, según una condición.

```
<s:div styleClass="actionButtons" rendered="#{empty from}">
    <s:button view="/FacturaVenta.xhtml"
        id="create"
        propagation="none"
        value="Nueva Factura">
    </s:button>
</s:div>
```

S:conversationId → Visualiza la conversación en la que encuentra la aplicación

```
<s:conversationId />
```

En la creación del proyecto hay aspectos como la generación de la vista en Seam que tiene una funcionalidad bastante interesante para el desarrollador ya que incluye conceptos bastante avanzados de visualización como lo son los facelets y los menús de Richfaces, estos elementos

ayudan a organizar la vista desde la programación y la parte visual respectivamente, lo cual mejora la experiencia del programador y del usuario.

Aspectos organizativos incluidos en Seam

Facelets en Seam

```
<ui:include src=”menu.xhtml”>
        <ui:param name=”projectName” value=”factSeam”/>
     </ui:include>
     <div class=”body”>
       <h:messages id=”messages” globalOnly=”true” styleClass=”message”
         errorClass=”errormsg” infoClass=”infomsg” warnClass=”warnmsg”
         rendered=”#{showGlobalMessages != ‘false’}”/>
       <ui:insert name=”body”/>
     </div>
```

Menús en Seam

```
<rich:toolbar
   xmlns=”http://www.w3.org/1999/xhtml”
   xmlns:ui=”http://java.sun.com/jsf/facelets”
   xmlns:h=”http://java.sun.com/jsf/html”
   xmlns:f=”http://java.sun.com/jsf/core”
   xmlns:s=”http://jboss.org/schema/Seam/taglib”
   xmlns:rich=”http://Richfaces.org/rich”>
   <rich:toolbarGroup>
      <h:outputText value=”FacturaciÃ³n Seam :”/>
      <s:link id=”menuHomeId” view=”/home.xhtml” value=”Inicio” propagation=”none”/>
   </rich:toolbarGroup>
   <rich:dropDownMenu showDelay=”250” hideDelay=”0” submitMode=”none”>
      <f:facet name=”label”>Modulos Usuario</f:facet>
   <rich:menuItem>
        <s:link view=”/FacturaCompra.xhtml”
                value=”Factura Compra”
                id=”FacturaCompraId”
                        includePageParams=”false”
                propagation=”none”/>
    </rich:menuItem>
    <!—Otros elementos del menu→
   </rich:dropDownMenu>
   </rich:toolbarGroup>
</rich:toolbar>
```

4.7.4 Capa del Modelo

Seam provee un modelo en donde se manejan varias formas de realizar la programación según se necesite un mantenimiento simple o estructuras más complejas donde se maneje una lógica del negocio más elaborada. Estos dos modelos son definidos por anotaciones, para aclarar al framework que son un bean y también para definir el alcance del mismo.

En el primer caso el modelo se puede codificar desde una implementación de la clase EntityHome para el mantenimiento de la tabla y de la clase EntityQuery, ambas provistas por Seam. Para ambos casos se implementan algunos de los métodos propios de estas clases para obtener los resultados de forma sencilla.

Manejo de formas simples en Seam

Mantenimiento de una tabla simple

```
@Name("proveedoresHome")
public class ProveedoresHome extends EntityHome<Proveedores> {

@Override
protected Proveedores createInstance() {
        Proveedores proveedores = new Proveedores();
        return proveedores;
}
public void load() {
        if (isIdDefined()) {
                wire();
        }
}
public void wire() {
        getInstance();
}
public 116lients isWired() {
        return true;
}
public Proveedores getDefinedInstance() {
        return isIdDefined() ? getInstance() : null;
}
}
```

Querys en Seam

```
@Name("proveedoresList")
public class ProveedoresList extends EntityQuery<Proveedores> {

private static final String EJBQL = "select proveedores from Proveedores proveedores";
private Proveedores proveedores = new Proveedores();
public ProveedoresList() {
        setEjbql(EJBQL);
        setRestrictionExpressionStrings(Arrays.asList(RESTRICTIONS));
        setMaxResults(25);
}
public Proveedores getProveedores() {
        return proveedores; } }
```

Al implementar diseños de negocio más complejos es necesaria una forma más flexible para la determinación de la lógica, es por esto que Seam también permite desarrollar las aplicaciones

desde la inyección del EntityManager, el cual es el encargado de realizar cualquier acción en la base de datos, ya sea de introducción de datos o de consultas de los mismos.

Manejo de formas complejas Seam

```
@Name("cabecerasFacturaBean")
@Scope(ScopeType.CONVERSATION)
public class CabecerasFacturaBean{
   @In(create=true)
   EntityManager entityManager;

public List<CabeceraFactura> getRegistros() {

   String sqlTiposUsuario = "FROM CabeceraFactura r WHERE r.estado='A'";
   Query queryTiposUsuario = entityManager.createQuery(sqlTiposUsuario);
   List<CabeceraFactura> registros = queryTiposUsuario.getResultList();
   return registros;
}
public String ingresar() throws JRException, SQLException, IOException {
        calculaSubtotal();
        actual = new CabeceraFactura();
        actual.setClientes(cliente);
        actual.setEstado("A");
        actual.setSubtotal0(subtotal_0);
        actual.setSubtotalIva(subtotal_iva);
        actual.setFechaEmision(new Date());
        entityManager.persist(actual);
        entityManager.flush();
        ingrsarDetalle();
        return "/home.xhtml";
}

public void anularFactura(){
        seleccionado.setEstado("X");
        entityManager.persist(seleccionado);
        entityManager.flush();
}

}
```

4.7.5 Capa del Controlador

El controlador en este framework está implícito como lo es en el caso de Richfaces, es decir este se encargará de gestionar los mapeos y las acciones que se ejecuten desde la vista, pero agrega además un concepto muy novedoso, ya que deja el paradigma de tener todo sobre la navegación y el paso de parámetros en un solo archivo de configuración, es así que Seam implementa un controlador en cada una de las páginas que lo necesite, es decir si desde la página clientes.xhtml se desea pasar un parámetros a otra página, entonces se ve la necesidad de crear un archivo llamado

clientes.page.xml en donde se codifica el paso de parámetros y otras opciones como el manejo de conversaciones y reglas de navegación según condiciones específicas.

Controladores por páginas

```
<?xml version="1.0" encoding="UTF-8"?>
<page xmlns="http://jboss.org/schema/Seam/pages"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://jboss.org/schema/Seam/pages
http://jboss.org/schema/Seam/pages-2.3.xsd"
      no-conversation-view-id="/ClientesList.xhtml"
      login-required="true">

   <begin-conversation join="true" flush-mode="MANUAL"/>

   <action execute="#{clientesHome.wire}"/>

   <param name="clientesFrom"/>
   <param name="clientesCedula" value="#{clientesHome.clientesCedula}"/>


   <navigation from-action="#{clientesHome.persist}">
      <rule if-outcome="persisted">
         <end-conversation/>
         <redirect view-id="/Clientes.xhtml"/>
      </rule>
   </navigation>

   <navigation from-action="#{clientesHome.update}">
      <rule if-outcome="updated">
         <end-conversation/>
         <redirect view-id="/Clientes.xhtml"/>
      </rule>
   </navigation>

   <navigation from-action="#{clientesHome.remove}">
      <rule if-outcome="removed">
         <end-conversation/>
         <redirect view-id="/ClientesList.xhtml"/>
      </rule>
   </navigation>

</page>
```

Algunos aspectos un poco más específicos del controlador pueden definirse también en los archivos de configuración pages.xml y web.xml, en el primero se pueden definir aspectos como redireccionamientos en el caso de errores típicos del sistema, estos direccionamientos funcionaran para todas las páginas, el segundo aspectos típicos del servidor y de los skins del proyecto.

Archivos de configuración del Controlador

Pages.xml

```
<exception class="javax.persistence.EntityNotFoundException">
      <redirect view-id="/error.xhtml">
        <message severity="warn">Registro no encontrado</message>
      </redirect>
  </exception>

  <exception class="javax.persistence.EntityExistsException">
      <redirect view-id="/error.xhtml">
        <message severity="warn">Registro duplicado</message>
      </redirect>
  </exception>

  <exception class="javax.persistence.OptimisticLockException">
      <end-conversation/>
      <redirect view-id="/error.xhtml">
        <119lients severity="warn">Otro usuario está cambiando los mismos datos</119lients>
      </redirect>
  </exception>
```

Web.xml

```
<!—Seam →
  <listener>
    <listener-class>org.jboss.Seam.servlet.SeamListener</listener-class>
  </listener>

  <filter>
    <filter-name>Seam Filter</filter-name>
    <filter-class>org.jboss.Seam.servlet.SeamFilter</filter-class>
  </filter>

  <filter-mapping>
    <filter-name>Seam Filter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
<!—JSF →
  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.Seam</url-pattern>
  </servlet-mapping>
```

4.7.6 Implementación de seguridad

Seam posee un modelo de seguridad muy elaborado basado en anotaciones, toma en cuenta aspectos de autentificación de usuarios, manejo de identidades por roles, autorizaciones a páginas basadas en permisos por roles y administración de permisos. Debido a la gran cantidad de funcionalidades de seguridad que posee el framework tiene un modelo de base de datos propio que se debe implementar para obtener la máxima funcionalidad del mismo.

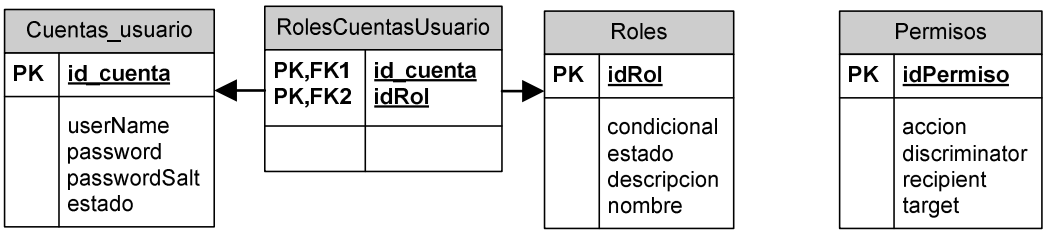

Figura 4.30: Diagrama Entidad Relación para autentificación en Seam.

Para la implementación del modelo de seguridad se implementan anotaciones en cada una de las persistencias de la base de datos, es ahí donde se indica cuáles son los campos de usuario, contraseña, roles y permisos. El campo password y passwordSalt son campos vinculados ya que el framework realiza una autentificación por llave, estando el password encriptado y siendo passwordSalt la llave.

La seguridad se ve también involucrada en los archivos de configuración, ya que dentro del page.xml de cada página se puede especificar que para visualizar se requiere estar registrado y en el archivo pages.xml se puede restringir accesos a páginas de acuerdo a los roles.

Configuraciones de seguridad
Encabezado de archivo de configuración
login-required=*"true"*
Archivo pages.xml
<page view-id=*"/mantUsuarios.Seam"*> <restrict>#{s:hasRole('admin')}</restrict> </page>

4.7.7 Generación de reportes

Seam pese a ser un framework muy completo es su versión estable actual no posee una herramienta de generación de reportes propia, para la siguiente versión está planeada una herramienta que ahora se encuentra en versión beta.

Reportes en Seam

```
@SuppressWarnings("unchecked")
public void generarReporteClientes() throws Exception {
InputStream inputStream = new FileInputStream
("/home/gerito/Desarrollo/121lients/tesis/reportesSeam/report_clientes_todos.jrxml");
List<Clientes> dataBeanList= new ArrayList<Clientes>();
dataBeanList=entityManager.createQuery("Select p from Clientes p").getResultList();
Map parameters = new HashMap();
JRBeanCollectionDataSource beanColDataSource = new
JRBeanCollectionDataSource(dataBeanList);
JasperDesign jasperDesign = JRXmlLoader.load(inputStream);
JasperReport jasperReport = JasperCompileManager.compileReport(jasperDesign);
JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport, parameters,
beanColDataSource);
byte[] pdf = null;
pdf=JasperExportManager.exportReportToPdf(jasperPrint);
FacesContext context = FacesContext.getCurrentInstance();
HttpServletResponse response = (HttpServletResponse)
context.getExternalContext().getResponse();
try {
    OutputStream os = response.getOutputStream();
    response.setContentType("application/pdf"); // fill in
    response.setContentLength(pdf.length);
    response.setHeader("Content-disposition", "attachment; filename=\""+
"reporteClientes.pdf\"");
    os.write(pdf); // fill in bytes
    os.flush();
    os.close();
    context.responseComplete();
} catch (IOException e) {
    e.printStackTrace();
}
}
```

4.6.8 Aplicación resultante Seam

Luego de la codificación de todos los aspectos nombrados en los puntos anteriores con Seam Framework se tiene como resultado la siguiente aplicación:

Figura 4.31: Vista de página de registro desarrollada con Seam.

Vista 4.32: Vista de página de inicio desarrollada con Seam.

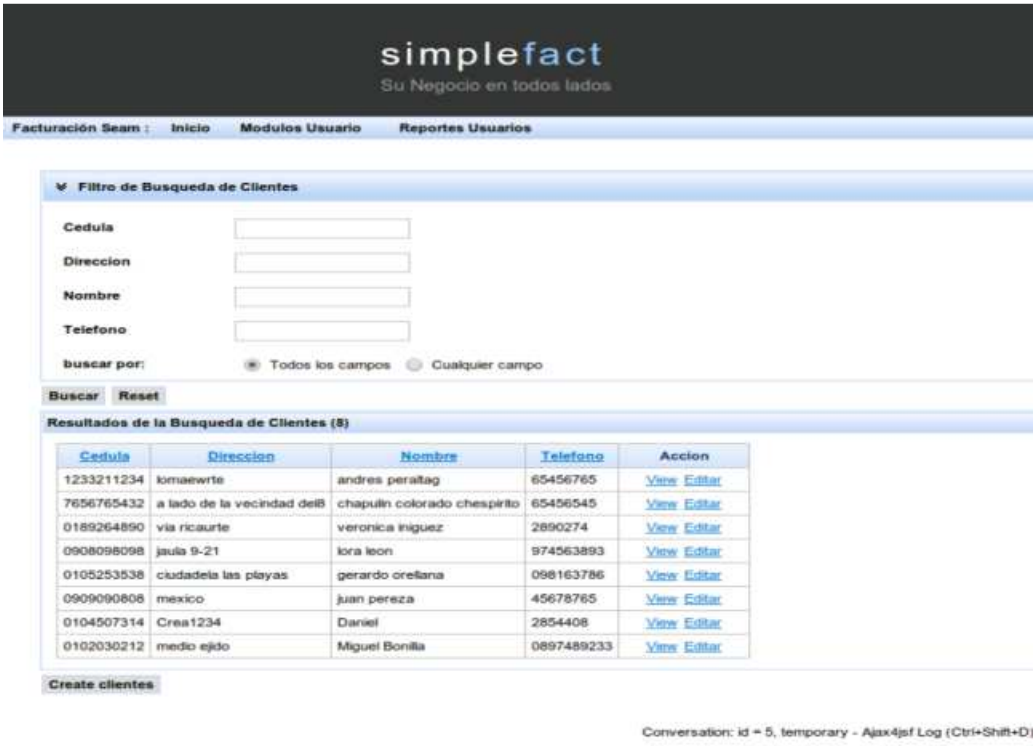

Figura 4.32: Vista de página de búsqueda desarrollada con Seam.

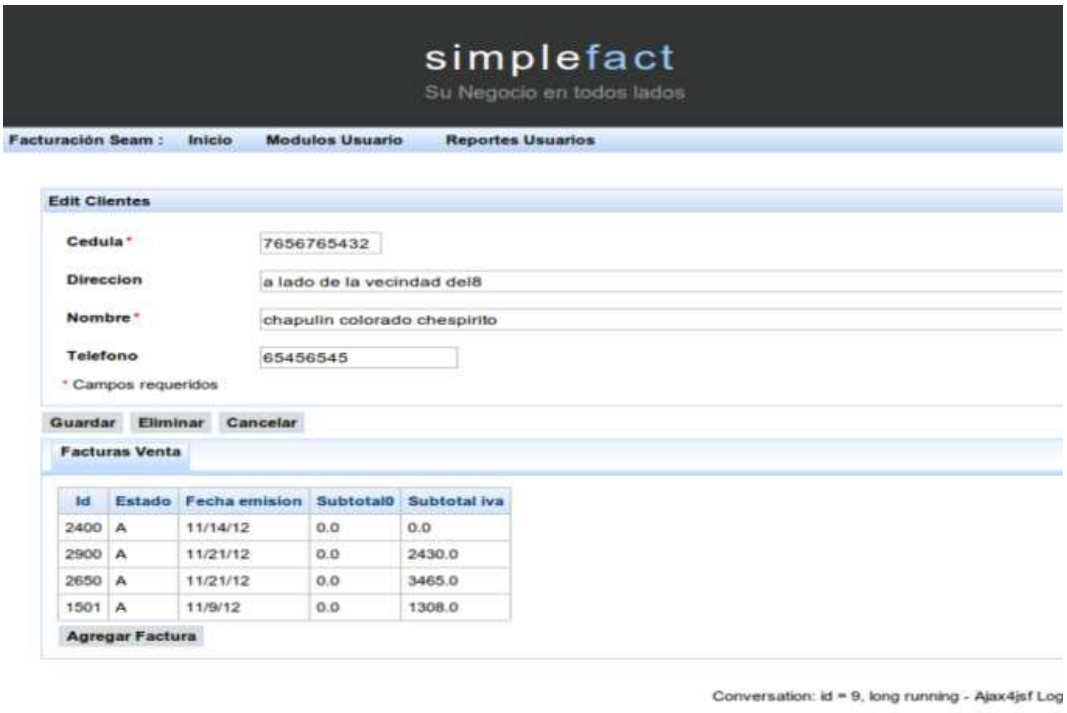

Figura 4.33: Vista de página de Mantenimiento desarrollada con Seam.

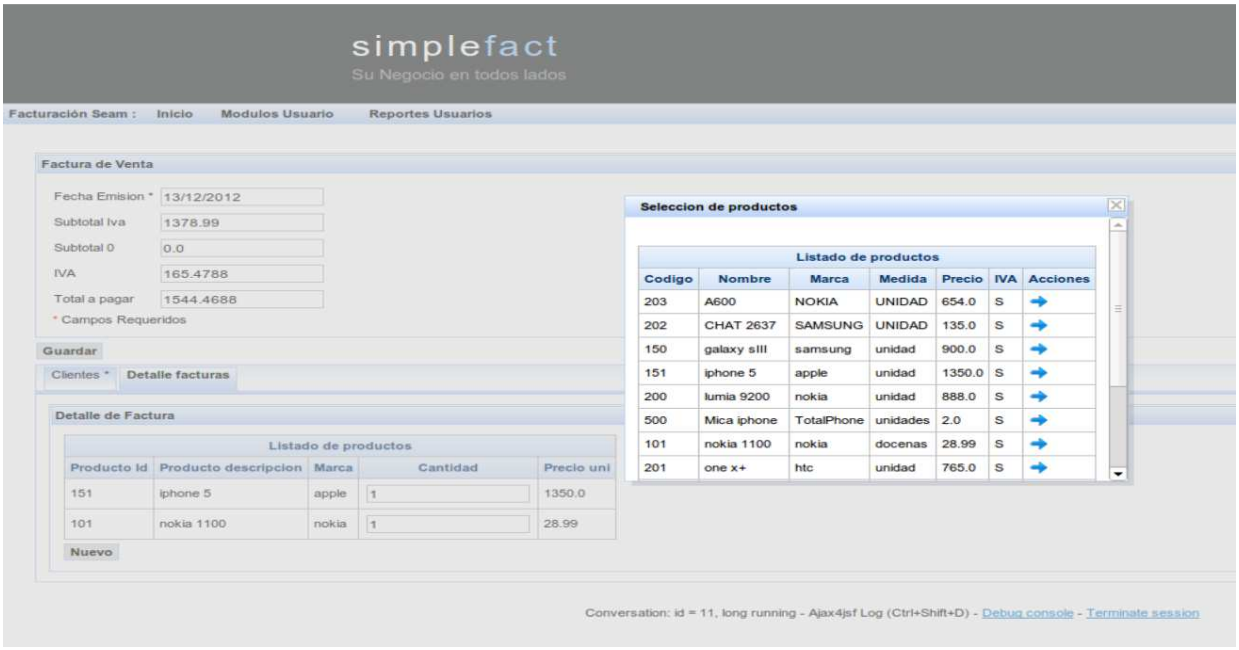

Figura 4.34: Vista de Factura desarrollada con Seam.

El desarrollo de aplicaciones en este framework es bastante dinámico, pero en ocasiones puede resultar confuso por lo que se considera adecuado facilitar un manual básico para el desarrollo de aplicaciones en Seam.

Ver Anexo 3

4.8 Conclusiones

Tener una metodología con la cual construir un software es un aspecto muy importante; ya que, desde este punto se adoptará una estructura para realizar los procesos que involucran la elaboración. La metodología en cascada es un enfoque lineal que proporciona una guía con los pasos básicos para construir un sistema.

El análisis del sistema es una herramienta fundamental para determinar de la mejor manera el dominio del problema, partiendo desde la especificación de requisitos los cuales proporcionan la base para la construcción de la aplicación. En el análisis aspectos como el modelado del contenido e interacción son también importantes ya que ayudan a determinar aspectos estructurales que puede tener la aplicación web así como la interacción que tendrá con el usuario final.

El diseño del sistema es la base del sistema final, proporciona aspectos indispensables para la codificación del programa como lo son el modelado de los datos, del contenido, la navegación y la interfaz. Un diseño bien realizado proporcionará a la futura aplicación una base sólida. El diseño del sistema es una fase crítica del proyecto ya que una falla en este puede resultar en grandes fallas en la aplicación final.

Un framework proporciona un modelo de desarrollo para una aplicación y un conjunto de facilidades y funcionalidades para la obtención de una aplicación robusta. Cada uno de los frameworks tiene fortalezas y debilidades al momento de la programación, la diferencia radica en el área en la cual el framework enfatiza su funcionalidad.

Richfaces framework provee un gran funcionamiento en la capa de presentación, trae consigo un conjunto de controles que hacen el implementar una página web rica sea sencillo. En la capa del modelo el framework no proporciona dificultades y al ser un estándar permite la integración de otras tecnologías de forma sencilla. Las falencias radican en la seguridad de la aplicación ya que no tiene ningún mecanismo para la misma.

Spring es un framework muy completo que tiene una gran cantidad de módulos que trabajando conjuntamente proveen a una aplicación web de una robustez envidiable. La falencia de este framework radica en la capa de presentación, ya que si bien utiliza el estándar JSP y JSTL estos pueden no ser suficientes para responder a las necesidades del desarrollador y del usuario final. El framework es muy eficiente en el manejo de persistencias, seguridad y reportería.

Seam es uno de los frameworks más completos del mercado, este no busca reinventar la rueda sino que integra tecnologías ya existentes de forma natural e incrementa su funcionalidad con

elementos propios del mismo. Debido a esta integración tiene grandes posibilidades en cada una de las capas que proporciona el paradigma MVC. La falencia de este framework reside en que se cierra en las tecnologías que usa, ya que si se desea integrar otras tecnologías en vez de las que viene por defecto puede resultar en un dolor de cabeza o simplemente no resultar.

Capítulo 5

Aplicación de Modelo de Calidad

Este capítulo pretende desarrollar un modelo de calidad de software específico para el dominio del presente trabajo de graduación, partiendo del hecho de que la temática de los frameworks web java es muy extensa y que la comparación entre frameworks de este tipo puede resultar muy compleja debido a la ausencia de una terminología común entre los distintos productos, es decir que cada uno de los productos puede estar determinado por distintas tecnologías y arquitecturas. Para materializar un modelo de calidad para este dominio específico se utilizará el método IQMC, que proporciona un conjunto de técnicas específicas para la construcción de un modelo de calidad basado en el estándar ISO 9126-1 seleccionado en el capítulo 3.

5.1 El método IQMC

El método IQMC proporciona un conjunto de directrices y técnicas, con la intención de ayudar en la definición de modelos de calidad de diversos dominios de software. Este método tiene un enfoque mixto ya que toma como punto de partida un marco de trabajo o framework de calidad y por otro lado se debe desarrollar un conjunto de atributos propios para el dominio estudiado. El método está formado por los siguientes 7. (10)

- Paso 0: Definición y estudio del dominio.
- Paso 1: Determinar las subcaracterísticas de calidad.
- Paso 2: Definir una jerarquía de subcaracterísticas.
- Paso 3: Descomponer las subcaracterísticas en atributos.
- Paso 4: Descomponer atributos derivados en básicos.
- Paso 5: Manifestar las relaciones entre las características de calidad.
- Paso 6: Determinar métricas para atributos básicos.

5.1.1 Paso 0: Definición y estudio del dominio

La definición del dominio es el aspecto más básico para el desarrollo del modelo de calidad ya que es imprescindible la profundización del tema tratado para la construcción del mismo. Para el caso actual en el capítulo 1 y 2 se han presentado aspectos que van desde conceptualización, tipos y arquitectura básica de los frameworks refiriéndonos al paradigma MVC, hasta a una profundización de tres de los frameworks determinados como los más populares. Estos dos capítulos determinan un entendimiento del dominio para el cual se desarrollará un modelo de calidad.

Según el enfoque propuesto por Carvallo en la obra “Systematic Construction of Quality Models For COST-Based Systems”, determina la necesidad de representar el dominio utilizando una notación formal en lenguaje UML, pues esto desencadena un mejor entendimiento del dominio, resolviendo problemas que puedan provenir del lenguaje natural. El siguiente diagrama muestra el dominio de los frameworks web java representado en lenguaje UML:

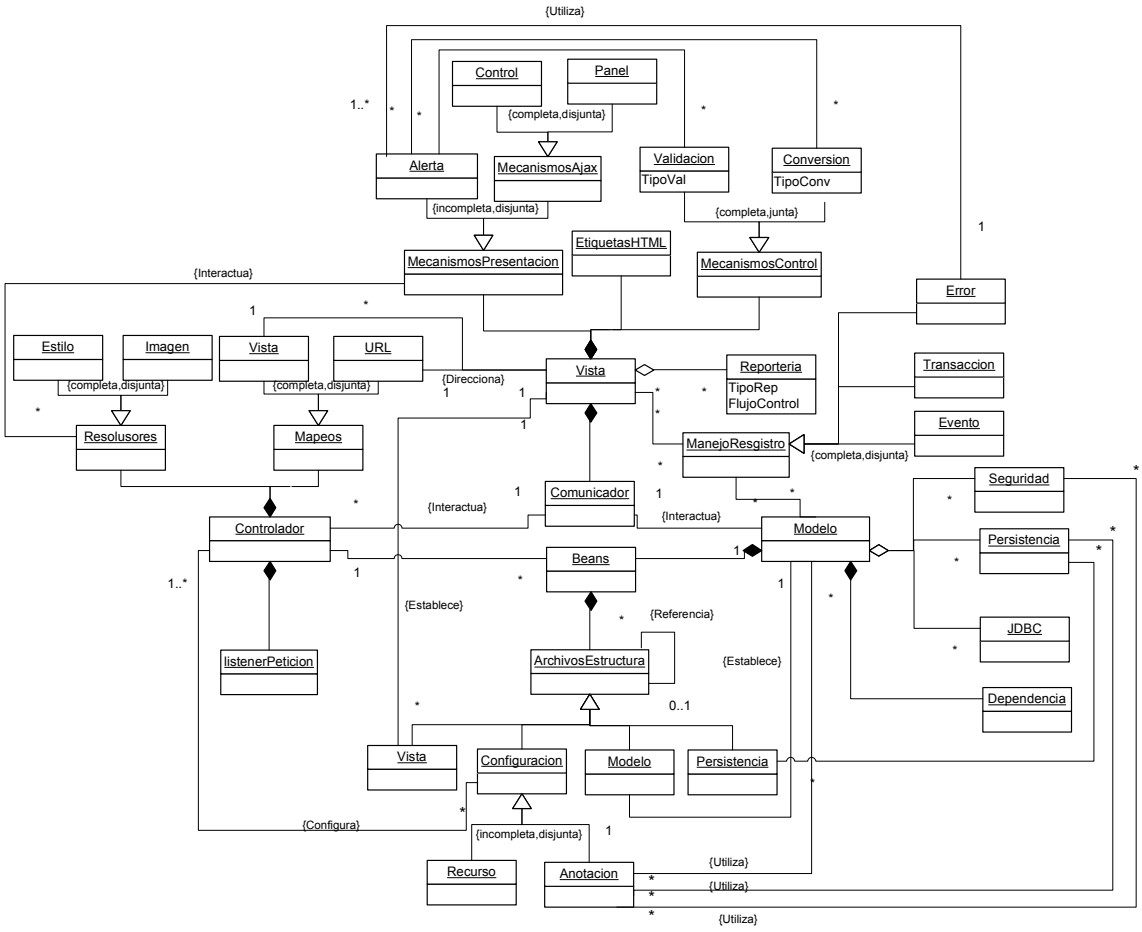

Figura 5.1: Representación del dominio en UML

La figura 5.1 describe de manera clara el domino evitando cualquier subjetividad que pueda darse en el lenguaje natural; es así que se observa gran parte de las funcionalidades provistas por un framework a partir del UML presentado. La representación del domino mediante esta técnica aporta en la posterior construcción de un modelo de calidad.

5.1.2 Paso 1: Determinar las subcaracterísticas de calidad

El método IQMC como ya se nombró en el punto 5.1 tiene un enfoque mixto, ya que parte de un modelo o estándar de calidad ya concebido, en el caso actual se parte del estándar ISO 9126-1, por ser este el seleccionado como más idóneo para el dominio planteado. El estándar trae un conjunto de subcaracterísticas que pueden ser modificadas si el evaluador lo considera conveniente al analizar el dominio. En el dominio de los frameworks web java se ha decidido tomar las características provistas por el estándar sin efectuar ningún cambio sobre las mismas.

5.1.3 Paso 2: Definir una jerarquía de subcaracterísticas

Las subcaracterísticas ofrecidas por el estándar ISO 9126-1 pueden ser extendidas para formar una jerarquía, de esta manera se establecerá una especialización del modelo en cuanto al dominio especificado. Una característica que debe ser respetada es que en la especialización las nuevas subcaracterísticas sean del mismo nivel de abstracción que las ofrecidas en el estándar, ya que de otra forma se convertirían en atributos de calidad. Como ejemplo de esta especialización se puede citar la subcaracterística de seguridad, en donde las especializaciones estarán dadas por la seguridad en la trasmisión de datos y el almacenamiento de datos, en donde se observa que conservan el mismo nivel de abstracción que su subcaracterística padre.

5.1.4 Paso 3: descomponer las subcaracterísticas en atributos

Las subcaracterísticas proporcionadas en el paso 2, si bien brindan un buen entendimiento del modelo su nivel de abstracción es muy alto como para poderlo evaluar directamente, es por esto que la descomposición agregando un nivel de detalle superior es esencial, es decir el paso de subcaracterísticas a atributos de calidad. “Un atributo de calidad realiza un seguimiento de una característica peculiar y observable de los componentes COST en el dominio (10), bajo esta premisa se puede determinar que en un dominio de software tan amplio como lo es el de los frameworks web se puede tener una gran cantidad de atributos.

5.1.5 Paso 4: Descomponer atributos derivados en básicos

Las subcaracterísticas de calidad en conjunto sus atributos, es en un buen nivel una abstracción detallada del dominio estudiado, pero estos atributos todavía pueden ser muy abstractos para ser medidos de manera clara y con una métrica definida. Al existir un nivel alto de abstracción en los atributos estos deben ser derivados en atributos básicos, es decir descomponerlos hasta hacerlos medibles. Para esto puede ser muy útil el modelo conceptual desarrollado en el punto 5.1.1, ya que a partir de este se pueden clarificar conceptos en especial a lo referente en la funcionalidad del framework.

5.1.6 Paso 5: Manifestar relaciones entre las características de calidad

El manifestar las relaciones de calidad entre las características tiene como objetivo obtener un modelo de calidad mucho más elaborado, es decir un paso más refinado que el nivel anterior. En el paso 4 se obtienen los atributos de calidad, y una vez obtenidos las relaciones que tienen entre si deben declararse. Existen algunos tipos de dependencias determinadas por Carvallo, las cuales determinando dos entidades de calidad A y B se pueden definir de la siguiente manera:

- Colaboración: El crecimiento de A implica el crecimiento de B.
- Perjuicio: El crecimiento de A implica el decrecimiento de B.
- Dependencia: Algunos valores de A requieren el cumplimiento de ciertas condiciones pro B.

5.1.7 Paso 6: Determinar métricas para los atributos básicos

Luego de la realización de los 6 pasos anteriores, se obtiene una lista de características con algunos atributos de calidad que debido a su nivel de abstracción son susceptibles a ser medidos de forma objetiva, es por esto que este paso tiene como meta determinar la mejor forma de hacerlo. Existen algunos tipos de métricas que pueden ser usadas desde simples hasta estructuradas, las siguientes son los distintos tipos de métricas que pueden ser usadas (10)

- Booleana: Manifiesta la presencia o ausencia de una característica de un producto.
- Numérica: El atributo expresa algún tipo de métrica ya sea esta un numero entero o un real.
- Etiqueta: El atributo medido registra un nombre.
- Fijado (Set): Tipo de métrica que puede registrar una colección de valores.
- Funciones: El valor resultante de la medición depende de otro, es decir el valor no es absoluto.

Las métricas tienen otro factor importante que es su escala y al hablar de estas lo más correcto es primero adoptar una definición de la misma, de esta manera: “Una escala de medición es un conjunto de valores que permite establecer relaciones entre medidas.” (19) Según Fenton y Pfleeger existen 5 tipos de escalas, de menor a mayor complejidad se describen a continuación:

- Escala nominal: Una escala de este tipo está formada por categorías en las cuales no existe ningún orden, por lo que la única relación que se puede aplicar es la de igualdad. Un ejemplo se puede observar en una clasificación de lenguajes de programación, es decir {Java, C++, Phyton, Ruby….}

- Escala Ordinal: En esta escala se definen también categorías, la diferencia con la anterior radica en que en una escala de este tipo debe haber una noción de orden, un ejemplo de esto se manifiesta en clasificaciones de valoración cualitativa como {Muy poco, Poco, Medio, Bastante, Mucho}
- Escala de intervalo: En este tipo de escala la distancia entre intervalos es conocida y siempre es la misma, ejemplos de estos son los test de inteligencia, la temperatura, o en el área del software la duración de un proyecto.
- Escala de ratio: Escala con un valor inicial referente a cero, permitiendo definir periodos o ratios, la escala proporciona información compleja y permite realizar análisis más complejos, como ejemplo se puede citar el tiempo de un producto de software en el mercado.
- Escala Absoluta: las escalas absolutas tienen características de las escalas anteriores, si bien simplemente consisten en la cuenta sin transformación del número de entidades. Como ejemplo se puede citar el número de programadores involucrados en un desarrollo, es decir algo que solo se puede medir contando.

Una vez citados todos los pasos para la realización del modelo de calidad, lo consiguiente es la materialización del mismo siguiendo cada uno de los pasos ya citados, para posteriormente involucrar a los productos que serán evaluados.

5.2 Modelo de calidad

El siguiente cuadro expresa el resultado del método IQMC para la construcción de modelos de calidad de software.

Modelo de Calidad para frameworks Web Java

Características / Subcaracterísticas	Atributos Nivel 1	Atributos Nivel 2	Atributos Nivel 3	Metrica	Descripción
1 Funcionalidad					ISO/IEC 9126-1
1 Adecuación					ISO/IEC 9126-1
1 Adecuacion de la vista					Atributos relacionados a la adecuación de los mecanismos de la vista
	1 Administracion Alertas			Funtion OBA	Atributos relacionados al envio de alertas al usuario
		1 Generación de alertas		Soportado: Nominal; Soportado=(True,False)	Generacion de alertas por parte del framework
			1 Tipos Alertas	alertas: Nominal = (noSoportado, paginadas, envueltas)	Atributo que determina el tipo de alerta por su redireccion en el flujo
			3 Alertas de validación	Soportado: Nominal; Soportado=(True,False)	Determina el soporte de alertas de validacion de campos
			4 Alertas via email	Soportado: Nominal; Soportado=(True,False)	Determina el soporte de alertas al usuario via e-mail
		2 Control de alertas		Soportado: Nominal; Soportado=(True,False)	Control del tipo de alerta
			1 Personalización de alertas	Soportado: Nominal; Soportado=(True,False)	Personalizacion del mensaje que envia la alerta
			2 Formato alertas	formato: Nominal = (Texto, Imagen,cuadroDialogo)	Personalizacion del contenido del formato de la alerta
	2 Mecanismos de Validacion			Soportado: Nominal; Soportado=(True,False)	Mecanismos en la vista para validar campos
		1 Tipos Validación		Soportado: Nominal; Soportado=(True,False)	Tipos de validacion existenten en la vista
			1 Validación de length	Soportado: Nominal; Soportado=(True,False)	Validacion del numero de caracteres en un campo
			2 Validacion de dates	Soportado: Nominal; Soportado=(True,False)	Falidacion de formatos de fechas
			3 Validación de numbers	Soportado: Nominal; Soportado=(True,False)	Validacion de mascaras en numeros
	3 Mecanismos de Conversion			Soportado: Nominal; Soportado=(True,False)	Lista de tipos de cuenta existentes
		1 Tipos de conversion		Conversion: Nominal; Conversion= (number, string, date, ...)	Nombre del tipo de cuenta gratuito
			1 Conversiones tipos numericos	Conversion:Set(Lables:Nominal); Labels=(noSoportado,toLong, toDouble,...)	Conversion de un campo imput a un number
			2 Conversiones tipo date	Conversion:Set(Lables:Nominal); Labels=(noSoportado,toTime, toTimeStamp)	Conversion de un campo imput a una fecha
			3 Conversiones tipo Byte	Conversion:Set(Lables:Nominal); Labels=((noSoportado,toByteArray,tobinary,...)	Conversion de un campo imput a un archivo
			4 Conversiones tipo string	Conversion:Set(Lables:Nominal); Labels=(noSoportado,(toNumber,toDate,...)	Conversion de un campo imput a un mapa de caracteres
	4 Mecanismos de control			Funtion OBA	Metodologías para comunicación con otras capas de aplicación
		1 Metodología comunicación controlador		metodología: Set(Lables: Nominal); Labels=(implicita, explicita)	Metodologia para comunicación con capa del controlador
		2 Metodología señalamiento modelo		Soportado: Nominal; Soportado=(True,False)	Metodologia para hacer refereencias a capas del modelo cuando el controlador es implicito
	5 Ajax Cointeiners			Funtion OBA	Mecanismos que facilitan porveer de funcionalidad Ajax automatico
		1 Contenedores Ajax		Conteenedores:set(Labels:Nominal);Labels=(reegion,outputpanel, ...)	Area contenedora de controles que seran actualizables
		2 Paneles Ajax		Paneles: Set(Labels: Nominal)Labels=(Modal, Popout, ...)	Paneles independientes proveedores de informacion
		3 Controles Ajax		Controles:Set(Labels:Nominal);Labels=(buttons, inputs, ...)	Controles que accionaran Ajax a paneles o contenedores
	6 Manejo Reportería			Funtion OBA	
		1 Mecanismo Reporteria propios		Soportado: Nominal; Soportado=(True,False)	Mecanismos para proveer manejo de reportes a traves de consultas propios del framework
		2 Reportería Externa		reportería:set(Labels:Nominal);Labels=((Jasper, Cristal Reports, ..)	Mecanismo que provee de funcionalidad de herramientas de reportes externas, integradas de forma automatica al framework
		3 Control flujo reporte		Soportado: Nominal; Soportado=(True,False)	Control de paginacion de reportes
	7 Etiquetas html			Funtion OBA	
2 Adecuación del controlador					Atributos relacionados a la adecuación de los mecanismos del controlador
	1 Metodología de l controlador			metodología: Set(Lables: Nominal); Labels=(implicita, explicita)	Determinacion de una metodología implicita o desarrollada del controlador
	2 Administración de la vista			Funtion OBA	Capacidad del controlador para manejar los reecursos de la capa de presentacion
		1 Resolucion de recursos		Soportado: Nominal; Soportado=(True,False)	Mecanismos de resolucion de recursos utilizados por la página
			1 Resolucion de imágenes	Soportado: Nominal; Soportado=(True,False)	Resolucion de imágenes dentro de la pagina
			2 Resolucion de estilos	Soportado: Nominal; Soportado=(True,False)	Resolucion de estilos CSS por la página
		1 Mecanismos de mapeo URL		Soportado: Nominal; Soportado=(True,False)	Mecanismos para mapeos URL de ser necesarios
			1 Anotaciones mapeado	Soportado: Nominal; Soportado=(True,False)	Mecanismos para anotaciones del tipo de controlador
3 Adecuación del modelo					Atributos relacionados a la adecuación de los mecanismos del modelo
	1 Mecanismos persistentes			Funtion OBA	Mecanismos de framework os de persistencias integrados
		1 Manejador de persistencias		Lenguages:Nominal=(Hibernate, OpenJPA, ..)	Implementacion o intefaz de un framework de presisteencia
			1 Anotaciones persistencia	Soportado: Nominal; Soportado=(True,False)	Anotaciones de ayuda para el control de persistencias
		2 Manejador de seguridad persistencia		Seguridad:Set(Lables:Nominal); Labels=(Anotaciones,Configuracion,...)	Mecanismos sobre persistencias para definir campos de seguridad
			1 Anotaciones seguridad	Soportado: Nominal; Soportado=(True,False)	Anotaciones definitorias de seguridad sobre campo
		3 Resolucion de datos		Funtion OBA	mecanismos resolusores de datos utilizando persistencias
			1 Insercion de datos	Soportado: Nominal; Soportado=(True,False)	Mecanismos de insercion de datos
			2 Modificación de datos	Soportado: Nominal; Soportado=(True,False)	Mecanismos para la modificacion de datos
			3 Eliminacion de datos	Soportado: Nominal; Soportado=(True,False)	Mecanismos para la eliminacion de datos
			4 Consulta de datos	Soportado: Nominal; Soportado=(True,False)	Mecanismos que proveen elguajes para la consulta de datos
	2 Mecanismos JDBC			Function OBA	Mecanismos sencillos de comunicación JDBC
		2 Resolucion de datos JDBC		Funtion OBA	Mecanismos resolusores de datos utilizando JDBC
			1 Insercion de datos	Soportado: Nominal; Soportado=(True,False)	Mecanismos de insercion de datos
			2 Modificación de datos	Soportado: Nominal; Soportado=(True,False)	Mecanismos para la modificacion de datos
			3 Eliminacion de datos	Soportado: Nominal; Soportado=(True,False)	Mecanismos para la eliminacion de datos
			4 Consulta de datos	Soportado: Nominal; Soportado=(True,False)	Mecanismos que proveen elguajes para la consulta de datos
		2 Anotaciones JDBC		Soportado: Nominal; Soportado=(True,False)	Mecanismos resolusores de datos utilizando anotaciones JDBc
	3 Inyeccion dependencias			Function OBA	Mecanismos para el manejo de dependencias dentro de POJOS
		1 Dependencias datos		Soportado: Nominal; Soportado=(True,False)	Manejo de dependencias desde repositorios de datos
		2 Dependencias de identidades		Soportado: Nominal; Soportado=(True,False)	Manejo de depencencias para administrar credenciales
		3 Dependencias de clases		Soportado: Nominal; Soportado=(True,False)	Manejo de dependencias de clases creadas por el desarrollador
		4 Anotaciones Inyeccion		Anotaciones:Set(Label:Nominal) Labels=(In, Inject, Aware,...)	Anotaciones creadas para el manejo de las dependencias
	4 Encapsuladores (beans)			Funtion OBA	Componentes que desarrollan la lógica del negocio
		1 Alcance de beans		Allcances bean :Nominal=(SESSION,REQUEST, ..)	Definicion del alcance de bean
		2 Anotaciones beans		Soportado: Nominal; Soportado=(True,False)	Lenguajes de programacion en los cuales se puede desarrollar el API
4 Adecuación de la estructura					Atributos relacionados a la adecuación de la estructura de proyectos
	1 Administración de archivos			Funtion OBA	Adecuacion del API de desarrollo para cumplir con requisitos de programadores
		1 Archivos de configuracion		Funtion OBA	Soporte de operaciones sobre archivos de configuracion
			1 Crear archivos configuración	Soportado: Nominal; Soportado=(True,False)	Crear archivos
			2 Modificar archivos configuración	Soportado: Nominal; Soportado=(True,False)	Modificar archivos
			3 Eliminar archivos configuración	Soportado: Nominal; Soportado=(True,False)	Eliminar archivos
		2 Archivos de vista		Funtion OBA	Soporte de operaciones sobre archivos de vista
			1 Crear archivos vista	Soportado: Nominal; Soportado=(True,False)	Crear archivos
			2 Modificar archivos vista	Soportado: Nominal; Soportado=(True,False)	Modificar archivos
			3 Eliminar archivos vista	Soportado: Nominal; Soportado=(True,False)	Eliminar archivos
		3 Archivos modelo		Funtion OBA	Soporte de operaciones sobre archivos de modelo
			1 Crear archivos modelo	Soportado: Nominal; Soportado=(True,False)	Crear archivos
			2 Modificar archivos modelo	Soportado: Nominal; Soportado=(True,False)	Modificar archivos
			3 Eliminar archivos modelo	Soportado: Nominal; Soportado=(True,False)	Eliminar archivos
		4 Archivos de persistencias		Funtion OBA	Soporte de operaciones sobre archivos de persistencias
			1 Crear archivos persistencias	Soportado: Nominal; Soportado=(True,False)	Crear archivos
			2 Modificar archivos persistencias	Soportado: Nominal; Soportado=(True,False)	Modificar archivos
			3 Eliminar archivos persistencias	Soportado: Nominal; Soportado=(True,False)	Eliminar archivos
	2 Perspectiva Integrada IDE			Soportado: Nominal; Soportado=(True,False)	Perspectiva organizativa de proyecto en IDE

	5	Adecuación del soporte				Atributos relacionados a la adecuación del soporte para el usuario
		1	Documentación y manuales		Documentacion:Ordinal=(Disponible, parcialmente disponible, no disponible)	Documentos provistos por el vendedor del servicio
		2	Foros de usuario		contenido:Ordinal=(NoProvisto, basico, medio, avanzado)	Adecuación del framework para proporcionar soporte
		3	FAQs		contenido:Ordinal=(NoProvisto, basico, medio, avanzado)	Adecuación del framework para proporcionar un sitio de preguntas frecuentes
		4	Manuales		contenido:Ordinal=(NoProvisto, basico, medio, avanzado)	Adecuación del framework para proporcionar un video o flash que explore la funcionalidad
		5	Show case		contenido:Ordinal=(NoProvisto, basico, medio, avanzado)	Adecuacion del framework para proveer una demostracion de sus componentes
	2	Exactitud				ISO/IEC 9126-1
	1	Verificabilidad				Precision del sistema para permitir la sincronizacion, respaldo e historiales del usuario
		1	Capacidades de registro			Capacidad del sistemar de proveer verificacion de los registros del sistema
			1	Registro de errores	Soportado: Nominal; Soportado=(True,False)	Posibilidad de un registro de errores
			2	Registro de eventos	Soportado: Nominal; Soportado=(True,False)	Posibilidad de un registro de eventos
			3	Registro de trasacciones	Soportado: Nominal; Soportado=(True,False)	Posibilidad de un registro de transacciones
		2	Historiales de servicios			cpacidad para proveer verificacion del historial de los servicios
			1	Bitácora de uso de servicios	Soportado: Nominal; Soportado=(True,False)	Sitio con las actividades del uso del servicios del framework
	2	Efectividad				Mecanismos para determinar el estado de los resultados.
		1	Resultados de pruebas publicadas			Resultados de las pruebas publicados por los desarrolladores del framework sobre la efectividad del mismo
		2	Lista de errores publicados			Lista de errores ´por version que el framework a tenido
	3	Interoperabilidad				ISO/IEC 9126-1
	1	Interoperabilidad Directa				Capacidad del servicio para proporcionar interoperabilidad con otras aplicaciones
		1	Protocolos internet			Capacidad de interaccion con clientes por medio de protocolos
			1	Protocolos web	Protocolos:Set(Label:Nominal); Label=(Http(s),ftp(s)),...)	Protocolos soportados por el framework
		2	IDE desarrollo			Capacidad de interaccion del framework con un IDE de desarrollo
			1	IDES soportados	IDE:SET(Label:Nomina);=Label=(Eclipse, Netbeans,...)	IDEs soportados por el framework
				1	Adecuaciones version	Soportado: Nominal; Soportado=(True,False)	Interoperabilidad entre el framework y las distintas versiones de un IDE
				2	Documentacion plugin	Soportado: Nominal; Soportado=(True,False)	Documentacion de la integracion entre un IDE y un framework
		3	Datos			Capacidad de interactuar con mecanismos relacionales
			1	Frameworks Persistencias	Persistencias:SET(Label:Nominal);Label=(Hibernate,OpenJpa,...)	Integracion con frameworks proveedores de persistencias
			2	Conectores bases de datos JDBC	Soportado: Nominal; Soportado=(True,False)	Integracion con JDBC para la interaccion directa con los datos
		4	Herramientas Reporteria			Capacidad de integracion con herramientas de reporteria externas
			1	Librerias Reportería	Reportería:set(Labels:Nominal);Labels=((Jasper, Cristal Reports, ..)	Capacidad de admision dee librerias de reporterias al framework
			2	Recursos Reportería	Soportado: Nominal; Soportado=(True,False)	Capacidad de redireccion de recursos utilizados por las herramientas de reporteria
		5	Pruebas unitarias			Capacidad de integracion con frameworks para pruebas unitarias de codigo
			1	Frameworks Pruebas Unitarias	Pruebas:SET(Label:Nominal);Label=(Junit,...)	Integracion con distintos frameworks y librerias de prueba unitarias
		6	Servidores aplicaciones web			Capacidad para funcionar en distintos servidores web/aplicaciones
			1	Servidores soportados	Servidorees:SET(Label:Nominal);Label=(Tomcat,GlshFish,...)	Servidores web soportados por el framework
	2	Interoperabilidad Indirecta				Capacidad del servicio para proporcionar interoperabilidad con otras aplicaciones
		1	Navegadores web			Capacidad para funcionar en distintos navegadores web
			1	Navegadores soportados	Funtion OBA	Navegadores web soportados con una completa funcionabilidad de todos los componentes
				1	Google Chrome	Soportado: Nominal; Soportado=(True,False)	Navegador google chrome soportado
				2	Mozilla Fireefox	Soportado: Nominal; Soportado=(True,False)	Navegado mozilla firefox soportado
				3	Internet Explorer	Soportado: Nominal; Soportado=(True,False)	Navegador Internet explorer soportado
				4	Safari	Soportado: Nominal; Soportado=(True,False)	Navegador Safari soportado
				5	Opera	Soportado: Nominal; Soportado=(True,False)	Navegador opera soportado
	4	Seguridad				ISO/IEC 9126-1
	1	Seguridad de la aplicación				Mecanismos de seguridad para evitar que los usuarios y sus datos sean hackeados
		1	Control de usuarios			Mecanismos de seguridad sobre control de usuarios
			1	Login y password	Soportado: Nominal; Soportado=(True,False)	Mecanismos de login y password
			2	Permisos por roles	Soportado: Nominal; Soportado=(True,False)	Mecanismos de permisos por roles
			3	Restricción a no autentificados	Soportado: Nominal; Soportado=(True,False)	Mecanismos de restriccion de paginas a no usuarios
			4	Protección de páginas por URL	Soportado: Nominal; Soportado=(True,False)	Proteeccion de paginas por roles
			5	Manejo de sesiones	Soportado: Nominal; Soportado=(True,False)	Manejos de sesion de usuarios
		2	Seguridad de los datos			Mecanismos de seguridad sobre los datos del servicio
		1	Encriptación de datos			Mecanismos de encriptacion de datos en bd
			1	Protocolos de encriptacion	Protocolos:Set(Label:Nominal); Label=(md5,token,...)	Protocolos soportados de encriptacion de contraseñas
		2	Transmición de datos			Mecanismos de seguridad al trasmitir datos
		1	Protocolos de tranferencia de datos		Protocols:Set(Labels: Nominal); Labels=(SSL, ...)	Protocolos de transferencia de datos seguros sobre la web
	5	Cumplimiento Funcional				ISO/IEC 9126-1
2	Confiabilidad					ISO/IEC 9126-1
	1	Madurez				ISO/IEC 9126-1
	1	Historial del producto				Historial del producto en el mercado
		1	Tiempo del producto en el mercado		Periodo: Ratio; Period = Float[Años]	Tiempo que el producto ha permanecido en el mercado
		2	Versiones producto y parches		Function OBA	Versiones de los productos y parches
			1	Versiones	Versiones: Ratio;Versiones=Float(Numero)	Numero de versiones estables en el mercado
			2	Parches por version	Parches:Set(<Version:Ordinal, numParches:ratio>); Version=Confiabilidad, madurez, historialProducto, numParches=(desconocido)	Numero de parches por version estable
			1	Fallas detectadas por version	Fallas Set(<Version: Ordinal, numFallas Ratio>); Version=Reliability.Matirity.ProductHystory.V, numFallas = Integer	Fallas conocidas por cada version
	2	Robustez				Mantenibilidad del producto y sus fallas
		1	Robustez Preoperacional			Mecanismos de medicion de confiabilidad pro-operaciona
			1	Tiempo medio entre fallos produccion	Periodo: Ratio; Periodo= Float[Hours]	Tiempo medio entre fallas en un ambiente de versiones beta
			2	Tiempo medio entre fallos Desarrollo	Periodo: Ratio; Periodo= Float[Hours]	Tiempo medio entre fallos en un ambiente de desarrollo de versiones beta
		2	Robustez Operacional			Mecanismos de historial de fallas una vez liberada una version del producto
			1	Tiempo medio entre fallos produccion	Periodo: Ratio; Periodo= Float[Hours]	Tiempo medio entre fallas en un ambiente de pruebas de produccion
			2	Tiempo medio entre fallos Desarrollo	Periodo: Ratio; Periodo= Float[Hours]	Tiempo medio entre fallos en un ambiente de desarrollo
	3	Transparencia				capacidad del sistema de funcionar con fallos sin que afecte el funcionamiento de los usuarios
		1	Navegacion continua tras un fallo		Soportado: Nominal; Soportado=(True,False)	Soporte al seguir con una navegacion transparentee ante un fallo
		2	Resolucion de dependencias fallidas en tiempo real		Soportado: Nominal; Soportado=(True,False)	Soporte de creacion de de dependencias ante un fallo de las mismas
		3	Re-instanciación de componente fallidos		Soportado: Nominal; Soportado=(True,False)	Soporte de re instanciacion de un componente del framework ante un fallo en el mismo
	4	Nivel de tolerancia				Capacidad del sistema para ser configurado con un nivel de tolerancia a fallos
		1	Numero de compilaciones por error de memoria		►Numero intentos: (Type: Ordinal, Value: Absolute); Type = (NotConfigurable, PartiallyConfigurable, FullyConfigurable); Value = Integer[Retry]	Posibilidad de configurar el numero maximo de compilaciones antes de un fallo de memoria java (perjem space)
	5	Capacidad de recuperación de fallos				Mecanismos de recuperacion en caso de fallos
		1	Reiniciacion de servidor de aplicaciones ante fallos		Soportado: Nominal; Soportado=(True,False)	Posibilidad de que el framework por si solo reinicie el servidor de aplicaciones ante un fallo

	2 Recuperabilidad							ISO/IEC 9126-1
	1 Recuperacion de sistema							Capacidad de recuperar los datos del usuario tras un fallo
		1 Registro de errores						Posibilidad de un registro de errores
		2 registro de eventos						Posibilidad de un registro de eventos
		3 Registro de transacciones						Posibilidad de un registro de transacciones
		4 Recuperación automática ante fallos						Posibilidad de recuperar los servicios del framework automaticamente tras un fallo
	3 Cumplimiento de Confiabilidad							ISO/IEC 9126-1
3 Usabilidad								ISO/IEC 9126-1
Usabilidad	1 Comprensibilidad							ISO/IEC 9126-1
	1 Comprension de la metodología							Esfuerzo necesario para entender como usar la metodología de programacion del framework
	1 Conprension de la vista							Esfuerzo necesario para entender los mecanismos y herramientas de la vista provista por el framework
		1 Mecanismos de comunicación						Comprension de los mecanismos para comunicar la vista y modelo
		2 Preedictabilidad de uso						Prediccion y comprension del uso de tecnologias de la vista al compararlo con estandares html o jsp
		3 Mecanismos ajax						Facilidad de comprension de los mecanismos Ajax en caso de tenerlos
	2 Comprension del modelo							Esfuerzo necesario para entender los mecanismos y herramientas del modelo provista por el framework
		1 Metodología de dependencias						Comprension de los mecanismos y herramientas de resolucion de dependencias en caso de tenerlos
		2 Metodología de encapsuladores (bean)						Conprension de la metodología de encapsuladores de codigo web
	3 Comprension del controlador							Esfuerzo necesario para entender los mecanismos y herramientas del controlador provista por el framework
		1 Metodos controlador						Comprension de la metodología de reesolucion que tiene el controlador, sean implicitos o programados
		2 Resolusores de recursos						Comprension de los mecanismos resolusores de imágenes u otros
		3 Resolusores de vistas						Comprension de los mecanismos resolusores de vistas
		4 Alcance controlador						Entendimiento de los mecanismos del controlador y la abastraccion del mismo en torno a configuracion y flujo
	2 Estructura Global							Definicion de una estructura para la creeacion de proyectos
		1 Arquitectura bien definida						Definicion de una arquiteectura MVC bien definida para la creacion de proyectos
			1 Vista definida					Mecanismos de vista definidos de forma clara
			2 Modelo definido					Mecanismos de modelo definidos de forma clara
			3 Contolador definido					Mecanismos de controlador dfinidos de forma clara.
		1 Flexibilidad de la estructura de proyectos						Flexibilidad en la estructura de proyectos realizados con el framework
	2 Aprendizaje							ISO/IEC 9126-1
	1 Entrenamiento							Mecanismos de entrenamiento provistos por el desarrollador del framework
		1 Videos demostrativos						Aprendizaje por medio de videos demostrativos del framework
		2 Soporte online						Soporte a desarrolladores/programadores en linea
		3 Tutoriales						Tutoriales en video o texto que determinen un entrenamiento del uso del framework
		4 Show case						Adecuacion del framework para proveer una demostracion de sus componentes en la web
	2 Documentación							Documentacion usada para el aprendizaje del servicio
	1 Documentación del desarrollador							Documentos provistos por el desarrollador del framework
		1 Documentación y manuales						Documentos provistos por el desarrollador del framework
		2 Foros de usuario						Adecuación del framework para proporcionar foros de desarrollo
		3 FAQs						Adecuación del framework para proporcionar un sitio de preguntas frecuentes
		4 Consultas						Adecuación del framework para proporcionar cconsultas on-line del producto.
		5 Show case						Adecuacion del framework para proveer una demostracion de sus componentes
		6 Proyectos ejemplos demostrativos						Adecuacion del framework para proporcionar ejemplos de proyectos desarrollados con el mismo
	2 Documentación externa							Documentacion externa para el uso de la aplicación
		1 Foros						Foros de usuario externos del framework
		2 Ayuda online						ayuda online de externos acerca del framework
		3 webs dedicadas al soporte						Paginas web dedicadas exlusivamente a la exploracion y soporte del framework
	3 Operabilidad							ISO/IEC 9126-1
	1 Configuración							Mecanismos de configuracion que permite el framework al desarrollador
	1 Configuración global							Mecanismos que permiten al desarrollador conigurar el servicio
		1 Administración del controlador						Configuracion de mecanismos del controlador
			1 Configuracion del view resolver					Configuracion del resolusor de vistas
				1 Patron de URL				Configuracion de los patrones que seran admisibles en la navegacion
				2 Pagina de inicio				Configuracion de la pagina de inicio
				3 Metodos de acceso				Configuracion de los metodos de acceso a los servicios de la pagina
			2 Configuracion del resource resolver					Configuracion de la ubicación de los recursos que usara la aplicación
			3 Configuracion del tipo de mapeo					Configuracion de los patrones de mapeo URL de la aplicación
		2 Administracion de beans						Configuracion de los encapsuladores o beans
			1 Habilitacion de anotaciones					Configuracion de las anotaciones que indican que una clase es un bean y el tipo del mismo
			2 Configuracion de beans					Configuracion de los beans de la aplicación
				1 Beans de modelo				Configuracion de los beans del modelo
				2 Beans de reporteria				Configuracion de los beans de reporteria
				3 paquetes base Beans				Configuracion de las ubicaciones de beans
		3 Administración de seguridad						Configuracion de la seguridad de la aplicación
			1 Mecanismos de login					Mecanismos de autentificacion de usuarios
				1 Determinacion login page				Configuracion de la vista de autentificacion de usuarios
			2 Configuracion de restricciones					Configuracion de restricciones de la aplicación web
				1 Patrones de restricciones				Configuracion de patrones de restricciones a usuarios
				2 Permisos por patrones de paginas				Configuracion de permisos a usuarios por patrones
				3 Permisos por paginas individuales				Configuracion de permisos por vistas individuales
				4 Permisos por roles				Configuracion de permisos por roles
		4 Administración persistencias						Mecanismos de configuracion de persistencias
			1 Configuracion de data source					Configuracion de la fuente de datos para el framework de persistencias
			2 Configuracion de persistencias					Configuracion de persistenciass

					1 Tipo de trasacciones	Configurable: Nominal; Configurable=(True,False)	Configuracion de transaccionalidad local o JTA (java transaction API)
					2 Lenguajes de consultas	Configurable: Nominal; Configurable=(True,False)	Configuracion del lenguaje que sera usado para las consultas
					3 Cambios automaticos bd	Configurable: Nominal; Configurable=(True,False)	Configuracion de cambios en la base de datos desde archivos de persistencias
					4 Dialecto de proveedor	Configurable: Nominal; Configurable=(True,False)	Configuracion del dialecto de comunicación entre el framework de persistencias y la bae de datos
	4 Atractividad						ISO/IEC 9126-1
	1 Navegabilidad						Mecanismos pprovistos por el framework para la clarificacion de la navegacion en el plugin de un IDE i IDE propio
		1 Iconos personalizados plugin IDE				Soportado: Nominal; Soportado=(True,False)	Provision de iconos de carpetas personalizados para llos componentes propios de la aplicación.
	5 Cumplimiento de usabilidad						ISO/IEC 9126-1
4 Eficiencia							ISO/IEC 9126-1
	1 Empleo de Recursos						ISO/IEC 9126-1
	1 Requerimientos						Reequerimientos del producto en cuanto a hardware y software
		1 Requerimientos Hardware				Recursos:Set(<Nombre:Nominal, Requerimiento:Nominal>); Nombre=(RAM, Processor, HD,...), Requerimiento=Lavel[Resource unit]	Requerimientos minimos de hardware necesarios para el framework
			1 Memoria RAM			Memoria: Ordinal; Memoria=(0-256,256-512,512-1024,1024-2048,2048>)	Requerimientos de memoria RAM para el correcto funcionamiento del framework
			2 CPU			CPU: Ordinal; CPU=(0-1Ghz,1-2Ghz,2-3Ghz,3Ghz>)	Requerimientos de CPU minimos para el correcto funcionamiento
		2 Requerimientos Software				Recursos:Set(Labels: Nominal); Labels=(OS, ...)	Requerimientos minimos de software necesarios para el framework
			2 Requerimientos jdk			Version:(Labels:Nominal); Labels=(1.4,1.5,1.6,...)	Reequerimientos del kit de desarrollo java
		2 Sitemas operativos				OS:Set(Labels: Nominal); Labels=(Windows, Unix, Linux, ...)	Requerimientos del sistema operativo
	2 Cumplimiento de eficiencia						ISO/IEC 9126-1
5 Mantenimiento							ISO/IEC 9126-1
	1 Capacidad de ser analizado						ISO/IEC 9126-1
	1 Análisis del producto						Datos disponibles para evaluar producto
	1 Capacidades de registro						Mecanismos de reegistro de framework en logs importantes para el analisis
		1 Registro de errores				Soportado: Nominal; Soportado=(True,False)	Posibilidad de un registro de errores
		2 registro de eventos				Soportado: Nominal; Soportado=(True,False)	Posibilidad de un registro de eventos
		3 Registro de transacciones				Soportado: Nominal; Soportado=(True,False)	Posibilidad de un registro de transacciones
		4 Recuperación automática ante fallos				Soportado: Nominal; Soportado=(True,False)	Posibilidad de recuperar los servicios del framework automaticamente tras un fallo
	2 Capacidad para ser cambiado						ISO/IEC 9126-1
	1 Flexibilidad en interfaz						Provision de mecanismos de interfaz
		1 Soporte de skins				Soportado: Nominal; Soportado=(True,False)	Capacidad de soporte de skins para cambiar una interfaz de manera sencilla
	2 Ambiente IDE Desarrollo						Ambiente de desarrollo que el framework provee
		1 IDE personalizado				Soportado: Nominal; Soportado=(True,False)	Mecanismos de personalizacion que extienen funcionalidad de un IDE
		2 Librerias de desarrollo				Soportado: Nominal; Soportado=(True,False)	Capacidad de tomar librerias externas para proyectos
	3 Documentacion Desarrollo						Documentacion para el desarrollador de aplicaciones
		1 Documentación y manuales				Documentacion:Ordinal(Disponible, parcialmente disponible, no disponible)	Documentos provistos por el desarrollador del framework
		2 Foros de usuario				Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	Adecuación del framework para proporcionar foros de desarrollo
		3 FAQs				Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	Adecuación del framework para proporcionar un sitio de preguntas frecuentes
		4 Consultas				Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	Adecuación del framework para proporcionar cconsultas on-line del producto.
		5 Show case				Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	Adecuacion del framework para proveer una demostracion de sus componentes
		6 Proyectos ejemplos demostrativos				Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	Adecuacion del framework para proporcionar ejemplos de proyectos desarrollados con el mismo
	3 Estabilidad						ISO/IEC 9126-1
	1 Estabilidad operacional						Estabilidad del liberamiento de versiones del producto
		1 Tiempo promedio entre liberación de versiones estables				Tiempo: Ratio; Tiempo = Float[meses]	Tiempo promedio entre la liberacion de nuevas versiones estables del framework
	2 Estabilidad Desarrollo						Estabilidad del liberamiento de versiones de prueba del producto
		1 Tiempo promedio entre liberación de versiones beta				Tiempo: Ratio; Tiempo = Float[meses]	Tiempo promedio entre la liberacion de nuevas versiones alfas y betas del framework
	4 Testeo						ISO/IEC 9126-1
		1 Frameworks Pruebas Unitarias				Soportado: Nominal; Soportado=(True,False)	Capacidad del produco para ser probado desde un framework de pruebas unitarias
		2 Herramientas de análisis				Soportado: Nominal; Soportado=(True,False)	Herramientas provistas por el framework para el analisis de la funcionalidad del mismo
	5 Cumplimiento de mantenimiento						ISO/IEC 9126-1
6 Portabilidad							ISO/IEC 9126-1
	1 Adaptabilidad						ISO/IEC 9126-1
		1 Sistemas operativos soportados				OS:Set(Labels: Nominal); Labels=(Window ,Liunux, ...)	Adaptacion del framework a distintos sistemas operativos sin errores
		3 Serviores web soportados				servidorees:SET(Label:Nominal);Label=(Tomcat,GlshFish,...)	Adaptacion del framework a distintos servidorees web
		4 Navegadores web soportados				Function OBA	Adaptacion del framework a distintos navegadores web
			1 Google Chome			Soportado: Nominal; Soportado=(True,False)	Navegador google chrome soportado
			2 Mozilla Fireefox			Soportado: Nominal; Soportado=(True,False)	Navegado mozilla firefox soportado
			3 Internet Explorer			Soportado: Nominal; Soportado=(True,False)	Navegador Internet explorer soportado
			4 Safari			Soportado: Nominal; Soportado=(True,False)	Navegador Safari soportado
			5 Opera			Soportado: Nominal; Soportado=(True,False)	Navegador opera soportado
	2 Instalación						ISO/IEC 9126-1
	1 Facilidades de instalación						Complejitud de la instalacion del producto
		1 Asistente de instalación				Herramientas:Nominal; Herramientas=(Installation wissards, configuration tools...)	Herramientas y asistentees provistas para facilitar la instalacion desktop
	2 Soporte de instalacion						Soporte para la instalacion del producto
		1 Documentación y manuales				Documentacion:Ordinal(Disponible, parcialmente disponible, no disponible)	Documentos provistos para la instalacion del framework
		2 FAQs				Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	Soporte de instalacion dentro de FAQS
		3 Archivos de ayuda				Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	Archivos de ayuda para instalacion
		4 Soporte para creacion de proyectos nuevos				Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	Documentos explicativos en donde se detallen los aspectos de la creacion de un proyecto nuevo
	3 Compatibilidad de la plataforma						Capacidad del sistema para ser instalado en cualquier plataforma
		1 Sistemas operativos soportados				OS:Set(Labels: Nominal); Labels=(Window ,Liunux, ...)	Adaptacion del framework a distintos sistemas operativos sin errores
	3 Coexistencia						ISO/IEC 9126-1
	2 Por el significado de las APIs(Conectores)						Capacidad de interactuar con otras aplicaciones por medio de APIs
		1 Protocolos internet					Capacidad de interaccion con clientes por medio de protocolos
		1 Protocolos web				Protocolos:Set(Label:Nominal); Label=(Http(s),ftp(s),...)	Protocolos soportados por el framework
		2 IDE desarrollo					Capacidad de interaccion del framework con un IDE de desarrollo
		1 IDES soportados				IDE:SET(Label:Nomina);=Label=(Eclipse, Netbeans,...)	IDEs soportados por el framework
			1 Adecuaciones version			Soportado: Nominal; Soportado=(True,False)	Interoperabilidad entre el framework y las distintas versiones de un IDE
			2 Documentacion plugin			Soportado: Nominal; Soportado=(True,False)	Documentacion de la integracion entre un IDE y un framework
		3 Datos					Capacidad de interactuar con mecanismos relacionales
		1 Frameworks Persistencias				Persistencias:SET(Label:Nominal);Label=(Hibernate,OpenJpa,...)	Integracion con frameworks proveedores de persistencias
		2 Conectores bases de datos JDBC				Soportado: Nominal; Soportado=(True,False)	Integracion con JDBC para la interaccion directa con los datos

4	Herramientas Reporteria			Capacidad de integracion con herramientas de reporteria externas
1	Librerias Reportería		Soportado: Nominal; Soportado=(True,False)	Capacidad de admision dee librerias de reporterias al framework
2	Recursos Reportería		Soportado: Nominal; Soportado=(True,False)	Capacidad de redireccion de recursos utilizados por las herramientas de reporteria
5	Pruebas unitarias			Capacidad de integracion con frameworks para pruebas unitarias de codigo
1	Frameworks Pruebas Unitarias		Pruebas:SET(Label:Nominal);Label=(Junit,...)	Integracion con distintos frameworks y librerias de prueba unitarias
6	Servidores aplicaciones web			Capacidad para funcionar en distintos servidores web/aplicaciones
1	Servidores soportados		servidorees:SET(Label:Nominal);Label=(Tomcat,GlshFish.)	Servidores web soportados por el framework
4	Cumplimiento de portabilidad			ISO/IEC 9126-1

Tabla 5.1 Modelo de calidad para Frameworks Web.

5.3 Porcentajes de importancia

Para el establecimiento de resultados en una evaluación de calidad de software, es necesario primero establecer un conjunto de porcentajes de cómo cada una de las características y atributos de calidad afectaran a la evaluación. Para el presente caso el establecimiento de estos porcentajes será instituido a partir de una visión del desarrollador de aplicaciones web, es decir se buscará un perfil en el cual se vean optimizadas las características que en base a la experiencia resultan más vitales para el desarrollador.

- Funcionalidad: La funcionalidad es uno de los aspectos más importantes a la hora de la selección de un framework, ya que a partir de esta el desarrollador tendrá las herramientas para el futuro trabajo, es por esto que la funcionalidad se dará una gran importancia según el siguiente cuadro:

Característica	Peso Total
Funcionalidad	35%
Subcaracterísticas	Peso sobre característica
Adecuación	40%
Exactitud	10%
Interoperabilidad	25%
Seguridad	25%

Tabla 5.2: Porcentajes asignados en torno a Funcionalidad.

- Confiabilidad: Esta es también una de las características que un desarrollador busca en un framework ya que el este debe mantener ciertos niveles de prestaciones en ambientes donde se vea exigido.

Característica	Peso
Confiabilidad	10%
Subcaracterísticas	Peso sobre característica
Madurez	50%
Recuperabilidad	50%

Tabla 5.3: Porcentajes asignados en torno a Confiabilidad.

- Usabilidad: La usabilidad de una herramienta, en este caso un framework es un aspecto fundamental, ya que debido a que el tiempo siempre es un aspecto importante en los proyectos, no se puede desperdiciarlo en una herramienta que posea una gran dificultad y que pueda no tener todos los medios para una correcta comprensión y aprendizaje.

Característica	**Peso**
Usabilidad	25%
Subcaracterísticas	Peso sobre característica
Comprensibilidad	30%
Aprendizaje	40%
Operatividad	25%
Atractividad	5%

Tabla 5.4: Porcentajes asignados en torno a Usabilidad.

- Eficiencia: Un framework web debe usar de forma racional los recursos que le son entregados, por lo cual el tema de la eficiencia es de gran importancia.

Característica	**Peso**
Eficiencia	5%
Subcaracterísticas	Peso sobre característica
Empleo de recursos	100%

Tabla 5.5: Porcentajes asignados en torno a Eficiencia.

- Mantenimiento: el mantenimiento del software es una de las tareas más costosas, es por esto que la posibilidad de cambiar el software o analizarlo es una de las actividades en las cuales el desarrollador determina menos recursos, ya que el eje central es el desarrollo de la aplicación.

Característica	**Peso**
Mantenimiento	15%
Subcaracterística	Peso sobre característica
Capacidad para ser analizado	34%
Capacidad para ser cambiado	33%
Testeo	33%

Tabla 5.6: Porcentajes asignados en torno a Mantenimiento.

- Portabilidad: la capacidad de un software para ser operado en distintos entornos siempre es importante para el desarrollador, pese a que los frameworks estudiados son portables de forma natural al ser estos desarrollados en java, existen otros aspectos importantes a tomar en cuenta.

Característica	Peso
Portabilidad	10%
Subcaracterísticas	Peso sobre característica
Adaptabilidad	33%
Instalación	33%
Coexistencia	34%

Tabla 5.7: Porcentajes asignados en torno a Portabilidad.

5.4 Cuantificación de métricas

Como se ha podido observar en el en el punto 5.2, un modelo de calidad tiene un conjunto de métricas de distinto tipo, estas métricas en la mayoría de los casos expresas un valor cualitativo por lo cual resulta complicado determinar cuál es el mejor framework, es por esto que se ve necesario el aplicar valores numéricos a cada una de estas, para que de esta forma poder determinar objetivamente que framework web java satisface de la mejor manera los porcentajes de importancia presentados en el punto 5.3. Para la asignación de valores a cada una de las métricas se tomará en cuenta el tipo de dato y la escala que la métrica tiene.

El modelo de calidad presentado en el punto 5.2 tiene los siguientes tipos distintos de métrica, de acuerdo a su escala se encuentran las Nominales, Ordinales, Ratio, pudiendo ser estas de distinto tipo como ya se ha determinado en el punto 5.1. La asignación de valores se realizará de la siguiente manera según el tipo:

- Métricas booleanas: La asignación de valores a esta métrica es bastante simple ya que sus valores solamente pueden ser verdadero o falso, lo cual implica que si el framework cumple con un valor verdadero se asignara un valor de 10, caso contrario se asignara un 0.
- Métricas Nominales Fijado (Set): La asignación de valores en este tipo de métricas puede resultar un tanto complicado, ya que un atributo con esta métrica puede retornar un conjunto de valores cuyo número no está determinado exactamente. De acuerdo a esto el valor que será fijado estará entre 0 y 10, será dependiente de un aproximado del número posible de opciones de valores.
- Métricas Ratio: Si bien estas métricas pueden proporcionar una gran cantidad de información sobre un atributo, estas son métricas que son muy difícilmente cuantificables, por lo que para el presente caso permanecerán como ítems informativos, que el desarrollador utilizará.
- Métricas ordinales: La cuantificación de las métricas ordinales se las realizará mediante una asignación de acuerdo a la posición del valor del atributo dentro de la métrica, por tanto el máximo valor de la métrica obtendrá un 10 y el menor un valor de 0.

5.5 Aplicación de modelo de calidad

El siguiente cuadro demuestra la aplicación del modelo de calidad desarrollado tomando en cuenta todos los lineamientos que se han representado en este capítulo, es decir la cuantificación de las métricas y el establecimiento de porcentajes de importancia ya están también expresados. En este proceso de evaluación se debe ser muy preciso para obtener los resultados más acercados a la realidad posibles para de esta manera tener la mejor idea posible de cuál es el framework más conveniente a las necesidades del desarrollador de aplicaciones web java.

Modelo de Calidad para frameworks Web Java								
Características	Metrica		Spring Framwork		Seam Framework		Richfaces Framework	
1 Funcionalidad								
1 Adecuación		29,10526316	553	35,84210526	681	29,57894737	562
1 Adecuacion de la vista			110		209		182
1 Administracion Alertas	Funtion OBA		40		70		60
1 Generación de alertas	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
1 Tipos Alertas	alertas: Nominal = (noSoportado, paginadas, envueltas)	noSoportado	0	paginadas, enueltas	10	paginadas, enueltas	10
3 Alertas de validación	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
4 Alertas via email	Soportado: Nominal; Soportado=(True,False)	False	0	True	10	False	0
2 Control de alertas	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
1 Personalización de alertas	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
2 Formato alertas	formato: Nominal = (Texto, Imagen,cuadroDialogo)	False	0	True	10	True	10
2 Mecanismos de Validacion	Funtion OBA		0		40		40
1 Tipos Validación	Soportado: Nominal; Soportado=(True,False)	False	0	True	10	True	10
1 Validación de length	Soportado: Nominal; Soportado=(True,False)	False	0	True	10	True	10
2 Validacion de dates	Soportado: Nominal; Soportado=(True,False)	False	0	True	10	True	10
3 Validación de numbers	Soportado: Nominal; Soportado=(True,False)	False	0	True	10	True	10
3 Mecanismos de Conversion	Funtion OBA		29		39		39
1 Tipos de conversion	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
1 Conversiones tipos numericos	Conversion:Set(Lables:Nominal); Labels=(noSoportado,toLong, toDouble,...)	toNumber	6	toInt, toLong, toDouble	10	toInt, toLong, toDouble	10
2 Conversiones tipo date	Conversion:Set(Lables:Nominal); Labels=(noSoportado,toDate,toTime, toTimeStamp)	noSoportado	0	toDate,toTimeStamp	6	toDate,toTimeStamp	6
3 Conversiones tipo Byte	Conversion:Set(Lables:Nominal); Labels=((noSoportado,toByteArray,tobinary,...)	toByteArray	3	toByteArray	3	toByteArray	3
4 Conversiones tipo string	Conversion:Set(Lables:Nominal); Labels=(noSoportado,toChar,toString)	toString	10	toString	10	toString	10
4 Mecanismos de control	Funtion OBA		16		20		16
1 Metodología comunicación controlador	metodología: Set(Lables: Nominal); Labels=(implicita, explicita)	explicita	6	explicita, implicita	10	implicita	6
2 Metodología señalamiento modelo	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
5 Ajax Cointeiners	Funtion OBA		0		30		17
1 Contenedores Ajax	Conteenedores:set(Labels:Nominal);Labels=(noSoportado,region,outputpanel, ...)	noSoportado	0	region, output panel, notification panel	10	region, output panel	6
2 Paneles Ajax	Paneles: Set(Labels: Nominal)Labels=(nosoportaddo, Modal, Popout, ...)	noSoportado	0	modal panel, popout panel, panelConfirmation, panelGroup	10	modal panel, popout panel	4
3 Controles Ajax	Controles:Set(Labels:Nominal);Labels=(noSoportaddo,buttons, inputs, ...)	noSoportado	0	comandbutton, command link, tree, select button, pickList, List, fileEntry, submitMonit	10	comandbutton, command link, tree, select button, pickList	7
6 Manejo Reportería	Funtion OBA		15		0		0
1 Mecanismo Reporteria propios	Soportado: Nominal; Soportado=(True,False)	False	0	False	0	False	0
2 Reportería Externa incluida	reportería:set(Labels:Nominal);Labels=((noSoportado,Jasper, Cristal Reports, ..)	Jasper	5	noSoportado	0	noSoportado	0
3 Control flujo reporte	Soportado: Nominal; Soportado=(True,False)	True	10	False	0	False	0
7 Etiquetas html	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
2 Adecuación del controlador			55		40		35
1 Metodología de l controlador	metodología: Set(Lables: Nominal); Labels=(implicita, explicita)	explicita	5	implicita, explicita	10	explicita	5
2 Administración de la vista	Funtion OBA		50		30		30
1 Resolucion de recursos	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
1 Resolucion de imágenes	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
2 Resolucion de estilos	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
1 Mecanismos de mapeo URL	Soportado: Nominal; Soportado=(True,False)	True	10	False	0	False	0
1 Anotaciones mapeado	Soportado: Nominal; Soportado=(True,False)	True	10	False	0	False	0
3 Adecuación del modelo			223		270		196
1 Mecanismos persistentes	Funtion OBA		103		120		92
1 Manejador de persistencias	Lenguages:Nominal=(Hibernate, OpenJPA, ..)	Hibernate, JDO, Oracle Toplink, iBatis, Jpa	8	Hibernate, JDO, Oracle Toplink, iBatis, Jpa, EclipseLink, OpenJpa	10	Hibernate	2
1 Anotaciones persistencia	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
2 Manejador de seguridad persistencia	Seguridad:Set(Lables:Nominal); Labels=(noSoportado,Anotaciones,Configuracion)	Configuracion	5	Configuracion, Anotaciones	10	noSoportado	0
1 Anotaciones seguridad	Soportado: Nominal; Soportado=(True,False)	False	0	True	10	False	0
3 Resolucion de datos	Funtion OBA		40		40		40
1 Insercion de datos	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
2 Modificación de datos	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
3 Eliminacion de datos	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
4 Consulta de datos	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
2 Mecanismos JDBC	Function OBA		80		90		90
1 Resolucion de datos JDBC	Funtion OBA		40		40		40
1 Insercion de datos	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
2 Modificación de datos	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
3 Eliminacion de datos	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
4 Consulta de datos	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
2 Anotaciones JDBC	Soportado: Nominal; Soportado=(True,False)	False	0	True	10	True	10
3 Inyeccion dependencias	Funtion OBA		40		40		0
1 Dependencias datos	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	False	0
2 Dependencias de identidades	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	False	0
3 Dependencias de clases	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	False	0
4 Anotaciones Inyeccion	Anotaciones:Set(Label:Nominal) Labels=(In, Inject, Aware,...)	Aware, Inject	10	In, Injecct 10	10	noSoportado	0
4 Encapsuladores (beans)	Funtion OBA		0		20		14
1 Alcance de beans	Allcances bean :Nominal=(NoSportado,SESSION,REQUEST, ..)	noSoportado	0	event, page, conversation, session, bussiness process, stateless	10	Session, Request, Application	4
2 Anotaciones beans	Soportado: Nominal; Soportado=(True,False)	False	0	True	10	True	10
4 Adecuación de la estructura			130		130		120
1 Administración de archivos	Funtion OBA		120		120		120
1 Archivos de configuracion	Funtion OBA		30		30		30
1 Crear archivos configuración	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
2 Modificar archivos configuración	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
3 Eliminar archivos configuración	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10

Funcionalidad

2 Archivos de vista	Funtion OBA		30		30		30
1 Crear archivos vista	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
2 Modificar archivos vista	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
3 Eliminar archivos vista	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
3 Archivos modelo	Funtion OBA		30		30		30
1 Crear archivos modelo	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
2 Modificar archivos modelo	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
3 Eliminar archivos modelo	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
4 Archivos de persistencias	Funtion OBA		30		30		30
1 Crear archivos persistencias	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
2 Modificar archivos persistencias	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
3 Eliminar archivos persistencias	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
2 Perspectiva Integrada IDE	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	False	0
5 Adecuación del soporte			35		32		29
1 Documentación y manuales	Documentacion:Ordinal=(Disponible, parcialmente disponible, no disponible)	Disponible	10	Disponible	10	Disponible	10
2 Foros de usuario	contenido:Ordinal=(NoProvisto, basico, medio, avanzado)	Avanzado	10	Avanzado	10	Medio	7
3 FAQs	contenido:Ordinal=(NoProvisto, basico, medio, avanzado)	Medio	5	Medio	5	Medio	5
4 Manuales	contenido:Ordinal=(NoProvisto, basico, medio, avanzado)	Avanzado	10	Medio	7	Basico	2
5 Show case	contenido:Ordinal=(NoProvisto, basico, medio, avanzado)	noProvisto	0	noProvisto	0	Medio	5
2 Exactitud		5	20	10	40	5	20
1 Verificabilidad			20		40		20
1 Capacidades de registro							
1 Registro de errores	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
2 Registro de eventos	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
3 Registro de trasacciones	Soportado: Nominal; Soportado=(True,False)	False	0	True	10	False	0
2 Historiales de servicios							
1 Bitácora de uso de servicios	Soportado: Nominal; Soportado=(True,False)	False	0	True	10	False	0
3 Interoperabilidad		21,16666667	127	18,33333333	110	15	90
1 Interoperabilidad Directa			77		70		50
1 Protocolos internet							
1 Protocolos web	Protocolos:Set(Label:Nominal); Label=(Http(s),ftp(s)),...)	http, https	10	http, https	10	http, https	10
2 IDE desarrollo							
1 IDES soportados	IDE:SET(Label:Nomina);=Label=(Eclipse, Netbeans,...)	Eclipse, NetBeans, Spring IDE	10	Eclipse	5	Eclipse, NetBeans	8
1 Adecuaciones version	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	False	0
2 Documentacion plugin	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	False	0
3 Datos							
1 Frameworks Persistencias	Persistencias:SET(Label:Nominal);Label=(Hibernate,OpenJpa,...)	Hibernate, JDO, Oracle Toplink, iBatis, Jpa	2	Hibernate, JDO, Oracle Toplink, iBatis, Jpa, EclipseLink, OpenJpa	10	Hibernate	2
2 Conectores bases de datos JDBC	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
4 Herramientas Reporteria							
1 Librerias Reportería	Reportería:set(Labels:Nominal);Labels=((Jasper, Cristal Reports, ..)	Jasper	5	noSoportado	0	noSoportado	0
2 Recursos Reportería	Soportado: Nominal; Soportado=(True,False)	False	0	False	0	False	0
5 Pruebas unitarias							
1 Frameworks Pruebas Unitarias	Pruebas:SET(Label:Nominal);Label=(Junit,...)	Junit	10	Junit	10	Junit	10
6 Servidores aplicaciones web							
1 Servidores soportados	Servidorees:SET(Label:Nominal);Label=(Tomcat,GlshFish,...)	Tomcat, Wass, Appserver, WebLogic, Jboss AS, Websphere, Glashfish	10	Tomcat, Glassfish, Jboss AS	5	Tomcat, WebLogic, Jetty, Sun Application server, Glassfish, Jboss AS, Websphere	10
2 Interoperabilidad Indirecta			50		40		40
1 Navegadores web							
1 Navegadores soportados	Funtion OBA		50		40		40
1 Google Crhome	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
2 Mozilla Fireefox	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
3 Internet Explorer	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
4 Safari	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
5 Opera	Soportado: Nominal; Soportado=(True,False)	True	10	False	0	False	0
4 Seguridad		25	70	25	70	0	0
1 Seguridad de la aplicación			50		50		0
1 Control de usuarios							
1 Login y password	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	False	0
2 Permisos por roles	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	False	0
3 Restricción a no autentificados	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	False	0
4 Protección de páginas por URL	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	False	0
5 Manejo de sesiones	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	False	0
2 Seguridad de los datos			20		20		0
1 Encriptación de datos							
1 Protocolos de encriptacion	Protocolos:Set(Label:Nominal); Label=(md5,token,...)	md5	10	token	10	noSoportado	0
2 Transmición de datos							
1 Protocolos de tranferencia de datos	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
Totales Funcionalidad		80,27192982		89,1754386		49,57894737	
Procentajes Funcion		28,09517544		31,21140351		17,35263158	
5 Cumplimiento Funcional							

2 Confiabilidad								
Confiabilidad	**1 Madurez**		16,91666667	29	28,58333333	49	13,41666667	23
	1 Historial del producto			9		4		3
	1 Tiempo del producto en el mercado	Periodo: Ratio; Period = Float[Años]	9	9	5	4	3	3
	2 Versiones producto y parches	Function OBA						
	1 Versiones estables	Versiones: Ratio;Versiones=Float(Numero)	4	-	6	-	4	-
	3 Deteccion de fallas y correciones	Function OBA						
	1 Fallas detectadas por version	Fallas Set(<Version: Ordinal, numFallas Ratio>); Version=Reliability.Matirity.ProductHystory.V, numFallas = Integer	noDefinido	-	noDefinido	-	noDefinido	-
	2 Robustez			0		0		0
	1 Robustez Preoperacional							
	1 Tiempo medio entre fallos produccion	Periodo: Ratio; Periodo= Float[Hours]	noDefinido	-	noDefinido	-	noDefinido	-
	2 Tiempo medio entre fallos Desarrollo	Periodo: Ratio; Periodo= Float[Hours]	noDefinido	-	noDefinido	-	noDefinido	-
	2 Robustez Operacional							
	1 Tiempo medio entre fallos produccion	Periodo: Ratio; Periodo= Float[Hours]	noDefinido	-	noDefinido	-	noDefinido	-
	2 Tiempo medio entre fallos Desarrollo	Periodo: Ratio; Periodo= Float[Hours]	noDefinido	-	noDefinido	-	noDefinido	-
	3 Transparencia			10		30		10
	1 Navegacion continua tras un fallo	Soportado: Nominal; Soportado=(True,False)	False	0	True	10	False	0
	2 Resolucion de dependencias fallidas en tiempo real	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
	3 Re-instanciación de componente fallidos	Soportado: Nominal; Soportado=(True,False)	False	0	True	10	False	0
	4 Nivel de tolerancia			10		5		10
	1 Numero promedio de compilaciones por error de mer	►Numero_compilacions (Type: Ordinal); Numero_compilaciones(0-50,50-100,100-500,Indefinidas)	Indefinidas	10	60	5	Indefinida	10
	5 Capacidad de recuperación de fallos			0		10		0
	1 Reiniciacion de servidor de aplicaciones ante fallos	Soportado: Nominal; Soportado=(True,False)	False	0	True	10	False	0
	2 Recuperabilidad		37,5	30	50	40	25	20
	1 Recuperacion de sistema			30		40		20
	1 Registro de errores	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
	2 registro de eventos	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
	3 Registro de transacciones	Soportado: Nominal; Soportado=(True,False)	False	0	True	10	False	0
	4 Recuperación automática ante fallos	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	False	0
	Totales Confiabilidad		54,41666667		78,58333333		38,41666667	
	Porcentajes Funcion		5,441666667		7,858333333		3,841666667	
	3 Cumplimiento de Confiabilidad							
3 Usabilidad								
Usabilidad	**1 Comprensibilidad**		26,30769231	114	27,23076923	118	24,23076923	105
	1 Comprension de la metodología			76		83		75
	1 Conprension de la vista							
	1 Mecanismos de comunicación	Comprension:(Labels:Ordinal); Labels=(Comprensible, MedianamenteComprensible, PocoComprensible, GranDificultad)	Comprensible	10	Medianamente Comprencible	8	Medianamente Comprencible	8
	2 Preedictabilidad de uso	Prediccion:(Labels:Ordinall); Labels=(Predicible, MedianamentePredicible, NoPredicible)	Predecible	10	Medianamente predecible	7	Medianamente predecible	7
	3 Mecanismos ajax	Comprension:(Labels:Ordinal); Labels=(Comprensible, MedianamenteComprensible, PocoComprensible, GranDificultad)	GranDificultad	0	Comprensible	10	Comprensible	10
	2 Comprension del modelo							
	1 Metodología de dependencias	Comprension:(Labels:Ordinal); Labels=(Comprensible, MedianamenteComprensible, PocoComprensible, GranDificultad)	Comprensible	10	Comprensible	10	GranDificultad	0
	2 Metodología de encapsuladores (bean)	Comprension:(Labels:Ordinal); Labels=(Comprensible, MedianamenteComprensible, PocoComprensible, GranDificultad)	Medianamente Comprensible	8	Comprensible	10	Comprensible	10
	3 Comprension del controlador							
	1 Metodos controlador	Comprension:(Labels:Ordinal); Labels=(Comprensible, MedianamenteComprensible, PocoComprensible, GranDificultad)	Medianamente Comprencible	8	Medianamente Comprencible	8	Comprensible	10
	2 Resolusores de recursos	Resolusores:(Labels:Ordinal); Labels=(Definidos, ParcialmenteDefinidos, NoDefinidos)	Definidos	10	Definidos	10	Definidos	10
	3 Resolusores de vistas	Resolusores:(Labels:Ordinal); Labels=(Definidos, ParcialmenteDefinidos, NoDefinidos)	Definidos	10	Definidos	10	Definidos	10
	4 Alcance controlador	Comprension:(Labels:Ordinal); Labels=(Comprensible, MedianamenteComprensible, PocoComprensible, GranDificultad)	Comprensible	10	Comprensible	10	Comprensible	10
	2 Estructura Global			38		35		30
	1 Arquitectura bien definida	Funtion OBA		30		30		20
	1 Vista definida	Definido: Nominal; Definido=(True,False)	True	10	True	10	True	10
	2 Modelo definido	Definido: Nominal; Definido=(True,False)	True	10	True	10	True	10
	3 Contolador definido	Definido: Nominal; Definido=(True,False)	True	10	True	10	False	0
	1 Flexibilidad de la estructura de proyectos	Flexibilidad:(Labels:Ordinal);Labels=(Inflexible, MedianamenteFlexible,Flexible,MuyFlexible)	Flexible	8	Medianamente Flexible	5	Muy Flexible	10
	2 Aprendizaje		28,61538462	93	30,15384615	98	24	78
	1 Entrenamiento			27		27		22
	1 Videos demostrativos	Entrenamiento: Ordinal; Entrenamiento=(NoProvisto, Basico, Medio, Avanzado)	Medio	7	Avanzado	10	NoProvisto	0
	2 Soporte online	Entrenamiento: Set(Nivel:Nominal), Level(noProvistosBasico,Medio,Avanzado)	Avanzado	10	Avanzado	10	Medio	8
	3 Tutoriales	Tutorials: Nominal; Tutorial=(Disponible,parcialmenteDisponible,NoDisponible)	Disponible	10	parcialmente Disponible	7	parcialmente Disponible	7
	4 Show case	Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	noProvisto	0	noProvisto	0	Medio	7
	2 Documentación			66		71		56
	1 Documentación del desarrollador							
	1 Documentación y manuales	Documentacion:Ordinal(Disponible, parcialmente disponible, no disponible)	Disponible	10	Disponible	10	Disponible	10
	2 Foros de usuario	Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	Avanzado	10	Avanzado	10	Medio	7
	3 FAQs	Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	Medio	5	Medio	5	Medio	5
	4 Consultas	Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	Basico	3	Avanzado	10	Medio	3
	5 Show case	Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	noProvisto	0	noProvisto	0	Medio	7
	6 Proyectos ejemplos demostrativos	Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	Medio	8	Medio	8	NoProvisto	0

	2 Documentación externa							
	1 Foros	Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	Avanzado	10	Avanzado	10	Medio	8
	2 Ayuda online	Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	Avanzado	10	Avanzado	10	Medio	8
	3 webs dedicadas al soporte	Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	Avanzado	10	Medio	8	Medio	8
3 Operabilidad			23,91304348	220	21,73913043	200	11,95652174	110
1 Configuración				220		200		110
1 Configuración global								
1 Administración del controlador		Function OBA		60		50		40
1 Configuracion del view resolver		Configurable: Nominal; Configurable=(True,False)	True	10	True	10	True	10
1 Patron de URL		Configurable: Nominal; Configurable=(True,False)	True	10	True	10	False	0
2 Pagina de inicio		Configurable: Nominal; Configurable=(True,False)	True	10	True	10	True	10
3 Metodos de acceso		Configurable: Nominal; Configurable=(True,False)	True	10	True	10	True	10
2 Configuracion del resource resolver		Configurable: Nominal; Configurable=(True,False)	True	10	False	0	False	0
3 Configuracion del tipo de mapeo		Configurable: Nominal; Configurable=(True,False)	True	10	True	10	True	10
2 Administracion de beans		Function OBA		40		20		20
1 Habilitacion de anotaciones		Configurable: Nominal; Configurable=(True,False)	True	10	True	10	True	10
2 Configuracion de beans		Function OBA		30		10		10
1 Beans de modelo		Configurable: Nominal; Configurable=(True,False)	True	10	True	10	True	10
2 Beans de reporteria		Configurable: Nominal; Configurable=(True,False)	True	10	False	0	False	0
3 paquetes base Beans		Configurable: Nominal; Configurable=(True,False)	True	10	False	0	False	0
3 Administración de seguridad		Function OBA		70		70		0
1 Mecanismos de login		Soportado: Nominal; Soportado=(True,False)	True	10	True	10	False	0
1 Determinacion login page		Configurable: Nominal; Configurable=(True,False)	True	10	True	10	False	0
2 Configuracion de restricciones		Soportado: Nominal; Soportado=(True,False)	True	10	True	10	False	0
1 Patrones de restricciones		Configurable: Nominal; Configurable=(True,False)	True	10	True	10	False	0
2 Permisos por patrones de paginas		Configurable: Nominal; Configurable=(True,False)	True	10	True	10	False	0
3 Permisos por paginas individuales		Configurable: Nominal; Configurable=(True,False)	True	10	True	10	False	0
4 Permisos por roles		Configurable: Nominal; Configurable=(True,False)	True	10	True	10	False	0
4 Administración persistencias		Function OBA		50		60		50
1 Configuracion de data source		Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
2 Configuracion de persistencias		Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
1 Tipo de trasacciones		Configurable: Nominal; Configurable=(True,False)	True	10	True	10	True	10
2 Lenguajes de consultas		Configurable: Nominal; Configurable=(True,False)	True	10	True	10	True	10
3 Cambios automaticos bd		Configurable: Nominal; Configurable=(True,False)	False	0	True	10	False	0
4 Dialecto de proveedor		Configurable: Nominal; Configurable=(True,False)	True	10	True	10	True	10
4 Atractividad			5	10	5	10	0	0
1 Navegabilidad								
1 Iconos personalizados plugin IDE		Soportado: Nominal; Soportado=(True,False)	True	10	True	10	False	0
Totales Usabilidad			83,8361204		84,12374582		60,18729097	
Porcentajes Funcion			20,9590301		21,03093645		15,04682274	
5 Cumplimiento de usabilidad								
4 Eficiencia								
1 Empleo de Recursos			85	34	77,5	31	85	34
1 Requerimientos				34		31		34
1 Requerimientos Hardware Minimos		Function OBA		16		13		16
1 Memoria RAM		Memoria: Ordinal; Memoria=(0-256,256-512,512-1024,1024-2048,2048>)	256-512	8	512-1024	5	256-512	8
2 CPU		CPU: Ordinal; CPU=(0-1Ghz,1-2Ghz,2-3Ghz,3Ghz>)	1-2Ghz	8	1-2Ghz	8	1-2Ghz	8
2 Requerimientos Software Minimos		Function OBA		8		8		8
2 Requerimientos jdk		Version:(Labels:Nominal); Labels=(1.4,1.5,1.6,1.7)	1.5	8	1.5	8	1.5	8
2 Sistemas operativos		OS:Set(Labels: Nominal); Labels=(Windows, Unix, Linux, ...)	Windows, Linux, Unix, Mac OS	10	Windows, Linux, Unix, Mac OS	10	Windows, Linux, Unix, Mac OS	10
Totales Eficiencia			85		77,5		85	
Porcentajes Funcion			4,25		3,875		4,25	
2 Cumplimiento de eficiencia								
5 Mantenimiento								
1 Capacidad de ser analizado			33	40	33	40	16,5	20
1 Análisis del producto				40		40		20
1 Capacidades de registro								
1 Registro de errores		Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
2 registro de eventos		Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
3 Registro de transacciones		Soportado: Nominal; Soportado=(True,False)	True	10	True	10	False	0
4 Recuperación automática ante fallos		Soportado: Nominal; Soportado=(True,False)	True	10	True	10	False	0
2 Capacidad para ser cambiado			21,15555556	56	27,57777778	73	19,64444444	52
1 Flexibilidad en interfaz				0		10		10
1 Soporte de skins		Soportado: Nominal; Soportado=(True,False)	False	0	True	10	True	10
2 Ambiente IDE Desarrollo				20		20		10
1 IDE personalizado		Soportado: Nominal; Soportado=(True,False)	True	10	True	10	False	0
2 Librerias de desarrollo		Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
3 Documentacion Desarrollo				36		43		32
1 Documentación y manuales		Documentacion:Ordinal(Disponible, parcialmente disponible, no disponible)	Disponible	10	Disponible	10	Disponible	10
2 Foros de usuario		Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	Avanzado	10	Avanzado	10	Medio	7
3 FAQs		Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	Medio	5	Medio	5	Medio	5
4 Consultas		Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	Basico	3	Avanzado	10	Medio	3
5 Showcase		Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	noProvisto	0	noProvisto	0	Medio	7
6 Proyectos ejemplos demostrativos		Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	Medio	8	Medio	8	NoProvisto	0
3 Estabilidad			0	0	0	0	0	0
1 Estabilidad operacional								
1 Tiempo promedio entre liberación de versiones estab		Tiempo: Ratio; Tiempo = Float[meses]	4	-	9	-	10	-
2 Estabilidad Desarrollo								
1 Tiempo promedio entre liberación de versiones beta		Tiempo: Ratio; Tiempo = Float[meses]	2	-	3	-	5	-
4 Testeo			33	20	33	20	16,5	10
1 Frameworks Pruebas Unitarias		Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
2 Herramientas de análisis		Soportado: Nominal; Soportado=(True,False)	False	10	True	10	False	0
Totales Eficiencia			87,15555556		93,57777778		52,64444444	
Porcentajes Funcion			13,07333333		14,03666667		7,896666667	
5 Cumplimiento de mantenimiento								

Portabilidad	6 Portabilidad							
	1 Adaptabilidad		33	70	25,92857143	55	28,28571429	60
	1 Sistemas operativos	OS:Set(Labels: Nominal); Labels=(Window ,Liunux, ...)	Windows, Linux, Unix, Mac OS	10	Windows, Linux, Unix, Mac OS	10	Windows, Linux, Unix, Mac OS	10
	2 Serviores web soportados	servidorees:SET(Label:Nominal);Label=(Tomcat,GlshFish,...)	Tomcat, Wass, Appserver, WebLogic, Jboss AS, Websphere, Glashfish	10	Tomcat, Glassfish, Jboss AS	5	Tomcat, WebLogic, Jetty, Sun Application server, Glassfish, Jboss AS, Websphere	10
	3 Navegadores soportados	Funtion OBA		50		40		40
	1 Google Chrome	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
	2 Mozilla Fireefox	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
	3 Internet Explorer	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
	4 Safari	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
	5 Opera	Soportado: Nominal; Soportado=(True,False)	True	10	False	0	False	0
	2 Instalación		20,9	38	29,15	53	22	40
	1 Facilidades de instalación							
	1 Asistente de instalación	Soportado: Nominal; Soportado=(True,False)	False	0	True	10	False	0
	2 Soporte de instalacion							
	1 Documentación y manuales	Documentacion:Ordinal(Disponible, parcialmente disponible, no disponible)	Disponible	10	Disponible	10	Disponible	10
	2 FAQs	Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	Medio	5	Medio	5	Medio	5
	3 Archivos de ayuda	Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	Medio	5	Avanzado	10	Medio	5
	4 Soporte para creacion de proyectos nuevos	Contenido:Ordinal; Contenido=(NoProvisto, basico, medio, avanzado)	Medio	8	Medio	8	Avanzado	10
	3 Compatibilidad de la plataforma							
	1 Sistemas operativos	OS:Set(Labels: Nominal); Labels=(Window ,Liunux, ...)	Windows, Linux, Unix, Mac OS	10	Windows, Linux, Unix, Mac OS	10	Windows, Linux, Unix, Mac OS	10
	3 Coexistencia		26,18	77	23,8	70	17	50
	2 Por el significado de las APIs(Conectores)			77		70		50
	1 Protocolos internet							
	1 Protocolos web	Protocolos:Set(Label:Nominal); Label=(Http(s),ftp(s)),...)	http, https	10	http, https	10	http, https	10
	2 IDE desarrollo							
	1 IDES soportados	IDE:SET(Label:Nomina);=Label=(Eclipse, Netbeans,...)	Eclipse, NetBeans, Spring IDE	10	Eclipse	5	Eclipse, NetBeans	8
	1 Adecuaciones version	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	False	0
	2 Documentacion plugin	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	False	0
	3 Datos							
	1 Frameworks Persistencias	Persistencias:SET(Label:Nominal);Label=(Hibernate,OpenJpa,...)	Hibernate, JDO, Oracle Toplink, iBatis, Jpa	2	Hibernate, JDO, Oracle Toplink, iBatis, Jpa, EclipseLink, OpenJpa	10	Hibernate	2
	2 Conectores bases de datos JDBC	Soportado: Nominal; Soportado=(True,False)	True	10	True	10	True	10
	4 Herramientas Reporteria							
	1 Librerias Reportería	Reportería:set(Labels:Nominal);Labels=((Jasper, Cristal Reports, ..)	Jasper	5	noSoportado	0	noSoportado	0
	2 Recursos Reportería	Soportado: Nominal; Soportado=(True,False)	False	0	False	0	False	0
	5 Pruebas unitarias							
	1 Frameworks Pruebas Unitarias	Pruebas:SET(Label:Nominal);Label=(Junit,...)	Junit	10	Junit	10	Junit	10
	6 Servidores aplicaciones web							
	1 Servidores soportados	Servidorees:SET(Label:Nominal);Label=(Tomcat,GlshFish,...)	Tomcat, Wass, Appserver, WebLogic, Jboss AS, Websphere, Glashfish	10	Tomcat, Glassfish, Jboss AS	5	Tomcat, WebLogic, Jetty, Sun Application server, Glassfish, Jboss AS, Websphere	10
	Totales Eficiencia		80,08		78,87857143		67,28571429	
	Porcentajes Funcion		8,008		7,887857143		6,728571429	
	4 Cumplimiento de portabilidad							
			79,23498332		90,579086		57,74858131	

Tabla 5.8: Modelo de calidad aplicado.

Como un resumen de los resultados de la aplicación expuestos en la tabla 5.8 del modelo de calidad se presenta el siguiente cuadro:

		Spring	Seam	Richfaces
	Máximo	Porcentajes		
Funcionalidad	**35**	**28.1**	**31.21**	**17.35**
Adecuación	40	29,11	35,84	29.58
Exactitud	10	5	10	5
Interoperabilidad	25	21,17	18,33	15
Seguridad	25	25	25	0
Totales	**100**	**80,28**	**89,17**	**49.58**

Confiabilidad	**10**	**5.42**	**7.86**	**3.84**
Madurez	50	16,92	28,58	13,42
Recuperabilidad	50	37,5	50	25
Totales	**100**	**54,42**	**78,58**	**38,42**
Usabilidad	**25**	**20.96**	**21.03**	**15.05**
Comprensibilidad	30	26,31	27,23	24,23
Aprendizaje	40	28,61	30,15	24
Operabilidad	25	23,91	21,74	11,96
Atractividad	5	5	5	0
Totales	**100**	**83,83**	**84,12**	**60,19**
Eficiencia	**5**	**4.25**	**3.88**	**4.25**
Empleo de recursos	100	84	77,5	85
Totales	**100**	**84**	**77,5**	**85**
Mantenibilidad	**15**	**13.07**	**14.04**	**7.90**
Capacidad para ser analizado	34	24,75	33	16,5
Capacidad para ser cambiado	33	21,16	27,58	19,64
Estabilidad	0	0	0	0
Testeo	33	16,5	33	16,5
23Totales	**100**	**62,41**	**93,58**	**52,64**
Portabilidad	**10**	**8.0**	**7.89**	**6.73**
Adaptabilidad	33	33	25,93	28,29
Instalación	33	20,9	29,15	22
Coexistencia	34	26,18	23,8	17
Totales	**100**	**80,08**	**78,88**	**67,29**
Porcentajes Totales	**100**	**79.83**	**85.9**	**55.12**

Tabla 5.9: Resultados globales de aplicación.

Como se puede observar en la tabla 5.9, Seam es el framework que cumple de mejor manera casi la totalidad de características por lo que se le considera el mejor framework para desarrollar aplicaciones web, ya que ofrece la funcionalidad más completa con referencia a los otros frameworks evaluados, además en otros aspectos importantes como la usabilidad, confiabilidad y mantenibilidad también da claras muestras de ser superior a los otros. Seam Framework es el marco de trabajo ideal para trabajar con aplicaciones robustas, ya que ofrece un modelo con la funcionalidad adecuada, confiable, portable y con gran cantidad de documentación lo que lo hace mucho uno de los más usables de mercado.

Spring Framework pese a no ser el framework que mejor cumple las características del modelo de calidad, es uno de los frameworks más populares y usados en el mercado de frameworks web java, ya que ofrece un entorno de desarrollo con una buena funcionalidad y con una curva de aprendizaje media, lo cual lo hace una muy buena opción a la hora de escoger un framework. Ofrece características flexibles y una metodología que pese a ser bastante trabajosa en la práctica es bastante poderosa a la hora de construir una aplicación web.

5.6 Conclusiones

Un modelo de calidad es una herramienta importante para determinar cuál es el software más conveniente para las necesidades de una empresa/desarrollador, pero para obtener los mejores resultados se debería usar un método de construcción con un enfoque mixto como lo es el método IQMC, este tipo de enfoques es ideal ya que especializan un modelo o estándar como el ISO 9126-1 para ajustarlo a un dominio específico.

Un proceso fundamental para la aplicación de un modelo de calidad es la asignación de porcentajes de importancia a las características y subcaracterísticas, de acuerdo con esto se determina qué características son más relevantes para el dominio específico. Sin la asignación de importancia es imposible determinar cuál es el mejor producto a un nivel general.

Se determina como el mejor framework para la construcción de aplicaciones empresariales web en java a Seam, ya que luego de la aplicación del modelo de calidad se determina que este cumple de la mejor manera con las características, denotando una funcionalidad, confiabilidad, usabilidad y mantenibilidad superiores al resto de frameworks evaluados.

Spring es una buena opción para la construcción de aplicaciones web ya que luego de la aplicación del modelo de calidad, se observa que mantiene porcentajes buenos en áreas importantes para el desarrollador como lo son la funcionalidad y usabilidad. Este framework proporciona un modelo completo para el desarrollo pese a ciertas falencias en la funcionalidad de la vista.

Richfaces luego de la evaluación ha demostrado ser un framework poderoso en la capa de la vista, pero con muchas falencias en el resto de capas. Se aconseja usar este framework solamente en aplicaciones dedicadas exclusivamente a la capa de la vista.

CONCLUSIONES

Todos los objetivos planteados en el diseño de esta tesis se cumplieron.

-En la presente tesis se evidencia que el modelo de calidad ISO 9126-1 se adecua de forma correcta a la evaluación de frameworks desarrollados en java, orientado a aplicaciones empresariales, ya que cumple con características deseadas como son:

- Ser un modelo mixto.
- Disfrutar de una gran cantidad de documentación.
- Tener una estructura jerárquica con características y subcaracterísticas definidas.
- Profundizar aspectos de funcionalidad, usabilidad, confiabilidad, eficiencia, mantenibilidad y portabilidad.
- Poseer una organización que respalda y actualiza el modelo.

-El mejor framework entre los evaluados en la presente tesis con el estándar ISO 9126-1 es Seam; debido a que, luego de aplicar el modelo de calidad obtuvo un porcentaje de cumplimiento total de 85.9%, superando a los otros en los siguientes aspectos: Funcionalidad, Usabilidad, Confiabilidad y Mantenibilidad.

-El software para reportería JasperReports 4.7 no se ejecuta correctamente en versiones superiores a la JDK 1.6.24 de Linux.

- El framework de persistencias Hibernate proporciona facilidad en la manipulación de datos, pero dificulta la construcción de consultas complejas, debido a que utiliza HQL como remplazo de SQL.

-El uso de Apache Maven como herramienta de construcción y compilación facilita el desarrollo de aplicaciones en Spring, ya que maneja una metodología de dependencias agilizando los cambios de versiones y el manejo de librerías.

-El desarrollar en un framework web java requiere conocer de ciertas herramientas y paradigmas tales como:

- Tener conocimiento medio-alto del lenguaje JAVA.
- Comprender la arquitectura MVC.
- Conocer el nivel de permisos que el framework requiere en cuanto al SO
- Entender el funcionamiento de herramientas de compilación y construcción tales como Apache Ant y Apache Maven

TRABAJO FUTURO

Como se ha podido evidenciar en esta tesis los frameworks web es un dominio de trabajo bastante extenso por lo que se puede considerar muchas líneas para un futuro trabajo, entre algunas se pueden mencionar las siguientes:

- La proliferación de tecnologías para la capa de presentación ha llegado a tener una gran importancia dentro del desarrollo móvil, en particular los frameworks y librerías de componentes visuales; estos adquieren importancia y son desarrollados en una variedad de herramientas y lenguajes, por lo que una evaluación de los mismos es un tema de gran interés.
- La identificación y descripción detallada de los actuales frameworks para el mapeo de objetos relacionales es importante; debido a que, actualmente la mayoría de aplicaciones java se ven potenciadas por estas herramientas.
- Analizar robustez, confiabilidad y disponibilidad de las aplicaciones desarrolladas con los frameworks descritos ayudaría a descubrir cuáles son las capacidades de los mismos a largo plazo.
- El desarrollo de un documento que presente las estrategias más eficaces para agilizar la selección de software para desarrollo, pudiendo este incluir aspectos de modelos de calidad así como de una base de conocimientos basados en la experiencia.

BIBLIOGRAFIA

1. **Johnson, Rod.** *Introduction to Spring.* 2003.

2. **S/A.** tu maestro web. [Online] [Citado: 02 01, 2012.] http://www.tumaestroweb.com/curiosidades/que-es-mvc/.

3. **Walls, Craig.** *Spring in Action.* s.l. : Manning, 2007.

4. **Filocamo, Demetrio.** *JBoss RichFaces 3.3.* Birmingham : Inglaterra, 2009.

5. **Loor, José Miguel.** *Java Server Faces.* Quito : s/editorial, s/año.

6. **oracle.** the j2ee tutorial. [Online] [Citado: 02 18, 2012.] http://docs.oracle.com/javaee/1.4/tutorial/doc/JSFIntro10.html.

7. **s/n.** RichFaces Developer Guide. *RichFaces.* [Online] s/a. www.richfaces.org.

8. **Farley, Jim.** *Practical JBoss Seam.* New York : Apress, 2007.

9. **QSOS.** QSOS. [Online] [Citado: 1 25, 2011.] www.qsos.org.

10. **Carvallo, Juan Pablo.** *Systematic Construction of Quality Models for COST-BASED Systems.* 2005.

11. **Scalone, Fernanda.** *Estudio Comparativo de los modelos y estandares de calidad de software.* Buenos Aires : s.n., 2006.

12. **Ramirez, Paola and Ramirez , Carolina.** Estudio de las prácticas de calidad del software implementadas en la mipymes desarrolladores de software de Pereira. Pereira : s.n., 2010.

13. **ISO and IEC.** *ISOIEC 9126-1.* 2001.

14. **Houman, Younessi.** *Achieving Quality Goals.* 2002.

15. **S/A.** zona programacion. [Online] [Citado: 04 06, 2012.] http://zonazeroradio.nixiweb.com/uml/?p=132.

16. **Pintado, Pablo.** *Ingeniería Web.* Cuenca : s.n., 2012.

17. **Comunity, JBoss.** Hibernate. *Hibernate.* [Online] [Citado: 09 01, 2012.] http://www.hibernate.org/.

18. **JasperSoft.** Jasper Reports. *Jasper Reports.* [Online] [Citado: 08 10, 2012.] http://community.jaspersoft.com/project/jasperreports-library.

19. **Rodriguez, Daniel.** *Medición en la Ingeniería del Software.* Madrid : Universidad deAlcala, 2003.

20. **Erazo, Lenin.** *Especificación de Requisitos Ej. Bodega.* Cuenca : s.n., 2011.

21. **java-source.** java-source.net. [Online] [Citado: 12 10, 2011.] http://java-source.net/.

22. **konda, Madhusudhan.** *Just Spring.* Gravenstein Higheay : O'Really Media, Inc, 2011.

23. **JBoss_Richfaces_Reference.** *Richfaces Reference.*

24. **Seam_Framework.** *Introductio to JBoss Seam.*

25. **Ortiz, Edison and Herrera, Francisca.** Guía para la aplicación de metricas para determinar la calidad de un sistema de software. Quito : s.n., 2010.

28. **Redondo, Manuel.** *MANUAL DE CALIDAD Y PROCEDIMIENTOS PARA LA GESTIÓN DEL SISTEMA DE CALIDAD DE UNA EMPRESA DE DESARROLLO DE SOFTWARE.* sa.

ANEXOS

Anexo 1: Hoja de evaluación del software. Método de evaluación QSOS

Durabilidad Intrínseca		Puntuación		
		0	1	2
Madurez	Edad	Menos de tres meses	Entre tres meses y un año	Más de tres Años
	Estabilidad	Software inestable con muchas versiones	Software estable con versiones pero viejas	Software estable. Las nuevas versiones tienen errores corregidos pero en especial nuevas funcionalidades
	Historial de problemas	Muchos problemas que pueden ser prohibitivos	No se le conoce grandes problemas o crisis	Historial de buena administración de situaciones criticas
	Probabilidad de bifurcación	El software muy probablemente se expandirá en otro	El software proviene de otro, pero hay pocas posibilidades de que se expanda en otro	El software tiene muy pocas posibilidades que de bifurque en otro
Adopción	Popularidad	Muy pocos usuarios identificados	Uso detectable en el internet	Numerosos usuarios y numerosas referencias
	Referencias	Ninguna	Pocas referencias, no es usos importantes	Implementado frecuentemente en aplicaciones críticas
	Calidad de la comunidad	No existe comunidad o con muy poca actividad	Existe comunidad con notable actividad	Gran comunidad, mucha actividad en foros, muchos contribuyentes
	Libros	No existen libre del software	Menos de 5 libros disponibles	Más de 5 libros y en muchos lenguajes
Estilo de liderazgo	Tamaño del equipo de desarrollo	1 o 2 individuos involucrados	entre 2 y 5 personas independientes	Más de 5 personas

	Estilo de la Administración del equipo	Completa dictadura	Despotismo culto	Consejo de desarrollo con un líder identificado
Actividad	Numero de Desarrolladores	Menos de 3 desarrolladores no identificados claramente	Entre 4 y 7 desarrolladores, o más no identificados.	Más de 7 desarrolladores claramente identificados, equipo estable
	Actividad en Errores	Poca actividad en foros o ninguna nota que arregle el error	Actividad detectada pero sin un proceso claramente expuesto	Alta actividad, basada en roles y asignación de tareas
	Actividad en funcionalidad	Ninguna o pocas funcionalidades	Evolución del producto introducido por el equipo de desarrollo pero sin un proceso formal.	Herramientas para administras peticiones de características, alta interacción con el roadmap
	Actividad en liberaciones de nuevas versiones	Poca actividad en equipos de producción y desarrollo	Actividad en producción y desarrollo, frecuentemente lanzamientos menores (parches de errores)	Actividad importante el lanzamientos menores y lanzamientos mayores planeados en relación con el roadmap

Solución Industrializada		Puntuación		
		0	1	2
Servicios	Formación	No existen ofertas de formación	Existe oferta pero está restringida geográficamente y solo en un lenguaje	Existe y es provista por muchas empresas, en varios lenguajes y dividen el aprendizaje en módulos

	Soporte	No ofrecen soporte, excepto vía foros	Existe soporte pero es provisto únicamente por una empresa, sin el compromiso necesaria del servicio	Múltiples proveedores del servicio con gran compromiso
	Consultoría	No ofrece servicio de consultoría	Existe oferta pero está restringida geográficamente y solo en un lenguaje	Proveedores de consultoría de diversas empresas y con varios lenguajes.
Documentación	Documentación	No existe documentación	Existe documentación pero esta desactualizada y en un solo lenguaje	La documentación esta siempre a la fecha y posiblemente adaptada para distintos usuarios
Método de calidad	Aseguramiento de calidad	No existe aseguramiento de calidad	Existe un proceso de calidad pero informal y sin herramienta	Proceso automatizado de calidad con publicación de resultados
	Herramientas	No existe herramienta para administración	Herramientas estándar	Herramientas especializadas

Disponibilidad de plataformas		Puntuación		
		0	1	2
Paquetes	Windows	No puede ser instalado en Windows	Existe un puerto para la instalación con problemas	Windows es soportado y existe paquete para la instalación
	Debian/Ubuntu	No está empaquetado para Debian/Ubuntu	Existe un puerto para la instalación con problemas	El software esta empaquetado para la distribución
	Red Hat/Fedora	No está empaquetado para Red Hat/Fedora	Existe un puerto para la instalación con problemas	El software esta empaquetado para la distribución

	Suse	No está empaquetado para Suse	Existe un puerto para la instalación con problemas	El software esta empaquetado para la distribución
	Mandriva	No está empaquetado para Mandriva	Existe un puerto para la instalación con problemas	El software esta empaquetado para la distribución
	Mac OS X	No está empaquetado para Mac OS X	Existe un puerto para la instalación con problemas	El software esta empaquetado para la distribución
	Solaris	No está empaquetado para Solaris	Existe un puerto para la instalación con problemas	El software esta empaquetado para la distribución

Adaptabilidad Técnica		Puntuación		
		0	1	2
Modularidad	Modularidad	Software Estructurado sin módulos	Presencia de módulos de alto nivel permitiendo adaptación en el primer nivel	Concepción modular, permite una fácil adaptación del software seleccionando modulo o desarrollando nuevos
	Facilidad de extensión de código	No provee facilidad de extensión	Se puede extender el código con dificultad	Arquitectura plug-in, diseñado para extensiones dinámicas

Estrategia		Puntuación		
		0	1	2
Licencia	Permisividad	Licencia muy estricta como l aGPL	Licencia moderadamente permisiva	Licencia muy permisiva como la Apache
	Protección contra extensiones del framework	Licencia muy permisiva como la Apache	Licencia moderadamente permisiva	Licencia muy estricta como l aGPL

Copyright	Dueños de Copyright	Derechos mantenidos por pocos individuos, los que fácilmente podrían cambiar la licencia	Derechos mantenidos por numerosos individuos, dueños del código en una forma homogénea	Derechos mantenidos por una entidad legal, de la cual la comunidad confía.
Modificación del código fuente	Modificación del código fuente	No existe forma práctica de modificar el código	Provee herramienta para modificar el código pero no es la misma usada para el desarrollo	El proceso de modificación del código es bien definido, expuesto y respetado
RoadMap	RoadMap (mapa de versiones y liberaciones)	Sin publicaciones de roadmap	existe un roadmap pero sin la debida planificación	Roadmap con las versiones, planificado y con detalles de medición
Patrocinador	Patrocinador	El software no tiene patrocinador, el equipo desarrollador no es pagado	El software solo tiene un patrocinador que puede determinar su estrategia	El software es patrocinado por la industria.
Independencia estratégica	Independencia estratégica	No se encuentra una estrategia para el desarrollo	Visión estratégica compartida con otros productos open.-source	Alta independencia del equipo de desarrollo una entidad legal sostiene los derechos.

Industrialización del Desarrollo		Puntuación		
		0	1	2
Herramientas	IDE	No tiene IDE para el Desarrollo	Tiene un IDE definido para el Desarrollo	Existe una gran cantidad de IDEs para el desarrollo

	Herramientas de Construcción	No cuenta con herramientas que simplifiquen la construcción del aplicativo	Cuenta con pocas herramientas para la construcción del aplicativo	Cuenta con herramientas de construcción e integración con otros marcos de trabajo
	Herramientas de Pruebas	So soporta la integración de herramientas de pruebas	Soporta integración de herramientas de pruebas	Soporta integración de herramientas de pruebas e incluye su propia herramienta de pruebas
	Herramienta gráfica para el desarrollo de interfaces	Interfaces desarrolladas sin herramientas	Posee una herramienta para interfaces con pocos componentes	Herramienta de interfaces con gran número de componentes disponibles

Anexo 2: Descripción de casos de uso

Para el correcto entendimiento de los casos de uso especificados se manejarán información descriptiva de los mismos, tomando en cuenta en una primera instancia el manejo de prioridades bajo la siguiente tabla:

Prioridad	Descripción
Existente	El caso de uso es una extensión de otro sistema ya implementado
Manual	La funcionalidad se realiza manualmente y no se registra inherentemente en el sistema.
Opcional	Se especificará explícitamente por parte de la persona responsable si se automatizará las funcionalidades, bajo la supervisión del responsable del control de aceptación de los ERS.
Deseable	El responsable del control de aceptación de los ERS definirá la automatización de estas funcionalidades
Necesario	Estas funcionalidades podrían ser implementadas de diferentes formas
Obligatorio	Estas funcionalidades serán automatizadas 100%

Priorización de casos de uso (20)

Caso de uso 1	Mantenimiento de clientes
Actor	Usuario
Descripción	Realizar un ingreso modificación y eliminación de cada cliente
Prioridad	Necesario
REQUISITOS ASOCIADOS	
R 1.1 El cliente debe ingresarse con los siguientes datos como obligatorios: -Cedula de identidad: Documento único de identificación -Nombre: Nombres del cliente -Apellidos: Apellidos del cliente -Dirección: Dirección de domicilio -Teléfono: Teléfono del cliente -Fecha de nacimiento R1.2 Para la modificación y eliminación de un cliente es necesario los siguientes campos como obligatorios -Cedula o id del cliente -Razón de la eliminación del cliente	

Caso de uso 2	Mantenimiento de productos
Actor	Usuario
Descripción	Realizar un ingreso, modificación de productos
Prioridad	Necesario
REQUISITOS ASOCIADOS	
R 2.1 Para el ingreso de un nuevo producto deben considerarse los siguientes datos como obligatorios -Nombre del producto -Descripción del producto -Unidad de medida -Precio de la unidad -Cantidad de la unidad R 2.2 Para la modificación de un producto es necesario el siguiente campos -id del producto	

Caso de uso 3	Generación de Facturas
Actor	Usuario
Descripción	Realizar un ingreso, modificación de productos
Prioridad	Obligatorio
REQUISITOS ASOCIADOS	
R 3.1 Para la generación de facturas son necesarios los siguientes requisitos -Estar registrado en el sistema -Haber seleccionado un producto de la base de datos -Seleccionar el cliente a facturar R 3.2 Si el cliente no existe en la base debe ingresarlo R 3.3 Un campo obligatorio en la factura es la cantidad del producto a vender	

Caso de uso 4	Anulación de Facturas
Actor	Usuario
Descripción	Anular una factura cuando el cliente requiera alguna devolución o hubo una equivocación del usuario
Prioridad	Necesario
REQUISITOS ASOCIADOS	

R 4.1 La factura que se desea anular debe estar ingresada en el sistema R 4.2 Se debe tener el número de factura a anula R 4.3 El usuario debe ingresar la razón por la cual anula la factura

Caso de uso 5	Ingreso de Factura de compra
Actor	Usuario
Descripción	Al momento de comprar alguna cantidad de un producto es necesario ingresarlo al sistema con su correspondiente factura de compra
Prioridad	Obligatorio
REQUISITOS ASOCIADOS	
R 5.1 El producto que se compra debe constar en la base de datos R 5.2 El número de factura de compra es un campo obligatorio R 5.3 Se deben ingresar las cantidades del producto que se está comprando	

Caso de uso 6	Generar reporte de clientes
Actor	Usuario
Descripción	Un reporte de clientes es generado cuando se necesita información para la administración del negocio
Prioridad	Deseable
REQUISITOS ASOCIADOS	
R 6.1 La tabla de clientes debe contar con registros R 6.2 El reporte debe ser usado por el administrador del negocio	
R 6.3 El reporte debe contar con datos como - Cedula, Nombre, Apellido - Fecha última compra -Monto total de compras	

Caso de uso 7	Generar reporte de productos
Actor	Usuario
Descripción	Un reporte de productos es generado cuando se necesita información para la administración del negocio, en cuanto a productos con sus cantidades y precio
Prioridad	Deseable
REQUISITOS ASOCIADOS	
R 7.1 La tabla de productos debe contar con registros R 7.2 El reporte debe ser usado por el administrador del negocio R 7.3 El reporte debe contar con datos como - Nombre y descripción del producto - Cantidades de los productos -Monto total de inversión en cada producto	

Caso de uso 8	Generar reporte de cuadre de caja
Actor	Usuario
Descripción	El reporte de cuadre de caja, concilia el dinero que tiene el usuario al final del día físicamente en caja, con el que debería tener de acuerdo a las ventas ingresadas en el sistema
Prioridad	Necesario
REQUISITOS ASOCIADOS	
R 8.1 El reporte de cuadre de caja debe tener los datos de la fecha inicial y final R 8.2 El usuario que realice el reporte debe tener permisos en el sistema	

Caso de uso 9	Generar reporte de facturas
Actor	Usuario
Descripción	Un reporte de facturas, es un reporte que representa las ventas que ha tenido la empresa en un cierto rango de fechas dado por el usuario, es usado por administración
Prioridad	Necesario

REQUISITOS ASOCIADOS
R 9.1 El reporte de cuadre de caja debe tener los datos de la fecha inicial y final R 9.2 El reporte de facturas tiene que tener en cuenta los siguientes aspectos -Número de factura -Productos vendidos con cantidad -Fecha de factura -Datos del cliente - Total de la factura

Anexo 3: Manual de desarrollador Seam Framework

Un manual completo suele ser una herramienta esencial para el desarrollador cuando necesita comenzar a usar un framework, ya que se necesitan de los lineamientos básicos en donde se expliquen cómo implementar ya en forma de código cada uno de los conceptos y constructos teóricos que se han estudiado. De esta forma a través de este manual se intentará transmitir los conceptos de aplicación más importantes del que se ha determinado como el mejor framework para la construcción de aplicaciones web orientados a la empresa.

Instalación de Seam Framework

El entorno del cual se describirá la instalación es en un sistema Linux Ubuntu 11.10, la arquitectura de hardware para el presente caso no es trascendente. Los siguientes son los pasos para la instalación de Jboss Seam desde cero. Descargar los siguientes programas y almacenarlos en la carpeta /home/pcLinux/Desarrollo:

Programa	Versión	URL
sun-java6-jdk	1.6 u35	http://www.oracle.com/technetwork/java/javase/downloads/index.html
eclipse	Indigo 3.7.2 SR2	http://www.eclipse.org/downloads/
jboss AS	7.1.1 GA	http://www.jboss.org/jbossas/downloads/
jboss tools	3.3.1 GA	https://www.jboss.org/tools/download/stable.html
jboss Seam	2.3 GA	http://Seamframework.org/Download
apache ant	1.8.4	http://maven.apache.org/download.html
apache maven	3.0.4	http://ant.apache.org/bindownload.cgi

Hay que tomar en cuenta que las versiones de java, apache ant y apache maven, pueden cambiar hacia versiones superiores sin ningún problema para la producción de Seam. Una vez descargados los paquetes los pasos para la instalación son los siguientes:

1) Abrir una ventana de comandos, y teclear

    cd /home/pcLinux/Desarrollo/

2) Descomprimir Eclipse con el comando:

    tar -xzvf eclipse-jee-indigo-SR2-linux-gtk.tar.gz

Moverlo a la ruta destino /opt con el comando:

sudo mv eclipse /opt

3) Descomprimir JBoss Application Server con el comando:

unzip jboss-5.1.0.GA-jdk6.zip

Moverlo a la ruta destino /opt con el comando:

sudo mv jboss-5.1.0.GA /opt

4) Descomprimir JBoss Seam con el comando:

tar -xzvf jboss-Seam-2.2.0.GA.tar.gz

Moverlo a la ruta destino /opt con el comando:

sudo mv jboss-Seam-2.2.0.GA /opt

5) Descomprimir Apache Ant con el comando:

tar -xzvf apache-ant-1.8.4-bin.tar.gz

Moverlo a la ruta destino /opt con el comando:

sudo mv apache-ant-1.8.4 /opt

6) Descomprimir Apache Maven con el comando:

tar -xzvf apache-maven-3.0.4-bin.tar.gz

Moverlo a la ruta destino /opt con el comando:

sudo mv apache-maven-3.0.4 /opt

7) Crear las variables de entorno para JAVA_HOME y ANT_HOME, para ello abrir el archivo profile con:

sudo gedit /etc/profile

Agregar al final los comandos de exportación y uso de esas respectivas variables como a continuación:

export JAVA_HOME=/usr/lib/jvm/jdk1.6.0_35

export PATH=$JAVA_HOME/bin:$PATH

export ANT_HOME=/opt/apache-ant-1.8.4

export PATH=$ANT_HOME/bin:$PATH

export MAVEN_HOME=/opt/apache-maven-3.0.4

export PATH=$MAVEN_HOME/bin:$PATH

Guardar el archivo y a continuación abrir una consola de comandos, en caso de que en dicha consola saliera algún error es porque hay errores de sintaxis en las líneas alteradas.

8) Crear una carpeta /home/pcLinux/projects, que será el lugar en donde se creen los proyectos.

9) Abrir eclipse desde la ruta /opt/eclipse/eclipse con Nautilus, y cuando pida carpeta de workspace, especificar la creada en el punto anterior (/home/pcLinux/projects)

10) Una vez abierto eclipse pulsar el botón de la esquina superior derecha que dice "Workbench"

11) Abrir del menu Help -> Install New Software, en esa pantalla pulsar el boton "Add", en Name dititar: "JBoss Tools 3.3.1 GA File", y en location ubicar el archivo desde la carpeta home/pcLinux/Desarrollo, pulsar "Add", en la siguiente ventana seleccionar todas las opciones y pulsar "Next", se procesaran los prerrequisitos, luego pulsar nuevamente "Next", aceptar la licencia y pulsar "Finish" y cuando termine reiniciar Eclipse.

12) Una vez abierto eclipse Abrir la Perspectiva "Seam", en ella en la ventana "Servers" dar click derecho y elegir New -> Server. En "Server Type", elegir JBoss Community -> JBoss AS 7.1 y pulsar "Next", en la siguiente pantalla elegir el Home Directory de JBoss AS como "/opt/jboss-7.1.1.GA" y elegir en configuración el perfil "Default", pulsar "Next" y luego "Finish"

Creación de un nuevo proyecto Seam

Seam posee un asistente para la creación de proyectos que facilita en gran manera los labores de desarrollo, este puede funcionar de tal manera que puede crear de forma dinámica las persistencias de la base de datos, la pantalla inicial de la aplicación, un proceso de autentificación de usuarios simple y por último los mantenimientos de todas las tablas de la base de datos. Para la realización de todas estas labores hay que especificar una serie de opciones en el asistente de creación de nuevos proyectos de Seam.

1) Correr el asistente de configuración de Seam

cd /opt/jboss-Seam-2.3.GA/

sh Seam setup

2) Utilizar esta configuración

Ruta Proyectos: /home/pcLinux/projects

Ruta JBoss AS: /opt/jboss-7.1.1.GA/

Dominio JBoss AS: default

Ruta GlasFish: Enter

Dominio GlasFish: Enter

Project Name: factSeam

Usar Icefaces en vez de Richfaces: n

Skin Richfaces: classic

Formato Empaquetado: war

Paquete para Java Clases: com.mydomain.factSeam

Paquete para Session Beans: com. factSeam.controllers

Paquete para los Entity Beans: com. factSeam.models

Paquete para Test Cases: com. factSeam.test

Ruta driver JDBC: /opt/jboss-as-7.1.1.Final/standalone/lib/postgresql-8.3-603.jdbc3.jar

Dialecto Hibernate: org.hibernate.dialect.PostgreSQLDialect

JDBC Class Driver: org.postgresql.Driver

JDBC DataSource Class: org.postgresql.jdbc3.Jdbc3ConnectionPool

JDBC URL: jdbc:postgresql://localhost:5432/nombre_base

Usuario: postgres

Clave: clave_postgres

Esquema BD: public

Catalogo BD: nombre_base

Utilizar tablas desde BD: y

Configuraciones iniciales

El desarrollador debe tener en cuenta que al crear un proyecto Seam, este viene pre configurado para que cualquier cambio que se realice en las persistencias afecte directamente a la base de datos, es decir si se borra algún atributo en alguna persistencia este cambio se replicará automáticamente. Estos cambios no son deseados generalmente por lo que hay que realizar cambios en dos archivos: Persistence-dev.xml y Persistence-prod.xml, en estos archivos hay que buscar la siguiente línea: <property name="hibernate.hbm2ddl.auto" value="update"/> y cambiarla por: <property name="hibernate.hbm2ddl.auto" value="none"/> de esta manera se modifica la conducta de replicación automática que puede resultar muy problemática.

Mecanismos de la vista

Seam Framework como ya se ha nombrado en capítulos anteriores es un framework que tiene una gran riqueza en la vista ya que integra soluciones JSF, tales como Richfaces y IceFaces, es por esto que en la vista se pueden usar gran cantidad de componentes, los siguientes se detallan como más importantes para el uso de los mecanismos de la vista:

Menús: Los menús en Seam pueden tener dos aproximaciones ya que hay dos mecanismos posibles para esto, se las puede usar desde Richfaces o IceFaces, por defecto Seam usa los menus del primero de la siguiente manera:

Menús Seam

```
<rich:toolbarGroup>
      <h:outputText value="FacturaciÃ³n Seam :"/>
      <s:link id="menuHomeId" view="/home.xhtml" value="Inicio" propagation="none"/>
   </rich:toolbarGroup>
   <rich:dropDownMenu showDelay="250" hideDelay="0" submitMode="none">
      <f:facet name="label">Modulos Usuario</f:facet>}
   <rich:menuItem>
        <s:link view="/FacturaCompra.xhtml"
              value="Factura Compra"
              id="FacturaCompraId"
                    includePageParams="false"
              propagation="none"/>
    </rich:menuItem>
        <rich:menuItem>
        <s:link view="/ClientesList.xhtml"
              value="Clientes"
              id="ClientesId"
                    includePageParams="false"
              propagation="none"/>
    </rich:menuItem>
        <rich:menuItem>
        <s:link view="/ProveedoresList.xhtml"
```

```
                value="Proveedores"
                id="ProveedoresId"
                        includePageParams="false"
                propagation="none"/>
    </rich:menuItem>
</rich:dropDownMenu>
```

Layouts: Seam usa los layouts de la misma manera de como se la realiza en Richfaces; es decir, se tiene una sola vista con la plantilla de la aplicación, el resto de vistas hereda del layout principal.

Layouts

Inclusión de un menú en el layout

```
<ui:include src="menu.xhtml">
        <ui:param name="projectName" value="factSeam"/>
     </ui:include>
```

Inclusión del layout el encabezado de las vistas

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
   xmlns:s="http://jboss.org/schema/Seam/taglib"
   xmlns:ui="http://java.sun.com/jsf/facelets"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:rich="http://Richfaces.org/rich"
   template="layout/template.xhtml">

<ui:define name="body">

</ui:define>
   <rich:panel>
      <f:facet name="header">Factura Compra</f:facet>

      <s:decorate id="id" template="layout/display.xhtml">
         <ui:define name="label">Id</ui:define>
         <h:outputText value="#{cabeceraCompraHome.instance.id}"/>
      </s:decorate>
    </rich:panel>
</ui:composition>
```

En la vista Seam provee de algunos componentes propios creados con el objetivo de aumentar funcionalidad a la vista, de esta manera tenemos algunos de los más importantes:

Componente	Funcionalidad
S:button	Etiqueta que soporta propagación de conversaciones Seam
S:conversationPropagation	Permite modificar la propagación de una conversación en un botón
S:cache	Etiqueta que instruye a Seam a almacenar en cache un fragmento de una página
S:convertDateTime	Permite convertir una fecha en una variedad de formatos definidos

	por el desarrollador
S:decorate	Etiqueta que rodea a un JSF y permite desplegar un mensaje de error luego de una validación.
S:div	Permite desplegar un div dinámico, bajo alguna condición dada.
S:fragment	Etiqueta que permite renderizar sin ninguna salida HTML
S:label	Etiqueta asociada con un JSF cercano
S:link	Permite incluir parámetros propios de Seam al ejecuta una acción.
S:Message	Despliega un mensaje en la vista
S:validate	Valida un campo de entrada JSF
S:validateAll	Valida un conjunto de campos de entrada JSF

De igual forma existen una gran variedad de elementos JSF, Richfaces, IceFaces que pueden ser usados en la vista, que por su gran cantidad no serán descritos en este manual.

Mecanismos del modelo

El modelo puede estar definido por dos tipos de manejo de datos, el manejo mediante un entityManager el cual es el manejador por defecto de JPA y por un mecanismo de implementación de clases propias de Seam, llamadas entityHome y entityQuery. Las dos últimas se implementas de forma muy sencilla:

EntityQuery

```
{
private static final String EJBQL = "select clientes from Clientes clientes";
private Clientes clientes = new Clientes();

        public ClientesList() {
                setEjbql(EJBQL);
                setMaxResults(25);
        }
}
```

Las consultas son sencillas y como se observa solamente se necesita de una consulta en lenguaje HQL, este es un lenguaje de consulta orientado a objetos que puede facilitar en gran medida el manejo.

Entity Home

```
@Name("clientesHome")
public class ClientesHome extends EntityHome<Clientes> {
        @Override
        protected Clientes createInstance() {
                Clientes clientes = new Clientes();
                return clientes;
```

```
        }
        public void load() {
                if (isIdDefined()) {
                        wire();
                }
        }
        public void wire() {
                getInstance();
        }
        public boolean isWired() {
                return true;
        }
        public Clientes getDefinedInstance() {
                return isIdDefined() ? getInstance() : null;
        }
}
```

Hay que recordar que esta clase extiende a la clase EntityHome provista por Seam, por lo que el manejo de datos no es necesario repetirlo debido a que esto está implementado de forma implícita, de la siguiente manera:

EntityHome

```
public class EntityHome<E> extends Home<EntityManager, E>
{
  private static final long serialVersionUID = -3140094990727574632L;

@Transactional
  public String update()
  {
    joinTransaction();
    getEntityManager().flush();
    updatedMessage();
    raiseAfterTransactionSuccessEvent();
    return "updated";
  }
@Transactional
  public String persist()
  {
    getEntityManager().persist( getInstance() );
    getEntityManager().flush();
    assignId( PersistenceProvider.instance().getId( getInstance(), getEntityManager() ) );
    createdMessage();
    raiseAfterTransactionSuccessEvent();
    return "persisted";
  }
@Transactional
  public String remove()
  {
    getEntityManager().remove( getInstance() );
    getEntityManager().flush();
```

```
    deletedMessage();
    raiseAfterTransactionSuccessEvent();
    return "removed";
  }
```

El desarrollo de los datos de la aplicación mediante EntityManager es sencillo; debido a que, existen unas pocas instrucciones con las que se pueden manejar las acciones de mantenimiento.

EntityManager

Ingreso de dato nuevo
Crear un objeto cliente con sus valores y luego:

```
    entityManager.persist(Cliente);
  entityManager.flush();
```

Para modificar un registro se lo hace mediante las mismas instrucciones
Para eliminar un registro:

```
    entityManager.remove(Cliente);
  entityManager.flush();
```

Contextos de aplicación

Al desarrollar una aplicación hay que recordar que Seam tiene una gran cantidad de distintos contextos que si bien suelen estar implícitos en las llamadas que el framework hace de las clases, hay veces que estos tienen que estar declarados por medio de anotaciones, los sontextos básicos de Seam son:

- Stateless context
- Event context
- Page context
- Conversation context
- Session context
- Business process context
- Application context

Manejo de seguridad

La seguridad es implementada y manejada principalmente mediante anotaciones y archivos de configuración, en una primera instancia se definen dentro de las persistencias los valores de seguridad que el framework deberá tomar:

Seguridad sobre archivos de persistencias

Persistencia CuentasUsuarios

```
@UserPrincipal
@Column(name = "user_name", unique = true, nullable = false)
@NotNull
public String getUserName() {
```

```
        return this.userName;
}
@UserPassword
@Column(name = "password", nullable = false)
@NotNull
public String getPassword() {
        return this.password;
}
@PasswordSalt
@Column(name = "password_salt", nullable = false)
@NotNull
public String getPasswordSalt() {
        return this.passwordSalt;
}
@UserEnabled
@Column(name = "estado", nullable = false)
public boolean isEstado() {
        return this.estado;
}
@UserRoles
@OneToMany(fetch = FetchType.LAZY, mappedBy = "cuentasUsuario")
public Set<RolesCuentaUsuario> getRolesCuentaUsuarios() {
        return this.rolesCuentaUsuarios;
}
```

Persistencia Roles

```
@RoleConditional
@Column(name = "condicional")
public Boolean getCondicional() {
        return this.condicional;
}
@RoleName
@Column(name = "nombre", nullable = false)
@NotNull
public String getNombre() {
        return this.nombre;
}
```

Persistencia Permisos

```
@PermissionAction
public String getAccion() {
        return this.accion;
}
@PermissionDiscriminator
public String getDiscriminator() {
        return this.discriminator;
}
@PermissionUser
@PermissionRole
public String getRecipient() {
        return this.recipient;
}
```

```
@PermissionTarget
public String getTarget() {
        return this.target;
}
```

La siguiente instancia de la seguridad se define al ingresar nuevos usuarios en la base de datos, debido a que la contraseña de usuario es almacenada con una contraseña encriptada, basada en un protocolo de encriptación por llave.

Ingreso de usuarios

```
public String guardarUsuario()
{
        if (password.equals(confirmarPassword)){
        final Roles rolSeleccionado = entityManager.getReference(Roles.class, tipoUsuario);
        Identity.setSecurityEnabled(false);
        new RunAsOperation() {
        public void execute() {
                identityManager.createUser(getUsername().trim(), getPassword());
                entityManager.flush();
               identityManager.grantRole(getUsername().trim(), rolSeleccionado.getNombre());
        }
}.addRole(rolSeleccionado.getNombre()).run();
entityManager.flush();
Identity.setSecurityEnabled(true);
}
else{
        facesContext.addMessage(null, new FacesMessage(FacesMessage.SEVERITY_INFO,"Error
al guardar usuario", "Contrasenias no coinciden"));
        return null;
}
return "/home.xhtml";
}
```

Por último para terminar con aspectos de seguridad, existen dos archivos de configuración en los que se deben ingresar ciertas líneas para tener una seguridad funcional con implementaciones de permisos de usuario por roles.

Configuración seguridad

Components.xml

```
<drools:rule-base name="securityRules">
      <drools:rule-files><value>/security.drl</value></drools:rule-files>
   </drools:rule-base>

   <security:rule-based-permission-resolver security-rules="#{securityRules}"/>

   <security:identity authenticate-method="#{authenticator.authenticate}"
remember-me="true"/>

     <security:identity-manager
          identity-store="#{jpaIdentityStore}"
          role-identity-store="#{jpaIdentityStore}"/>
```

```
   <security:jpa-identity-store
       user-class="com.factSeam.models.CuentasUsuario"
       role-class="com.factSeam.models.Roles"
   />

   <security:jpa-permission-store user-permission-
class="com.factSeam.models.Permisos"/>

   <event type="org.jboss.Seam.security.notLoggedIn">
      <action execute="#{redirect.captureCurrentView}"/>
   </event>
   <event type="org.jboss.Seam.security.loginSuccessful">
      <action execute="#{redirect.returnToCapturedView}"/>
   </event>
```

Page.xml

```
<page view-id="/mantUsuarios.Seam">
      <restrict>#{s:hasRole('admin')}</restrict>
     </page>
<exception class="org.jboss.Seam.security.AuthorizationException">
<redirect view-id="/error.xhtml">
     <message severity="error">You don't have permission to access this
resource</message>
</redirect>
</exception>
```

Manejo de reportes

Seam no provee una herramienta propia de reportería en la versión utilizada para el desarrollo, aunque en una versión que todavía se encuentra en estado beta se está utilizando dicha herramienta, para la visualización de reportes se puede utilizar el siguiente código:

Reportes en Seam

```
SuppressWarnings("unchecked")
public void generarReporteClientes() throws Exception {
     InputStream inputStream = new FileInputStream
("/home/gerito/Desarrollo/reportes/tesis/reportesSeam/report_clientes_todos.jrxm
l");
     List<Clientes> dataBeanList= new ArrayList<Clientes>();
     dataBeanList=entityManager.createQuery("Select p from Clientes
p").getResultList();
     Map parameters = new HashMap();
     JRBeanCollectionDataSource beanColDataSource = new
     JRBeanCollectionDataSource(dataBeanList);
     JasperDesign jasperDesign = JRXmlLoader.load(inputStream);
     JasperReport jasperReport =
JasperCompileManager.compileReport(jasperDesign);
     JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport,
parameters, beanColDataSource);
     byte[] pdf = null;
     pdf=JasperExportManager.exportReportToPdf(jasperPrint);
     FacesContext context = FacesContext.getCurrentInstance();
     HttpServletResponse response = (HttpServletResponse)
```

```
context.getExternalContext().getResponse();
     try {
          OutputStream os = response.getOutputStream();
          response.setContentType("application/pdf"); // fill in
          response.setContentLength(pdf.length);
          response.setHeader("Content-disposition", "attachment; filename=\""+
"reporteClientes.pdf\"");
          os.write(pdf); // fill in bytes
          os.flush();
          os.close();
     context.responseComplete();
     } catch (IOException e) {
          e.printStackTrace();
     }
}
```

Mecanismos del controlador

Seam es un framework mixto en el lado del controlador, ya que si bien implementa la mayor parte de las características del mismo de forma implícita, existen otras que se pueden implementar de forma manual, de esta manera el framework permite cosas fundamentales como el paso de parámetros entre paginas manteniendo así una conversación fluida entre las mismas, sin necesidad de implementar complejos mecanismos. La forma de mantener conversaciones es mediante un archivo extra por cada una de las vistas; es decir, una vista xhtml tiene asociado un archivo page.xml con el mismo nombre, en este archivo se determinan parámetros, condiciones y conversaciones para el despliegue de la vista.

ClientesList.page.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<page xmlns="http://jboss.org/schema/Seam/pages"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://jboss.org/schema/Seam/pages
http://jboss.org/schema/Seam/pages-2.3.xsd"
    login-required="true">

  <param name="firstResult" value="#{clientesList.firstResult}"/>
  <param name="sort" value="#{clientesList.orderColumn}"/>
  <param name="dir" value="#{clientesList.orderDirection}"/>
  <param name="logic" value="#{clientesList.restrictionLogicOperator}"/>

  <param name="from"/>
  <param name="cedula" value="#{clientesList.clientes.cedula}"/>
  <param name="direccion" value="#{clientesList.clientes.direccion}"/>
  <param name="nombre" value="#{clientesList.clientes.nombre}"/>
  <param name="telefono" value="#{clientesList.clientes.telefono}"/>

</page>
```