UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS COMPUTACIONALES

TEMA:

ESTUDIO COMPARATIVO DE FRAMEWORKS RIA PARA EL DESARROLLO DE APLICACIONES WEB CON JAVA SERVER FACES (JSF)

APLICATIVO:

SISTEMA DE CONTROL MICROCURRICULAR PARA LA CARRERA DE INGENIERÍA EN SISTEMAS

AUTOR:

ALEX RAÚL CAIZA MORILLO

DIRECTOR:

Ing. MIGUEL ORQUERA

Ibarra – Ecuador
2011

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

La Universidad Técnica del Norte dentro del Proyecto Repositorio Digital Institucional determina la necesidad de disponer de textos completos en formato digital con la finalidad de apoyar a los procesos de investigación, docencia y extensión de la Universidad.

Por medio del presente documento dejamos sentada nuestra voluntad de participar en éste proyecto, para lo cual ponemos a disposición la siguiente información.

DATOS DEL AUTOR

CÉDULA DE IDENTIDAD	100255643-7
APELLIDOS Y NOMBRES	CAIZA MORILLO ALEX RAÚL
DIRECCIÓN	LA PRIMAVERA - IBARRA
E-MAIL	araul_ar@yahoo.com
TELÉFONO MÓVIL	091505871

DATOS DE LA OBRA

TÍTULO	ESTUDIO COMPARATIVO DE FRAMEWORKS RIA PARA EL DESARROLLO DE APLICACIONES WEB CON JAVA SERVER FACES (JSF)
AUTOR	CAIZA MORILLO ALEX RAÚL
FECHA	31 de Octubre del 2011
PROGRAMA	PREGRADO
TÍTULO POR EL QUE OPTA	INGENIERO EN SISTEMAS
DIRECTOR	ING. MIGUEL ORQUERA

2. AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

Yo, Alex Raúl Caiza Morillo, con cédula de identidad N: 100255643 – 7, en calidad de autor y titular de los derechos patrimoniales de la obra o trabajo de grado descrito anteriormente, hago entrega del ejemplar respectivo en forma digital y autorizó a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y usos del archivo digital en la Biblioteca de la Universidad con fines académicos, para ampliar la disponibilidad del material y como apoyo a la educación, investigación y extensión, en concordancia con la ley de Educación Superior, Artículo 143.

UNIVERSIDAD TÉCNICA DEL NORTE

CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

Yo, Alex Raúl Caiza Morillo, con cédula de identidad N: 100255643 – 7, manifiesto mi voluntad de ceder a la Universidad Técnica del Norte, los Derechos Patrimoniales consagrados en la Ley de Propiedad Intelectual del Ecuador, artículo 4, 5 y 6 en calidad de Autor de la Obra o trabajo de grado denominado: ESTUDIO COMPARATIVO DE FRAMEWORKS RIA PARA EL DESARROLLO DE APLICACIONES WEB CON JAVA SERVER FACES (JSF), que ha sido desarrollada para optar por el título de INGENIERO EN SISTEMAS, en la UNIVERSIDAD TÉCNICA DEL NORTE, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En calidad de autor me reservo los derechos morales de la obra antes citada. En concordancia suscribo este documento en el momento que hago la entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Técnica del Norte.

Firma.....

NOMRES: ALEX RAÚL CAIZA MORILLO

CÉDULA: 100255643 – 7

Ibarra, a los treinta y un días del mes de octubre del 2011.

DECLARACIÓN

Yo, Alex Raúl Caiza Morillo, con cédula de identidad N: 100255643 – 7, declaro bajo juramento que el trabajo aquí descrito es de mi autoría, y que este no ha sido previamente presentado para ningún grado o calificación profesional.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo a la Universidad Técnica del Norte, según lo establecido por las leyes de Propiedad Intelectual y Normatividad vigente de la Universidad Técnica del Norte.

.....

Alex Raúl Caiza Morillo

CI.: 100255643 – 7

CERTIFICACIÓN

Certifico que el desarrollo de la presente tesis: "**ESTUDIO COMPARATIVO DE FRAMEWORKS RIA PARA EL DESARROLLO DE APLICACIONES WEB CON JAVA SERVER FACES (JSF)"** fue realizada en su totalidad por el Egresado Alex Raúl Caiza Morillo, bajo mi dirección.

.....

Ing. Miguel Orquera

CI.:

DIRECTOR

DEDICATORIA

Dedico este proyecto de tesis a Dios y a mis padres Raúl Caiza y Matilde Morillo pilares fundamentales en mi vida. A Dios porque ha estado conmigo a cada paso que doy, cuidándome y dándome fortaleza para continuar, a mis padres, quienes a lo largo de mi vida han velado por mi bienestar y educación siendo mi apoyo en todo momento. Depositando su entera confianza en cada reto que se me presentaba sin dudar ni un solo momento en mi inteligencia y capacidad. Su tenacidad y lucha insaciable han hecho de ellos el gran ejemplo a seguir y destacar, no solo para mí, sino para mis hermanos y familia en general.

Alex Raúl Caiza Morillo

AGRADECIMIENTO

Esta tesis está dedicada a mi Padres, a quienes agradezco de todo corazón por su amor, cariño y comprensión. En todo momento los llevo conmigo.

Agradezco a Dios por llenar mi vida de dicha y bendiciones.

Agradezco haber encontrado el amor y compartir mi existencia con ella.

Agradezco a los amigos por su confianza y lealtad.

A la Universidad Técnica del Norte, Facultad de Ingeniería en Ciencias Aplicadas, Escuela de Ingeniería en Sistemas Computacionales, por haberme dado la oportunidad de cursar el estudio y cumplir con la meta de llegar a ser Ingeniero en Sistemas.

A todos los docentes por brindarme sus conocimientos para un buen desempeño profesional, y de manera especial al Ing. Miguel Orquera Director de esta investigación quien con sus conocimientos y sabiduría de catedrático universitario me supo guiar para poder llegar a la culminación de este trabajo, que servirá como guía para los estudiantes de la carrera de sistemas.

A todos, mi mayor agradecimiento y gratitud.

Alex Raúl Caiza Morillo

RESUMEN

Debido a la creciente interacción de usuarios con sistemas web, surge la necesidad de combinar las funcionalidades e interfaces de usuario de las aplicaciones clásicas de escritorio, con la accesibilidad y bajo costo de publicación de las aplicaciones basadas en la Web; dando origen a las Aplicaciones Enriquecidas en Internet (RIA).

El presente proyecto se encuentra encaminado a la evaluación de tres implementaciones de la Tecnología JAVASERVER FACES 1.2 con soporte para AJAX, en el Sistema de Control Microcurricular para la Carrera de Ingeniería en Sistemas.

La evaluación de las diferentes Tecnologías se realizó mediante investigación y experimentación directa con cada una de las implementaciones seleccionadas para el estudio, RichFaces, IceFaces, Apache My Faces Trinidad.

Luego se seleccionará la implementación RichFaces de Jboss (RedHat), por la madurez, robustez y la suma de componentes y funcionalidades adicionales que este framework pone a disposición de los desarrolladores. Una de las características principales de RichFaces es la gran cantidad y calidad de recursos de información que posee y la continuidad en la publicación de nuevas versiones, que además de corregir errores, añaden funcionalidades.

Finalmente, se apreciará todas las bondades que brinda una aplicación RIA (Aplicaciones de Internet Enriquecidas) frente a una aplicación Web de tipo tradicional y en especial, las ventajas que ofrece la utilización del framework RichFaces.

ABSTRACT

Due to the increasing interaction of users with Web systems, the need to combine the functionality and user interfaces for desktop applications classics, accessibility and low cost of publication of Web-based applications, giving rise to the Applications rich Internet (RIA).

This project is aimed at the evaluation of three implementations of the JavaServer Faces technology 1.2 with support for AJAX in Microcurricular Control System for Systems Engineering Degree.

The evaluation of different technologies is achieved through research and direct experience with each of the implementations selected for study, RichFaces, ICEfaces, Apache Trinidad My Faces.

Then select JBoss RichFaces implementation (RedHat), the maturity, strength and the sum of components and additional functionality that this framework offers developers. One of the main features of RichFaces is a great resource quantity and quality of information held and continuity in the publication of new versions, as well as correct errors, add functionality.

Finally, we appreciate all the benefits offered by an RIA (Rich Internet Applications) compared to a traditional Web application type and in particular the advantages of using the RichFaces framework.

PRELIMINARES

ÍNDICE TEMÁTICO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO I

Frameworks RIA para Java Server Faces

1.1.Rich Internet Applications

Las Rich Internet Applications, o RIA en español "Aplicaciones de Internet Enriquecidas", son aplicaciones web que tienen la mayoría de las características de las aplicaciones de escritorio tradicionales. Estas aplicaciones utilizan un navegador web estandarizado para ejecutarse y por medio de complementos o mediante una máquina virtual se agregan las características adicionales.

Esta surge como una combinación de las ventajas que ofrecen las aplicaciones Web y las aplicaciones tradicionales. Buscan mejorar la experiencia del usuario

Normalmente en las aplicaciones Web, hay una recarga continua de páginas cada vez que el usuario pulsa sobre un enlace. De esta forma se produce un tráfico muy alto entre el cliente y el servidor, llegando muchas veces, a recargar la misma página con un mínimo cambio.

En los entornos RIA, en cambio, no se producen recargas de página, ya que desde el principio se carga toda la aplicación, y sólo se produce comunicación con el servidor cuando se necesitan datos externos como acceso a una Base de Datos o de otros ficheros externos.

Esta solución está orientada a hacer más amigable el desarrollo de aplicaciones web, mediante el uso del potencial ofrecido por JavaScript en la capa de presentación, con un gran conjunto de componentes AJAX y el estilo para desarrollo de aplicaciones propuesto.

1.1.1. Definición de RIA

Las aplicaciones RIA son un nuevo tipo de aplicaciones con más ventajas que las tradicionales aplicaciones Web. Esta surge como una combinación de las ventajas que ofrecen las aplicaciones Web y las aplicaciones tradicionales.

Hay muchas herramientas para la creación de entornos RIA. Entre estas se puede mencionar las plataformas Adobe Flash, Adobe Flex y Adobe AIR de Adobe, AJAX, OpenLaszlo, Silverlight de Microsoft, JavaFX Script de Sun Microsystems, GWT y Javascript, Rich Faces de JBoss, IceFaces de IceSoft, Trinidad del Apache Software Fundation.

1.1.2. Características de RIA

Las aplicaciones RIA (Rich Internet Applications) nacen de la convergencia de dos paradigmas, las aplicaciones de escritorio y las aplicaciones web; gracias a la evolución de ambos cada día se hace más difícil distinguir entre uno y otro.

Se puede notar, de esta manera, que es posible determinar que existen aplicaciones RIA que corren sobre un navegador de Internet y otras no. Las aplicaciones RIA basadas en un navegador son sitios web que mejoran la experiencia de los usuarios aumentando facilidades de personalización, tiempo de respuesta que facilita el manejo de grandes volúmenes de información, como ejemplos de estas tenemos gMail, Facebook y Google Maps. Las tecnologías comprendidas por la definición de RIA, varían en un rango amplio entre las aplicaciones tradicionales de escritorio y las simples páginas estáticas, tal como se muestra en la siguiente figura:

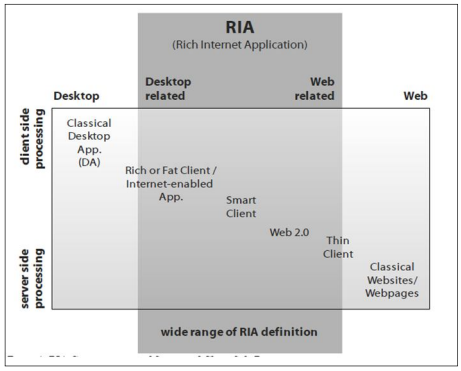


Figura 1. Dominio de la Aplicaciones RIA

Aunque existen varias tecnologías y aplicaciones que pueden ser consideradas dentro de la categoría de aplicaciones RIA dependiendo del punto de vista, la característica común es que una aplicación RIA junta las bondades de las aplicaciones de escritorio con los beneficios de las aplicaciones web.[1]

De las aplicaciones web se toman las siguientes características:

- Tiempo corto para la obtención de un producto final gracias a la estandarización de scripts y etiquetas
- Las aplicaciones que corren sobre un servidor no necesitan instalación de parches ni actualizaciones.
- Independencia de la plataforma desde la cual son accedidas
- Alta disponibilidad
- Interfaz simple y estandarizada aumentando la curva de aprendizaje.

De las aplicaciones de escritorio se toman las siguientes características:

- No existe la necesidad de recargar la aplicación, (recarga de página).
- Soporte online y offline.
- Mayor complejidad en las aplicaciones.
- Gran familiaridad con los usuarios.

Se considerará ahora los aspectos fundamentales que distinguen a las aplicaciones RIA:

1. Proporcionan un gran rendimiento, esto debido a que el procesamiento se puede hacer en el lado del cliente, en lugar que todos los procesos se realicen en el lado del servidor. De igual manera mientras se usa la aplicación el usuario no experimenta ninguna clase de refresco de página.
   Sin embargo la decisión de que procesos se realizan en la parte del cliente y que en la parte del servidor, dependen de varios aspectos, en la figura siguiente se puede notar la separación de las capas de una aplicación, junto con la distribución de procesamiento.
2. Priorizan la interacción con el usuario, por la gran riqueza de recursos proporcionan una experiencia más atractiva y entretenida.

[1] Análisis comparativo de los Frameworks Adobe Flex, Java RichFaces y Extjs http://bibdigital.epn. edu.ec /bitstream/15000/ 2110/1/CD-2887.pdf

A continuación se describen algunos ejemplos:

- Gracias a las herramientas gráficas para la visualización de datos, permite al nivel gerencial interactuar con datos complejos y realizar análisis más profundos.
- Permite a los usuarios personalizar un producto en línea para satisfacer sus necesidades específicas.

3. Acoplan de manera natural varias tecnologías para conseguir la aplicación final que cumpla con las necesidades del usuario. Esto se hace sin olvidar la consigna de independencia de plataforma de acceso del usuario. En este punto es necesario considerar que el uso de varias tecnologías versus el uso de una sola, determina que la complejidad y tiempo de desarrollo sufren un claro incremento.

1.1.3. Arquitectura para RIA

Generalmente se tiene una aplicación cliente "stateful" y una capa de servicios separada. RIA se apoya más sobre un desarrollo "cliente-servidor" en vez de un desarrollo web tradicional, en donde el estado se mantiene en el servidor en sesiones. El cliente sabe acerca de sí mismo y el tipo de datos que está solicitando y únicamente solicita los datos que necesita sin ninguna otra información.

- ***Cliente***. Se maneja la interacción entre el usuario y la "interfaz del usuario", el usuario invoca comandos, actualiza vistas y carga datos. Aquí se mantiene el estado de la aplicación, se manejan todas las peticiones de datos hacia el servidor y se controla como se presentan los datos.
- ***Servidor***. Aquí se manejan y se procesan todas las peticiones de la aplicación cliente y delega las acciones en el servidor, estas pueden ser, guardar datos en la base de datos, actualizar los archivos del sistema, retornar datos al servidor, o algún tipo de proceso analítico. Determina y le da formato a los datos que son retornados al cliente.

1.1.4. El entorno RIA

RIA[2] es un tipo de aplicación Web que provee una mayor riqueza interactiva que las aplicaciones Web tradicionales, incorporando características muy similares a las

[2] Rich Internet Application, o Aplicación de Internet Enriquecida.

que poseen las aplicaciones de escritorio, las cuales proveen una interacción dinámica y rica en elementos de interfaz.

El siguiente diagrama muestra los cambios fundamentales entre la arquitectura de la Web 1.0 y la arquitectura de la Web 2.0.

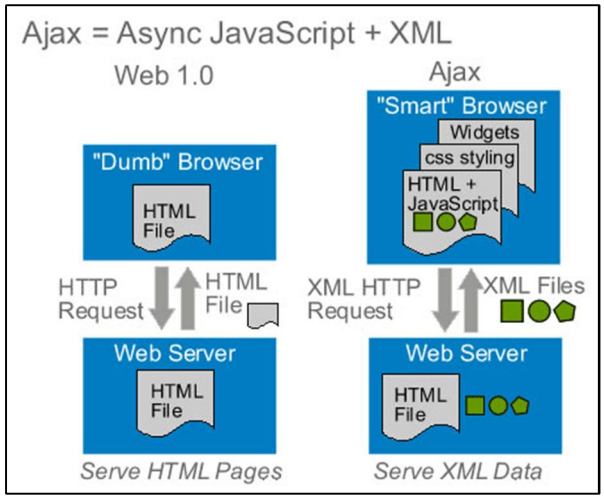


Figura 2. Cambio de Arquitectura en la Web

La web 1.0 principalmente trata lo que es el estado estático, es decir los datos que se encuentran en ésta no pueden cambiar, se encuentran fijos, no varían, no se actualizan.

Los conceptos para la Web 1.0 desde el 2.002-2004 son:

- HTML
- FLASH
- JavaScript 1.0
- CGI
- Diseño de páginas por (Marcos, Tablas, hipervínculos).
- Buscadores
- Portales
- Páginas personalizadas
- Conceptos como (E-commerce, E-procurement, E-learning)
- Foros de discusión
- IRC , chateos, contactos on-line
- E-Mail, Web Mail

La Web 2.0 está comúnmente asociada con aplicaciones web que facilitan el compartir información, la interoperabilidad, el diseño centrado en el usuario y la colaboración en la World Wide Web.

Para la Web 2.0, a partir del año 2002, se han asignado los siguientes conceptos:

- AJAX
- Wikis
- RSS
- XML, XHTML,DHTML
- Blogs
- Audio , Video
- Estándares de Web 3WC
- Datos Inteligentes SEO
- Movilidad (Móvil, PDA)
- Proyectos Open Source (PHP, MySql, Perl, etc.…)
- Hojas de Estilo (CSS)
- Programación en etiquetas div
- DOM, JAVA, JAVASCRIPT 2.0, APPLETS
- El mundo Google

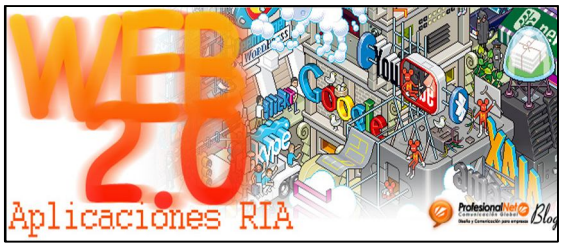


Figura 3. Aplicaciones RIA para la Web 2.0

El término Rich Internet Application fue introducido en marzo de 2002 por Macromedia; sin embargo, este mismo concepto ya había sido manejado bajo otros nombres:

- Remote Scripting, por Microsoft, en 1998
- X Internet, por Forrester Research, en octubre de 2000
- Rich Web Clients
- Rich Web Application

Como se mencionó anteriormente, el modelo tradicional de aplicaciones Web tiene una serie de limitaciones, como la poca capacidad multimedia que posee y la recarga continua de páginas. Todo esto se debe a que el cliente en las aplicaciones Web tradicionales sólo se limita a desplegar el contenido HTML[3]. En cambio, las RIAs incorporan un motor como una nueva capa del lado del cliente, que sirve como intermediaria entre la interacción del cliente con el servidor, y tiene la responsabilidad de realizar los cambios sobre la interfaz de usuario.

Algunos de los beneficios que provee el uso de este entorno son los siguientes:

- Se aprovecha la capacidad de procesamiento del lado del cliente. El motor del lado del cliente puede tener la capacidad de realizar actividades solicitadas por el usuario sin la necesidad de realizar una petición al servidor. Estas actividades podrían ser cálculos matemáticos, cambios en la IU[4] (reposicionar objetos), etc.
- El motor del lado del cliente puede interactuar con el servidor asincrónicamente; por ejemplo, el motor podría solicitar datos al servidor para futuras peticiones del usuario, mejorando así el tiempo de respuesta para dichas peticiones.
- Las RIAs no necesitan hacer recargas continuas de toda la IU; en vista que el cliente es más sofisticado, es posible recargar la sección específica de la página que presente algún cambio.

A pesar de los múltiples beneficios que provee este entorno, las aplicaciones RIA presentan algunas limitaciones:

- Debido a que las RIAs se ejecutan en un sandbox[5], éstas tienen acceso restringido a los recursos del sistema. Una aplicación podría necesitar algún tipo de recurso particular, por ejemplo, tener acceso al sistema de archivos del cliente; en ciertas RIAs este acceso puede estar restringido por el sandbox, por lo tanto, su utilización para el desarrollo de esta aplicación no sería viable ya que no se podría satisfacer dicho requerimiento.
- Las RIAs son dependientes de un componente tecnológico para su correcto funcionamiento, el cual en muchos casos es el soporte de JavaScript o el uso de

[3] Acrónimo de HyperText Markup Language (Lenguaje de Marcado de Hipertexto).
[4] Interfaz de Usuario: vínculo entre el usuario y el programa de computadora.
[5] Sandbox es un entorno seguro donde las aplicaciones tienen acceso restringido a determinados recursos del sistema.

algún plugin específico. En algunos casos también son dependientes de la plataforma tecnológica por ejemplo, el navegador.

- Según la compañía IBM[6] (Gadge, 2006), existe una serie de aspectos que son necesarios considerar a la hora de escoger una tecnología o un enfoque para desarrollar RIAs. A continuación se describe cada uno de estos aspectos.

1.1.4.1. Usabilidadad

Los usuarios esperan que el navegador continúe trabajando de la misma forma como si se estuviera interactuando con una aplicación Web tradicional. Esto quiere decir que el usuario espera que los botones atrás e historial del navegador mantengan el comportamiento que presentan en las aplicaciones Web tradicionales, así como también que las teclas de acceso rápido (cortar y pegar, buscar contenido en la página, entre otros) se puedan utilizar en la misma.

1.1.4.2. Reducción de la recarga continua de páginas

Algunas solicitudes por parte del usuario podrían generar cambios en una pequeña parte de la interfaz de usuario; por consiguiente, la tecnología RIA debe facilitar la disminución del número de recargas de la página, permitiendo restringir los cambios únicamente a los sectores que deban ser modificados en la misma.

1.1.4.3. Complejidad e interoperabilidad

La tecnología para desarrollar la RIA debe ser fácil de aprender y de usar. También debe tener la capacidad de interoperar con tecnologías Web existentes.

1.1.4.4. Seguridad

Que provea características o mecanismos de seguridad que permitan mantener cierto control sobre la aplicación. Por ejemplo, sería conveniente estar al tanto de todas las conexiones que se establecen por medio del cliente, para así conocer la forma en la cual interactúa el motor del lado del cliente con el servidor y evitar así conexiones no generadas por acciones por parte del usuario de la aplicación (intrusiones).

[6] International Business Machine. Empresa multinacional estadounidense de tecnología y consultoría con sede en Armonk, Nueva York. IBM fabrica y comercializa hardware y software para computadoras.

1.1.4.5. Soporte para los paradigmas básicos de la Web

La tecnología debe soportar los paradigmas básicos existentes en la Web, tales como internacionalización, independencia del dispositivo de acceso, independencia del navegador, entre otros.

1.1.4.6. Utilidades

Verificar qué utilidades se encuentran disponibles para el desarrollo de RIAs. Estas utilidades pueden presentarse como plugins para los IDE[7] de desarrollo, pudiéndose mencionar depuradores, herramientas de asistencia para la codificación, entre otras.

1.2. JavaServer Faces

La tecnología JavaServer Faces es un framework de interfaz de componentes de usuarios del lado del servidor para las aplicaciones web basadas en la tecnología Java.

Los principales componentes de la tecnología JSF son:

- Una API[8] para:
  - Representar componentes de Interfaz de Usuario (UI) y gestionar su estado.
  - Manejar eventos, validar en el servidor y conversión de datos.
  - Definir la navegación de páginas.
  - Soporte de internacionalización y accesibilidad.
- Dos librerías de etiquetas JSP[9] personalizadas para expresar componentes en una página JSP y enlazar los componentes a objetos del servidor.

El modelo de programación bien definido y las librerías de etiquetas facilitan la construcción y mantenimiento de las aplicaciones web con Interfaces de Usuario (UI) de servidor. Con un mínimo esfuerzo se podría:

- Poner componentes en una página mediante etiquetas de componentes.

[7] Integrated Development Environment o Entorno de desarrollo integrado.
[8] Application Programming Interface o Interfaz de programación de aplicaciones.
[9] JavaServer Pages es la tecnología para generar páginas web de forma dinámica en el servidor, desarrollado por Sun Microsystems, basado en scripts que utilizan una variante del lenguaje java.

- Enlazar eventos generados por componentes con código de la aplicación en el servidor.
- Relacionar componentes UI en una página con datos del servidor.
- Construir una UI con componentes reutilizables y extensibles.
- Salvar y restaurar el estado de la UI más allá de la vida de las peticiones.

JSF es una especificación desarrollada por la Java Community Process. Actualmente existen cuatro versiones de esta especificación:

Versión	Fecha	Características
JSF 1.0	11-03-2004	Lanzamiento inicial de las especificaciones de JSF.
JSF 1.1	27-05-2004	Lanzamiento que solucionaba errores. Sin cambios en las especificaciones ni en el renderkit de HTML.
JSF 1.2	20-03-2007	Lenguaje de Expresión Unificado Integración con JSTL Asociación de mensajes mensaje con un componente específico en la página
JSF 2.0	06-03-2011	Último lanzamiento. Expansión del ciclo de vida para brindar soporte a peticiones Ajax. Pretende mejorar la interoperabilidad entre librerías de diferentes proveedores.

Tabla 1.Versiones de JSF

1.2.1. Objetivos de JSF

Estos objetivos de diseño representan el foco de desarrollo de JSF:

- Definir un conjunto simple de clases base de Java para componentes de la interfaz de usuario, estado de los componentes y eventos de entrada. Estas clases tratarán los aspectos del ciclo de vida de la interfaz de usuario, controlando el estado de un componente durante el ciclo de vida de su página.
- Proporcionar un conjunto de componentes para la interfaz de usuario, incluyendo los elementos estándares de HTML para representar un formulario. Estos componentes se obtendrán de un conjunto básico de clases base que se pueden utilizar para definir componentes nuevos.
- Proporcionar un modelo de JavaBeans para enviar eventos desde los controles de la interfaz de usuario del lado del cliente a la aplicación del servidor.

- Definir APIs para la validación de entrada, incluyendo soporte para la validación en el lado del cliente.
- Especificar un modelo para la internacionalización y localización de la interfaz de usuario.
- Automatizar la generación de salidas apropiadas para el objetivo del cliente, teniendo en cuenta todos los datos de configuración disponibles del cliente, como versión del navegador.

1.2.2. Definición de JSF

JSF es una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario para crear aplicaciones java J2EE basadas en el patrón MVC Modelo Vista Controlador (Model View Controller).

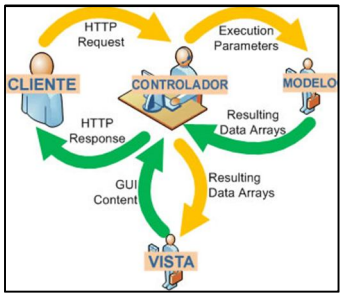


Figura 4.Patrón de diseño MVC

Para el desarrollo de aplicaciones de negocio se utiliza frecuentemente el patrón de diseño MVC que además es sencillo de implementar en las aplicaciones web. En este patrón el modelo es modificable por las funciones de negocio. Estas funciones son solicitadas por el usuario mediante el uso de un conjunto de vistas de la aplicación que solicitan dichas funciones de negocio a través de un controlador, que es el módulo que recibe las peticiones de las vistas y las procesa.

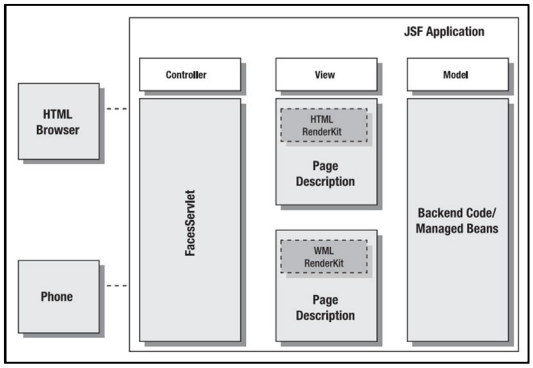


Figura 5. Arquitectura MVC con JSF

JSF nos ofrece un marco de trabajo que facilita el desarrollo de aplicaciones, separando las diferentes capas de una arquitectura: presentación, reglas y entidades de negocio. En su mayor parte, una aplicación JSF es como cualquier otra aplicación web de Java. Una típica aplicación JSF incluye lo siguiente:

- Un API para representar componentes de interfaz de usuario y la gestión de su estado, manejo de eventos, validación en el servidor, la conversión de datos, la accesibilidad, y proporcionar extensibilidad para todas estas características.
- Un conjunto de páginas JSP (aunque no se limita al uso de páginas JSP como tecnología de presentación).
- Un conjunto de beans de respaldo, los cuales son JavaBeans que definen propiedades y funciones para los componentes de interfaz de usuario en una página.
- Un archivo de configuración de recursos de la aplicación, que define las reglas de navegación de la página y configura los beans y otros objetos como los componentes típicos.
- Posiblemente un conjunto de objetos personalizados creados por el desarrollador de la aplicación. Estos objetos pueden incluir componentes típicos, validadores, convertidores o detectores.
- Un conjunto de etiquetas para representar los típicos objetos en la página.
- Un modelo de eventos en el lado del servidor.
- Administración de estados.

1.2.3. Características de JSF

Este modelo de programación bien definido y la librería de etiquetas para componentes UI, facilitan la tarea de la construcción y mantenimiento de aplicaciones web con UIs del lado del servidor.

- Utiliza páginas JSP para generar las vistas, añadiendo una biblioteca de etiquetas propia para crear los elementos de los formularios HTML.
- Asocia a cada vista con formularios un conjunto de objetos java manejados por el controlador (managed beans) que facilitan la recogida, manipulación y visualización de los valores mostrados en los diferentes elementos de los formularios.

- Introduce una serie de etapas en el procesamiento de la petición, como por ejemplo la de validación, reconstrucción de la vista, recuperación de los valores de los elementos, etc.
- Utiliza un sencillo fichero de configuración para el controlador en formato xml.
- Es extensible, pudiendo crearse nuevos elementos de la interfaz o modificar los ya existentes.
- Y lo que es más importante: forma parte del estándar J2EE[10]. En efecto, hay muchas alternativas para crear la capa de presentación y control de una aplicación web java, como Struts y otros frameworks, pero solo JSP forma parte del estándar.

1.2.4. El entorno JSF

El entorno JSF es un conjunto ampliable de componentes de interfaz de usuario que va acompañado de un API para gestionar de forma dinámica el estado de las aplicaciones, el manejo de eventos, la validación de datos de entrada, el desplazamiento por las páginas y las funciones de accesibilidad y adaptación a entornos internacionales. AJAX[11] es una nueva técnica de desarrollo de aplicaciones web que utiliza JavaScript en el lado cliente para proporcionar una experiencia de uso más rica y ágil que la que puede obtenerse si se utiliza únicamente la lógica del lado servidor.

1.2.4.1. Usabilidad

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

JSF ofrece una serie de ventajas:

- El código JSF con el que creamos las vistas o etiquetas jsp es muy parecido al HTML estándar. Lo pueden utilizar fácilmente desarrolladores y diseñadores web.
- JSF se integra dentro de la página JSP y se encarga de la recogida y generación de los valores de los elementos de la página

[10] Java 2 Enterprise Edition

[11] Acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications).

- JSF resuelve validaciones, conversiones, mensajes de error e internacionalización (i18n)
- JSF permite introducir javascript en la página, para acelerar la respuesta de la interfaz en el cliente (navegador del usuario).
- JSF es extensible, por lo que se pueden desarrollar nuevos componentes a medida, También se puede modificar el comportamiento del framework mediante APIs que controlan su funcionamiento.

Desde el punto de vista técnico podemos destacar lo siguiente:

- JSF forma parte del estándar JEE, mientras que otras tecnologías para creación de vistas de las aplicaciones no lo forman, como por ejemplo Struts.
- JSF dispone de varias implementaciones diferentes, incluyendo un conjunto de etiquetas y una interfaz de programación de aplicaciones (APIs) estándar que forman el núcleo del framework. Entre estas implementaciones cabe destacar la implementación de referencia de Sun Microsystems, actualmente desarrollada como un proyecto open source.
- El desarrollo de JSF está realmente empezando. Las nuevas versiones del framework recogen la funcionalidad de versiones anteriores siendo su compatibilidad muy alta, de manera que el mantenimiento de aplicaciones no se ve penalizado por el cambio de versiones.

1.2.4.2. Riesgos

Antes de comenzar el desarrollo con JSF debemos conocer aquellos puntos que lo pueden hacer más largo de lo realmente necesario. Entre ellos la experiencia nos muestra los siguientes:

- Conocimientos básicos. JSF es una herramienta y como tal tiene una forma de uso. Si nos empeñamos en seguir desarrollando las páginas como siempre, intentando adaptar JSF al modo al que habitualmente desarrollamos en vez de adaptarnos a JSF complicaremos el desarrollo.
- Abuso del javascript. JSF permite utilizar Javascript para hacer más rápida una página html, evitando peticiones al servidor. Sin embargo la introducción de javascript en la página complica y alarga los desarrollos con JSF, y en general con jsp. La capa javascript añade etapas adicionales a la aplicación, que hace más difícil su depurado. La página debería poderse ejecutar sin pérdida de funcionalidad (sólo de rendimiento si se desactiva el javascript).

- La maquetación compleja también complica el desarrollo ya que obliga a utilizar muchas etiquetas y atributos, especialmente en los datatables. Si la maquetación de nuestras páginas es compleja deberíamos pensar en crear componentes JSF a medida que simplifiquen dicho trabajo.

1.2.4.3. Funcionamiento de JSF

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

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

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

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

Las aplicaciones JSF están formadas por los siguientes elementos principales:

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

1.2.4.4. Etiquetas JSF

JSF dispone de un conjunto básico de etiquetas que permiten crear fácilmente componentes dinámicos en las páginas web. Estas etiquetas son:

- **<h:commandButton>:** Un botón al que podemos asociar una acción.
- **<h:commandLink>:** Un enlace hipertexto al que podemos asociar una acción.
- **<h:dataTable>:** Crea una tabla de datos dinámica con los elementos de una propiedad de tipo Array o Map del bean.
- **<h:form>:** Define el formulario JSF en la página JSP.
- **<h:graphicImage>:** Muestra una imagen jpg o similar.
- **<h:inputHidden>:** Incluye un campo oculto del formulario.
- **<h:inputSecret>:** Incluye un campo editable de tipo contraseña (no muestra lo que se escribe)
- **<h:inputText>:** Incluye un campo de texto normal.
- **<h:inputTextarea>:** Incluye un campo de texto multilínea.
- **<h:message>:** Imprime un mensaje de error en la página (si se ha producido alguno).
- **<h:messages>:** Imprime varios mensajes de error en la página, si se han producido.
- **<h:outputFormat>:** Muestra texto parametrizado. Utiliza la clase de formateo java.text.MessageFormat.
- **<h:outputLabel>:** Muestra un texto fijo.
- **<h:outputLink>:** Crea un enlace hipertexto.
- **<h:outputText>:** Es la etiqueta más común utilizada para mostrar texto plano, y no genera ningún elemento HTML extra.
- **<h:panelGrid>:** Crea una tabla con los componentes incluidos en el panelGrid.
- **<h:panelGroup>:** Agrupa varios componentes para que cierto componente los trate como un único componente por ejemplo para meter varios componentes en una celda de un panelGrid.
- **<h:selectBooleanCheckbox>:** Crea una casilla con dos estados: activado y desactivado.
- **<h:selectManyCheckbox>:** Crea un conjunto de casillas activables.
- **<h:selectManyListbox>:** Crea una lista que permite seleccionar múltiples elementos.
- **<h:selectManyMenu>:** Crea una lista desplegable de selección múltiple.
- **<h:selectOneListbox>:** Crea una lista en la que se puede seleccionar un único elemento.

- **<h:selectOneMenu>:** Crea una lista desplegable de selección.
- **<h:selectOneRadio>:** Crea una lista de botones, redondos normalmente, excluyentes.

1.2.4.5. Los Managed-Bean

A las clases java que se asocian a los formularios JSF se les denomina backend beans ya que son los beans (clases java) que están detrás del formulario. Estos beans se referencian en el fichero de configuración de JSF en el apartado de managed beans, ya que son beans gestionados por el controlador JSF este se encarga de su construcción y destrucción automáticas cuando es necesario.

1.2.4.6. Estructura de las páginas

En su versión más sencilla, cada página JSF está formada por una página JSP que contiene un formulario (HTML FORM) y un backbean.

El controlador JSF registra en el servidor de aplicaciones un tipo especial de petición, típicamente *.jsf, que estará asociado a estas páginas.

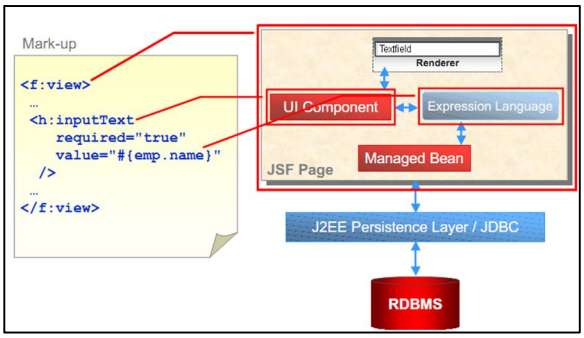


Figura 6. Procesamiento de una página JSF

El primer caso comienza cuando el usuario realiza en su navegador una petición de navegación a una url de tipo *.jsf. Cuando al servidor web llega una petición del tipo página JSF, el controlador JSF entra en funcionamiento.

Primero comprueba si es la primera vez que se accede a dicha página. Si es así, carga la página jsp asociada pagina.jsp y la procesa construyendo en memoria la representación de los controles de la página. Tras esta etapa JSF sabe cómo construir el código HTML de salida y la lista de controles de usuario que la cumplen, es decir, sabe lo que contiene y cómo pintarla.

El siguiente paso es asociarle los managed beans. Para ello, del procesamiento de la página jsp, el controlador ha obtenido la lista de managed beans asociados, por lo que procede a buscarlos en sus correspondientes ámbitos como: aplicación, request, session. Los beans que no existan se crean llamando a los constructores de sus clases, definidos en la sección de managed beans del fichero de configuración de JSF.

El tercer paso es dar valores a las propiedades de los elementos JSF de la página. Aquí juega un papel fundamental el lenguaje de expresiones de JSF, que es parecido al lenguaje de expresiones que se permite en las páginas jsp normales.

En su versión más sencilla una expresión JSF sería del tipo:
#{managedbean.propiedad}.

Finalmente el servidor devuelve al usuario una página creada a partir de una página JSP que incluye normalmente etiquetas JSF, cuyos valores se extraerán del backbean asociado, ahora ya actualizados.

1.2.4.7. Respuesta a acciones del usuario

Una vez que el usuario ve la página web que se ha construido con JSF, está listo para comenzar a interactuar con ella. El método más sencillo de interacción es el uso de formularios web. Un formulario web simple consta de:

- Etiquetas que muestran information
- Campos editables
- El botón de envío del formulario

El controlador JSF dispone de un conjunto de etiquetas que permiten definir formularios JSF. Las más sencillas son:

- h:form. Esta etiqueta sustituye al form de HTML, añadiendo la funcionalidad JSF al formulario
- h:outputText. Sirve para imprimir valores en la página
- h:inputText. Sirve para crear campos editables en los que introducir los datos
- h:commandButton. Crea botones que envían el formulario.

Cuando la página JavaServer Faces contiene elementos que incluyen acciones se ejecuta una fase más en el procesamiento de la petición al servidor. Si en nuestro formulario hay botones u otros elementos que tienen una propiedad action, si se pulsa sobre el elemento cuando la petición sea procesada por el servidor se ejecutará la lógica de la acción asociada a este elemento. Este es el mecanismo JavaServer Faces habitual para ejecutar la lógica de la aplicación. Esto se hace ejecutando los métodos del backbean asociado a la página.

1.2.4.8. Navegación entre páginas

Cuando se ejecuta una petición que incluye una acción se ejecuta el mecanismo de navegación de JavaServer Faces. Tras la ejecución de la acción, el controlador determina cómo se debe mostrar al usuario el resultado de la petición. Hay varias posibilidades:

- Finalizar la petición mostrando la página jsp que originó la petición, que es la opción por defecto.
- Mostrando otra página jsp diferente.
- Enviando al usuario una petición de redirección, por lo que el navegador del usuario se dirigirá automáticamente a otra página cuando reciba la respuesta a su petición.

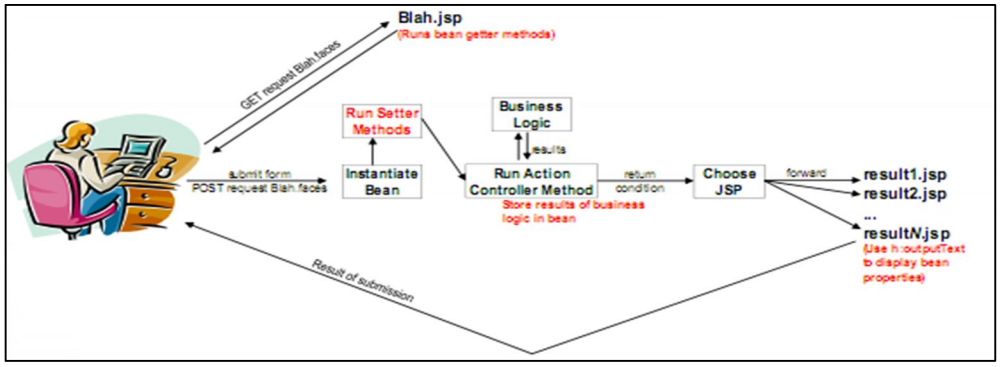


Figura 7. Navegación entre páginas

Este mecanismo de navegación se implementa de manera sencilla en la página JavaServer Faces. Cuando el controlador JavaServer Faces llama al método asociado a la acción, este devuelve un valor de tipo String. Este valor es utilizado junto con las reglas de navegación creadas en el fichero de configuración de JavaServer Faces para determinar la página que se debe enviar como respuesta al usuario.

Las reglas de navegación definen:

- **La página de origen.** Indica el jsp que originó la petición.
- **La etiqueta de destino.** Es la cadena que identifica el destino. Esta cadena es devuelta por el método del backbean que procesa la acción.
- **La página de destino para cada etiqueta.** Normalmente es el jsp el que procesará la petición de salida, utilizando los datos que hay en la request y en la sesión.
- **Si es un envío directo interno o una redirección externa.** En el primer caso la respuesta se generará en la misma petición mediante una redirección interna a otro jsp o servlet. En el segundo caso se enviará como respuesta al navegador una instrucción de redirección para que el navegador realice una nueva petición de otra página.

Además las direcciones de origen admiten el * para que una misma regla sirva para múltiples páginas. También se pueden poner reglas por defecto que se aplican a todas las peticiones.

1.2.4.9. Etapas de procesamiento de una petición

Para entender el procesamiento de una página JSF hay que entender el ciclo de vida de la petición dentro del controlador JSF. Este ciclo de vida está compuesto de 6 fases.

Durante el procesamiento de una petición el controlador JSF realiza las siguientes etapas:

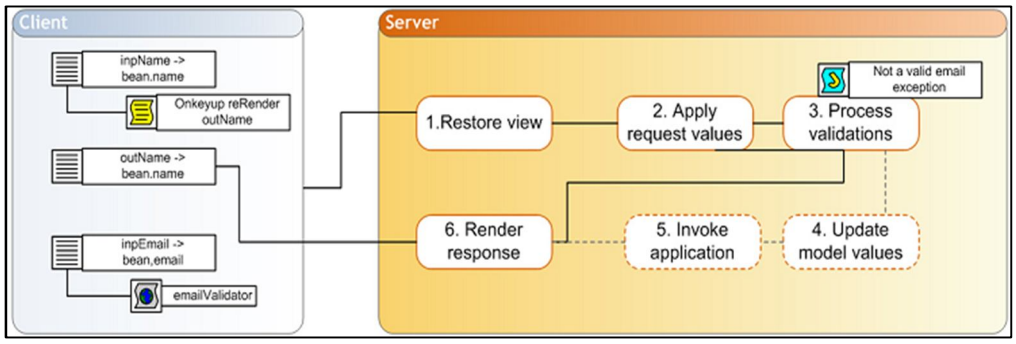


Figura 8. Ciclo de Vida JSF

1. **Restaurar los componentes de la vista** (restore view). En esta etapa el controlador construye en memoria la estructura de componentes de la página.

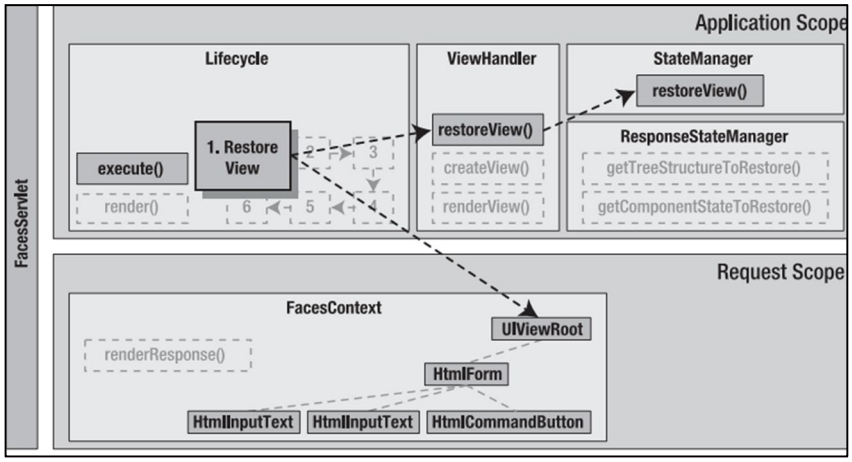


Figura 9. Restaurar los componentes de la vista

2. **Aplicar los valores de la petición** (apply request values). En esta etapa se recuperan los valores de la request y se asignan a los beans de la página.

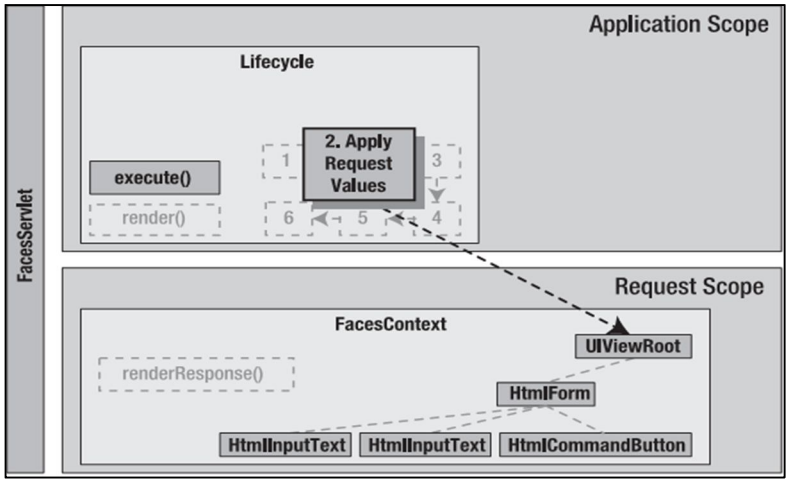


Figura 10. Aplicar los valores de la petición

3. **Procesamiento de las validaciones** (process validations). Se verifican los parámetros de entrada según un conjunto de reglas definidas en un fichero de configuración.

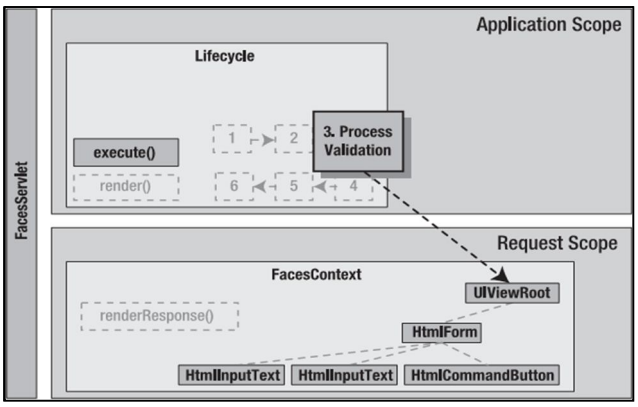


Figura 11. Procesamiento de las validaciones

4. **Actualizar los valores del modelo** (update model values). Los valores leídos y validados son cargados en los beans.

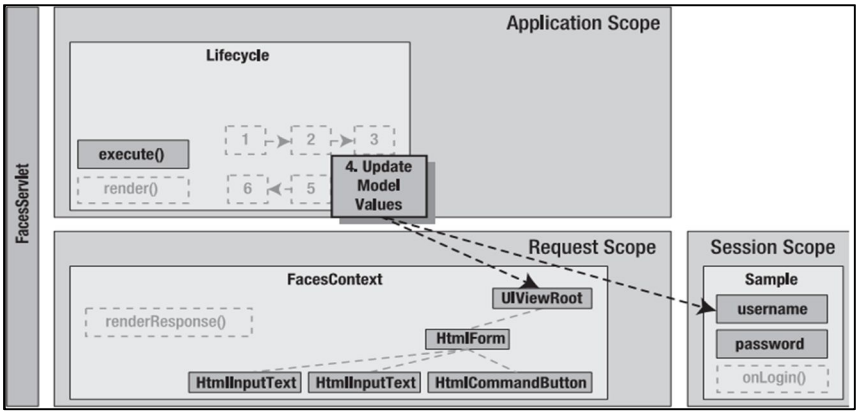


Figura 12. Actualizar los valores del modelo

5. **Invocación a la aplicación** (invoke application). Se ejecutan las acciones y eventos solicitados para la página.

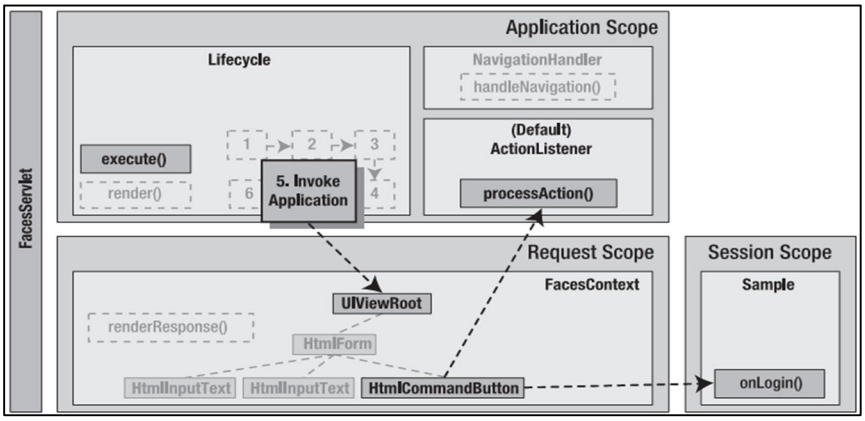


Figura 13. Invocación a la aplicación

Devolución de datos con sistema de navegación (Postback with Navigation): Si es necesario se realiza la navegación.

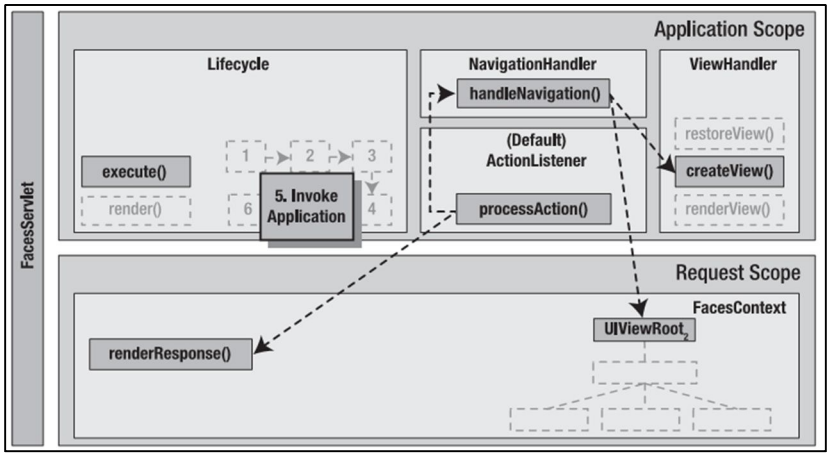


Figura 14. Devolución de datos con navegación

6. **Generación de la página** (render response). En esta fase se genera la página que será enviada al usuario con todos sus elementos y valores actualizados.

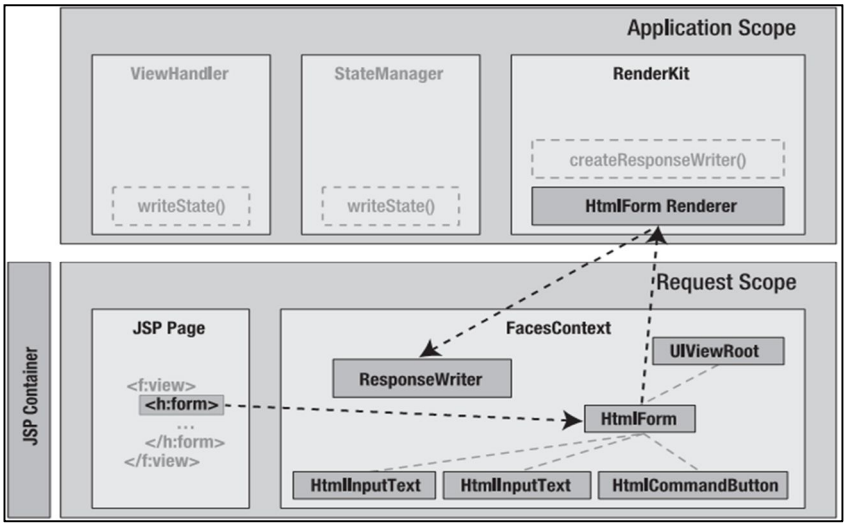


Figura 15. Generación de la página

1.2.4.10. Gestión de los beans

JSF gestiona automáticamente la creación y el acceso a los beans[12] que utilizan las páginas jsp. Para ello el controlador determina qué beans utiliza la página y dónde debe almacenarlos. El fichero de configuración JSF mapea los nombres cortos de los beans utilizados en las páginas con las clases java que los definen.

¿Cuándo se crean los beans?. JSF busca el bean cada vez que se menciona en la página, en el orden en que aparecen en la página. Si el bean no existe en el ámbito, lo crea. Por tanto el orden de las expresiones EL determinan el orden de la creación de los beans, si usamos más de un bean en la página.

¿Cómo se hace esto internamente?. Al procesar la página JSP, las etiquetas JSF añaden código que busca el bean mencionado en cada expresión EL. Si el bean no existe en el ámbito elegido (request, session o application) se crea uno nuevo, llamando a su constructor por defecto, y se asocia al ámbito requerido. Este mecanismo es fundamental para la comprensión del procesamiento de la página, sobre todo si trabajamos con beans de ámbito request. Es recomendable poner siempre logs a los constructores para saber el momento exacto de su ejecución.

[12] Componentes de software que tiene la particularidad de ser reutilizable y así evitar la tediosa tarea de programar los distintos componentes uno a uno.

1.2.4.11. Lenguaje de expresiones EL

Para facilitar la visualización y recogida de los datos de los formularios JSF introduce un lenguaje de expresiones similar al introducido en jsp. De hecho a partir de JSF 1.2, ambos lenguajes de expresiones se han unificado.

El lenguaje de expresiones permite acceder de manera muy sencilla a las propiedades de los backbeans.

En una forma más sencilla una expresión EL se puede escribir como: **#{backbean.propiedad}**

Esto permite asignar o leer valores de las etiquetas JSF. Por ejemplo para escribir y leer valores se pueden usar las etiquetas JSF:

<h:outputText value = "**#{backbean.propiedad}**" />
<h:inputTtxt value = "**#{backbean.propiedad}**" />

1.2.4.12. El entorno FacesContext

Un backbean es una clase que no conoce nada del resto de la aplicación. Para acceder al entorno JSF y en general al entorno de ejecución en el que la clase se está ejecutando JSF prevé el mecanismo de contexto JSF FacesContext. El FacesContext es una clase que sirve al bean de puente al exterior, ya que le permite acceder no solo al contexto JSF sino también al contexto HTTP[13]. Esto permite al bean el acceso a los demás beans de la aplicación, a las propiedades de la aplicación e incluso a la petición HTTP que se está ejecutando.

El uso del contexto FacesContext se realiza desde el bean, ya que la clase FacesContext proporciona un método que devuelve una referencia a la instancia JSF asociada a la petición.

[13] HTTP es un protocolo sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores. El desarrollo de aplicaciones web necesita frecuentemente mantener estado. Para esto se usan las cookies, que es información que un servidor puede almacenar en el sistema cliente. Esto le permite a las aplicaciones web instituir la noción de "sesión", y también permite rastrear usuarios ya que las cookies pueden guardarse en el cliente por tiempo indeterminado.

1.2.5. Relación de JSF y Ajax.

JSF es un framework que lanza muchas peticiones al servidor. Para optimizar dicho dialogo están empezando a aparecer implementaciones de JSF que incorporan AJAX en sus etiquetas. Esto permite actualizar los componentes en el navegador del usuario de manera selectiva, sin necesidad de recargar la página completa. La combinación JSF AJAX dota a las páginas de gran dinamismo sin complicar el desarrollo, evitando el uso de javascript codificado a mano asegurando un mayor soporte a los navegadores web.

AJAX es un conjunto de tecnologías de desarrollo web para la creación de aplicaciones interactivas o RIA (Rich Internet Applications), las mismas que se ejecutan en el cliente (navegador de los usuarios), mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. Permitiendo así, realizar cambios sobre las páginas sin necesidad de recargarlas, lo que incrementa la interactividad, velocidad y usabilidad en las aplicaciones.

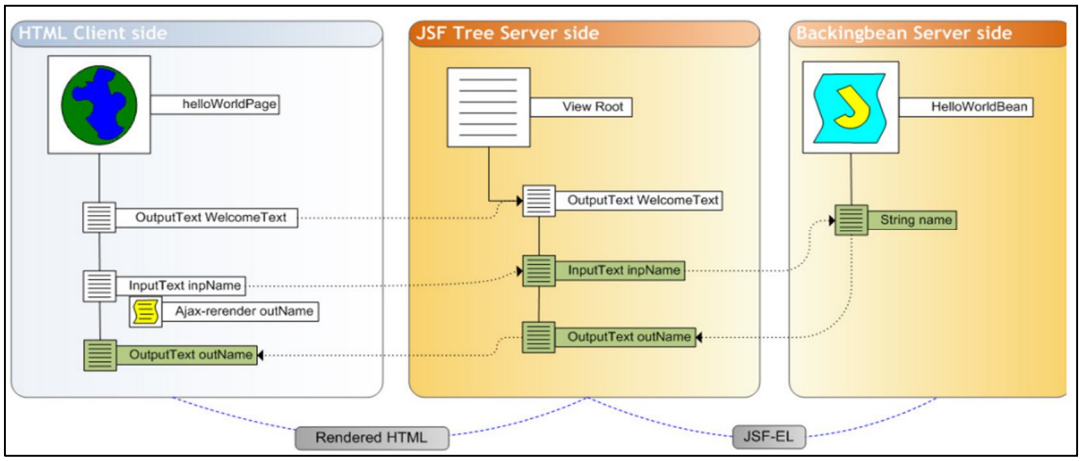


Figura 16. Petición a través de AJAX

1.3. Integración de JSF y EJB3

JSF es un framework de desarrollo de Aplicaciones Web Java, cuyo diseño implementa el patrón MVC al completo, un desarrollador puede implementar una aplicación Web completa utilizando sólo componentes JSF. El problema es que JSF le deja al desarrollador casi todo el trabajo de implementar la capa de modelo sin conocer detalles de los servicios de negocio que brinda esta.

Para un desarrollador realizar una aplicación JSF + EJB3, la forma correcta es crear el modelo en EJB3, y realizar las llamadas a éste modelo desde los Managed Beans de la aplicación.

1.3.1. Enterprise Java Beans

Enterprise Java Beans (también conocidos por sus siglas EJB[14]) es una plataforma para construir aplicaciones de negocio portables, reusables y escalables usando el lenguaje de programación Java.

Los Enterprise JavaBeans son una de las API[15] que forman parte del estándar de construcción de aplicaciones empresariales JEE.

Desde el punto de vista del desarrollador, un EJB es una porción de código que se ejecuta en un contenedor EJB, que provee determinados componentes de servicio.

Los EJB proporcionan un modelo de componentes distribuido estándar del lado del servidor. El objetivo de los EJB es dotar al programador de un modelo que le permita abstraerse de los problemas generales de una aplicación empresarial (concurrencia, transacciones, persistencia, seguridad, etc.) para centrarse en el desarrollo de la lógica de negocio en sí. El hecho de estar basado en componentes permite que éstos sean flexibles y sobre todo reutilizables.

Los EJBs pueden ser vistos como componentes que encapsulan comportamiento y permite reutilizar porciones de código, ya que, desplegados en un contenedor, proveen servicios para el desarrollo de aplicaciones enterprise, servicios que son invisibles para el programador y no ensucian la lógica de negocio con funcionalidades trasversales al modelo de dominio. En la especificación 3.0, los EJB son POJOs[16] es decir para clases Java normales, convencionales que se activan cuando son ejecutados en un contenedor de EJBs.

[14] Acrónimo de Enterprise Java Beans.
[15] Acrónimo de Application Programming Interface
[16] Acrónimo de Plain Old Java Object, es una sigla creada por Martin Fowler, Rebecca Parsons y Josh MacKenzie en septiembre de 2000 y utilizada por programadores Java para enfatizar el uso de clases simples y que no dependen de un framework en especial.

No hay que confundir los Enterprise JavaBeans con los JavaBeans. Los JavaBeans también son un modelo de componentes para la construcción de aplicaciones, pero no pueden utilizarse en entornos de objetos distribuidos al no soportar nativamente la invocación remota (RMI).

1.3.2. Contenedor EJB

Un contenedor de aplicaciones es un entorno que provee diferentes servicios a una aplicación que se desea ejecutar, gestionándolo por nosotros. Dichos servicios incluyen la creación/mantenimiento/destrucción de nuestros objetos de negocio, así como los servicios mencionados en el punto anterior, entre otros. Aunque el contenedor es responsable de la gestión y uso de dichos recursos/servicio.

Una vez escrita nuestra aplicación EJB, podemos desplegarla en cualquier contenedor compatible con EJB, beneficiándonos de toda el trabajo tras bastidores que el contenedor gestiona por nosotros. De esta manera la lógica de negocio se mantiene independiente de otro código que pueda ser necesario, resultando en código que es más fácil de escribir y mantener (además de ahorrarnos mucho trabajo).

El cliente nunca invoca directamente los métodos que poseen las instancias de los enterprise beans, si no que la invocación es recibida por el container EJB y luego delegada a la instancia.

Servicios a los que da soporte el Contendor EJB

- Persistencia mediante JPA[17]
- Invocación remota de métodos mediante RMI/IIOP
- Procesamiento de transacciones y control de concurrencia
    - Cada invocación de un método de un EJB forma parte de un transacción.
- Procesamiento de eventos: JMS Java Message Service
- Servicios de directorio: JNDI Java Naming and Directory Interface
- Seguridad: autenticación, control de acceso
- Publicación de los métodos de negocio de los EJBs como servicios Web mediante JAX-WS

[17] Acrónico de Java Persistence API

Además de estas tareas, el container EJB realiza la administración de las instancias existentes para los objetos, es decir, administra los diferentes ciclos de vida.

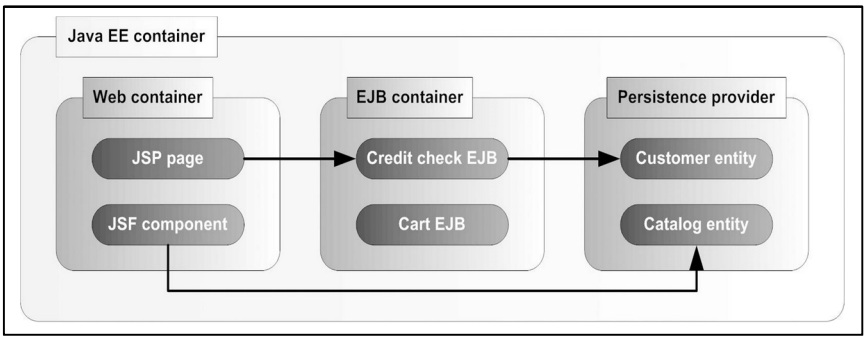


Figura 17. Contenedor EJB

1.3.3. Patrones de Diseño

"Una arquitectura orientada a objetos bien estructurada está llena de patrones. La calidad de un sistema orientado a objetos se mide por la atención que los diseñadores han prestado a las colaboraciones entre sus objetos. Los patrones conducen a arquitecturas más pequeñas, más simples y más comprensibles".

1.3.3.1. Business Delegate

Este patrón es utilizado en sistemas multi-capa distribuidos cuando es necesaria la invocación de métodos remotos para enviar y recibir datos entre las distintas capas del sistema.

Evita que los componentes de la capa de presentación interactúen directamente con los servicios de negocio y sean vulnerables a los cambios en la capa de negocio. Si no se empleara este patrón sería necesario exponer la implementación de la interfaz del servicio de negocio a la capa de presentación obteniendo una fuerte dependencia entre ambas capas.

Este patrón reduce el acoplamiento entre los clientes y la capa de negocio, ocultando los detalles de la implementación de los servicios de negocios y búsquedas y accesos asociados en los sistemas distribuidos.

Otras de las ventajas del uso de este patrón es la posibilidad de dotarlo de la capacidad de almacenar resultados y referencias a servicios remotos que puede mejorar considerablemente el rendimiento.

Puede utilizarse como interfaz entre distintas capas, ya que no se restringe su uso a la separación de las capas de presentación y modelo.

Normalmente el Business Delegate oculta a otro patrón, el Session Facade, ocultando la tecnología utilizada en el modelo.

1.3.3.2. Session Facade

Este patrón representa una capa controladora entre los clientes y la capa de negocio, abstrae la complejidad de esta última y le presenta al cliente una interfaz sencilla reduciendo el acoplamiento y dependencia entre ellos. Encapsula la complejidad de las interacciones entre objetos exponiendo sólo las interfaces requeridas y proporcionando un acceso uniforme a los clientes.

Se trata de una capa sencilla en el modelo que da soporte directo para implementar los casos de uso y oculta la tecnología de acceso a datos realizando las funciones de controlador dentro del modelo.

Las capas inferiores no tienen conocimiento de la capa Facade. Este punto de acceso a las capas inferiores del modelo proporciona un desacoplo que facilita la posibilidad de cambios en las capas inferiores del modelo sin que los clientes se vean afectados.

El patrón Facade, o Fachada, sustituye las interfaces de una serie de clases bajo una sola interfaz por lo que provee un acceso unificado a un subsistema de una aplicación.

Los sistemas se estructuran en subsistemas formados por patrones y clases que los implementan, con dependencias entre ellos. Con la utilización de este patrón se intenta evitar que un cliente, al acceder un sistema (o subsistema) necesite acceder a más de una clase provocando dependencia de cada una de ellas.

1.3.3.3. Service Locator

En más de una ocasión un cliente deberá hacer uso de JNDI[18] ya sea para obtener una conexión a la base de datos, una referencia a la clase home de un enterprise bean, o una referencia a los canales de mensajería. Al ser algo común para muchas componentes, tiende a aparecer código similar múltiples veces y la creación repetida del objeto InitialContext que puede tomar cierto tiempo.

Utilizar un Service Locator permite abstraer todo los usos de JNDI simplificando el código del cliente, creando un único punto de control y mejorando el performance de la aplicación.

En sistemas distribuidos se utiliza para abstraer la utilización de JNDI y ocultar las complejidades asociadas a la creación del contexto, búsquedas de objetos y creación de objetos tipo EJB.

1.4. Implementaciones RIA para JSF.

Actualmente existen muchas librerías de etiquetas JSF que pueden complementar a la implementación de la especificación oficial. A continuación se describen las principales implementaciones RIA para JSF.

1.4.1. RichFaces

RichFaces es un framework de código abierto que añade capacidad Ajax dentro de aplicaciones JSF sin recurrir a JavaScript. RichFaces incluye ciclo de vida, validaciones, conversores y la gestión de recursos estáticos y dinámicos. Los componentes de RichFaces están construidos con soporte Ajax y un alto grado de personalización del "look-and-feel" que puede ser fácilmente incorporado dentro de las aplicaciones JSF.

[18] Acrónimo de "Java Naming Directory Interface": Es una Interfaz de Programación de Aplicaciones que provee interfaces estándar para localizar usuarios, máquinas, recursos y objetos en la red a través de servicios de Nombre y Directorio.

1.4.2. Icefaces

Icefaces es un Framework Ajax que permite desarrollar Aplicaciones RIA proporciona un entorno de presentación web para aplicaciones JSF que mejora el framework JSF estándar y el ciclo de vida con características interactivas basadas en Ajax.

1.4.3. Apache MyFaces Trinidad

Apache MyFaces Trinidad es un framework JSF que incluye una biblioteca de componentes de alta calidad, con soporte para características como: la accesibilidad, idiomas escritos de derecha a izquierda.

Los componentes del proyecto Trinidad también se los denomina ADF Faces, fueron donados por Oracle al proyecto Apache. Tras una evolución puede que sea una de las librerías con más componentes JSF.

CAPÍTULO II

RichFaces

2.1. Historia

RichFaces se crea a partir del framework Ajax4jsf, el cual fue diseñado y desarrollado por Alexander Smirnov en el año 2005. En 2006 Smirnov formando parte de Exadel, continua con el proyecto y para Marzo se da a conocer la primera versión de Ajax4jsf; más adelante el mismo año Exadel decide dividir al framework obteniendo de esta manera Ajax4jsf y RichFaces.

Ajax4jsf se convierte en un proyecto open source, mientras RichFaces se convierte en una librería de componentes comercial.

En el 2007 JBoss (desde 2006 JBoss es una división de Red Hat) y Exadel firman un contrato de colaboración, donde los proyectos Ajax4jsf y RichFaces serán mantenidos por JBoss. JBoss convierte a RichFaces en un proyecto open source, y para septiembre de 2007 por una decisión bilateral se unifican Ajax4jsf y RichFaces bajo el único nombre de RichFaces.

JBoss y Red Hat mantienen a RichFaces como open source[19] hasta la actualidad, sin embargo el soporte técnico y parches se dan mediante la subscripción como desarrollador o como consumidor de los productos en la versión empresarial.

Las diferencia entre el desarrollo de las librerías que componen Rich Faces ocasionó un conjunto de dificultades en la integración, estas dificultades fueron resueltas y la última versión de Rich Faces es la versión 3.3.0 que fue lanzada en enero de 2009.

RichFaces se integra con JSF mediante el empleo de 2 librerías de componentes, la primera Ajax4jsf, la cual permite integrar funcionalidad Ajax sin la necesidad de

[19] Código Abierto es un término que empezó a utilizarse en 1998 por algunos usuarios de la comunidad del software libre.

escribir código JavaScript, la segunda es la librería de interfaz de usuario RichFacesUI diseñada para mejorar las características de JSF en lo referente a interfaz de usuario, permitiendo además una integración con otras librerías de componentes.

2.2.Conceptos básicos del framework RichFaces.

RichFaces es un framework muy útil de código abierto que le permite añadir capacidades de Ajax a sus aplicaciones JSF (usando los componentes estándar JSF), sin la necesidad de escribir código JavaScript y administrar la compatibilidad de JavaScript entre navegadores. Se integra con el ciclo de vida de JSF y otras características de JSF estándar como la validación, la conversión y administración de recursos.

Por otra parte, RichFaces ofrece el gran poder de la skinnability[20]. Esta personaliza la apariencia de las aplicaciones JSF. Se pueden definir diferentes esquemas de color para crear temas personalizados o usar las predefinidas. Además, puede administrar los estilos predefinidos CSS[21] o añadir los propios para cambiar la apariencia de los componentes de interfaz de usuario de la biblioteca de una manera coherente incluso se puede utilizar de forma dinámica XCSS para personalizar los estilos CSS. La característica de skinnability de RichFaces pueden aplicar estilos a los elementos estándar de HTML, como input, select, textarea, fieldset, y así sucesivamente.

RichFaces proporciona dos librerías de componentes:

- **Core Ajax:** la biblioteca principal contiene componentes que son útiles para "ajaxizar" páginas JSF y componentes estándar de JSF. Es muy sencillo definir áreas de Ajax y para invocar peticiones Ajax con el fin de actualizar las áreas de la página que se desee.
- **La interfaz de usuario:** La biblioteca RichFaces interfaz de usuario es un conjunto de componentes avanzados de JSF que Ajax utiliza para agregar características de la interfaz de usuario ricas a sus aplicaciones. Los componentes Ajax fuera de la caja

[20] Skinnability es la habilidad que tiene RichFaces de darle un mismo estilo a todos sus componentes.
[21] Cascading Style Sheets es el lenguaje de hojas de estilo más utilizado en páginas web.

se integran perfectamente con la biblioteca principal. Asimismo, apoyan plenamente los estilos y pueden ser adaptados según las necesidades de los usuarios.

2.3.Características y Arquitectura RichFaces.

2.3.1. Características

Como características de RichFaces tenemos que nos permite:

- Intensificar el conjunto de beneficios JSF al trabajar con Ajax. RichFaces está completamente integrado en el ciclo de vida de JSF. RichFaces permite acceder al action y al valor del listener así como a las acciones de los managed bean, así como invocar a validadores y convertidores durante el ciclo de petición-respuesta de Ajax.
- Añadir capacidad Ajax a aplicaciones JSF. El framework proporciona dos librerías de componentes (Core Ajax y la interfaz de usuario). La librería Core nos permite agregar la funcionalidad Ajax en las páginas que queramos sin necesidad de escribir nada de código JavaScript. RichFaces permite definir eventos en la propia página. Un evento invoca a una petición Ajax, sincronizándose así zonas de la página y componentes JSF después de recibir la respuesta del servidor por Ajax.
- Crear rápidamente vistas complejas basándose en la caja de componentes. La librería UI (Interfaz de usuario) que contiene componentes para agregar características de interfaz de usuario a aplicaciones JSF.

  Se amplía el framework de Rich Faces incluyendo un gran conjunto de componentes "habilitación de Ajax" que extiende el soporte de la página. Además, los componentes de RichFaces están diseñados para ser usados sin problemas con otras librerías de componentes en la misma página, de modo que existen más opciones para el desarrollo de aplicaciones.
- Escribir componentes propios con función soportada por Ajax. El CDK o Kit de Desarrollo de Componentes basado en maven, incluye un generador de código para plantillas JSP utilizando una sintaxis similar. Estas capacidades ayudan a evitar un proceso de rutina de un componente de creación.
- Proporciona un paquete de recursos con clases de aplicación Java. Además de su núcleo, la funcionalidad de Rich Faces para Ajax proporciona un avanzado soporte a la gestión de diferentes recursos: imágenes, código JavaScript y hojas de estilo CSS.

El framework de recursos hace posible empaquetar fácilmente estos recursos en archivos jar junto con el código de los componentes personalizados.

- Generar fácilmente recursos binarios sobre la marcha. Los recursos del framework pueden generar imágenes, sonidos, hojas de cálculo de Excel, etc.
- Crear una moderna interfaz de usuario 'look-and-feel' basadas en tecnología de skins.

  Rich Faces proporciona una función que permite definir y administrar fácilmente diferentes esquemas de color y otros parámetros de la interfaz de usuario, con la ayuda de los parámetros del skin.

  Por lo tanto, es posible acceder a los parámetros del skin desde el código JSP y el código de Java (por ejemplo, para ajustar las imágenes generadas sobre la marcha basadas en la interfaz de usuario). RichFaces viene con una serie de skins predefinidas para empezar, pero también se pueden crear fácilmente skins propios.
- Permite gracias a una herramienta del framework generar casos de prueba para los componentes que estas creando (actions, listeners, etc.).

Los componentes de la interfaz de usuario de Rich Faces vienen preparados para su uso fuera del paquete, así los desarrolladores ahorrarán tiempo y podrán disponer de las ventajas mencionadas para la creación de aplicaciones Web. Como resultado, la experiencia puede ser más rápida y fácil de obtener.

2.3.2. Arquitectura

El framework está implementado como una biblioteca de componentes que agrega capacidad de AJAX en páginas existentes, por lo que no es necesario escribir ningún código JavaScript o sustituir componentes existentes a los nuevos “AJAX widgets”.

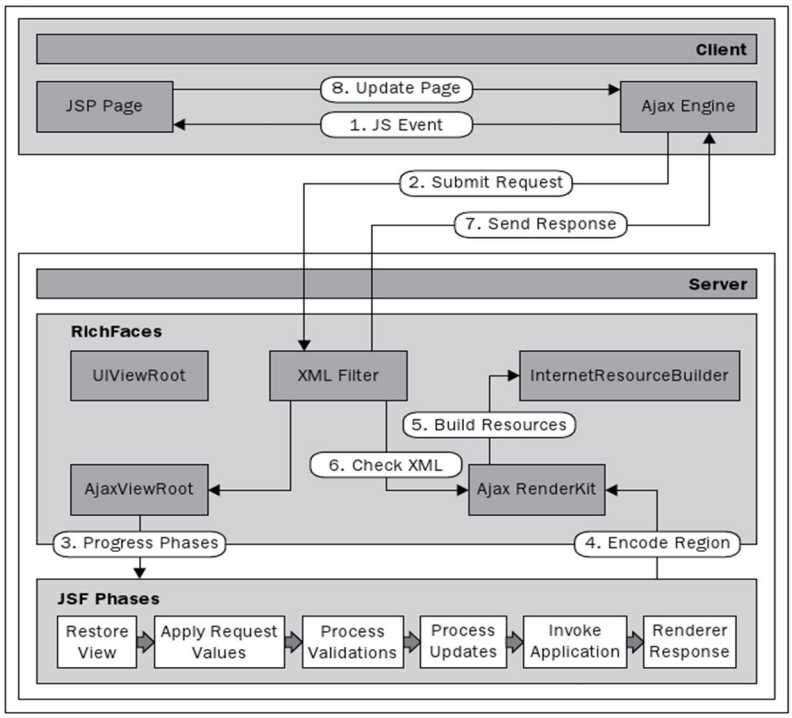


Figura 18. Arquitectura de RichFaces

La arquitectura del framework RichFaces está compuesta de 5 partes:

- Filtro AJAX
- Componentes de filtro y de acción de AJAX
- Contenedores AJAX
- Skinnability
- Un motor de JavaScript

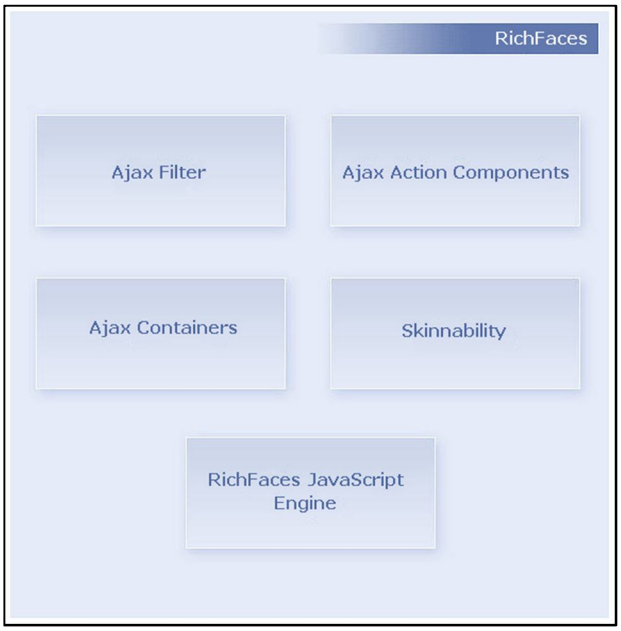


Figura 19. Estructura de Componentes Core Ajax

- **Filtro Ajax (Ajax Filter):** Para obtener todos los beneficios de RichFaces, se debe registrar un filtro en el archivo de configuración de la aplicación web. Este filtro reconoce múltiples tipos de peticiones. El diagrama de secuencia de la siguiente figura muestra la diferencia en el tratamiento de una página JSF regular y una solicitud Ajax. En el primer caso todo el árbol completo de JSF será codificado, mientras que en la segunda opción depende del tamaño de la región Ajax. Como se puede ver, en el segundo caso el filtro analiza el contenido de la respuesta Ajax antes de enviarlo al cliente.

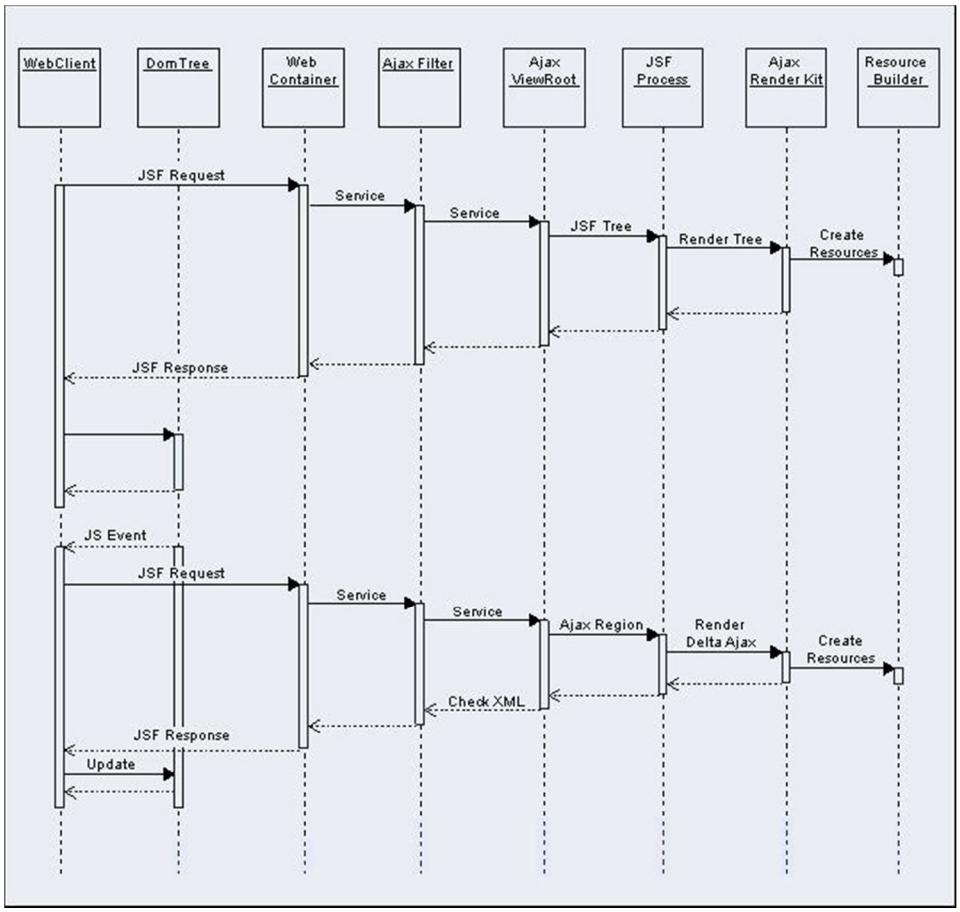


Figura 20. Diagrama de secuencia de una página JSF regular y una solicitud Ajax

En ambos casos, la información necesaria sobre recursos estáticos o dinámicos que pide la aplicación será registrada por la clase "ResourceBuilder". Cuando llega una petición de un recurso, el filtro de RichFaces comprueba la caché de recursos para ese recurso y si está, el recurso se envía al cliente.

De lo contrario, el filtro busca el recurso dentro de los que estén registrados por el ResourceBuilder. Si el recurso está registrado, el filtro de RichFaces enviará una petición al ResourceBuilder para crear (entregar) el recurso. La figura siguiente muestra la forma de procesar la petición de un recurso.

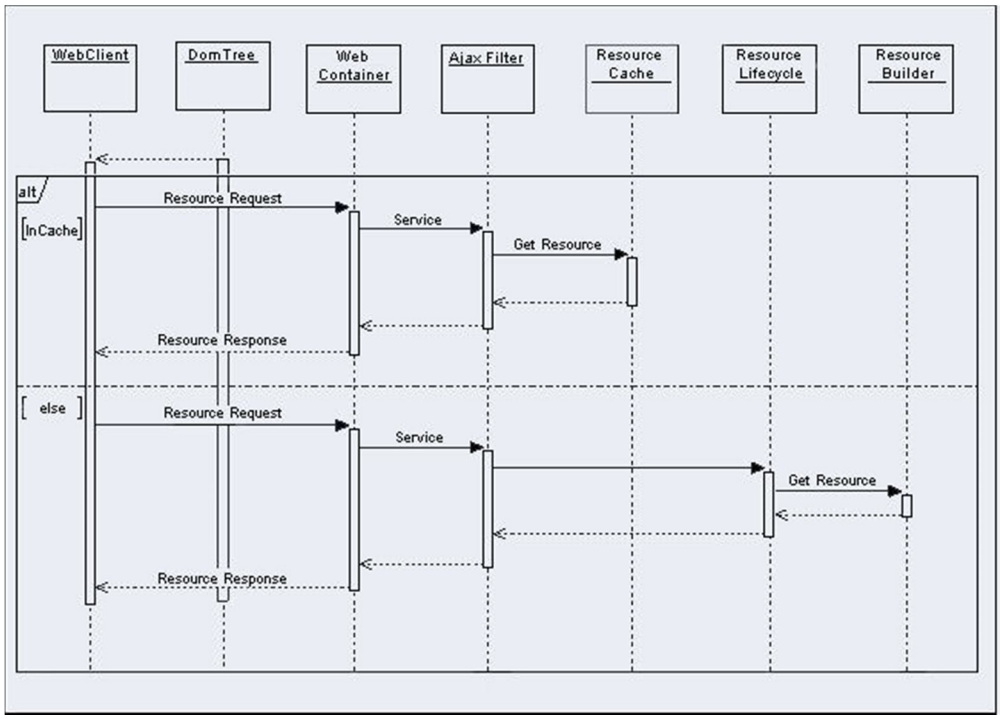


Figura 21. Petición de recursos

- **Componentes de acción AJAX (Ajax Action Components)**. Los componentes de acción más comunes que se utilizan en AJAX son: AjaxCommandButton, AjaxCommandLink, AjaxPoll, AjaxSupport, etc. Pueden utilizar para enviar peticiones Ajax desde el cliente.
- **Contenedor AJAX (Ajax Containers)**. El contenedor Ajax es una interfaz que describe un área de la página JSF que debería ser decodificado durante la solicitud Ajax. AjaxViewRoot y AjaxRegion son implementaciones de esta interfaz.
- **Motor de JavaScript (JavaScript Engine)**. Es un componente de Rich Faces que corre del lado del cliente y permite actualizar las diferentes áreas de la página JSF basado en la información de la respuesta del servidor.
- **Skinnability.** Este componente permite establecer características visuales para los componentes de las páginas JSF, extienden los beneficios de las CSS (Cascade Style Sheets) permitiendo definir estilos visuales para

ventanas emergentes, colores para cabeceras de secciones, etc. que no son posibles de lograr con una CSS.

Rich Faces usa Skinnability para facilitar el trabajo de ajustes de efectos visuales sobre la marcha para una interfaz de usuario, cada componente en Rich Faces puede obtener su estilo de un skin predefinido o de una CSS definida por el usuario.

Skinnability no puede reemplazar definitivamente las CSS, ni elimina su uso, más bien pueden ser usadas de manera conjunta para obtener una mejor apariencia visual en lo referente a combinación de colores y fuentes.

Otra característica importante son los themes. Cualquier número de temas definidos por medio de un archivo de propiedades pueden ser creados con diversos esquemas de color.

Cuando se fija un tema particular, el componente se referirá a ese tema y generará los colores y los estilos basados en ese tema. Esto significa que se puede cambiar fácilmente la apariencia y la sensación del uso simplemente cambiando a otro tema. RichFaces proporciona un sistema de temas predefinidos: default, plain, emeraldTown, blueSky, wine, japanCherry, ruby, classic, deepMarine

Cada componente tiene un archivo de XCSS un formato de archivo especial que combina la flexibilidad de XML y del CSS que realiza el trazado de los selectores del CSS a las características de un desarrollo particular.

2.4. Patrones de Diseño

Debido a que RichFaces extiende las capacidades de las JSF, implementa el patrón de diseño MVC (Model View Controller), para el patrón de diseño MVC, el Modelo es la representación de la persistencia (datos) de la aplicación, la vista corresponde a la interfaz de usuario, mientras el controlador sirve como nexo entre el modelo y la vista.

RichFaces como framework se corresponde de manera específica a la sección de Vista -Controlador con las JSP y los Backing Beans.

2.5. Integración de RichFaces con JSF.

RichFaces al integrarse con JSF usa el lenguaje Java para su funcionamiento, la manera en la que lo hace es por medio de tag libraries. Una tag library no es más que un conjunto de clases compiladas en un archivo de extensión .jar, que se incluyen dentro del código HTML mediante la etiqueta <@taglib uri=" " >, con la finalidad de extender las capacidades de las JSP (Java Server Pages) e invocar funcionalidad de las clases que conforman la mencionada tag library.

RichFaces se integra de manera nativa con las implementaciones de la especificación JSF 1.2.

```
<%@ taglib prefix="a4j" uri="http://richfaces.org/a4j"%>
<%@ taglib prefix="a4j" uri=" http://richfaces.org/rich "%>
```

RichFaces permite definir por medio de etiquetas de JSF diferentes partes de una página JSF que se desee actualizar con una solicitud Ajax, proporcionando así varias opciones para enviar peticiones Ajax al servidor. El esquema de la solicitud de procesamiento de flujo sería el que se muestra en la siguiente figura:

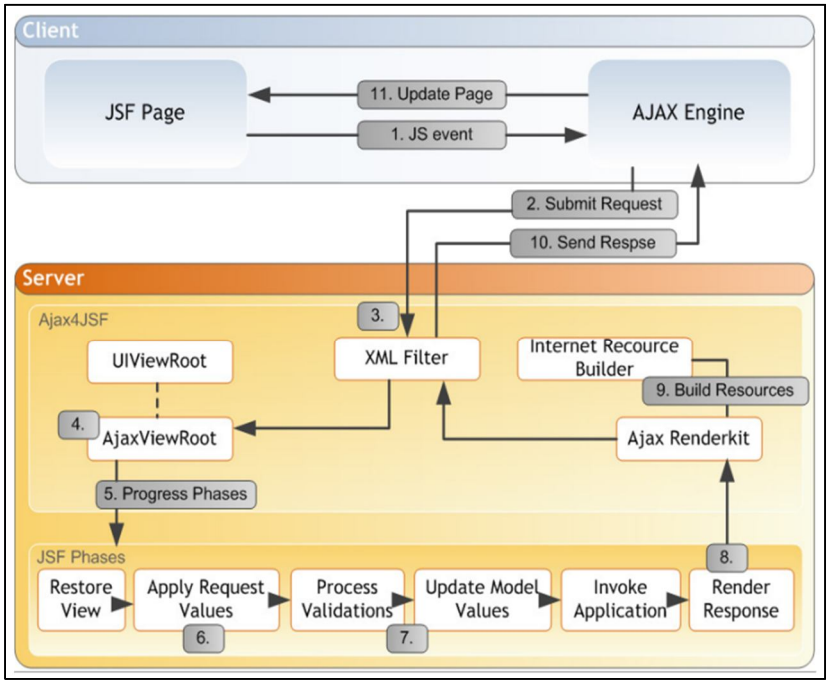


Figura 22. Flujo de Solicitud de Procesamiento

2.6. Versiones

La versión de Richfaces 3.3.0.GA, publicada el 13 de enero del 2009, es un lanzamiento importante con nuevas características y mejoras, incluyendo dos nuevos componentes: <rich:editor> y <a4j:queue>.

La versión de Richfaces 3.3.2.GA, publicada el 14 de octubre del 2009 incorpora un conjunto considerable de mejoras y nuevas funcionalidades como las etiquetas para layout (disposición de los elementos) de la página.

La versión de Richfaces 3.3.3.GA, publicada el 13 de Abril del 2010 incluye soporte básico de Java Server Faces (JSF) 2.0.

El 30 de marzo del 2011, salió la versión 4.0.0 de RichFaces, la cual se basa al 100% en JSF 2.

2.7. Requisitos

RichFaces fue desarrollado con una arquitectura de código abierto para ser compatible con la más amplia variedad de entornos.

Para poder trabajar con RichFaces es necesario disponer de una aplicación JavaServer Faces. Además hay que tener en cuenta la versión de los diferentes componentes implicados.

2.7.1. Versiones de Java soportadas:

- JDK 1.5 y superiores.

2.7.2. Implementaciones de JavaServer Faces soportadas:

- Sun JSF 1.2 – 2.x
- MyFaces 1.1.1 - 1.2
- Facelets JSF 1.1.1 - 1.2
- Seam 1.2. - 2.0

2.7.3. Servidores soportados

- Apache Tomcat 5.0 - 6.0
- IBM WebSphere 5.1 - 6.0
- Oracle AS/OC4J 10.1.3
- Sun Application Server 8 (J2EE 1.4)
- Glassfish (J2EE 5)
- JBoss 3.2 - 4.2.x

2.7.4. Navegadores admitidos

- Internet Explorer 6.0 - 7.0
- Firefox 1.5 - 2.0
- Opera 8.5 - 9.0
- Netscape 7.0
- Safari 2.0

2.8.Costos de Licencias

Todas las licencias de RichFaces son del tipo LGPL (GNU Lesser General Public License), con lo cual le permiten integrarse en proyectos de software no libre.

2.9.Componentes RichFaces

2.9.1. Etiquetas de Soporte Ajax de RichFaces

El componente de esta sección le permite agregar fácilmente capacidades AJAX a otros componentes, así como gestionar las peticiones Ajax.

- **<a4j:ajaxListener>:** Añade un detector de acción para un componente principal y funciona como los componentes de JSF <f:actionListener> o <f:valueChangeListener> pero con la diferencia de que la petición se hace al contenedor Ajax.

- **<a4j:actionparam>:** Combina la funcionalidad de <f:param> y <f:actionListener> y permite asignar el valor de la propiedad del managed bean directamente utilizando el atributo "assignTo".
- **<a4j:form>:** Es similar al <h:form> con la diferencia de que se puede enviar previamente el contenido al contenedor Ajax.
- **<a4j:region>:** Determina un área a decodificar en el servidor después de la petición Ajax.
- **<a4j:support>:** Etiqueta que se puede añadir a cualquier otra etiqueta JSF para dotarla de funcionalidad Ajax. Permite al componente generar peticiones asíncronas mediante eventos (onclick, onblur, onchange,...) y actualizar campos de un formulario de forma independiente, sin recargar toda la página.
- **<a4j:commandButton>:** Botón de envío de formulario similar a <h:commandButton> de JSF. La principal diferencia es que se puede indicar que únicamente actualice ciertos componentes evitando la recarga de todo el formulario.
- **<a4j:commandLink>:** Comportamiento similar a <a4j:commandButton> pero en un link.
- **<a4j:jsFunction>:** Se utiliza para pasarle un valor automáticamente a una función Javascript tras recibirlo del servidor.
- **<a4j:poll>:** Realiza cada cierto tiempo una petición al servidor.
- **<a4j:push>:** Realiza envío de mensajes de forma asíncrona a diferencia del <a4j:poll>.
- **<a4j:queue>:** Es un conjunto de peticiones Ajax en cola enviado desde el cliente. Los componentes RichFaces con una función de Ajax puede hacer referencia a la cola para optimizar las peticiones Ajax.
- **<a4j:status>:** Muestra el estado de la petición Ajax. Hay 2 estados posibles: procesando petición y petición terminada. Por ejemplo mientras dure el proceso de la llamada al servidor y la evaluación de la petición se puede mostrar el texto " procesando..." y cuando termine la petición y se devuelva la respuesta a la página se cambia el texto por "petición finalizada".

2.9.2. Etiquetas de Recursos/Beans Handling de RichFaces

- **<a4j:loadBundle>:** Este componentes es similar al mismo componente de la biblioteca JSF Core. Carga un paquete de recursos para el Locale de la vista actual.
- **<a4j:loadScript >:** Inserta en la página las funciones Javascript contenidas en un archivo .js

- **<a4j:loadStyle>:** Este componente se encarga de insertar hojas de estilo .css

2.9.3. Validadores Ajax

- **<rich:ajaxValidator>:** Es un componente diseñado para proporcionar la validación Ajax en el interior de las entradas JSF
- **<rich:beanValidator>:** Componente diseñado para proporcionar la validación usando el modelo Hibernate basado en restricciones.
- **<rich:graphValidator>:** Componente permite registrar validadores Hibernate para los componentes de entrada múltiple.

2.9.4. Etiquetas de Salida de Ajax de RichFaces

- **<a4j:include>:** Se utiliza para incluir en la página el contenido de otra de acuerdo a la definición que se haga en las reglas de navegación del faces-config. Es decir la siguiente página a cargar de acuerdo a la navegación especificada se cargaría en la vista actual.
- **<a4j:mediaOutput>:** Componente que permite mostrar contenido multimedia como imágenes, vídeos, archivos sonoros, etc.
- **<a4j:outputPanel>:** Se utiliza para agrupar componentes para aplicarles similares propiedades, por ejemplo a la hora de actualizar sus valores tras la petición Ajax.

2.9.5. Etiquetas Misceláneas de Ajax de RichFaces

- **<a4j:page>:** Es un componente obsoleto, utilizado para resolver los problemas de incompatibilidad entre Ajax4jsf y MyFaces. El componente codifica toda la estructura de la página html
- **<a4j:htmlCommandLink>:** Este componente es similar al mismo componente de la biblioteca JSF, diferenciándose en la generación de enlaces y la resolución de problemas que se produce cuando se utiliza un componente original.
- **<a4j:log>:** Carga en la página una consola que muestra las trazas de los logs que devuelve el contenedor Ajax.

2.9.6. Etiquetas de Iteración de Datos de RichFaces

- **<a4j:repeat>:** Etiqueta para iterar sobre una colección y mostrar todos sus campos.

- **<rich:column>:** El componente está destinado para la representación de las filas de los componentes UIData.
- **<rich:columnGroup>:** El componente combina columnas en una fila para organizar subpartes complejas de una tabla.
- **<rich:columns>:** Es un componente, que le permite crear una columna dinámica.
- **<rich:dataDefinitionList>:** Es un componente que presenta una lista de definición de un modelo.
- **<rich:dataFilterSlider>:** Un control basado en la acción, Este componente se utiliza para crear un filtro de los datos de una tabla.
- **<rich:dataGrid> :** Este componente presenta los datos en una cuadrícula con una función de apoyo para las actualizaciones de Ajax.
- **<rich:dataList> :** El componente presenta una lista desordenada de datos.
- **<rich:dataOrderedList>:** El componente presenta una lista ordenada de datos.
- **<rich:datascroller>:** El componente es diseñado para proporcionar la funcionalidad de los cuadros de desplazamiento utilizando solicitudes Ajax.
- **<rich:dataTable>:** Este componente nos permite crear tablas de datos.
- **<rich:subTable>:** El componente se utiliza para la inserción de subtablas.
- **<rich:extendedDataTable>:** Se extiende del componente estándar <rich:dataTable>, dando características como el soporte integrado de datos de desplazamiento, una función de selección, agrupación de filas, ordenar, filtrar, y así sucesivamente.
- **< rich:scrollableDataTable >:** Añade características adicionales al componente de tabla estándar, como el desplazamiento de datos integrada, fila dinámica a buscar, ordenar, filtrar, soporte para arrastrar y soltar, y así sucesivamente.

2.9.7. Etiquetas de soporte Drag-and-drop de RichFaces

- **<rich:dndParam>:** Este componente se utiliza para pasar parámetros durante las operaciones arrastrar y soltar.
- **<rich:dragIndicator>:** Este componente define lo que aparece bajo el cursor del ratón durante las operaciones de arrastrar y soltar. El indicador drag puede mostrar información sobre los elementos arrastrados.
- **<rich:dragSupport>:** Este componente define una "zona de arrastre" que puede arrastrar a cualquier componente que apoya las operaciones de soltar que es una "zona de soltar".

- **<rich:dropSupport>:** Se define una "zona de soltar" para operaciones de arrastrar y soltar. Cuando un elemento arrastrable se mueve y se dejó caer en el área de colocación, una petición Ajax para este evento es iniciado.
- **<rich:dragListener>:** El componente se utiliza para ejecutar un método del backend beans después de una operación de arrastre.
- **<rich:dropListener>:** El componente se utiliza para ejecutar un método del backend beans después de una operación de soltar.

2.9.8. Etiquetas de Menú de RichFaces

- **<rich:contextMenu>:** Se utiliza para crear un menú contextual, que se activa después de un evento definido por el usuario en cualquier elemento de la página.
- **<rich:dropDownMenu>:** Este componente se utiliza para crear múltiples menús desplegables.
- **<rich:menuGroup>:** Este componente se utiliza para definir un grupo de elementos expandibles.
- **<rich:menuItem>:** Este componente se utiliza para la definición de un único punto dentro de una lista emergente.
- **<rich:menuSeparator>:** Este componente se utiliza para la definición de un separador horizontal que puede ser colocado entre los grupos o los temas del programa.

2.9.9. Etiquetas de Árboles de RichFaces

- **<rich:tree>:** Este componente presenta una estructura de árboles de datos con capacidad de arrastrar y soltar.
- **<rich:treeNode>:** Este componente describe un nodo en el componente <rich:tree>.
- **<rich:changeExpandListener>:** Se utiliza para ejecutar un método del backbeans sobre un evento expandir / colapsar en un nodo del árbol.
- **<rich:nodeSelectListener>:** Se utiliza para ejecutar un método del backbeans después de la selección de un nodo de árbol.
- **<rich:recursiveTreeNodesAdaptor>:** En él se definen los modelos de datos y nodos de proceso de forma recursiva. Es una extensión de <rich:treeNodesAdaptor>.

- **<rich:treeNodesAdaptor>:** En él se definen los modelos de datos y crea representaciones para ellos.

2.9.10. Etiquetas de salida de RichFaces

- **<rich:modalPanel>:** Es un componente de RichFaces que básicamente es una ventana que se superpone a la ventana principal. La ventana principal se desactiva mientras esté el modalPanel en uso.
- **<rich:paint2D>:** Lo mismo que "pintar" (Graphics2D) en componentes "SWING", se genera una imagen mediante el uso de un método del managed bean.
- **<rich:panel>:** Se trata de un panel (con o sin cabecera).
- **<rich:panelBar>:** El componente está diseñado para agrupar una o más <rich:panelBarItem> en un panel interactivo.
- **<rich:panelBarItem>:** Es un componente que implementa un panel secundario dentro de un <rich:panelBar>.
- **<rich:panelMenu>:** El componente presenta un menú en línea vertical en una página.
- **<rich:panelMenuGroup>:** Este implementa un grupo expandible de ítems dentro de un panelMenu o de otros grupos.
- **<rich:panelMenuItem>:** Este es un item de panelMenu.
- **<rich:progressBar>:** El componente <rich:progressBar> muestra una barra de progreso que muestra el estado actual de un proceso.
- **<rich:separator>:** Es una línea horizontal personalizable para usar como separador en una presentación.
- **<rich:simpleTogglePanel>:** Un panel plegable que muestra/oculta el contenido después de la activación de un control de encabezado.
- **<rich:spacer>:** Este componente presenta una imagen transparente para usar como separador en una presentación.
- **<rich:tabPanel>:** Un panel que contiene otros paneles en forma de tabs.
- **<rich:tab>:** Un panel tab hijo dentro de un componente <rich:tabPanel>.
- **<rich:togglePanel>:** Este componente es un contenedor llamado faceta que puede mostrar una faceta específica invocándolo con el componente <rich:toggleControl>. Sólo una faceta a la vez se muestra.
- **<rich:toggleControl>:** Se utiliza para cambiar entre los <rich:togglePanel>.
- **<rich:toolBar>:** Una barra horizontal que acepta cualquiera de los componentes de JSF como hijos.

- **<rich:toolBarGroup>:** Un grupo de elementos de la barra de herramientas.
- **<rich:toolTip>:** Este presenta un componente que muestra una ventana emergente no modal para un evento específico.

2.9.11. Etiquetas de entrada de RichFaces

- **<rich:calendar>:** Presenta un calendario mensual para la fecha y hora.
- **<rich:comboBox>:** Proporciona un comboBox editable.
- **<rich:fileUpload>:** Permite subir un archivo al servidor.
- **<rich:inplaceInput>:** Permite desplegar y editar información.
- **<rich:inplaceSelect>:** Se utiliza para seleccionar algo así como un dropDown.
- **<rich:inputNumberSlider>:** Es una barra deslizante de selección de valor.
- **<rich:inputNumberSpinner>:** Genera un input numérico con botones de ajuste de valor.
- **<rich:suggestionbox>:** Genera sugerencias para un inputText.

2.9.12. Etiquetas de selección de RichFaces

- **<rich:listShuttle>:** Este componente se utiliza para mover los temas elegidos de una lista a otra con su facultativo reordenamiento.
- **<rich:orderingList>:** Es útil para ordenar los elementos en una lista.
- **<rich:pickList>:** Se utiliza para seleccionar elementos de una lista a otra. Las diferencias con <rich:listShuttle> son que es simple y no utiliza un modelo personalizado para los elementos, sólo la etiqueta <f: SelectItem>.

2.9.13. Etiquetas Misceláneas de RichFaces

- **<rich:componentControl>:** Este permite llamar a funciones API de JavaScript en los componentes definidos después de los acontecimientos.
- **<rich:effect>:** Utiliza la librería script.aculo.us de JavaScript para que reconozca los efectos de los elementos de otra página.
- **<rich:gmap>:** Este presenta el código necesario para mostrar un elemento del Google map
- **<rich:virtualEarth>:** Este presenta el código necesario para mostrar un elemento del virtualEarth map.

- **<rich:hotKey>:** Este componente permite al usuario registrar las teclas de acceso para la página o elementos particulares y por lo tanto ejecutar el código del lado del cliente.
- **<rich:insert>:** Se utiliza para insertar código fuente y destacar que esté de acuerdo con el tipo (por ejemplo Java, XHTML, C, Groovy, y así sucesivamente).
- **<rich:jQuery>:** Se instala la funcionalidad del framework de JavaScript jQuery y lo conecta con otros objetos DOM (o componentes).
- **<rich:message>:** El componente se utiliza para hacer un solo mensaje a un componente específico.
- **<rich:messages>:** El componente presenta todos los mensajes de los componentes de JSF (similar a <h:messages>, pero con soporte para Ajax y otras características).

CAPÍTULO III

ICEfaces

3.1.Introducción

Icefaces nace como una implementación de JSF, que surgió en el año 2004 como la oferta de Sun ante el despropósito de J2EE al programar aplicaciones web. Las dificultades de integración y las limitaciones para desarrollar en un "frontend" fueron las causas que motivaron a IceSoft desarrollar una librería de componentes ricos para JSF con un framework avanzado para la integración sencilla de las funcionalidades AJAX dentro del desarrollo de aplicaciones de negocio, con características similares a RichFaces.

3.2.Conceptos básicos del framework ICEFaces.

ICEFaces es un marco integrado AJAX de aplicaciones Java que permite a los desarrolladores de aplicaciones AJAX JEE, crear y desplegar aplicaciones RIA, en Java puro o ampliar las existentes sin costo alguno.

ICEFaces aprovecha los estándares JEE para el desarrollo de aplicaciones, así como las herramientas y entornos de ejecución.

Es un framework de desarrollo web creado sobre la especificación JSF, con capacidad de procesamiento de solicitudes AJAX, que permite a los desarrolladores web construir aplicaciones con contenido enriquecido, utilizando el lenguaje Java.

Provee un ambiente de presentación amplio de Java Server Faces (JSF) que mejora la estructura de un JSF normal y el ciclo de desarrollo, basado en las características interactivas de AJAX. Reemplaza los servicios de JSF normal con

servicios DOM[22], e incluye un puente de AJAX para entregar los cambios que presenta el navegador del cliente y comunicar los eventos de interacción de usuario al servidor de las aplicaciones JSF.

Proporciona una colección de componentes AJAX extensa que facilita el desarrollo rápido de aplicaciones basadas en características interactivas. Permite al programador incluir una serie de etiquetas AJAX en sus páginas JSP o XHTML de tal manera que el código AJAX es generado por el propio framework automáticamente.

3.3.Características y Arquitectura ICEfaces.

3.3.1. Características

ICEfaces es considerado un framework que integra funcionalidad AJAX y permite a los desarrolladores Java EE crear aplicaciones RIA (Rich Internet Applications) entre sus características principales tenemos:

- Es de código abierto y se basa en estándares.
- Extiende las funcionalidades de Java Server Faces.
- Desarrollo en Java puro: Permite desarrollar aplicaciones web en Java puro sin JavaScript.
- Actualización incremental de la página: Facilita la actualización de la página de manera incremental.
- Transparencia de la perspectiva de desarrollo: Estos rasgos de la presentación de ICEFaces son completamente transparentes de la perspectiva de desarrollo de la aplicación.
- Actualización de la presentación de forma asíncrona: Las aplicaciones JSF normales pueden entregar sólo cambios de la presentación en la respuesta a un evento iniciado por el usuario. ICEFaces introduce un mecanismo accionador que permite a la lógica de aplicación residente en el servidor realizar los cambios de presentación en el cliente, en respuesta a los cambios en el estado de la

[22] Document Object Model o Modelo de Objetos del Documento es una interfaz de programación de aplicaciones para acceder, añadir y cambiar dinámicamente contenido estructurado en documentos con lenguajes como JavaScript.

aplicación. Esto posibilita a los desarrolladores diseñar sistemas que entreguen datos al usuario en un tiempo real.

- Integración AJAX con JEE: Integra funcionalidad AJAX y permite a los desarrolladores JEE crear aplicaciones RIA (Rich Internet Applications) de una manera sencilla.
- Aparta completamente al desarrollador de realizar peticiones AJAX: No hacen falta etiquetas especiales, ICEFaces se encarga de enviar sólo la información necesaria entre el cliente y servidor.

3.3.2. Arquitectura

ICEfaces es un framework integrado que permite crear y desplegar aplicaciones JEE utilizando el framework JSF la arquitectura propuesta por ICEFaces para integrar la tecnología AJAX al ciclo de vida JSF:

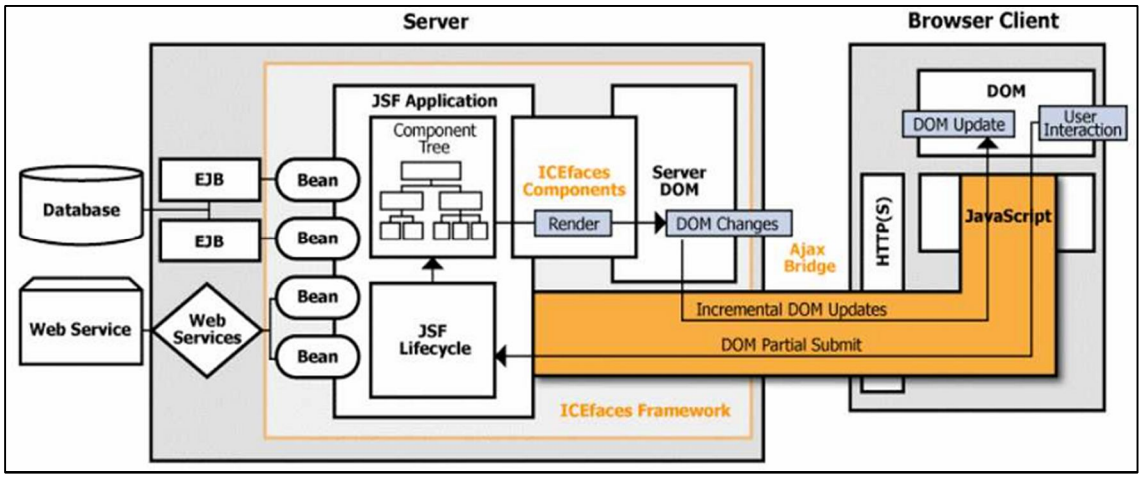


Figura 23. Arquitectura de ICEFaces

Claramente sobresalen 3 elementos principales en la arquitectura ICEFaces:

- **ICEFaces Framework** es la extensión de JSF poniendo la atención en la fase de renderizado. Utiliza la técnica direct-to-DOM para realizar las actualizaciones incrementales lo que resulta en una mejora de la experiencia para el usuario.
- **AJAX Bridge** es el encargado de establecer la comunicación entre el cliente y el servidor, y para ello dispone de componentes en ambos extremos.

- **ICEFaces Suite Components** que permiten realizar el submit parcial según las acciones del usuario sobre ellos, liberando al desarrollador de introducir código JavaScript de bajo nivel.

Se presenta una figura con los principales elementos de la arquitectura de una aplicación en JSF integrada con ICEFaces:

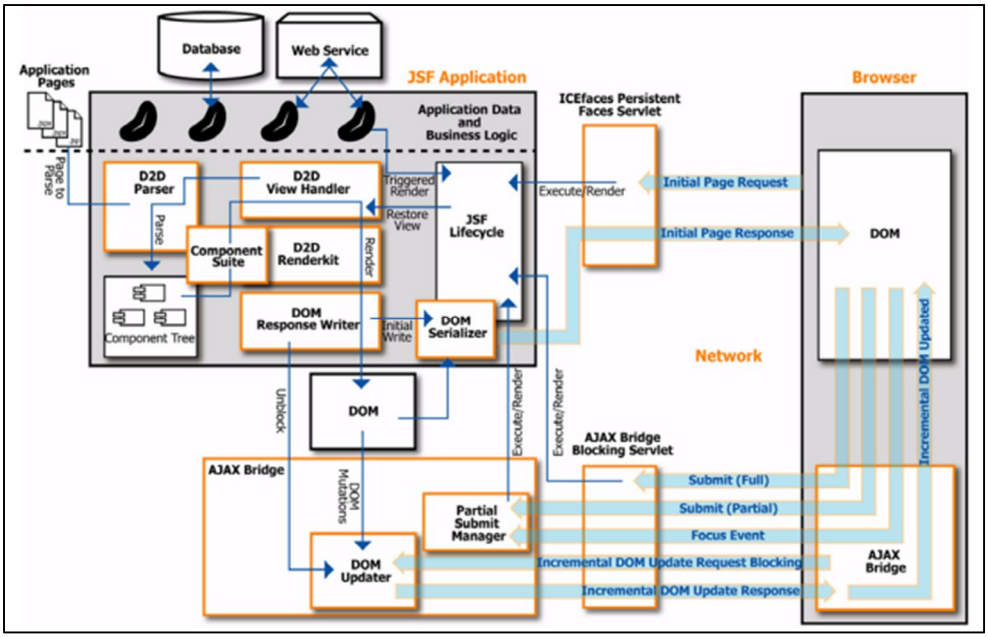


Figura 24. Principales elementos de la Arquitectura ICEFaces

Los principales elementos de la arquitectura ICEFaces son:

- **Persistent Faces Servlet:** Las URLs con extensión ".iface" son mapeadas por el servlet "Persistent Faces Servlet". Cuando se realiza una petición de la página inicial en la aplicación, éste servlet se hace responsable de la ejecución del ciclo de vida JSF para la petición asociada.
- **Blocking Servlet:** Responsable de gestionar el bloqueo y desbloqueo de las peticiones después de iniciado un evento en la página.
- **D2D ViewHandler:** Responsable de establecer el Direct-to-DOM (D2D), incluyendo la inicialización de la 'DOM Respuesta-Escritura'. También invoca al Parser para analizar el árbol de componentes JSF en la página inicial.
- **El D2D Parser:** Responsable de agrupar el árbol de componentes de un documento JSP.
- **El D2D RenderKit:** Responsable de establecer un árbol del componente en el DOM vía el DOM Respuesta-Escritura durante un paso normal de un JSF.

- **DOM Response Writer:** Responsable de escribir en el DOM. También inicia la serialización del DOM para la primera prestación, y desbloquea el DOM Updater para actualizaciones de DOM incrementales.
- **DOM Updater:** Responsable de agrupar las mutaciones de DOM en una sola actualización de DOM incremental. DOM Updater bloquea el DOM incremental para actualizar las demandas hasta que estén completas.
- **Client-side AJAX Bridge:** Responsable de la actualización DOM en curso generada por la solicitud y la respuesta del proceso. También es el encargado de centrar la gestión y de presentar el proceso.
- **DOM Serializer:** Responsable de la fabricación en serie del DOM para la respuesta de la página.
- **Component Suite:** Proporciona componentes 'rich JSF' con influencia AJAX, dando los elementos básicos para aplicaciones ICEFaces del lado del Cliente.

3.4.Integración de ICEFaces con JSF.

ICEFaces es un framework AJAX de Software Libre que utiliza las tecnologías JEE para crear los componentes. Permite utilizar técnicas AJAX de forma transparente. Una de las características más interesantes es que ICEFaces mejora el estándar JSF y el ciclo de vida. ICEFaces reemplaza la el renderers HTML estándar de JSF con Direct-to-DOM (D2D) renderers y introduce un ligero Ajax bridge para presentar los cambios en el cliente cliente desde la parte del servidor.

3.5.Requisitos

A continuación identificamos las plataformas que soportan el desarrollo basado en ICEFaces.

3.5.1. Integración con IDEs.

ICEFaces aporta integración básica con una alto número de interfaces de desarrollo adaptados por la comunidad de desarrolladores. Destacamos los siguientes:

- Eclipse.
- NetBeans.
- MyEclipse Enterprise Workbench.
- Oracle JDeveloper 10g Studio Edition.

3.5.2. Servidores de aplicaciones.

ICEFaces proporciona integración con los siguientes servidores de aplicaciones J2EE:

- Apache Tomcat.
- BEA Weblogic Server.
- JBoss Application Server.
- IBM Websphere Application Server.
- Oracle Application Server Container for J2EE (OC4J).
- Sun GlassFish.
- Sun Java System Application Server.

3.5.3. Portales y frameworks Java EE.

ICEFaces soporta los siguientes portales y frameworks de desarrollo JEE:

- Liferay Portal 4.3.
- JBoss Seam 1.3.
- JavaServer Faces (JSF) 1.2 – 2.x
- Facelets.

3.5.4. Navegadores soportados.

Se ha comprobado la compatibilidad de ICEFaces en los siguientes navegadores:

- Chrome
- Internet Explorer 6.x+, 7.0, 8.0
- Firefox 3.x +
- Opera 9.x +

3.6.Componentes ICEFaces

3.6.1. Etiquetas de comandos de ICEFaces

Etiquetas de comandos cubren una amplia gama de usos, tales como el envío de formularios, navegación y otros tipos de interacción del usuario.

- **<ice:commandButton**>: Muestra un botón que puede estar vinculado a una acción en un método de backing bean mediante la especificación del bean y el método en el atributo ActionListener. El botón puede ser sustituido por una imagen especificando la dirección URL de la imagen en el atributo de la imagen.
- **<ice:commandLink>:** Muestra un enlace que puede ser vinculado a una acción de la aplicación o método detector de acción en un backing bean mediante la especificación del bean en los atributos de acción y actionListener.
- **<ice:commandSortHeader>:** Este componente es usado junto con un datatable. El commandSortHeader se utiliza como complemento facet del encabezado de una columna que permite al usuario hacer clic para cambiar el orden de los datos de la tabla, ya sea ascendente o descendente según los valores de la columna.

3.6.2. Etiquetas de datos de ICEFaces

ICEFaces incluye una serie de etiquetas de datos útiles para representar los datos de manera diferente, incluyendo tablas, listas y conjuntos de resultado de desplazamiento.

- **<ice:dataTable>:** Este componente muestra los objetos de una colección java, donde cada objeto es una fila (row) de la tabla y cada columna (column) coincide con cada atributo del objeto. En un DataTable, se puede configurar opciones como el valor del número de filas a mostrar en una página o agregar otras funcionalidades como el ordenamiento de las filas, añadiendo un link en la(s) cabecera(s) del DataTable.
- **<ice:column>:** Muestra una fila y columna de información del backing bean DataModel en un formato de tabla. Esta etiqueta se puede utilizar en la misma forma que h:column.
- **<ice:columns>:** Muestra una fila y columna de información del backing bean DataModel en un formato de tabla. El atributo rows especifica el número de filas que se mostraran a partir de la fila especificada en el atributo first. El valor por defecto para el atributo first es 0.
- **<ice:dataPaginator>:** Se utiliza en conjunto con un DataTable. El dataPaginator puede ser utilizado para presentar un conjunto de datos en varias páginas en base al número de filas que son presentadas en un datatable.
- **<ice:rowSelector>:** Permite la selección de filas para un DataTable.

3.6.3. Etiquetas Generales de ICEFaces

- **<ice:effect>:** Añade efectos al componente principal.
- **<ice:form>:** Presenta un elemento "form" de HTML.
- **< ice:graphicImage >:** Presenta un elemento "img" de HTML.
- **<ice:message>:** Muestra un solo mensaje para un componente específico.
- **<ice:messages>:** Muestra todos los mensajes

3.6.4. Etiquetas de Entrada de ICEFaces

Además de las versiones extendidas de las etiquetas de entrada estándar JSF, ICEFaces incluye etiquetas especializadas de entrada de texto en los formularios.

- **<ice:inputFile>:** Presenta un archivo de entrada de elementos HTML. El componente inputFile se puede utilizar para proporcionar un archivo especificado por el usuario. Los usuarios especifican el archivo para subir haciendo clic en el botón Examinar para abrir una ventana de navegación del sistema de archivos.
- **<ice:inputText>:** Presenta un elemento de entrada de HTML de tipo "text".
- **<ice:inputTextarea>:** Presenta un elemento "textarea" de HTML.
- **<ice:inputHidden>:** Presenta un elemento de entrada de HTML de tipo "hidden".
- **<ice:inputSecret>:** Presenta un elemento de entrada de HTML de tipo "password".

3.6.5. Etiquetas de Menú de ICEFaces

Dado que la navegación entre las vistas es una tarea común de JSF, ICEFaces introduce algunas variables para ayudar a los diseñadores en la creación de atractivos y funcionales menús de navegación para aplicaciones web.

- **<ice:menuBar>:** Proporciona un sistema de menús que soporta las orientaciones del menú horizontal y vertical. Para definir los submenús del menú se lo puede realizar de manera dinámica con el atributo binding o atreves de las etiquetas menuItem.
- **<ice:menuItem>:** Es el elemento del menú contenida una barra de menú. El atributo value define la etiqueta de muestra para el MenuItem. El atributo icon se puede utilizar para especificar una imagen que se muestra en el lado izquierdo de la menuItem. Los atributos action y actionListener operan en la misma forma que los atributos de un componente estándar del mismo nombre.

- **<ice:menuItems>:** Es el componente submenú para utilizar, si se desea proporcionar una jerarquía potencialmente dinámica de objetos MenuItem.
- **<ice:menuItemSeparator>:** Es el componente de separación de nodos del menú.

3.6.6. Etiquetas de Salida de ICEFaces

ICEFaces se extiende de las etiquetas de salida básicos proporcionados por la librería JSF, y agrega etiquetas adicionales para la prestación de texto estático y dinámico en forma condicional, basado en roles de usuario.

- **<ice:outputChart>:** utiliza el JCharts de código abierto para crear gráficos. Los dos tipos principales de gráficos que se pueden crear con este componente son axis charts and pie charts.
- **<ice:outputConnectionStatus>:** Este componente que sirve para configurar los mensajes de conexión activa o conexión expirada. El componente muestra uno de los 4 estados posibles:
    1. *Active:* Existe conexión y hay una petición pendiente.
    2. *Inactive*: Existe conexión y no hay ninguna actividad pendiente.
    3. *Caution:* La conexión request/response ha superado el umbral configurado.
    4. *Disconnected:* La conexión se ha perdido, ya sea debido a un error de red o aplicación (sesión expirada, etc.)
- **<ice:outputDeclaration>:** Hace una declaración DOCTYPE que se coloca al principio del documento de salida.
- **<ice:outputFormat>:** Muestra el texto parametrizado.
- **<ice:outputLabel>:** Presenta un elemento "label" de HTML.
- **<ice:outputLink>:** Presenta un elemento “a” de HTML.
- **<ice:outputProgress>:** Se puede utilizar para el informe de progreso a los usuarios en los casos en que exista una ejecución larga de tareas en el servidor.
- **<ice:outputStyle>:** Se utiliza para vincular un hoja de estilo a una página para los componentes de ICEFaces.
- **<ice:outputText>:** Muestra un texto.

3.6.7. Etiquetas de panel de ICEFaces

Ayudan a organizar el diseño de componentes en su punto de vista.

- **<ice:panelBorder>:** Es un componente contenedor que organiza y cambia el tamaño especificado de contenedores secundarios en cinco regiones: norte, sur, este, oeste y centro. Estas regiones se definen mediante facets.
- **<ice:panelCollapsible>:** Muestra un panel que puede contraerse.
- **<ice:panelGrid>:** Presenta un elemento "table" de HTML.
- **<ice:panelGroup>:** Es un componente contenedor, que agrupa mas de un componete. Está diseñado para utilizar en situaciones en las que sólo se puede añadir un solo componente, como en el caso de los facets.
- **<ice:panelPopup>:** Es un componente contenedor que genera una ventana o panel que flota en la parte superior de una página web. El popupPanel contiene dos regiones que se definen mediante facets. Los nombres de las facetas son: header y body.
- **<ice:panelPositioned>:** Este componente genera una serie de componentes repetitivos dentro de un panel. Cada uno de estos componentes de hijos es posible arrastrar y pueden intercambiar posiciones entre ellos.
- **<ice:panelSeries>:** Proporciona un mecanismo para la generación dinámica de una serie de componentes repetitivos dentro de un panel. Este componente hace que sus componentes hijos de forma iterativa similar a la forma en que el componente dataTable presenta filas de datos.
- **<ice:panelStack>:** Es un componente contenedor que contiene a varios panel groups. Un solo contenedor del grupo de paneles es visible al mismo tiempo, llenando el área cubierta por el componente panelStack. El grupo de paneles especificado en el atributo selectPanel será visible, con los otros que son ocultos.
- **<ice:panelTabSet>:** Es un componente contenedor que a su vez contiene uno o más componentes panelTab, que son componentes de contenedor. El componente panelTabSet muestra un solo panelTab activo, ocultando el contenido de los demás. Los usuarios pueden seleccionar qué panelTab hacer visible haciendo clic en el encabezado de la ficha de la panelTab que desea mostrar.
- **<ice:panelTab>:** Es un componente contenedor Muestra una tab panel individual del componente panelTabSet.
- **<ice:tabChangeListener>:** Registra un TabChangeListener con un componente padre <ice:panelTabSet>. Esta etiqueta tiene un solo atributo "type" que especifica el nombre completo de la clase que implementa la interfaz: com.icesoft.faces.component.paneltabset.TabChangeListener.
  Una instancia de esta clase notificara cuando el usuario cambie a otra pestaña.

3.6.8. Etiquetas de selección de ICEFaces

Los componentes de la selección son una parte vital de una aplicación JSF. ICEFaces incluye una serie de estas etiquetas para los escenarios de selección comunes, tales como la selección de un idioma, el país, y otras opciones.

- **<ice:selectBooleanCheckbox>:** Presenta un elemento de entrada de de tipo "checkbox".
- **<ice:selectInputDate>:** Este componente es utilizado para representar valores de tipo fecha , el componente renderiza un calendario . Los usuarios pueden seleccionar una fecha haciendo clic sobre una fecha en el mes mostrado. El componente selectInputDate se puede utilizar en modo ventana emergente o en un simple input field.
- **<ice:selectInputText>:** Proporciona un componente inputText con auto-completado. A medida que el usuario introduce texto en el componente que proporciona una lista desplegable de posibles valores coincidentes de que el usuario puede seleccionar. El componente prevé una palabra o frase que un usuario quiere teclear sin que el usuario realmente complete escribiendo. El componente selectInputText obliga a los desarrolladores a implementar el algoritmo de búsqueda que coinciden con la lista en su backing bean.
- **<ice:selectManyCheckbox>:** Presenta una lista de casillas de verificación de HTML.
- **<ice:selectManyListbox>:** Muestra una lista en un cuadro de texto y permite al usuario seleccionar varios elementos en la lista.
- **<ice:selectManyMenu>:** Muestra un menú desplegable y permite al usuario seleccionar varios elementos en la lista.
- **<ice:selectOneListbox>:** Muestra una lista en un cuadro de texto y permite al usuario seleccionar sólo un elemento de la lista.
- **<ice:selectOneMenu>:** Muestra un menú desplegable que permite al usuario seleccionar una sola opción.
- **<ice:selectOneRadio>:** Muestra una serie de botones de radio y permite al usuario seleccionar un solo botón.

3.6.9. Etiquetas de árbol de ICEFaces

Los desarrolladores de software están acostumbrados a trabajar con los controles de árbol para mostrar datos estructurados de una manera intuitiva. ICEFaces incluye

controles de árboles que hacen todas las funciones de los componentes de árbol de HTML en sus páginas JSF.

- **<ice:tree>:** Muestra datos jerárquicos como un árbol. Opcionalmente, el árbol también puede mostrar los controles de navegación para la expansión dinámica y contraer los nodos del árbol. Los eventos de navegación del árbol están disponibles para que una aplicación puede responder a estos eventos. Los nodos también pueden ejecutar eventos de acción.
- **<ice:treeNode>:** Proporciona la plantilla que se aplicará en la presentación de cada nodo, la etiqueta TreeNode es compatible con dos facets: icon y content.

CAPÍTULO IV

Apache MyFaces Trinidad

4.1.Historia

MyFaces Trinidad es un subproyecto de MyFaces donados por Oracle a la Apache Software Foundation el proyecto era conocido con ADF Faces y renombrado a MyFaces Trinidad después de un largo proceso. Trinidad es más que una biblioteca de componentes, contiene muchas extras que resuelven los problemas comunes de desarrollo y agregar capacidades mediante la ampliación de JSF

Oracle se encaminó al desarrollo de un framework web basado en componentes el cual evolucionó en una librería de componentes básicos para aplicaciones JSF y posteriormente estos mismos en frameworks JSF. Hoy en día JSF es una de las tecnologías centrales cuando se trata de implementar aplicaciones web a mediana y gran escala.

Oracle tomó la decisión de donar el framework originalmente conocido como Oracle ADF Faces a la empresa Apache Software Foundation, para tener un framework que conste solamente de código fuente abierto pero ahora las nuevas versiones están desarrolladas por Apache MyFaces.

Apache MyFaces es un proyecto de Apache Software Foundation que es responsable de la implementación JSF con el mismo nombre que incluye un número de subproyectos tales como Trinidad, Tomahawk, etc.

4.2.Características y Arquitectura Apache MyFaces Trinidad.

4.2.1. Características

- Representación parcial de Página (PPR): La tecnología JSF-Ajax es parte de prácticamente todas las etiquetas de Trinidad.

- Nomenclatura de atributos consistente, incluyendo conjuntos de atributos periódicos por todo el universo de las etiquetas de Trinidad.
- Un amplio número de etiquetas JSF están disponibles como una versión de Trinidad con refinamientos enfocados en tecnologías específicas de Trinidad.
- Framework de Cuadros de Diálogo: El framework de Cuadros de Dialogo de Trinidad permite a la aplicación web trabajar con ventanas emergentes encajables sin restricción de contenido.
- Trinidad además viene con sus propios ámbitos, como "pageFlowScope" para soportar Cuadros de Diálogo y flujo de datos de las páginas.

4.2.2. Arquitectura

A diferencia de los frameworks habilitados para AJAX donde la mayoría de la lógica de la aplicación reside en el cliente, con la lógica de MyFaces la aplicación reside en su mayoría en el servidor, se ejecuta en el ciclo de vida de JSF. El modelo de datos Java también permanece en el servidor, el framework de MyFaces efectúa la presentación inicial de sus componentes en el servidor, la generación de contenido HTML que se consume directamente por los navegadores.

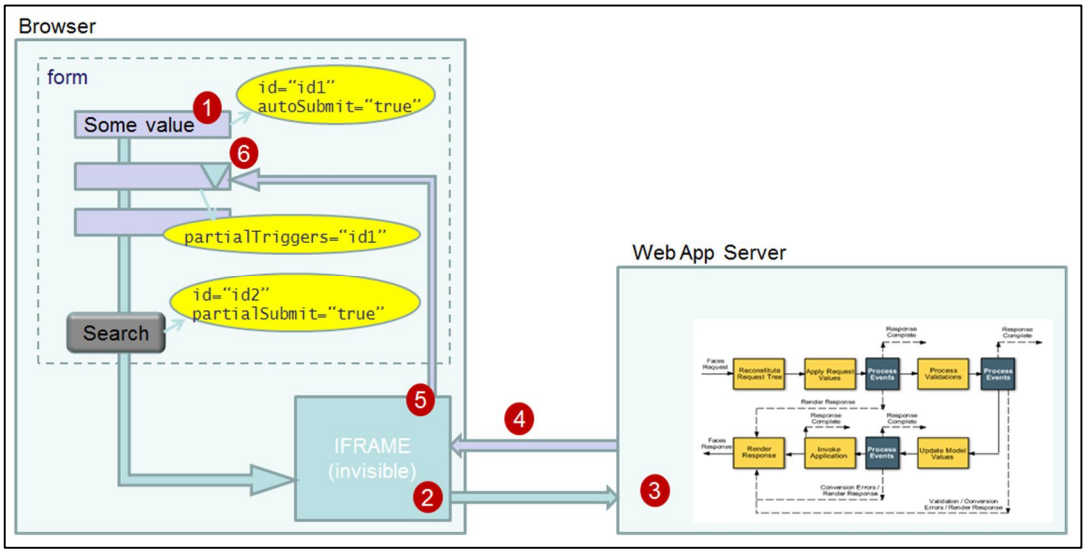


Figura 25. Arquitectura de MyFaces

Para poder implementar Ajax sobre Trinidad utilizaremos tres atributos de los distintos elementos que serán:

- **partialSubmit:** Asignándole valor "true" establece el elemento que envía la petición de Ajax. Este elemento debe poseer el atributo "id" para poder hacer referencia a él.
- **partialTrigger(s):** Será un listado de los id's de los elementos que solicitan las peticiones Ajax que afectarán al elemento con esta propiedad.
- **autoSubmit:** Todos los atributos de trinidad los soportan. Este atributo hace que cuando el elemento se modifica envíe una petición Ajax.

En la mayoría de casos para que las peticiones de Ajax funcionen bien, los elementos deben estar dentro de un contenedor. Por ejemplo, <tr:paneGroupLayout>.

4.3.Evolución de Apache MyFaces Trinidad

Fecha	Versión	Nuevas Características
14/04/2011	MyFaces Trinidad 2..0.0	Public API for the Apache MyFaces Trinidad project for JSF 2. Partial support
16/12/2009	MyFaces Trinidad 1.2.12	Version 1.2.12 resolves a long list of issues and provides several improvements and new features.
25/02/2009	MyFaces Trinidad 1.2.11	Librería de componentes de código abierto JSF 1.2
18/11/2008	MyFaces Trinidad 1.0.10	MyFaces Trinidad component library in version 1.0.10 (for JSF 1.1)

Tabla 2. Versiones de MyFaces Trinidad

4.4.Integración de Apache MyFaces Trinidad con JSF.

Apache MyFaces Trinidad es un framework JSF que incluye una biblioteca de componentes de alta calidad, Trinidad no es sólo una colección de JSF que contiene más de 100 componentes basados en Ajax con soporte para características críticas. También incluye un conjunto de componentes para:

- Partial page rendering: Renderizado parcial de la página para todo el conjunto de componentes.
- Ajax API (client-side and server-side).
- Dialog support.

- PageFlowScope: para comunicación entre páginas.
- CSS Skinning Framework

4.5.Requisitos

4.5.1. Implementaciones de Java soportadas

- JDK 1.5 y superiores.

4.5.2. Implementaciones de JavaServer Faces soportadas:

- Sun JSF 1.2 – 2.x
- MyFaces 1.1.1 - 1.2
- Facelets JSF 1.1.1 - 1.2
- Tomahawk 12-1.x

4.5.3. Servidores Soportados

- Tomcat 5.x - 6.x
- Jrun 4 (SP1a)
- JBoss 4.2.x
- BEA Weblogic 8.1
- Jonas 3.3.6 w/Tomcat
- Resin 2.1.x
- Jetty 4.2.x - 5.1.x
- Websphere 5.1.2
- OC4J

4.5.4. Navegadores Soportados

- Internet Explorer
- Firefox
- Chrome
- Safari

4.6.Componentes Apache MyFaces Trinidad.

4.6.1. Etiquetas de comandos de MyFaces Trinidad

Etiquetas de comandos cubren una amplia gama de usos, tales como el envío de formularios, navegación y otros tipos de interacción del usuario.

- **<tr:commandButton**>: Etiqueta que produce un botón que se puede utilizar para enviar los datos o producir una acción.
- **<tr:commandLink>:** Etiqueta que produce un enlace que puede ser utilizado para enviar datos o producir una acción.
- **<tr:commandNavigationItem>**: Produce un componente (botón, enlace, opción), que se puede utilizar para navegar.
- **<tr:goButton>:** Etiqueta que navega directamente a otra página.
- **<tr:goLink>:** Crea un hipervínculo.
- **<tr:resetButton>:** Muestra un botón que restablece los campos de un formulario.

4.6.2. Etiquetas de componentes de MyFaces Trinidad

Estas etiquetas representan diferentes componentes de la interfaz de usuario y convertidores en la biblioteca de etiquetas Trinidad.

- **<tr:group>:** Se utiliza para agrupar elementos relacionados. Esta etiqueta no hace nada visualmente es un control invisible.
- **<tr:poll>:** Este componente lanza eventos después de un intervalo de tiempo.
- **<tr:switcher>:** Este componente permite mostrar una de los diferentes facets en la pantalla.

4.6.3. Etiquetas de conversión de MyFaces Trinidad

Trinidad incluye una serie de etiquetas útiles de conversión.

- **<tr:convertColor>:** Se utiliza para conviertir una cadena a objeto java.awt.Color y de nuevo a una cadena.
- **<tr:convertDateTime>:** Se utiliza para conviertir una cadena a objeto java.util.Date y de nuevo a una cadena basada en el patrón o la fecha y los estilos de tiempo.

- **<tr:convertNumber>:** se utiliza para convertir una cadena a un objeto de número y volver a una cadena basada en el patrón o tipo.

4.6.4. Etiquetas de datos de MyFaces Trinidad

Trinidad incluye una serie de etiquetas de datos para representar datos en una variedad de maneras, incluyendo tablas, listas y conjuntos de resultados de desplazamiento.

- **<tr:column>:** Debe ser el primer componente hijo del componente de la tabla para producir cada columna de la tabla.
- **<tr:forEach>:** Se utiliza para recorrer una lista de objetos.
- **<tr:iterator>:** Procesa cada elemento durante la petición.
- **<tr:showDetail>:** Este componente permite mostrar y ocultar una sección de una página.
- **<tr:showDetailHeader>:** Permite a un grupo de componentes de una cabecera mostrar y ocultar.
- **<tr:showDetailItem>:** Muestra y oculta un elemento que el usuario puede seleccionar.
- **<tr:table>:** Muestra los datos en una tabla.

4.6.5. Etiquetas de documento de MyFaces Trinidad

Las etiquetas de documento permiten construir documentos de forma dinámica a partir de fragmentos de código.

- **<tr:document>:** Crea los elementos básicos (<HTML>, <HEAD>, <BODY>) de una página.
- **<tr:form>:** Crea un elemento form
- **<tr:importScript>:** Se utiliza para importar archivos javascript.
- **<tr:page>:** Muestra una página donde los elementos de navegación están limitadas a un modelo de menú.
- **<tr:separator>:** Muestra una barra horizontal.
- **<tr:spacer>:** Muestra un espacio en blanco.
- **<tr:subform>:** Permite que parte de una página sea enviada. Uno de los componentes dentro de un subformulario.

4.6.6. Etiquetas para Manejo de Eventos (Event Handling) de MyFaces Trinidad

Estas etiquetas son de apoyo tanto en el cliente y el servidor para el manejo de eventos.

- **<tr:resetActionListener>:** Utiliza una sintaxis para restablecer los valores antes de que se ejecute una acción.
- **<tr:returnActionListener>:** Permite una acción para devolver un valor desde un cuadro de diálogo o proceso.
- **<tr:setActionListener>:** Utiliza una sintaxis para asignar valores antes de que se ejecute una acción.

4.6.7. Etiquetas gráficas de MyFaces Trinidad

Trinidad incluye una serie de etiquetas gráficas para representar las imágenes en sus puntos de vista.

- **<tr:chart>:** Se utiliza para mostrar los datos en el formato de los distintos tipos de gráficos.
- **<tr:icon>:** Este componente se utiliza para presentar un icono.
- **<tr:image>:** Crea elementos tipo img de html.
- **<tr:legend>:** se utiliza para describir un mesaje de leyenda.
- **<tr:progressIndicator>:** muestra el progreso de una tarea.

4.6.8. Etiquetas de Entrada de MyFaces Trinidad

Además de las versiones extendidas de las etiquetas de entrada estándar JSF, Trinidad incluye etiquetas especializadas de entrada para la entrada de texto y la entrada de datos en los formularios.

- **<tr:inputColor>:** Este componente maneja la selección de colores en una aplicación JSF.
- **<tr:inputDate>:** Este componente presenta un campo de texto con un botón al lado de él aparece una ventana de calendario para que el usuario interactúe con él. Cuando esta etiqueta se combina con la etiqueta <tr:chooseDate>, el calendario se puede presentar en la misma página el campo de texto sin necesidad de una ventana pop-up.

- **<tr:inputFile>:** Se crea un componente para subir archivos. Debe tener <tr:formusesUpload="true"> y debe tener un filtro de Apache Trinidad instalado.
- **<tr:inputHidden>:** crea un valor que se enviará, pero no presenta en la página.
- **<tr:inputListOfValues>:** Crea un campo de texto con un botón al lado de él y pone en marcha una nueva ventana del navegador con una lista de valores predefinidos para el usuario.
- **<tr:inputNumberSpinbox>:** Es un componente de interfaz de usuario numérico que restringe la entrada de datos a un rango determinado en el lado del cliente y proporciona un componente de interfaz de usuario de interés para nuestras aplicaciones JSF que se extienden más allá del conjunto limitado de componentes HTML apoyado por el navegador.
- **<tr:inputText>:** Crea un campo donde se pueden introducir datos. Pueden ser líneas múltiples de datos o una contraseña oculta.

4.6.9. Etiquetas de Navegación de MyFaces

Dado que la navegación entre las vistas es una tarea común de JSF, Trinidad introduce algunas variables útiles para ayudar a los diseñadores en la creación de atractivos y funcionales menús de navegación para aplicaciones web.

- **<tr:breadCrumbs>:** Especifica la ruta de vuelta a la página raíz de una jerarquía de vínculos. Vínculos jerárquicos se pueden añadir como hijos o el uso de un modelo de menú que se enlaza con el componente de breadCrumbs.
- **<tr:navigationPane>:** Crea una serie de enlaces para un nivel en una jerarquía. Los enlaces se pueden añadir como los hijos o un modelo de menú se puede enlazar a la navigationPane.
- **<tr:navigationTree>:** Se utiliza para mostrar los datos de manera jerárquica. Las partes del árbol se puede expandir y contraer y el enfoque de la ruta es resaltada.
- **<tr:processChoiceBar>:** Muestra un botón anterior, una barra de selección y un botón de siguiente.
- **<tr:singleStepButtonBar>:** Muestra un botón anterior, el texto "Stage", con el paso actual y las pasos máximos y un botón al lado que permite al usuario navegar entre los elementos.
- **<tr:train>:** Muestra una serie de enlaces que permiten al usuario navegar por un proceso de varias páginas.

4.6.10. Etiquetas de Salida de MyFaces Trinidad

Trinidad se extiende de las etiquetas de salida básicos proporcionados por la librería JSF, y agrega etiquetas adicionales para la prestación de texto estático y dinámico en forma condicional, basado en roles de usuario.

- **<tr:message>:** Muestra un mensaje para el componente. Para los componentes de entrada, establezca el atributo simple en verdadero.
- **<tr:messages>:** se utiliza para mostrar los mensajes añadido a la FacesContext en la parte superior de la página
- **<tr:outputDocument>:** se utiliza para mostrar texto con estilo incluyendo saltos de línea y se puede convertir objetos Java.
- **<tr:outputFormatted>:** se utiliza para interpretar y mostrar una cadena que contiene un conjunto limitado de formato html.
- **<tr:outputLabel>:** Se utiliza para mostrar una etiqueta y un icono indicando un tipo de mensaje .
- **<tr:outputText>:** Muestra texto con estilo y se puede convertir objetos Java.

4.6.11. Etiquetas de panel de MyFaces Trinidad

Ayudan a organizar el diseño de componentes en su punto de vista.

- **<tr:panelAccordion>:** Muestra un conjunto de barras con un panel debajo de cada barra para cada showDetailItem a conocer. Los productos que se pueden mostrar uno a uno, o varios elementos a la vez.
- **<tr:panelBorderLayout>:** Muestra todos los elementos secundarios en forma consecutiva.
- **<tr:panelBox>:** Muestra la información en una página compensado por un color determinado.
- **<tr:panelButtonBar>:** Muestra un conjunto de botones.
- **<tr:panelChoice>:** Muestra un conjunto de nodos ShowDetailItem.
- **<tr:panelFormLayout>:** Muestra los controles de entrada con las etiquetas y los controles alineados verticalmente. Los separadores se mostrarán alrededor de los elementos agrupados.
- **<tr:panelGroupLayout>:** Muestra hijos en forma consecutiva, en una línea horizontal, vertical o en un grupo.

- **<tr:panelHeader>:** Muestra una etiqueta y un icono opcional, o mensajes en la parte superior de cada sección.
- **<tr:panelHorizontalLayout>:** Muestra elementos hijos horizontalmente.
- **<tr:panelLabelAndMessage>:** Muestra la etiqueta, el tip y el mensaje con cada elemento secundario. Para los elementos de entrada de atributo simple se debe establecer en true. Si esta etiqueta se encuentra dentro de un panelForm, las etiquetas se alinearán.
- **<tr:panelList>:** Muestra una lista con viñetas de los elementos secundarios. Si el atributo de filas no está definida, entonces todos los elementos secundarios se muestran en una columna, aunque el atributo maxColumns se establece. Si el atributo se establece en las filas y el atributo maxColumns no se establece, entonces el número de columnas es un múltiplo del número de filas con el resto que componen la columna final.
- **<tr:panelPage>:** Se utiliza para diseñar la navegación dentro de una página.
- **<tr:panelPageHeader>:** Se utiliza para diseñar la parte superior de la página.
- **<tr:panelPartialRoot>:** Muestra parte de la página cuando el documento de Trinidad y etiquetas de cuerpo no se puede utilizar.
- **<tr:panelRadio>:** Muestra un conjunto de nodos ShowDetailItem como botones de radio. Si la posición del atributo está establecido en superior, y luego los atributos de la alineación inicial y final afectará a la posición de los botones de radio. Si la posición está configurado para iniciarse, los atributos de alineación centro, arriba y abajo afectan a la posición de los botones de radio.
- **<tr:panelSideBar>:** Muestra una barra de navegación en un lado de la página.
- **<tr:panelTabbed>:** Muestra nodos ShowDetailItem como fichas.
- **<tr:panelTip>:** muestra una página o tips de sección de nivel para el usuario.

4.6.12. Etiquetas de selección de MyFaces Trinidad

Los componentes de la selección son una parte vital de una aplicación JSF. Trinidad incluye una serie de estas etiquetas para los escenarios de selección comunes, tales como la selección de un idioma, el país, y otras opciones.

- **<tr:chooseColor>:** Este componente presenta la paleta de colores estándar o colores personalizados en la misma página como el campo de texto, y por lo tanto no requiere el uso de una ventana externa para la selección de color.
- **<tr:chooseDate>:** Se utiliza con selectInputDate para permitir una rápida selección de fecha en línea.

- **<tr:selectBooleanCheckbox>:** Muestra una casilla de verificación.
- **<tr:selectBooleanRadio>:** Muestra un botón de radio.
- **<tr:selectItem>:** Es un elemento secundario que se utiliza en una lista, opción, radio o etiqueta volante.
- **<tr:selectManyCheckbox>:** Muestra las casillas de verificación que permiten al usuario seleccionar muchos elementos.
- **<tr:selectManyListbox>:** Muestra una lista en un cuadro de texto y permite al usuario seleccionar muchos elementos en la lista.
- **<tr:selectManyShuttle>:** Muestra dos cuadros de texto con una lista de elementos y permite al usuario mover elementos de una lista a la otra.
- **<tr:selectOneChoice>:** Muestra un menú desplegable que permite al usuario una sola opción.
- **<tr:selectOneListbox>:** Muestra una lista en un cuadro de texto y permite al usuario seleccionar sólo un elemento de la lista.
- **<tr:selectOneRadio>:** Muestra una serie de botones de radio y permite al usuario seleccionar un solo botón.
- **<tr:selectOrderShuttle>:** Muestra dos cuadros de texto con una lista de elementos y permite al usuario mover elementos de una lista a la otra y mover los elementos arriba y abajo en la lista.
- **<tr:selectRangeChoiceBar>:** Muestra un botón anterior y posterior que permite al usuario moverse entre una amplia gama de elementos de una lista de datos.

4.6.13. Etiquetas de árbol de MyFaces Trinidad

Los desarrolladores de software están acostumbrados a trabajar con los controles de árbol para mostrar datos estructurados de una manera intuitiva. Trinidad incluye controles de árboles que hacen todas las funciones de los componentes del árbol HTML en sus páginas JSF.

- **<tr:tree>:** Muestra un árbol de los enlaces que permitir al usuario navegar por un modelo de menú.
- **<tr:treeTable>:** Muestra un árbol de enlaces en una tabla para que el usuario pueda navegar por un modelo de tabla de árbol.

4.6.14. Etiquetas de Validación de MyFaces Trinidad

Validar la entrada del usuario es un requisito fundamental para las aplicaciones web de hoy. Trinidad se extiende la construcción en las etiquetas de validación JSF mediante la introducción de nuevas etiquetas basado en la librería de validación de Jakarta Commons para la validación de fechas, números, expresiones regulares y mucho más.

- **<tr:validateByteLength>:** Valida la longitud en bytes de una cadena.
- **<tr:validateDateRestriction>:** Valida un campo de fecha.
- **<tr:validateDateTimeRange>:** Comprueba que la fecha introducida esté dentro de un rango especificado.
- **<tr:validateDoubleRange>:** Comprueba que el valor (del tipo doble) se encuentra dentro del rango especificado.
- **<tr:validateLength>:** Comprueba si la cadena es la longitud especificada.
- **<tr:validateLongRange>:** Comprueba que el valor (de tipo long) se encuentra dentro del rango especificado.
- **<tr:validateRegExp>:** Comprueba que el valor coincide con el patrón especificado.
- **<tr:validator>:** Añade un validador a la etiqueta principal.

CAPÍTULO V

Esquemas de comparación de los Frameworks ICEFaces, RichFaces, Apache MyFaces Trinidad

5.1.Introducción

La gran proliferación de sistemas informáticos propia de la Sociedad de la Información y del Conocimiento ha impulsado el desarrollo de frameworks o plataformas que faciliten de desarrollo, mantenimiento y distribución de aplicaciones, la mayoría de ellas distribuidas.

Estos frameworks JSF con soporte para AJAX se pueden encontrar en el mercado, la mayoría de ellos son gratuitos, sin embargo otros son pagados y tienen dependencia con ciertas herramientas, presentan facilidades relacionadas con una gran diversidad de aplicaciones, con requerimientos diferentes en cuanto a disponibilidad, seguridad, eficiencia.

La selección de tres implementaciones JSF con soporte para AJAX en el estudio preliminar precisó la revisión de pruebas y evaluaciones realizadas por varios expertos de la comunidad de desarrolladores JEE, las cuales han sido estudiadas a profundidad en el capítulo anterior. Estas evaluaciones del estudio preliminar, también han influido en la selección final del framework JSF con soporte para AJAX a ser implementado en el desarrollo del Sistema de Control Microcurricular para la Carrera de Ingeniería en Sistemas.

5.2.Arquitectura de los Frameworks JSF

El estándar JSF ofrece un conjunto de elementos web para la construcción de interfaces que actualmente tienen implementación en varios proyectos de soporte (RichFaces, ICEFaces, Apache MyFaces, entre otros). Estos componentes permiten

la elaboración de interfaces enriquecidas con elementos desde formularios sencillos hasta menús jerárquicos, agendas y tablas dinámicas.

Al investigar sobre componentes JSF se encontró que algunos frameworks semi-comerciaIes o ex-comerciales como ICEFaces o Trinidad ofrecen un gran conjunto de componentes AJAX habilitados, cada uno con su propio conjunto de componentes de comportamiento específico, muchos de estos componentes soportan funcionalidades AJAX ofreciendo características que van mucho más allá del estándar.

Existen frameworks que soportan la adición de una sola etiqueta para proporcionar soporte AJAX, la cual puede ser anidada dentro de otros componentes JSF agregando a la misma funcionalidad AJAX. Normalmente ciertas funciones o eventos JavaScript del componente padre (como onCIick o onChange) desencadenará una petición AJAX y la respuesta será usada para actualizar algunos elementos de la página HTML. Este tipo de arquitectura tiene una gran oportunidad de perdurar en el futuro desarrollo de la especificación JSF, debido a las mejoras de las etiquetas estándar y los nuevos componentes especiales que pueden ser activados fácilmente por AJAX.

La arquitectura de los frameworks JSF con soporte AJAX tienen una estrategia de presentación parcial del contenido de una página JSF. Ajax permite ejecutar una petición a través del ciclo de vida JSF y desplegar la respuesta parcial de la petición realizada al servidor. A este tipo se estrategia se la llama "Partial Refreshes", esta estrategia reduce la cantidad de código HTML que viaja entre el cliente y el servidor y reduce el tiempo consumido para el despliegue de los componentes de la pantalla.

5.3.Tabla Comparativa

La especificación JSF es utilizada en la actualidad para el desarrollo de la mayoría de aplicaciones empresariales bajo la plataforma JEE, muchos miembros de la comunidad de desarrolladores JEE se han preocupado por la selección del framework adecuado para el desarrollo de sus aplicaciones, pero sobre la base de los siguientes parámetros podemos elegir el mejor framework adecuado:

- Componentes (Components)
- Documentación (Documentation)
- Integración de AJAX (AJAX Integration)
- Compatibilidad con navegadores (Browser support)
- Comunidad (Community)
- Herramienta de apoyo(Eclipse, NetBeans) (Tool support: Eclipse, NetBeans)

A continuación se incluye una matriz elaborada por el Dr. Thomas Latka y Juergen Kniephoff, quienes se han encargado de probar las características de los diferentes frameworks JSF, esta matriz es actualizada constantemente por sus autores, tomando en cuenta también la opinión de muchas personas quienes colaboran con sus comentarios y opiniones.

Esta matriz pretende mostrar en forma comparativa las diferentes características que cada uno de los frameworks JSF posee. Al haber seleccionado para este proyecto la evaluación de los frameworks RichFaces, ICEFaces y MyFaces Trinidad, la matriz que se expone a continuación únicamente hace referencia a los mismos, si se desea acceder a la matriz completa con las opiniones de los diferentes colaboradores la misma puede ser encontrada en la siguiente dirección web: http://www.jsfmatrix.net.[23]

	RICHFACES	**ICEFACES**	**MYFACES Trinidad**
Tecnología	JSF	JSF	JSF
Página Principal	www.jboss.org/rich faces	www.icefaces. org	myfaces.apache.org /trinidad
Ejemplos de Uso	livedemo.exadel.com/richfaces-demo/index.jsp	http://component-showcase.icefaces.org/component-showcase/showcase.iface	
Cantidad y calidad de Información	ALTA	ALTA	MEDIA

[23] Evaluación de tres implementaciones Java Server Faces 1.2 http://www3.espe.edu.ec:8700/ bitstream/ 21000/ 309/1/T-ESPE-027448.pdf

Aplicación de Ejemplo descargable	X	X	X
URL de Documentación	http://www.jboss.org/richfaces/docs	http://www.icefaces.org/main/resources/	http://myfaces.apache.org/trinidad/devguide/index.html
URL de Foros de Discusión	www.jboss.org/rich faces	http://www.icefaces.org/JForum/forums/list.page	
Componentes			
DATATABLE	X	X	X
TREE	X	X	X
TREETABLE			X
TAB	X	X	X
MENU	X	X	X
HTMLEDITOR	X	X	
INPLACEEDITOR	X		
CALENDAR	X	X	X
CHART	x	X	X
UPLOAD	X	X	X
PROGRESSBAR	X	X	X
DRAG & DROP	X	X	
AUTOCOMPLETE	X	X	
POPUPDIALOG	X	X	X
MODAL DIALOG	X	X	
GOOGLE MAPS	X	X	
Características Adicionales en Tablas de Datos			
COLUMN DRAG&DROP	X		
SORT	X	X	X
FILTER	X		
GROUPING	X	X	
ROW SELECTION	X	X	
DATABASE PAGINATION	X		
Librerías Javascript			
PROTOTYPE	X	X	X
JQUERY	X		X
SCRIPTACULOUS	X	X	X
DOJO		X	
Estrategia de Actualización			
Partial Page Rendering	X	X	X
Server	X		
Client	X	X	X
AJAX	X	X	X
Compatibilidad			

Coexistencia con otros Frameworks JSF	Seam, Spring, Tomahawk	Seam, Spring, Webflow, (Tomahawk)	Orchesta
JSF 1.2	X	X	X
JSF 2	X	X	X
FACELETS	X	X	X
JAVA 5	X	X	X
PORTLET	X	X	
Soporte IDE			
ECLIPSE TAG SUPPORT	X	X	
NETBEANS TAG SUPPORT	X	X	
JDEVELOPER	X	X	X
Navegador			
INTERNET EXPLORER	X	X	X
FIREFOX	X	X	X
CHROME	X	X	X
SAFARI	X	X	X
OPERA	X	X	
Misceláneos			
SERVERPUSHING	X	X	
TOOLTIP	X	X	X
SKINNING	X	X	X
Características Especiales	Multifile upload, Ajax Queue, Colorchooser, Google Maps		inputNumberSpinbox, selectManyShuttle y un proceso de entrenamiento y
Licenciamiento			
LICENCIA	LIBRE	LIBRE & Commercial	LIBRE

Tabla 3. Tabla Comparativa de Frameworks JSF[24]

5.4.Caso de Aplicación para los Prototipos

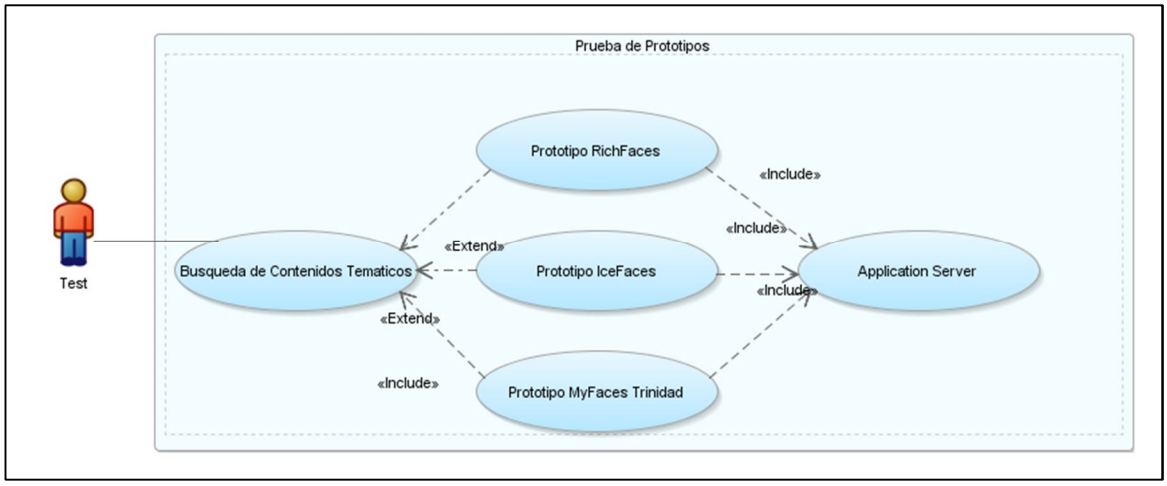
5.4.1. Propósito

El caso de aplicación consiste en proporcionar la búsqueda de Contenidos Temáticos de una materia, la automatización de este proceso de búsqueda lo podemos realizar mediante diferentes criterios con el fin de evaluar los prototipos de cada framework.

Para la selección del caso de aplicación, se consideró la implantación de tres prototipos desarrollados con los diferentes frameworks de estudio: RichFaces, IceFaces, MyFaces Trinidad.

[24] http://www.jsfmatrix.net

5.4.2. Caso de Uso



Especificación del Caso de Uso

Realizar las pruebas respectivas de cada uno de los frameworks de estudio, para los cual se implementa un proceso de búsqueda de contenidos temáticos para probar los diferentes criterios de evaluación que tienen las aplicaciones web como: Apariencia, Rendimiento, Facilidades para el desarrollo, Soporte.

Prototipo de RichFaces

Prototipo de IceFaces

Prototipo MyFaces Trinidad

5.5.Criterios de Evaluación Considerados[25]

La selección de los criterios para la evaluación de los frameworks JSF con soporte AJAX es un proceso en donde depende de los resultados finales para la selección del framework JSF que sería utilizado en el desarrollo de la aplicación con una plataforma JEE.

Para la definición de los criterios de evaluación se procedió a realizar una investigación sobre las principales características de los frameworks JSF, para finalmente llegar a definir diferentes criterios de evaluación.

Los criterios considerados para la evaluación de los frameworks se resumen en la tabla siguiente.

Categoría	Criterio	Valoración
1. Apariencia	Apariencia en los diferentes navegadores	1-3
	Independencia de la resolución	1-2
2.Rendimiento	Tiempo de descarga o tiempo de inicialización	1-4
	Consumo de canal de red	1-4
3. Facilidades para el desarrollo	Curva de aprendizaje	1-9
	Herramientas para el desarrollo de software	1-4
	Soporte en los servidores de aplicaciones	1-4
4. Soporte	Soporte del producto	1-4
	Prospectiva Tecnológica	1-4
	Tipo de Licenciamiento	1-4

Tabla 4. Resumen de Criterios de Evaluación

1. APARIENCIA

1.1 Apariencia en los diferentes navegadores

La documentación oficial de cada implementación indica la compatibilidad con los navegadores soportados para que las aplicaciones desarrolladas con JSF conserven el look & feel de una aplicación, independientemente del navegador del cual ha sido accedido.

[25] Análisis comparativo de los Frameworks Adobe Flex, Java RichFaces y Extjs http://bibdigital.epn.edu.ec /bitstream/ 15000/2110/1/CD-2887.pdf

El criterio se refiere además a evaluar la necesidad de configuración adicional por parte del usuario para mantener el estado de su aplicación en los diferentes navegadores.

VALORACIÓN

Para la valoración se consideraron los siguientes navegadores:

SERVIDOR	RICHFACES	ICEFACES	TRINIDAD
Internet Explorer	x	x	x
Firefox	x	x	x
Chrome	x	x	x
Safari	x	x	x
Opera	x	x	
TOTAL	5	5	4

La valoración de este criterio se dará en función del mayor número de navegadores.

Valor	Significado	RICHFACES	ICEFACES	TRINIDAD
3	Completamente compatible. No presenta ningún problema ni de JavaScript ni de CSS al mostrar los controles.	x	X	x
2	Parcialmente compatible en algunas versiones de navegadores.			
1	La apariencia de la aplicación es totalmente diferente en los navegadores considerados. Incompatibilidad por el objeto XMLHttpRequest y las diferentes versiones de JavaScript.			

Tabla 5. Apariencia en los diferentes navegadores

1.2 Independencia de la resolución

De manera complementaria al criterio anterior (Apariencia en los diferentes navegadores), el criterio considerado se refiere a la capacidad de los frameworks para representar correctamente los controles gráficos, de manera que estos conserven su apariencia independientemente de la resolución de la pantalla.

VALORACIÓN

Valor	Significado	RICHFACES	ICEFACES	TRINIDAD
2	Los controles conservan sus características gráficas independientemente de la resolución de pantalla desde la cual se accede a la aplicación	x	X	x
1	Los controles se distorsionan al cambiar la resolución de la pantalla en todos los exploradores.			

Tabla 6. Independencia de resolución

2. RENDIMIENTO

2.1 Tiempo de descarga o tiempo de inicialización

Se trata de uno de los aspectos más importantes desde el punto de vista del usuario, se refiere al tiempo que debe esperar antes de poder utilizar la aplicación. Para las aplicaciones RIA basadas en un navegador este factor depende de manera directa de la velocidad de la conexión.

Las páginas HTML simples tienen siempre un tiempo de descarga menor al de aplicaciones con alto contenido dinámico, sin embargo con las velocidades de las conexiones actuales esta diferencia se vuelve menos evidente.

VALORACIÓN

La valoración de este criterio se dará en función del menor tiempo de descarga obtenido en la evaluación de prototipos, asignándole el valor más alto al que tome menor tiempo en la inicialización.

Valor	CRITERIO	RICHFACES	ICEFACES	TRINIDAD
4	0.00 – 0.50 sec			
3	0.51 – 1.00 sec			
2	1.01 – 1.50 sec			x
1	1.51 – 2.00 sec	x	x	

Tabla 7. Tiempo de inicialización

2.2 Consumo de canal de red

Se refiere al ancho de banda necesario para poder interactuar con la aplicación, considera el tráfico cliente-servidor y el procesamiento que se lleva a cabo en cada uno de los componentes necesarios para el funcionamiento de las aplicaciones.

VALORACIÓN

La valoración de este criterio se dio en función del menor consumo de canal de red obtenido en la evaluación de prototipos, asignándole el valor más alto al que tome menor consumo de canal de red.

Valor	CRITERIO	RICHFACES	ICEFACES	TRINIDAD
4	100 – 105 kbps			
3	106 – 110 kbps			
2	111 – 115 kbps		x	x
1	116 – 120 kbps	x		

Tabla 8. Consumo de canal de red

3. FACILIDADES PARA EL DESARROLLO

3.1 Curva de aprendizaje

Determina la facilidad e inversión de tiempo necesarios para obtener un nivel de conocimiento aceptable del uso de los componentes de cada uno de los frameworks.

VALORACIÓN

Criterio	Valor	Significado	RICHFACES	ICEFACES	TRINIDAD
Documentación	3	Muy Buena	x		
	2	Buena		x	
	1	Regular			x
Facilidad de aprendizaje	3	Alta	x	x	x
	2	Media			
	1	Baja			
Difusión del aprendizaje	3	Alta	x	x	
	2	Media			x
	1	Baja			

Tabla 9. Curva de aprendizaje

3.2 Herramientas para el desarrollo de software

Los IDE's proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación. Este criterio busca determinar la productividad en el desarrollo de aplicaciones Java, fijándose en la existencia de IDE's, herramientas para depuración de código y herramientas para realización de pruebas.

VALORACIÓN

Se dio mayor valoración al framework en donde existen más IDE's para desarrollar aplicaciones.

IDE Plugins	RICHFACES	ICEFACES	TRINIDAD
Eclipse	x	x	
Netbeans	x	x	
MyEclipse	x	x	x
JDeveloper	x	x	x
TOTAL	4	4	2

Valor	Significado	RICHFACES	ICEFACES	TRINIDAD
4	Existe 4 o más IDE's que permite el desarrollo de aplicaciones a nivel gráfico además de la edición de código.	x	x	
3	Existen 3 IDE's que permite el desarrollo de aplicaciones a nivel gráfico además de la edición de código.			
2	Existen 2 IDE's que permite el desarrollo de aplicaciones a nivel gráfico además de la edición de código.			x
1	Existen 1 IDE que permite el desarrollo de aplicaciones a nivel gráfico además de la edición de código.			

Tabla 10. Herramientas para el desarrollo

3.3 Soporte en los servidores de aplicaciones

Criterio que considera la complejidad para que una aplicación pueda ser puesta en un servidor para ser accedida desde los clientes. Considera la diversidad de servidores de aplicaciones que soportan.

VALORACIÓN

La valoración para este criterio se dio al framework que permite desplegar la aplicación en el mayor número de servidores que se listan en la tabla siguiente.

SERVIDOR	RICHFACES	ICEFACES	TRINIDAD
Apache Tomcat 5.5 - 6.0	x	x	x
BEA WebLogic 9.1 - 10.0	x	x	x
Resin 3.1.x	x		x
Jetty 5.1.x -6.1.x	x	x	x
Sun Java System Application Server	x	x	
Glassfish V2, V3	x	x	
Websphere 7.0. and higher	x	x	
Geronimo 2.0 and higher	x		
JBoss 4.2.x – 5 and higher	x	x	x
Oracle Application Server Container[OC4J]		x	x
Jrun 4 (SP1a)			x
TOTAL	8	8	7

Tabla 11. Facilidad de puesta en producción

4. SOPORTE

4.1 Soporte del producto

Criterio en cual que se toma en cuenta las responsabilidades que cada que cada organización a cargo del desarrollo del framework tiene en la solución de problemas o bugs propios de sus productos, a la vez se refiere al acceso a documentación, cursos de capacitación y posibilidades de certificación.

VALORACIÓN

Valor	Significado	RICHFACES	ICEFACES	TRINIDAD
4	Existe soporte formal para los errores propios del producto, además de una comunidad que respalda el uso, esta comunidad recibe apoyo de la empresa desarrolladora.	x		x
3	El soporte es exclusivo de la empresa desarrolladora, y no existe apoyo formal para la comunidad de desarrolladores.		x	
2	El soporte lo brinda únicamente la comunidad de desarrolladores.			
1	No existe soporte alguno para el producto.			

Tabla 12. Soporte del producto

4.2 Prospectiva Tecnológica

Este criterio considera la planificación tecnológica del producto, considera el desenvolvimiento desde la salida al mercado del producto y las perspectivas al próximo año, considera la planificación de nuevas versiones o cambios que permitan adaptar al producto para ajustarse a nuevas necesidades.

VALORACIÓN

Valor	Significado	RICHFACES	ICEFACES	TRINIDAD
4	El producto cuenta con una planificación de versiones, implementación de requerimientos, mantenimiento de la documentación. Las perspectivas del producto están establecidas formalmente y se encuentran documentadas	x	x	x

3	El producto ha pasado por varias modificaciones, acorde a necesidades del cliente y nuevos requerimientos de estos. Sin embargo no existe documentación que respalde y verifique los cambios			
2	El producto ha pasado por varias versiones, sin embargo no se conocen perspectivas de nuevas modificaciones y adaptaciones			
1	La primera versión del producto no ha sufrido modificaciones y la documentación no ha sido actualizada, y no existen perspectivas de modificaciones al producto.			

Tabla 13. Prospectiva Tecnológica

4.3 Tipo de Licenciamiento

Considera las diferencias en licenciamiento de los productos de software desarrollados a partir del uso de los frameworks, las limitaciones y las perspectivas para comercializar dichos productos. Permite hacer una diferenciación entre la orientación de los frameworks con respecto a software libre y software privativo.

VALORACIÓN

Valor	Significado	RICHFACES	ICEFACES	TRINIDAD
4	Software libre sin obligación de generar software libre (LGPL)	x		x
3	Software libre GPL			
2	Mozilla Public License 1.1 (MPL)		x	
1	Software de código cerrado			

Tabla 14. Tipo de Licenciamiento

5.6.Análisis Comparativo de las tres tecnologías

5.6.1. Proceso de Comparación

Para determinar el framework más adecuado para el caso de aplicación utilizando los criterios que serán considerados y sus ponderaciones se utilizará una matriz de priorización.

Una matriz de priorización permite clasificar opciones, problemas o asuntos basados en un conjunto de criterios que permiten fijar la prioridad de los mismos, estos criterios son particulares dependiendo de la situación considerada.

La forma general de una matriz de priorización es la que se describe en la figura siguiente:

Opciones	Criterios				Total
	C#1	C#2	C#3	C#4	
Opción 1					
Opción 2					
Opción 3					

Figura 26. Estructura general de una Matriz de Priorización

Las aplicaciones de ejemplo demuestran los usos y características de cada uno de los componentes que conforman el framework, es por esto que su estudio y experimentación fue fundamental en la obtención de resultados.

Una vez que se ha investigado, estudiado y experimentado con cada framework, se procedió a evaluar objetivamente a cada uno de ellos en todos los criterios de evaluación definidos en el punto anterior.

5.6.2. Evaluación de Prototipos

Para la evaluación de los prototipos desarrollados, se considerará que todos hacen uso de una única capa de backend desarrollada utilizando tecnología Java, la cual está expuesta como una capa de negocio con EJB3.

5.6.2.1. Ambiente para el Proceso de Desarrollo y Evaluación

Durante el proceso de desarrollo de los prototipos se utilizó una máquina con las siguientes características.

Sistema Operativo:	Microsoft Windows 7
Procesador:	Intel(R) Core(TM)2 Duo CPU 2.0.0 GHz
Memoria RAM:	4GB
Versión de JDK:	1.5 update 12
IDE para RichFaces:	Eclipse 3.4
IDE para IceFaces:	Eclipse 3.4
IDE para MyFaces Trinidad:	Eclipse 3.4
Servidor de Aplicaciones	Jboss 5.1.0-GA

Tabla 15. Características de la Máquina de Desarrollo

5.6.2.2. Herramientas para la Evaluación

Con el fin de simular un ambiente de producción adecuado en el cual pone a prueba el rendimiento de los prototipos, en los aspectos referentes al consumo de canal de red y tiempo de inicialización se seleccionó la herramienta que se describe a continuación.

Charles Web Debuggin Proxy

Esta aplicación puede ser muy útil para los desarrolladores web, ya que permite ver las peticiones de aplicaciones, sus respuestas, información sobre cookies o de la caché sobre todo lo que ocurre entre el navegador y el servidor. Charles nos ofrece una completa información sobre las imágenes que contiene una página, el contenido de los volúmenes de información de una aplicación web.

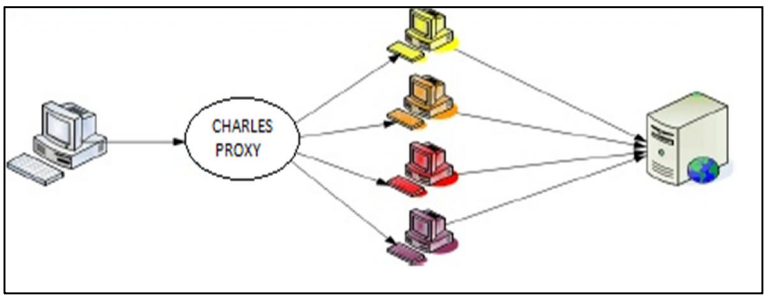


Figura 27. Simulación de múltiples conexiones con CHARLES PROXY

5.6.3. Cuadro de Evaluación de los Prototipos

Se utilizará a continuación la matriz de priorización para determinar cuál de los frameworks es el más adecuado para la aplicación considerada.

	Criterio	RICHFACES	ICEFACES	TRINIDAD
1	Apariencia en los diferentes navegadores	3	3	3
2	Independencia de la resolución	2	2	2
3	Tiempo de descarga o tiempo de inicialización	1	1	2
4	Consumo de canal de red	1	2	2
5	Curva de aprendizaje	9	8	6
6	Herramientas para el desarrollo de software	4	4	2
7	Soporte en los servidores de aplicaciones	4	4	3
8	Soporte del producto	4	3	4
9	Prospectiva Tecnológica	4	4	3
10	Tipo de Licenciamiento	4	2	4
	TOTAL	**36**	**33**	**31**

Tabla 16. Evaluación de Prototipos

5.6.4. Análisis de Resultados

Realizadas las mediciones y tabulados los datos en la matriz de priorización se obtienen los siguientes análisis a partir de los resultados:

- El framework más adecuado para la aplicación propuesta focalizada en un área de negocio específica es RichFaces, no solo por su riqueza visual sino por las facilidades que brinda al desarrollador.
- El punto más fuerte para RichFaces es el uso de patrones de diseño, aceptados y definidos que pueden ser identificados en sus componentes, lo cual le da gran valor.
- Para el desarrollo de aplicaciones con RichFaces se requiere tener conocimiento sobre la tecnología JEE ya que funciona de manera interrelacionada con componentes de esta arquitectura.
- En el caso de las librerías de componentes adicionales, es RichFaces la que tiene mayor puntuación que ICEFaces, destacando como puntos fuertes la madurez, la portabilidad entre servidores, la integración con la capa de negocio.

5.6.5. Selección del Framework

La selección del framework JSF que fue utilizado para el desarrollo del Sistema de Control Microcurricular para la Carrera de Ingeniería en Sistemas, se realizó en base a los parámetros de evaluación que se aplicó en cada uno de los frameworks. De la evaluación podemos seleccionar RichFaces como framework que se utilizará en la capa de presentación para la implementación del sistema como parte práctica de este proyecto en el siguiente capítulo.

5.6.6. Implementaciones en el Área Empresarial

Actualmente a nivel nacional RichFaces es utilizado en el desarrollo de sus aplicaciones de negocios en las siguientes empresas:

Empresa	Versión	Servidor
Petro Comercial	RichFaces 3.3.1	Websphere application server 7.0
SRI	RichFaces 3.3.0	jboss-4.2.2.GA
Correos del Ecuador	RichFaces 3.3.3	jboss-5.1.0.GA
Superintendencia de Compañías	RichFaces 3.3.0	Websphere application server 7.0

Tabla 17. Implementación de RichFaces en Empresas Ecuatorianas

5.7.Ventajas y desventajas de los Frameworks JSF

5.7.1. RichFaces

Algunas ventajas que aporta la utilización de RichFaces son:

- **Soporte de Ajax:** Tenemos que hacer uso de Ajax4JSF, que no es tan transparente para el desarrollador, puesto que, además de introducir los componentes de RichFaces, tenemos que añadir componentes no visuales de la librería Ajax4JSF.
- **Librerías en las que se basan:** Usa el soporte de prototypejs y script.aculo.us, aunque soporta también jquery.
- **Personalización de la interfaz de usuario:** Incorpora el concepto de skins y distribuye 12 temas para ajustar la apariencia y comportamiento de los componentes.

- **Número de componentes:** Tiene una gran variedad de componentes entre los propios de RichFaces y los de Ajax4JSF. Con RichFaces todos los componentes son OpenSource.
- **Licencia:** LGPL V 2.1. en su totalidad.
- **Relevancia:** Es la librería de componentes visuales de Jboss, se integra por defecto con Jboss Seam.

Inconvenientes detectados:

- No se puede realizar una aplicación RichFaces combinándolo con ICEFaces.
- Si no se va a utilizar RichFaces con Seam, y no es demasiado importante el tema de AJAX, ICEFaces es interesante, porque trae un set de componentes mucho más rico que el de RichFaces. ICEFaces tiene actualmente una buena integración con Seam.

5.7.2. ICEFaces

Las ventajas del uso de ICEFaces para enriquecer las aplicaciones web son numerosas. Debido a la cantidad de frameworks existentes, se procede a mostrar una serie de características diferenciadoras respecto a otros frameworks:

- **Experiencia de usuario enriquecedora:** crea una experiencia de usuario superior además de utilizar las ventajas de aplicaciones JEE. Esto se consigue gracias a los componentes que vienen incluidos dentro de la distribución de ICEFaces.
- **Librerías en las que se basan:** Usa el soporte de prototypejs, aunque la parte de Ajax la han rescrito y para los efectos visuales utilizan script.aculo.us.
- **Basado en estándares:** ICEFaces es una solución basada en Java, así que los desarrolladores pueden continuar trabajando de la misma forma que lo hacen. Hay multitud de plugins desarrollados para que ICEFaces sea integrado con multitud de IDEs Java.
- **El Ajax es transparente:** ICEFaces aporta a los programadores un desarrollo con mínimo esfuerzo en la sección JSF.
- **Compatibilidad:** ICEFaces soporta la mayoría los servidores de aplicaciones actuales, aporta plugins para distintos IDEs.
- **Seguridad:** ICEFaces es una de las soluciones Ajax más seguras, es compatible con SSL.

- **Escalabilidad:** El servidor asíncrono HTTP (AHS) aporta una alta escalabilidad para aplicaciones ICEFaces y pueden ser utilizadas por un gran número de usuarios concurrentes.
- Carga de páginas incremental con edición de secciones y sin recargas de página completas.
- Se preserva el contexto del usuario durante la actualización de la página, incluyendo posición del foco y scroll.
- En aplicaciones de tiempo real, las recargas de páginas son asíncronas.
- La comunidad de ICEfaces está creciendo continuamente, este es sin duda un indicador de la calidad del framework. La combinación de JSF y AJAX tiene sus propios problemas. Aunque la arquitectura detrás de ICEfaces es notable, la puesta en práctica tiene sus desafíos. El gestor de incidencias en http://jira.icefaces.org/ puede ayudar a comprobar si el estilo de desarrollo es el problema, o si es la versión actual de ICEfaces. A menudo, el foro ICEfaces puede ofrecer una solución para cualquier inconveniente.

En una comparativa con un producto similar, se opta por compararlo con RichFaces. Es la competencia de RichFaces, tiene tantos o más componentes que ésta y visualmente tiene la misma calidad. La documentación posiblemente es peor que la de RichFaces. El modo en que RichFaces hace funcionar Ajax permite interactuar con otras librerías e incluso añadirle Ajax a componentes que no lo tuvieran, mientras que ICEFaces limita a trabajar con sólo aquellos componentes para los que da soporte.

Inconvenientes detectados:

- **Número de componentes:** Tiene 79 componentes en la versión básica, a los que hay que sumar 32 de la versión empresarial, esta última es de pago. La percepción es que estan inviertiendo esfuerzos en mejorar la versión empresarial y, como es lógico, esperan obtener beneficio económico por ello, como por ejemplo si queremos utilizar el componente Dual List pagamos o lo implementamos nosotros.
- **Licencia:** MPL 1.1, que cubre la LGPL V 2.1. Si bien disponen de una versión empresarial con licencia comercial.

5.7.3. MyFaces Trinidad

Trinidad es una biblioteca de componentes para Java Server Faces (JSF) altamente adaptable a las necesidades diversas de las actuales aplicaciones de web modernas. La adaptación de Ajax hace que su implementación sea transparente, Trinidad cuenta con una colección de componentes muy completa. Su apariencia es fácilmente intercambiable gracias al skinnable.

Como ventajas importantes de Trinidad destacaría:

- **Madurez**: Trinidad es una biblioteca de componentes bastante maduro ya que se ha desarrollado desde principios de 2000 y ya ha aprobado una serie de ciclos de prueba y, además, su continuo desarrollo y los antecedentes comerciales, apuntan hacia un mayor grado de madurez.
- **Soporte**: El soporte se lo realiza principalmente a través de la lista de distribución, lo cual es bastante bueno teniendo en cuenta que todo es libre.
- **La calidad del diseño**: Esto se puede medir por la flexibilidad de uso de cada componente, que es realmente muy alto, como se puede ver fácilmente observando los atributos de control de los componentes.
- Es compatible con otras librerías de componentes, como Tomahawk, Facelets.
- **Skinning** - Soporte para ajustar la apariencia y comportamiento de los componentes.

CAPÍTULO VI

Desarrollo del Aplicativo

6.1.Introducción

En el presente capitulo se explica el desarrollo del Sistema de Control Microcurricular de la Carrera de Ingeniería en Sistemas Computacionales. El desarrollo del proyecto se lo ha realizado tomando los puntos más importantes de la metodología RUP.

En la sección de Gestión del Proyecto se muestran las planificaciones de desarrollo del proyecto, así como el cronograma de ejecución del proyecto, de construcción de la aplicación y cumplimiento de los plazos estimados.

En la sección de **Modelado del Negocio** se encuentran los artefactos utilizados de la metodología para definir un modelo del negocio, modelos de objetos del negocio y el modelo del dominio.

En la sección **Requisitos** se encuentra los artefactos definidos según la metodología, es decir, el documento plan de desarrollo de software, el documento visión, el documento glosario, matrices de atributos de todos los requerimientos, los casos de uso y sus especificaciones.

En la sección **Análisis/Diseño** se muestran el modelo de datos (modelo entidad – relación).

En la sección **Implementación** se muestran los prototipos de interfaces de usuario de la aplicación.

A continuación se detalla las Tecnologías y Herramientas usadas en el Desarrollo del Sistema.

- **Base de Datos**: PostgreSQL versión 8.4
- **Plataforma de Desarrollo**: Java con JDK 1.5
- **IDE de Desarrollo**: Eclipse 3.4
- **Framework de Desarrollo**: JSF 1.2, RichFaces 3.3.3 con EJB3
- **Servidor de Aplicaciones**: Jboss 5.1.0.GA

6.2.Gestión del Proyecto

En esta sección se detalla la planificación inicial del proyecto para la fase de inicio y la fase de elaboración.

6.2.1.1. Alcance

El Plan de Desarrollo de Software describe el plan global usado para el desarrollo del "**SISTEMA DE CONTROL MICROCURRICULAR PARA LA CARRERA DE INGENIERÍA EN SISTEMAS**". Durante el proceso de desarrollo en el artefacto "Visión" se definen las características del producto a desarrollar, lo cual constituye la base para la planificación de las iteraciones.

Para la versión 1.0 del Plan de Desarrollo de Software nos hemos basado en la captura de requisitos por medio del participante en el proyecto representante para hacer una estimación aproximada, una vez comenzado el proyecto y durante la fase de Inicio se generará la primera versión del artefacto "Visión", el cual se utilizará para refinar este documento.

Posteriormente, el avance del proyecto y el seguimiento en cada una de las iteraciones ocasionará el ajuste de este documento produciendo nuevas versiones actualizadas.

6.2.1.2. Resumen

Después de esta introducción, el resto del documento está organizado en las siguientes secciones:

Vista General del Proyecto: Proporciona una descripción del propósito, alcance y objetivos del proyecto, estableciendo los artefactos que serán producidos y utilizados durante el proyecto.

Organización del Proyecto: Describe la estructura organizacional del equipo de desarrollo.

Gestión del Proceso: Explica los costos y planificación estimada, define las fases e hitos del proyecto y describe cómo se realizará su seguimiento.

Planes y Guías de aplicación: Proporciona una vista global del proceso de desarrollo de software, incluyendo métodos, herramientas y técnicas que serán utilizadas.

6.2.2. Vista General del Proyecto

6.2.2.1. Propósito, Alcance y Objetivos

La información que a continuación se incluye ha sido extraída de las diferentes reuniones que se han celebrado con el participante en el proyecto desde el inicio del proyecto.

El proyecto debe proporcionar una respuesta para el desarrollo de todos los módulos implicados en el "**SISTEMA DE CONTROL MICROCURRICULAR PARA EL SEGUIMIENTO DE LA EJECUCION DE LA PLANIFICACION DE LAS MATERIAS DE LA CARRERA DE INGENIERIA EN SISTEMAS COMPUTACIONALES**".

- Módulo de Administración del Sistema
- Módulo de Seguimiento de la Ejecución del Plan Microcurricular de las Materias
- Módulo de Seguridades de usuario, roles y permisos
- Módulo de Reportes

6.2.3. Organización del Proyecto

6.2.3.1. Participantes en el Proyecto.

El personal participante en el proyecto se encuentra formado por los siguientes puestos de trabajo y personal asociado:

- **Jefe de Proyecto**: Ing. Miguel Orquera.
- **Desarrollador**: Egdo. Alex Raúl Caiza.

6.2.3.2. Plan de Fases

El desarrollo se llevará a cabo en base a fases con una o más iteraciones en cada una de ellas. La siguiente tabla muestra la distribución de tiempos y el número de iteraciones de cada fase.

FASE	Nro INTERACIONES	DURACIÓN
Fase de Inicio	1	4 semanas
Fase de Elaboración	2	3 semanas
Fase de Construcción	3	12 semanas
Fase de Transición	2	2 semanas

Tabla 18. Plan de Fase del Proyecto

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

DESCRIPCIÓN	HITOS
Fase de Inicio	En esta fase se desarrollará los requisitos del producto desde la perspectiva del usuario, los cuales serán establecidos en el artefacto Visión. Los principales casos de uso serán identificados y se hará un refinamiento al Plan de Desarrollo de Software.
Fase de Elaboración	En esta fase se analizan los requisitos y se desarrolla un prototipo de arquitectura (incluyendo las partes más relevantes y/o críticas del sistema). Al final de esta fase, todos los casos de uso correspondientes a requisitos que serán implementados en la primera fase de Construcción deben ser analizados y diseñados (en el Modelo de Análisis/Diseño).

Fase de Construcción	Durante la fase de construcción se terminan de analizar y diseñar todos los casos de uso, refinando el Modelo de Análisis/Diseño. Se comienza la elaboración del material de apoyo al usuario. El hito que marca el fin de esta fase es la versión del release del producto, con la capacidad operacional parcial del producto que se haya considerado como crítica.
Fase de Transición	En esta fase se prepararán los ejecutables de la aplicación para distribución, asegurando una implantación y cambio del sistema previo de manera adecuada. El hito que marca el fin de esta fase incluye, la documentación del proyecto con los manuales de instalación y todo el material de apoyo al usuario.

Tabla 19. Hitos de las Fases de un Proyecto

6.3.Plan de Desarrollo de Software

A continuación se presentan las herramienta utilizadas para declarar los requisitos del software, es decir, el documento del plan de desarrollo de software, el documento visión, el documento glosario y tanto las especificaciones de los casos de uso.

6.3.1. Visión

6.3.1.1. Introducción

El propósito de este documento es definir los requisitos de la aplicación SCMCIS en términos de las necesidades de los usuarios finales.

El **"SISTEMA DE CONTROL MICROCURRICULAR PARA LA CARRERA DE INGENIERIA EN SISTEMAS"**, está diseñado para reducir el esfuerzo manual de los Docentes y contribuir al control y verificación por parte del Director de la Carrera y el Jefe de Área, del cumplimiento de la Planificación de Unidades Temáticas por materia.

El **"SISTEMA DE CONTROL MICROCURRICULAR PARA LA CARRERA DE INGENIERIA EN SISTEMAS"**, es una aplicación Web que permitirá controlar el Seguimiento de la Ejecución del Plan Microcurricular de las Materias.

6.3.1.2. Alcance

El sistema de Control Microcurricular para el Seguimiento de la Ejecución del Plan Microcurricular de las Materias de la CISIC será una aplicación WEB a la que podrán acceder todos los Usuarios que se encuentren registrados en la Aplicación, y que realicen las actividades de seguimiento del plan Microcurricular de las materias de la Carrera de Ingeniera en Sistemas Computacionales, la cual será instalada en la infraestructura informática de la FICA que administrará la aplicación.

Mantener almacenado el material académico (documentos, diapositivas, ejercicios, deberes, referencias, bibliografía) que se utilizaron en cada clase para el desarrollo del contenido Microcurricular.

Permitir el acceso a los recursos académicos utilizados en cada uno de los contenidos de las Unidades Temáticas.

6.3.2. Posicionamiento

6.3.2.1. Definición del Problema

El problema de	No tener conocimiento del avance del Seguimiento de la Ejecución del Plan Microcurricular de las Materias de la Carrera de Ingeniería en Sistemas emitidas por los Docentes. Actualmente no existe un sistema que centralice esta información para facilitar el control del proceso de enseñanza de los Contenidos Temáticos de las Unidades de cada materia.
afecta a	Todos los usuarios involucrados con los procesos del Seguimiento de la Ejecución del Plan Microcurricular: Director de Escuela, Docentes, Estudiantes.
el impacto de ello es	Tener un control sobre el seguimiento de la Ejecución del Plan Microcurricular de cada una de las Materias.
una solución exitosa debería	Permitir implementar una solución informatizada diseñada, construida y soportada por la última tecnología de desarrollo de aplicaciones JEE, facilitando: - Aportar a la operación del negocio sin riesgos y adecuada a

	las necesidades de todos los participantes - Mantener, agregar, cambiar y mejorar el producto - Administrar centralizadamente la aplicación - Escalar sin inconvenientes si la demanda lo requiere - Cubrir las necesidades operativas de los usuarios
Para	Aquellos que necesiten interactuar con el seguimiento de la Ejecución el Plan Microcurricular de Materias
Quienes	Tengan soluciones que no se ajustan a sus necesidades.
El (Nombre del producto)	Sistema de Control Microcurricular para la Carrera de Ingeniera en Sistemas Computacionales. (SCM-CISIC)
Que	Es una herramienta que está desarrollada con tecnología JEE basada en java.
A diferencia de	La forma de gestionar y controlar actualmente el Seguimiento de la Ejecución del Plan Microcurricular de las Materias.
Nuestro producto	Permitirá Principalmente: - Registrar el Contenido Temático las Unidades de las Materias. - Realizar el seguimiento de la Ejecución del Plan Microcurricular de las Unidades Temáticas de las Materias. - Obtener información de los temas dictados en Clase por los Docentes. - Registrar el porcentaje de avance de la Planificación de las Unidades Temáticas dictados en clase. - Adjuntar la documentación utilizada en los temas dictados en clase. - Generar reportes de avance de la Planificación de unidades temáticas por Materia.

Tabla 20. Definición del Problema

6.3.2.2. Oportunidad de Negocio

Este sistema permitirá informatizar el seguimiento del proceso de enseñanza, lo cual supondrá un acceso rápido y sencillo a los datos, gracias a interfaces gráficas sencillas y amigables. Además los datos accedidos estarán siempre actualizados, lo cual es un factor muy importante.

6.3.2.3. Descripción de Participantes en el Proyecto y Usuarios

Para proveer de una forma efectiva productos y servicios que se ajusten a las necesidades de los usuarios, es necesario identificar e involucrar a todos los participantes en el proyecto como parte del proceso de modelado de requerimientos. También es necesario identificar a los usuarios del sistema y asegurarse de que el conjunto de participantes en el proyecto los representa adecuadamente. Esta

sección muestra un perfil de los participantes y de los usuarios involucrados en el proyecto, así como los problemas más importantes que éstos perciben para enfocar la solución propuesta hacia ellos. No describe sus requisitos específicos ya que éstos se capturan mediante otro artefacto. En lugar de esto proporciona la justificación de por qué estos requisitos son necesarios.

6.3.2.4. Resumen de Usuarios

Nombre	Descripción	Responsabilidades
Administrador	Responsable de la gestión de roles y permisos, agregar, borrar y modificar información de las cuentas de usuarios, reestablecer contraseñas, etc.	Seguridad
Personal Administrativo	Responsable del correcto ingreso de datos como de estudiante, docentes, materias, cursos y sílabos.	Ingreso de Datos
Director de Carrera	Responsable de controlar el avance del contenido temático dictado por cada uno de los docentes.	Control
Jefe de Área	Responsable de la revisión del seguimiento de las materias del área asignada.	Revisión
Docente	Responsable del registro del seguimiento de la planificación de la materia por curso.	Ingreso del seguimiento por materia dictada.
Estudiante	Responsable del registro de la aprobación de los temas de clase dictados por el docente.	Aprobación de los temas de clase

Tabla 21. Resumen de los Usuarios del Proyecto

6.3.2.5. Entorno de Usuarios

Los usuarios entrarán al sistema mediante un navegador web, identificándose de acuerdo al tipo de usuario con su usuario y password, tras este paso accederán a la aplicación diseñada de acuerdo a los requerimientos de cada usuario. Los reportes serán generados en pdf, lo cual le resultará familiar.

6.3.3. Descripción Global de Producto

6.3.3.1. Perspectiva del Producto

El producto a desarrollar es el **"SISTEMA DE CONTROL MICROCURRICULAR PARA LA CARRERA DE INGENIERÍA EN SISTEMAS"**, con la intención de agilizar su funcionamiento. Las áreas a tratar por el sistema son:

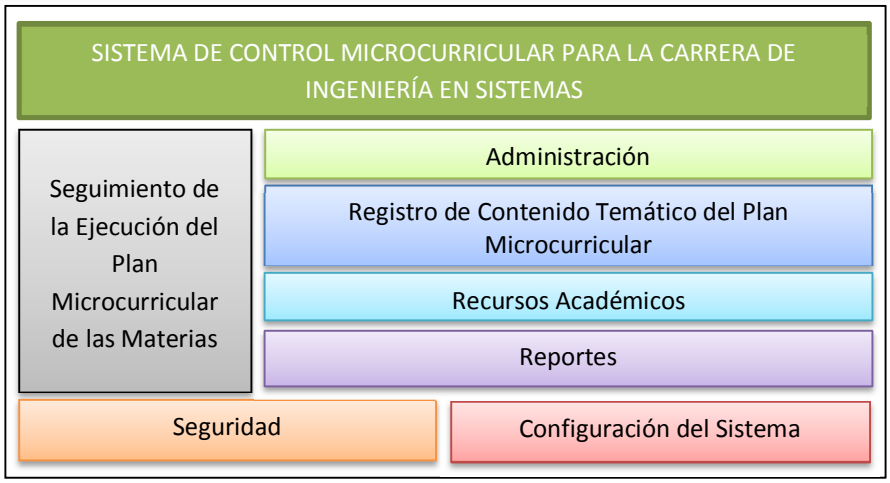


Figura 28. Bloques Modulares del Sistema de Control Microcurricular

El Sistema para Control Microcurricular tiene como objetivos:

1. Realizar un Seguimiento del Plan Microcurricular de los contenidos de cada materia de la carrera de ingeniería en sistemas computacionales.
2. Permitir el acceso a los recursos académicos utilizados en cada uno de los contenidos temáticos de las materias.
3. Mantener almacenado el material académico (documentos, diapositivas, ejercicios, referencias, bibliografía) que se utilizó en cada clase para el desarrollo del contenido Microcurricular.

Módulos y Funcionalidad

El sistema está estructurado de los siguientes módulos:

1. **Módulo de Administración**

   - Administración de Docentes
   - Administración de Estudiantes
   - Administración de Materias
   - Administración de Cursos
   - Administración de Personal Administrativo
   - Administración de Periodos Académicos

2. **Registro de Contenido Temático del Plan Microcurricular**

   - Registrar el Contenido Temático del Plan Microcurricular de cada Materia.

- Registrar de los Contendidos Temáticos dictados en Clase.
- Confirmar que los Contenidos Temáticos fueron dictados en clase

3. Recursos Académicos

- Guardar los recursos académicos que han sido utilizados para dictar Clase.

4. Reportes

- Permite generar los diferentes reportes para realizar el seguimiento de la ejecución del Plan Microcurricular de las Materias.

5. Seguridad y Configuración del Sistema

- Creación y edición de Permisos
- Creación y edición de Roles
- Creación y edición de Usuarios
- Edición de los parámetros del sistema

6.3.3.2. Resumen de las Características

A continuación se mostrará un listado de los beneficios que obtendrá el cliente a partir del producto.

Beneficio del cliente	Características que lo apoyan
Mayor agilidad y rapidez para realizar el seguimiento del Plan Microcurricular de los contenidos de cada materia de la carrera de ingeniería en sistemas	Aplicación web.
Mayor control de información.	Una base de datos para realizar el registro de la información del Seguimiento del Plan Microcurricular de los contenidos de cada materia de la carrera de ingeniería en sistemas computacionales como si fuera en el sitio mismo.
Seguridad.	El ingreso del sistema se controla por medio de un usuario y contraseña, se controla el acceso a las opciones a través de roles y permisos.

Tabla 22. Resumen de las características del Proyecto

6.3.4. Requisitos

La Facultad de Ingeniería en Sistemas Computacionales dispone con los recursos necesarios para la construcción del sistema: Comunicaciones y Hardware.

6.3.4.1. Requisitos de Desempeño

El mayor requisito de desempeño es la facilidad y rapidez para el acceso de datos debido a que sus datos de encuentran distribuidos evitando los cuellos de botellas. Y el sistema es agradable a la vista del usuario.

6.3.4.2. Requisitos de Entorno

El hardware debe estar conectado de manera adecuada. Se debe mantener siempre los dispositivos de red y servidores web para el perfecto ingreso al sistema. Dispone de un Equipo servidor que permite el desempeño de la Base de Datos y el Servidor de Aplicaciones.

6.3.4.3. Requisitos de Documentación

- **Manual de Usuario**

  El manual de usuario se encuentra en los anexos el cual contendrá información como: instalación del sistema, modo de acceder a cada rol.

- **Guías de Instalación, Configuración**

  Las guías de instalación, configuración se encuentra en los anexos.

6.4.Modelado del Negocio

A continuación se presentan los modelos definidos como modelo del negocio, modelo de datos y modelo de análisis y diseño de los módulos de la aplicación:

- Módulo de Administración del Sistema
- Módulo de Seguimiento de la Ejecución del Plan Microcurricular de las Materias
- Módulo de Seguridades
- Módulo de Reportes

6.4.1. Use Case UC1: Módulo de Administración del Sistema

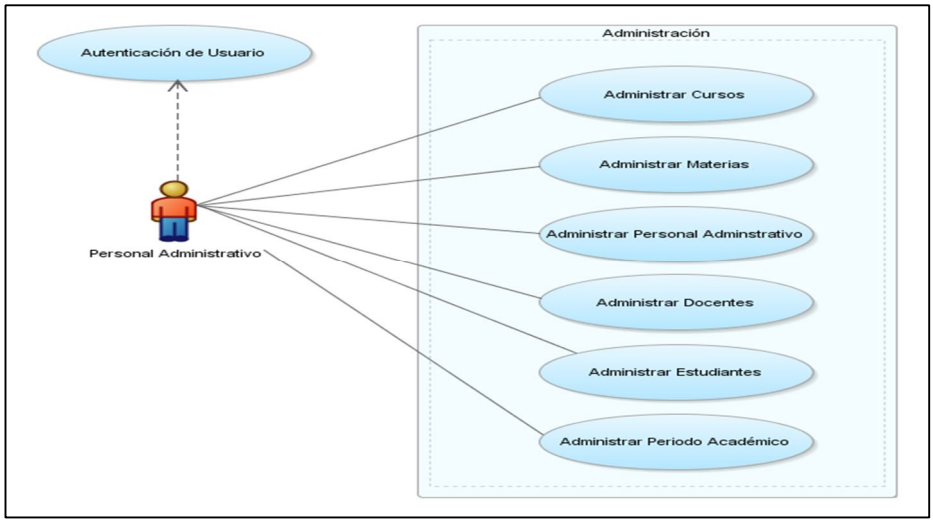


Figura 29. UC1: Módulo de Administración del Sistema

Especificación del Caso de Uso

Este caso de uso describe el proceso para Registrar la información acerca de Cursos, Materias, Docentes, Estudiantes, Periodo Académico. Este proceso se lo realiza mediante el uso de una interfaz gráfica donde se registran la información.

Objetivo

Registrar la información acerca de Cursos, Materias, Docentes, Estudiantes, Periodo Académico.

Precondiciones

El Usuario debe estar ingresado y autenticado en el Sistema.

Poscondiciones de éxito

El Registro de la información de Cursos, Materias, Docentes, Estudiantes, Periodo Académico fue realizado satisfactoriamente.

Poscondiciones de falla

El Registro de la información de Cursos, Materias, Docentes, Estudiantes, Periodo Académico no será registrado.

Se indicará el motivo por el cual no se pudo registrar la información

Actor Principal

Actor	Evento
Personal Administrativo	Es el usuario encargado de Registrar la información acerca de Cursos, Materias, Docentes, Estudiantes, Periodo Académico.

Flujo de eventos

Flujos Básicos

Administración de Docentes

El usuario puede crear modificar los datos de un Docente

Administración de Estudiantes

El usuario puede crear modificar los datos de un Estudiante

Administración de Materias

El usuario puede crear modificar los datos de una Materia

Administración de Cursos

El usuario puede crear modificar los datos de un Curso

El usuario puede asignar estudiantes a un determinado Curso

Administración de Personal Administrativo

El usuario puede crear modificar los datos del Personal Administrativo

Administración de Periodos Académicos

El usuario puede crear, cerrar un Periodo Académico.

6.4.2. Use Case UC2: Registrar el Plan Microcurricular de una Materia

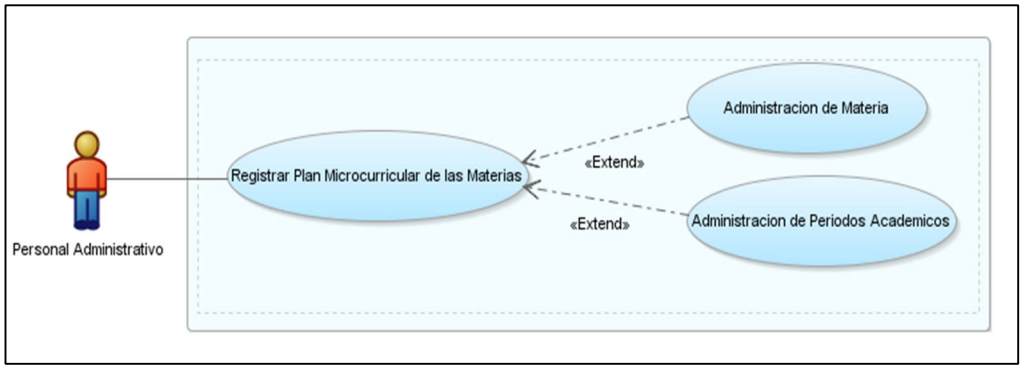


Figura 30. UC2: Registrar el Plan Microcurricular de una Materia

Especificación del Caso de Uso

El Personal Administrativo inicia el caso de uso. Una vez ingresado al sistema este le muestra las diferentes opciones que ella puede administrar. Por ejemplo, Registrar los Planificación de Unidades Temáticas de una Materia, Periodos Académicos, Crear Paralelos, Docentes y Estudiantes, etc. Podrá crear, editar y eliminar según sea el criterio o las necesidades. Para su entendimiento vamos a explicar cómo crear Planificación de Unidades Temáticas de una Materia.

Objetivo

Registrar el Plan Microcurricular de una Materia.

Precondiciones

El Personal Administrativo debe ingresar con su cuenta y contraseña para poder acceder a los diferentes módulos. El usuario debe tener los permisos necesarios para acceder al módulo de Registro de Planes Microcurriculares de las Materias.

Poscondiciones de éxito

El Registro de los Planes Microcurriculares de las Materias se realiza correctamente.

Poscondiciones de falla

El Registro de los Plan Microcurricular no será registrado.

Se indicará el motivo por el cual no se pudo realizar el registro del Plan Microcurricular de una Materia

Flujo de eventos

Registrar Planificación Temática

Flujos Básicos

1. El caso de uso comienza cuando el Usuario ingresa en el Sistema y hace clic en Registrar Planificación Temática.
2. El Sistema muestra el formulario para ingresar los datos para crear una Nueva Planificación Temática.

3. El usuario selecciona la Materia para registrar las Unidades y Contenidos Temáticos del Plan Microcurricular.
4. Una vez ingresado todos los datos debe elegir la opción de "Guardar" para registrar la información del nuevo Plan Microcurricular de una Materia.
5. El Usuario presiona el botón Guardar para registrar la información en las correspondientes entidades de datos.
6. El sistema despliega el resultado del proceso de registro.
7. Una vez guardado la Planificación Temática de la Materia se presenta un resumen de los datos ingresados.
8. Termina el C.U.

Flujos Alternativos

Editar Planificación Temática

Flujos Básicos

1. El Usuario define la búsqueda ya sea mediante materia, periodo académico.
2. Una vez localizada la Planificación Temática que se desea editar, procede a editar los campos que se crea modificar.
3. Una vez actualizados los datos debe elegir la opción de "Guardar" para guardar los datos modificados de la Planificación Temática de la Materia seleccionada.
4. El sistema se encarga de guardar los datos editados de la Planificación Temática.
5. El sistema despliega el resultado del proceso de registro.
6. Termina el C.U.

6.4.3. Use Case UC3: Registro de Contenidos Temáticos Dictados en Clase

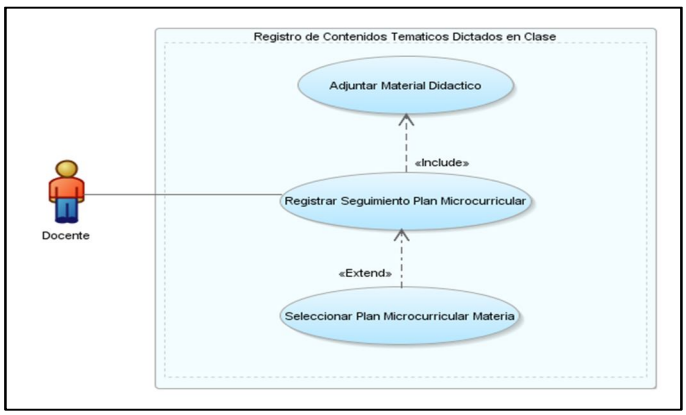


Figura 31. UC3: Registro de Contenido Temático Dictado en Clase de las Materias

Especificación del Caso de Uso

Este caso de uso describe el proceso para Registrar el Contenido Temático dictado en Clase del Plan Microcurricular de las Materias. Este proceso lo realiza un docente, mediante el uso de una interfaz gráfica donde se registran los contenidos temáticos dictados en clase por parte de docente de una materia en particular.

Objetivo

Registrar los contenidos temáticos del Plan Microcurricular dictados en Clase de las Materias.

Precondiciones

El docente debe estar ingresado y autenticado en el Sistema.

El docente debe tener asignado las materias que va dictar en el periodo académico actual.

Debe existir información en la base de datos del sistema respecto a Cursos, Estudiantes, Docentes, Materias, Plan Microcurricular de la Materia.

Poscondiciones de éxito

El Registro de los contenidos temáticos del Plan Microcurricular dictados en clase fue realizado satisfactoriamente.

Poscondiciones de falla

El Registro de los contenidos temáticos del Plan Microcurricular dictados en Clase no será registrada.

Se indicará el motivo por el cual no se pudo registrar los contenidos temáticos del Plan Microcurricular dictados en Clase

Actor Principal

Actor	Evento
Docente	Es el usuario encargado del registro los contenidos temáticos del Plan Microcurricular dictados en Clase.

Flujo de eventos

Flujos Básicos

1. El Docente ingresa al sistema con su cuenta con su respectivo nombre de usuario, ingresando también su contraseña mediante la interfaz del sistema ("Ingreso al Sistema").
2. El Docente selecciona la Opción de Menú "Registrar Contenido Temático" del Menú principal "Plan Microcurricular".
3. El sistema despliega las materias del Periodo Académico Actual del Docente.
4. El Docente selecciona la Materia para registrar los contenidos dictados en clase.
5. El sistema despliega el formulario del Plan Microcurricular de la Materia Seleccionada para el registro de los contenidos temáticos dictados en clase.
6. El Docente selecciona los contenidos dictados en clase.
7. El Docente puede adjuntar el Material Didáctico que es utilizado en el desarrollo de la clase, ingresar observaciones a cada uno de los contenidos temáticos si lo hubiese.
8. El docente presiona el botón Guardar para registrar la información en las correspondientes entidades de datos.
9. El sistema despliega el resultado del proceso de registro .
10. Termina el C.U.

Flujos Alternativos

En el paso 4 del flujo básico

El docente puede seleccionar la opción "Fecha de Clase" para registrar los contenidos dictados en la fecha seleccionada.

El sistema despliega el calendario de fechas de las materia selecciona.

El Docente selecciona la fecha de Clase para registrar los contenidos dictados en esa fecha de clase.

6.4.4. Use Case UC4: Aprobar Contenido Temático Dictado en Clase

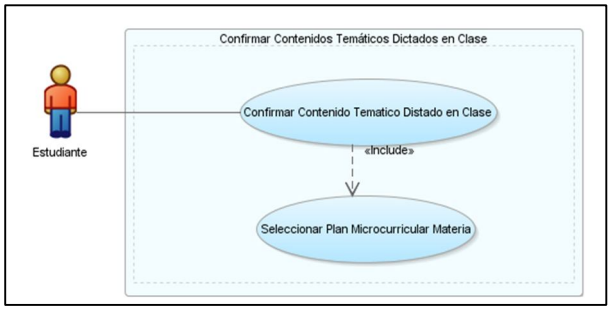


Figura 32. UC4: Confirmar Contenidos Temáticos Dictados en Clase

Especificación del Caso de Uso

Este caso de uso describe el proceso para Confirmar los Contenidos Temático dictados en Clase del Plan Microcurricular de las Materias. Este proceso lo realiza un estudiante, mediante el uso de una interfaz gráfica donde se realiza la confirmación de los contenidos temáticos dictados en clase por parte de un docente de una materia en particular.

Objetivo

Confirmar los Contenidos Temáticos del Plan Microcurricular dictados en Clase de las Materias.

Precondiciones

El estudiante debe estar ingresado y autenticado en el Sistema.

El estudiante debe tener asignado a los cursos de las materias del periodo académico actual.

Debe existir información en la base de datos del sistema respecto a Cursos, Estudiantes, Docentes, Materias, Plan Microcurricular de la Materia.

Poscondiciones de éxito

La confirmación de los contenidos temáticos del Plan Microcurricular dictados en clase se realizado satisfactoriamente.

Poscondiciones de falla

La confirmación de los contenidos temáticos del Plan Microcurricular dictados en Clase no será registrada.

Se indicará el motivo por el cual no se pudo realizar la confirmación de los contenidos temáticos del Plan Microcurricular dictados en Clase

Actor Principal

Actor	Evento
Estudiantes	Es el usuario encargado de la confirmación de los contenidos temáticos del Plan Microcurricular dictados en Clase.

Flujo de eventos

Flujos Básicos

1. El Estudiante selecciona la Opción de Menú "Confirmar Contenido Temático" del Menú principal "Plan Microcurricular".
2. El sistema despliega las materias del Periodo Académico Actual del Estudiante.
3. El Estudiante selecciona la Materia para confirmar los contenidos dictados en clase que fueron dictados por el Docente.
4. El sistema despliega el formulario de los Contenidos del Plan Microcurricular de la Materia Seleccionada para la confirmación de los contenidos temáticos dictados en clase.
5. Debe seleccionar los Contenidos Temáticos que se Dictaron en clase para realizar la confirmación.
6. El Estudiante presiona el botón Guardar para registrar la información en las correspondientes entidades de datos.
7. El sistema despliega el resultado del proceso de registro .
8. Termina el C.U.

Flujos Alternativos

N/A

6.4.5. Use Case UC5: Administración de Seguridades

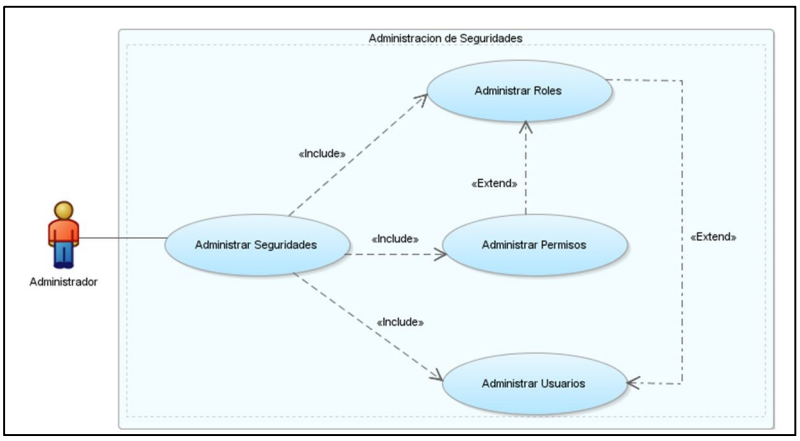


Figura 33. UC5: Administrar Seguridades

Especificación del Caso de Uso

Este caso de uso describe el proceso para administrar el Módulo de seguridades para el acceso a los recursos de la aplicación web. Define un esquema de permisos sobre recursos de la aplicación Web por medios Roles Permisos y Usuarios.

En la aplicación Web existen diferentes módulos disponibles que desean ser controlados. Con la funcionalidad de seguridad es posible definir quienes tienen acceso a qué recursos de la aplicación Web a partir de un determinado Rol.

Objetivo

Administrar el Modulo de Seguridades para controlar el acceso de los usuarios a la aplicación web.

Precondiciones

El usuario administrador de la aplicación debe estar activo en la aplicación (admin).

Poscondiciones de éxito

El usuario administrador pude crear, modificar, eliminar Permisos, Roles, Usuarios correctamente.

Poscondiciones de falla

La administración de Roles Permisos Usuarios no se puede realizar.

Se indicará el motivo por el cual no se pudo realizar la operación sobre el módulo de seguridades

Actor Principal

Actor	Evento
Administrador	Es el usuario encargado de administrar el Modulo de Seguridades para controlar el acceso de los usuarios a la aplicación web.

Flujo de eventos

Flujos Básicos

1. El Administrador selecciona la Opción de Menú "Roles" del Menú principal "Seguridades".

   El Administrador puede realizar las siguientes tareas sobre el Módulo de Roles:

   - Crear Roles
   - Modificar Roles
   - Eliminar Roles
   - Asignar Permisos al Rol

2. El Administrador selecciona la Opción de Menú "Permisos" del Menú principal "Seguridades".

   El Administrador puede realizar las siguientes tareas sobre el Módulo de Permisos:

   - Crear Permisos
   - Modificar Permisos
   - Eliminar Permisos

3. El Administrador selecciona la Opción de Menú "Usuarios" del Menú principal "Seguridades".

El Administrador puede realizar las siguientes tareas sobre el Módulo de Usuarios:

- Crear Usuarios
- Modificar Usuarios
- Eliminar Usuarios
- Asignar Roles al Usuarios

Flujos Alternativos

En el paso 1 del flujo básico

1.1 Para crear un nuevo Rol

Seleccionar la opción Nuevo.

El sistema despliega el Formulario para ingresar los datos del Nuevo Rol.

- Nombre
- Descripción
- Seleccionar los permisos del Rol

1.2 El Administrador presiona el botón Guardar.

1.3 El sistema despliega el resultado del proceso de registro.

En el paso 2 del flujo básico

2.1 Para crear un nuevo Permiso

Seleccionar la opción Nuevo.

El sistema despliega el Formulario para ingresar los datos del Nuevo Permisos.

- Nombre del Permiso
- Descripción del Permiso
- URL: Dirección de un recurso de la aplicación
- Orden del Menú
- Grupo: Seleccionar el grupo que pertenece

2.2 El Administrador presiona el botón Guardar.

2.3 El sistema despliega el resultado del proceso de registro.

En el paso 3 del flujo básico

3.1 Para crear un nuevo Usuario

Seleccionar la opción Nuevo.

El sistema despliega el Formulario para ingresar los datos del Nuevo Usuario.

- Usuario
- Password
- Nombre
- Tipo de Usuario
- Seleccionar los Roles del nuevo Usuario

3.2 El Administrador presiona el botón Guardar.

3.3 El sistema despliega el resultado del proceso de registro.

6.4.6. Use Case UC6: Reportes

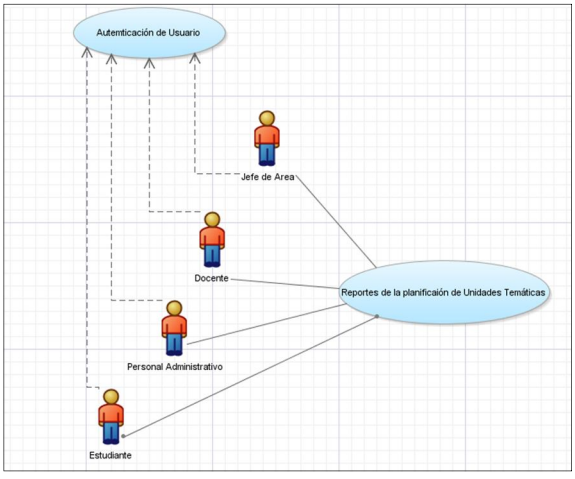


Figura 34. UC6: Reportes

Especificación del Caso de Uso

Este caso de uso describe el proceso para generar los diferentes reportes para realizar el seguimiento de la ejecución del Plan Microcurricular de las Materias.

Objetivo

Administrar el Modulo de Seguridades para controlar el acceso de los usuarios a la aplicación web.

Precondiciones

Se haya ingresado al sistema con éxito y exista todo tipo de información respecto al Seguimiento de la Ejecución del Plan de Clase de las Materias.

Poscondiciones de éxito

Presentación de Reportes de acuerdo a las opciones seleccionadas.

Poscondiciones de falla

No se presentan los datos de los Reportes

Actor Principal

Actor	Evento
Todos	Generación de Reportes de los Módulos del Sistema

Flujo de eventos

Flujos Básicos

1. Generar Reporte del Plan Microcurricular de una Materia.
2. Generar Reporte de Contenidos Distados en Clase por Materia.
3. Generar Reporte de Porcentaje de Avance por Materia
4. Generar Reporte de Porcentaje de Avance por Área.

Flujos Alternativos

N/A

6.4.7. Use Case UC7: Autenticación en el Sistema

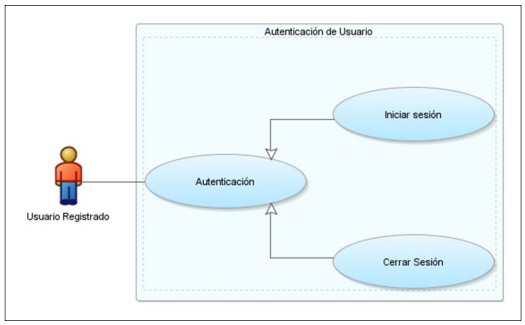


Figura 35. UC7: Autenticación en el Sistema

Especificación del Caso de Uso

En este caso de uso permite al usuario poder ingresar al sistema según el rol y los privilegios asignados. Este puede ser: Personal Administrativo, Jefe de Área, Docente o Estudiante.

Objetivo

Permitir al Usuario iniciar sesión en el sistema.

Precondiciones

El usuario debe tener previamente creado su cuenta. En caso de no tenerla deberá hablar con el administrador para que le proporcione una cuenta.

Poscondiciones de éxito

La autenticación del usuario se realiza correctamente.

Poscondiciones de falla

No se realiza la autenticación del Usuario en el sistema

Flujo de eventos

Flujos Básicos

1. El sistema presenta la pantalla de autentificación de usuarios para ingresar a la aplicación.
2. El sistema solicita al usuario los datos de su cuenta este puede ser: Docente o Estudiante, Personal Administrativo.
3. El usuario debe ingresar los datos de su cuenta: Cedula y Password.
4. Hacer clic en el botón “Ingresar” para realizar la autentificación del usuario con los datos ingresados.
5. Si los datos del Usuario son correctos el Sistema permite el acceso al sistema.
6. El sistema presenta los recursos a los que tiene permiso el Usuario en base a los Roles que tiene asignado el Usuario

7. Si los datos del Usuario son incorrectos el Sistema solicita nuevamente los datos del usuario para realizar la autentificación.
8. Termina C.U.

Flujos Alternativos

N/A

6.5.Análisis de Diseño

A continuación se presentan el modelo de datos (modelo entidad - relación).

6.5.1. Diagrama Entidad - Relación

6.6.Arquitectura del Sistema

6.6.1. Introducción

Una Arquitectura de Software, también denominada Arquitectura lógica, consiste en un conjunto de patrones que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información.

La arquitectura de la aplicación permite definir la organización interna de la misma estableciendo una funcionalidad en capas independientes que permiten tener mejor control de la ubicación de las reglas de negocio y por lo tanto facilita las tareas de mantenimiento de la mismas.

6.6.2. Objetivo

El objetivo de este documento es presentar la descripción de cada uno de los componentes arquitecturales para la implementación del Sistema de Control Microcurricular.

6.6.2.1. Objetivos del Proyecto

El objetivo del proyecto es la implementación de un Sistema Web que permita realizar el Seguimiento de la Ejecución del Plan Microcurricular de las Materias de la Carrera de Ingeniería en Sistemas Computacionales.

6.6.3. Arquitectura de la Solución

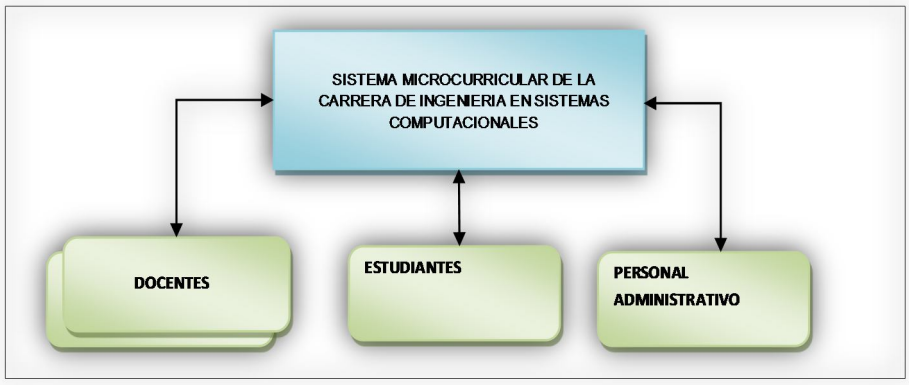
6.6.3.1. Diagrama general de Arquitectura

En este apartado se muestra un diagrama general de la arquitectura del SISTEMA DE CONTROL MICROCURRICULAR.

Por un lado tenemos a los docentes responsables de registrar los contenidos dictados en clase, puede adjuntar la documentación utilizada en cada uno de los temas de clase. Además el estudiante asignado para la aprobación de los temas que fueron dictados por el docente pude confirmar los temas dictados.

Generar Reporte del Avance de Contenidos Temáticos para ver el estado de avance de cada una de las materias.

El estudiante pude ingresar al sistema para descargar la información utilizada en cada uno de los temas de clase.



6.6.4. Arquitectura JEE

La arquitectura de la aplicación va a estar basada en la arquitectura JEE y en patrones de diseño ampliamente usados, así como estándares de desarrollo abiertos. Una arquitectura multicapa particiona el sistema en distintas unidades funcionales: cliente, presentación, lógica de negocio, integración, y sistema de información empresarial (EIS). Esto asegura una división clara de responsabilidades por funcionalidad y hace que el sistema sea mantenible y extensible.

6.6.4.1. Capa del cliente

La capa del cliente es donde se consumen y presentan los modelos de datos. Para una aplicación Web, la capa cliente normalmente es un navegador web.

6.6.4.2. Capa de presentación

La capa de presentación expone los servicios de la capa de lógica de negocio a los usuarios. Sabe cómo procesar una petición de cliente, cómo interactuar con la capa de lógica de negocio, y cómo seleccionar la siguiente vista a mostrar.

Esta capa resuelve la presentación de datos al usuario, esta capa se encarga de presentar las pantallas de la aplicación al usuario, y tomar los eventos que el cliente genere. También es conocida como interfaz gráfica y debe tener la característica de

ser amigable, entendible y fácil de usar para el usuario. Esta capa se comunica con la capa de negocio.

En la capa de presentación está el navegador que permite visualizar la aplicación web, el mismo que se comunica con el servidor de aplicaciones conformando la lógica de negocio y posteriormente se accede a la base de datos.

Existen numerosas tecnologías para esta capa (JSF, Struts, Flex, GWT, etc.)

6.6.4.3. Capa de Negocio

La capa de negocio conocida como capa de la lógica de negocio contiene los objetos y servicios de negocio de la aplicación.

Esta capa resuelve la lógica de la aplicación. Contiene los algoritmos, validaciones y coordinación necesaria para brindar los servicios de negocio de la aplicación. Los elementos fundamentales de esta capa son los objetos de dominio que representan los principales objetos del negocio. (Por ejemplo, un Docente, un Silabo, un Plan de Clase, Un Estudiante). La lógica para manipularlos se encuentra en los llamados objetos de negocio (Business Object). Estos objetos contienen la lógica del negocio, y manipulan a los objetos de dominio.

La capa de negocio se encuentra en el servidor de aplicaciones aquí es donde se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio e incluso lógica de negocio porque aquí es donde se establecen todas las reglas que se deben cumplir.

6.6.5. Capa de datos

Esta capa resuelve el acceso a la información. Es donde residen los datos y es la encargada de acceder a los mismos. Reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Para implementar la arquitectura JEE se usará el servidor JBoss Application Server jboss-5.1.0.GA con soporte para JEE 1.5.

6.6.6. Arquitectura de la aplicación

- Arquitectura JEE
  - Capa de Presentación.
    - JSF 1.2
    - Facelets
    - RichFaces 3.3.3
  - Capa de Negocio
    - Implementación de la lógica de negocio mediante componentes EJB 3.0 (Enterprise JavaBeans).
  - Capa de Persistencia
    - Hibernate 3 / JPA

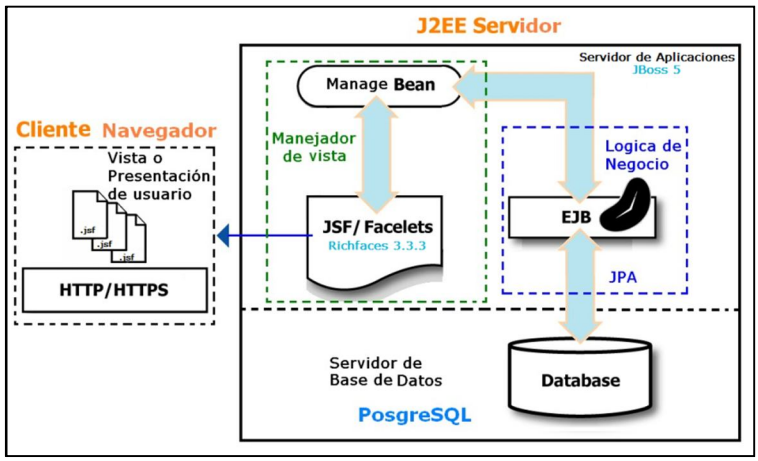


Figura 36. Arquitectura de la Aplicación

Para implementar la capa de presentación se utiliza el patrón MVC, se usará una combinación entre el Framework JavaServer Faces JSF 1.2, y la librería de componentes RichFaces 3.3.3 que permiten crear un ambiente Ajax de manera fácil, rápida y limpia.

Esta arquitectura permite separar en capas la aplicación dividiendo responsabilidades por funcionalidad. Esto proporciona un alto grado de flexibilidad ya que se consigue que al hacer modificaciones en unas capas.

Model (Modelo): Encapsula los datos y las funcionalidades, siendo independiente del comportamiento de entrada y de la representación de salida.

View (Vista): Muestra la información al usuario, recogiendo los datos del modelo.

Controller (Controlador): La principal función del controlador es asociar a las pantallas, clases java que recogen la información introducida y que disponen de métodos que responden a las acciones del usuario.

Para implementar la capa de negocio se utiliza EJB 3.0 EJB (Enterprise JavaBeans) que son componentes que encapsulan la lógica de negocio de una aplicación Se trata de construir los procesos de negocio que luego serán llamados por la vista.

Una de las nuevas características interesantes de EJB 3 es la forma en que maneja la persistencia. La persistencia es la capacidad de tener los datos en objetos Java para que se almacenen automáticamente en una base de datos relacional**,** como PostgreSQL. La persistencia en EJB 3 es administrada por JPA**,** automáticamente persiste los objetos Java usando una técnica llamada mapeo objeto-relacional (ORM). ORM es esencialmente el proceso de mapeo de datos realizada en los objetos Java a tablas de bases de datos. Se le libera de la tarea de escribir código JDBC de bajo nivel, y el complejo de la persistencia de objetos en una base de datos.

6.6.7. Estructura de la aplicación

La aplicación será construida en archivos ejecutables: prjSMCModel.jar y prjSMCWeb.war para ser desplegada en el servidor jboss-5.1.0.GA.

6.6.8. Almacenamiento de Datos

Toda la información relativa a los Docentes, Estudiantes, Materias, Contenidos Temático, etc., son almacenadas en una base de datos relacional. El servidor de base de datos que se va a utilizar es PostgreSQL 8.4. El modelo de datos que representa la información que se guarda en la base de datos está definido en el "Diagrama Entidad-Relación".

6.7.Implementación

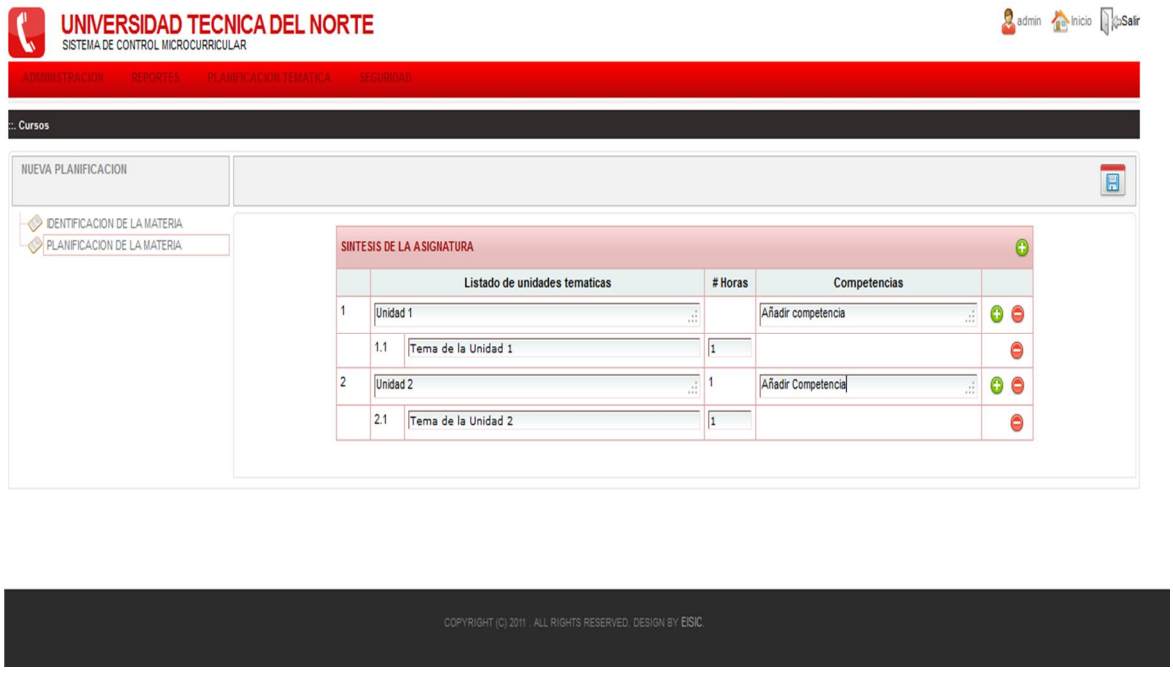
A continuación se presentan los modelos definidos como prototipos de interfaces gráficas de usuario diseñados para la aplicación final.

6.7.1. Interfaces de Usuarios

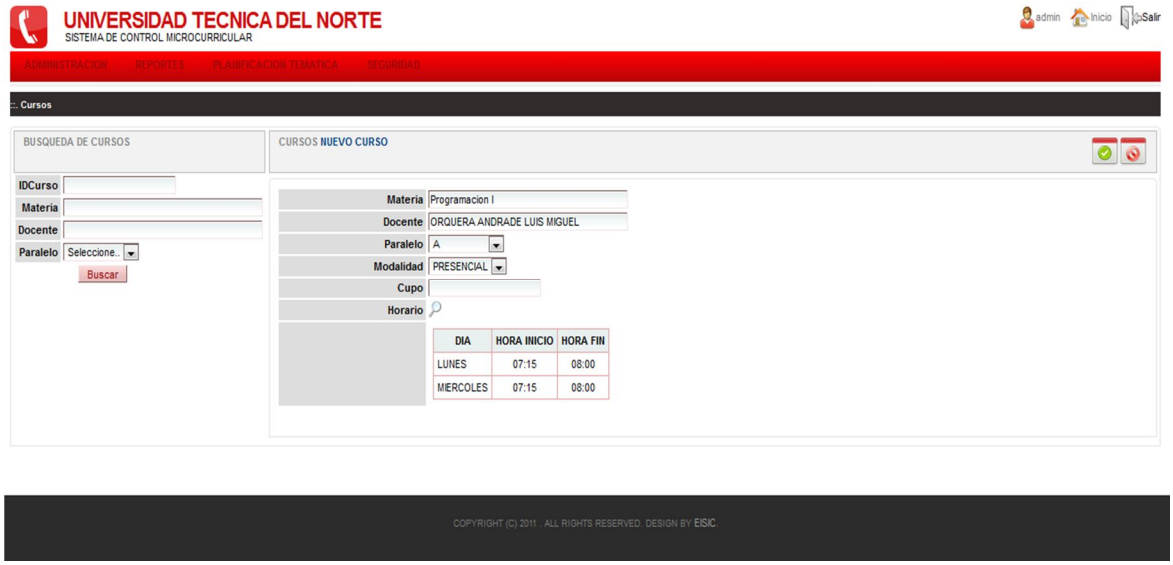
6.7.1.1. Autenticación Usuario

6.7.1.2. Menú Principal

6.7.1.3. Planificación Temática



6.7.1.4. Creación y Edición



CAPÍTULO VII

Conclusiones y Recomendaciones

7.1.Verificación de la hipótesis

7.1.1. Hipótesis

La implementación de Java Server Faces con frameworks que dan soporte AJAX, facilitan enormemente el desarrollo de interfaces ricas e interactivas para el desarrollo de aplicaciones RIA. Los componentes JSF que incluyen funcionalidades AJAX permiten ejecutar aplicaciones ricas en el cliente. JSF provee el control del lado del servidor, lo cual reduce la exagerada dependencia de JavaScript, que a menudo se encuentra en aplicaciones Ajax típicas.

7.1.2. Verificación

Se pueden establecer parámetros de comparación entre los principales framework que dan soporte AJAX, en base a los cuales se establece cual es el mejor para el desarrollo de aplicaciones en n capas.

- De la evaluación realizada entre las implementaciones de Java Server Faces 1.2 con soporte AJAX se determinó, a través del estudio comparativo, que los frameworks que ofrecen mejores características y funcionalidades, y que además se adaptan eficazmente a las exigencias de las aplicaciones web actuales como: RichFaces, IceFaces y Myfaces Trinidad.
- Todas las funcionalidades y características adicionales que cada framework JSF con soporte AJAX ofrece, pueden ser estudiadas y experimentadas en las aplicaciones de ejemplo provistas por los desarrolladores de los frameworks.
- Durante la investigación podemos decir que la implementación de JavaServer Faces con frameworks que dan soporte AJAX, si facilitan enormemente el desarrollo de interfaces ricas e interactivas para el desarrollo de aplicaciones RIA.

- La experiencia de usuario en la navegación es mucho más rica, ya no se refresca la página constantemente al interactuar con ella, el tiempo de espera para una petición se reduce, al realizar una nueva petición al servidor no se envía toda la página reduciendo el trafico al servidor.

7.2.Conclusiones

- Las aplicaciones RIA mejoran la experiencia visual del usuario, haciendo de la aplicación algo muy sencillo de uso, además de ofrecer mejoras en la conectividad, despliegue instantáneo de la aplicación, agilidad de acceso, permitiendo el uso de la aplicaciones web desde cualquier ordenador conectado a Internet y sobre cualquier sistema operativo..

- Java Server Faces es un Framework para aplicaciones Web basadas en tecnología JAVA usado generalmente para aplicaciones basadas en el modelo "Tres Capas (Modelo-Vista-Controlador)" del lado del servidor, que al complementarse con una librería de componentes ricos en Ajax esta tecnología permite realizar peticiones asincrónicas sin refrescar la página actual navegador web, para proveer al usuario una sensación de aplicación de escritorio.

- RichFaces es una framework potente y robusto a la hora de desarrollar aplicaciones RIA, permite añadir capacidad Ajax dentro de aplicaciones JSF y un avanzado framework para la integración de Ajax con facilidad en la capacidad de desarrollo de aplicaciones de negocio, el desarrollador no necesita escribir funciones asincrónicas para realizar peticiones al servidor. No hacen falta etiquetas especiales: se ponen los controles en la pantalla y RichFaces se encarga de enviar entre cliente y servidor sólo la información necesaria. Es decir, ya no se envían los formularios a la antigua usanza, en un POST de HTTP, sino que solo se envían los cambios que ha hecho el usuario del cliente al servidor, y los cambios en la pantalla del servidor al cliente. Eliminando recargas completas de pantallas en cada una de las peticiones realizadas al servidor.

- Con la evolución de las tecnologías de información en el desarrollo de aplicaciones para Internet, se debería tomar en cuenta que el uso de nuevos frameworks ayudan al desarrollo e implementación de las aplicaciones web dinámicas de manera fácil y rápida, aprovechando las características de los componentes para crear paginas

complejas que proporcionan mejoras en gran medida la experiencia del usuario más fiable y más rápidamente.

- La utilización de estos frameworks es sencilla mediante sus propias etiquetas que generan eventos y envían peticiones Ajax al servidor, esto significa que el desarrollador no tiene que preocuparse de crear el código JavaScript para enviar la petición al servidor ya que estos nuevos frameworks realizan este trabajo.

- La sistema de Control Microcurricular permite realizar el seguimiento de la ejecución del plan Microcurricular de las materias para lo cual el docente puede estructurar lo que va a enseñar en cada una de las unidades temáticas, seleccionar, secuenciar y jerarquizar la temática a tratar y su estructura lógica, adjuntar los recursos utilizados en cada uno de los temas de clase dictados.

7.3.Recomendaciones.

- Tras el desarrollo del Sistema de Control Microcurricular para la Carrera de Ingeniería en Sistemas con RichFaces, se pudo apreciar las excelentes funcionalidades que esta implementación ofrece, por lo que se recomienda utilizar este tipo de frameworks en desarrollo de aplicaciones JEE de una forma rápida y eficiente.

- La evolución de la especificación JSF, ha significado un gran esfuerzo de parte de cada una de las casas desarrolladoras de software que auspician los frameworks que implementan la especificación, por lo cual se recomienda visitar continuamente los sitios web de cada producto, para revisar las actualizaciones que se van liberando continuamente.

- Las implementaciones Ajax para JSF como RichFaces, IceFaces, MyFaces requieren de un conocimiento medio por lo menos de la especificación de JSF ya que tiene un alto nivel de abstracción que permite una mayor separación entre la presentación y la lógica de negocio.

- Los múltiples cambios originados por la revolución de las TIC ha evolucionado en el área de desarrollo de software, las competencias requeridas a los graduados de Carrera de Ingeniería en Sistema han cambiado y exigen el uso de nuevas tecnologías innovadoras en el área de desarrollo de sistema para internet. La carrera

debe atender esas nuevas demandas para que los estudiantes que pasan por ella estén mejor habilitadas para llevar una vida profesional, productiva y valiosa luego de graduarse.

- Se recomienda realizarlas siguientes investigaciones como posibles temas de tesis.

    - Desarrollo de Aplicaciones Web dirigida por modelos Model-Driven Development con soporte Ajax.
    - Jboss Seam, es un framework de aplicaciones empresariales JEE que realiza la integración de JSF y EJB3.

BIBLIOGRAFÍA

LIBROS

- **Ian Hlavats.** "JSF 1.2 Components", 1ra Edición (2009)
- **Jboss.org.** "RichFaces Developer Guide", (2008).
- **Demetrio Filocamo.** "JBoss RichFaces 3.3", 1ra Edición (2009)
- **Max Katz.** "Practical RichFaces", (2008), 1ra Edición (2009)
- **ICEsoft Technologies.** "Getting Started Guide" version 1.8, (2009).
- **ICEsoft Technologies.** "Developer's Guide" version 1.8, (2009).
- **Rainer Eschen.** "ICEfaces 1.8: Next Generation Enterprise Web Development", 1ra Edición (2009).
- **Rozanski Uwe.** "Enterprise Java Beans 3.0 con Eclipse y Jboss", 1ra Edición (2009).
- **David Thomas.** "Apache MyFaces Trinidad 1.2: A Practical Guide", (2009)
- **Zubin Wadia, Martin Marinschek, Hazem Saleh, and Dennis Byrne.** "The Definitive Guide to Apache MyFaces and Facelets", (2009)

INTERNET

- **Título:** "Rich Internet applications"
  Url: http://www.adobe.com/resources/business/rich_internet_apps/#open
  Fuente: adobe .com

- **Título:** "Rich Faces"
  Url: http://docs.jboss.org/richfaces/latest_3_3_X/en/devguide/html_single/
  Fuente: jboss.org

- **Título:** "Rich Faces Live Demo"
  Url: http://livedemo.exadel.com/richfaces-demo/index.jsp
  Fuente: jboss.org

- **Título:** "RichFaces"
  Url: http://en.wikipedia.org/wiki/RichFaces
  Fuente: Wikipedia

- **Título:** "Introducción a RichFaces"
  Url:
  http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=richFacesJsfIntro
  Fuente: adictosaltrabajo.com

- **Título:** "ICEfaces"
  Url: http://www.icefaces.org/main/resources/documentation.iface
  Fuente: icefaces.org

- **Título:** "ICEfaces Demo"
  Url: http://component-showcase.icefaces.org/component-showcase/showcase.iface
  Fuente: icefaces.org

- **Título:** "Trinidad"
  Url: http://myfaces.apache.org/trinidad
  Fuente: myfaces.apache.org

- **Título:** "MyFaces Core 1.2 Project"
  Url: http://myfaces.apache.org/core12/index.html
  Fuente: myfaces.apache.org

- **Título:** "AJAX JSF Matrix"
  Url: http://www.jsfmatrix.net/
  Fuente: jsfmatrix.net

- **Título:** "Análisis comparativo de los Frameworks Adobe Flex, Java RichFaces y Extjs"
  Url: http://bibdigital.epn.edu.ec/bitstream/15000/2110/1/CD-2887.pdf
  Fuente: bibdigital.epn.edu.ec

- **Título:** "Evaluación de tres implementaciones Java Server Faces 1.2"
  Url: http://www3.espe.edu.ec:8700/bitstream/21000/309/1/T-ESPE-027448.pdf
  Fuente: www3.espe.edu.ec

- **Título:** "AJAX JSF Frameworks Review"
  Url: http://www.theserverside.com/news/1363997/AJAX-JSF-Frameworks-Review
  Fuente: theserverside.com

- **Título:** "JAVA SERVER FACES"
  Url: http://www2.elo.utfsm.cl/~elo326/Presentaciones/Ronda4/JSF.pdf
  Fuente: Departamento de Electrónica de la Universidad Técnica Federico Santa María

- **Título:** "RichFaces e IceFaces"
  Url: http://www.slideshare.net/israelalcazar/introduccin-jsf-richfaces-e-icefaces-2267358
  Fuente: slideshare.net

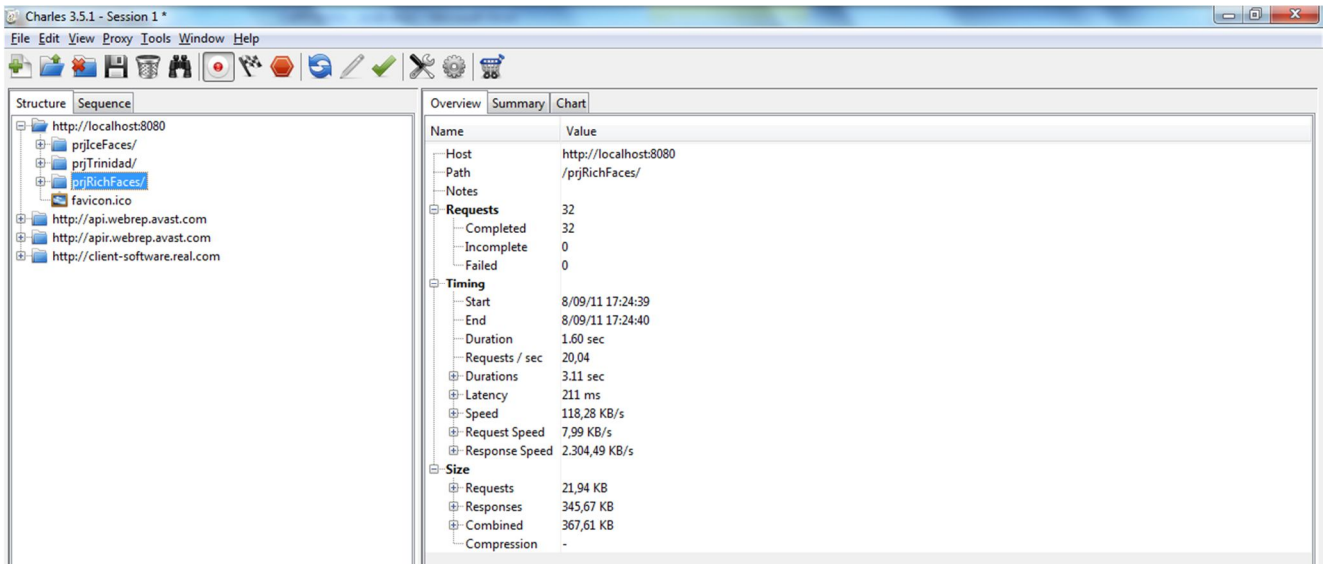
- **Título:** "Apache Trinidad"
  Url: http://www.slideshare.net/mwessendorf/apache-trinidad
  Fuente: slideshare.net

Anexos

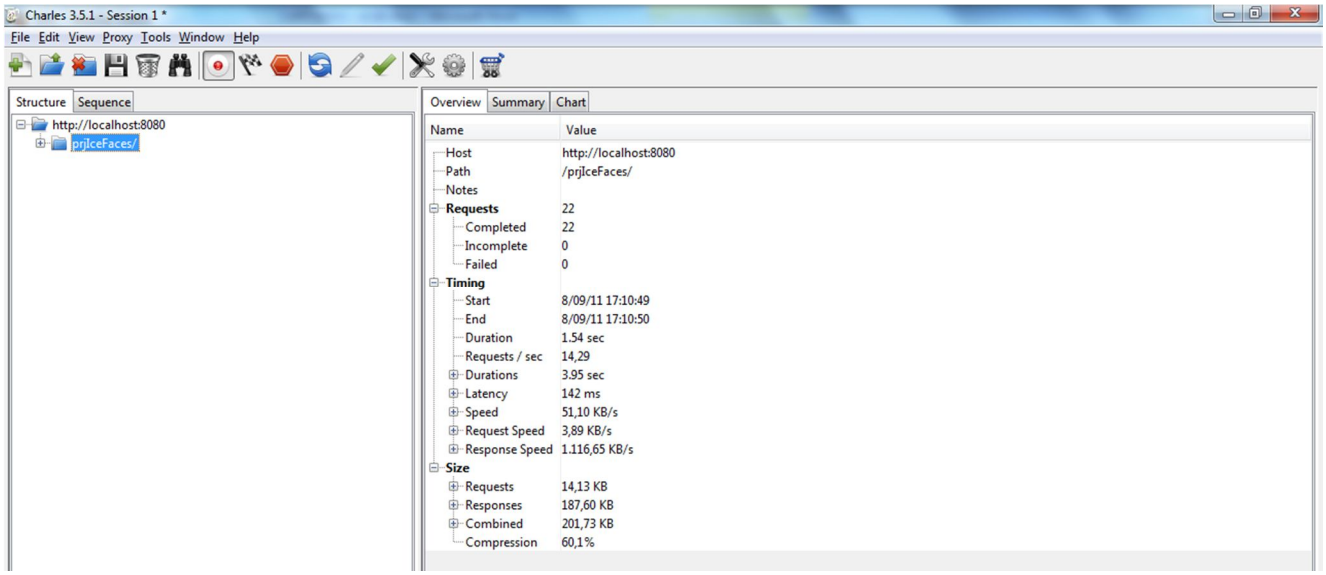
ANEXO 1

Evaluación de los Prototipos con Charles Proxy

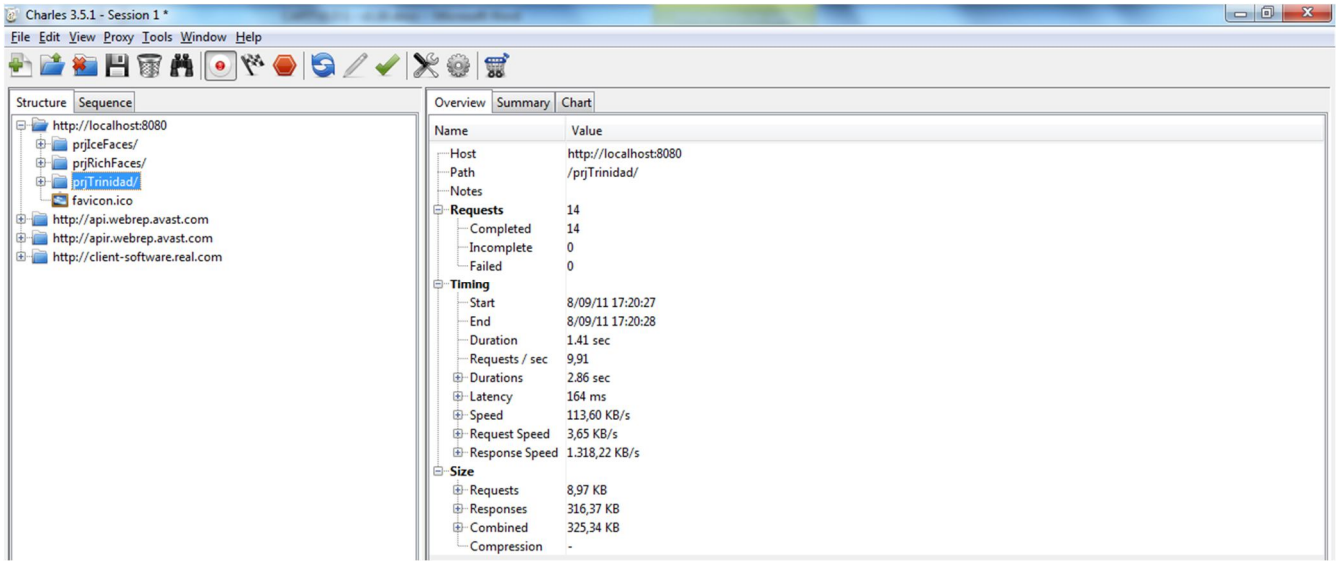
Trafico de red del prototipo de RichFaces



Trafico de red del prototipo de IceFaces



Trafico de red del prototipo de MyFaces



ANEXO 2

Manual de Instalación

La instalación de todos los paquetes necesarios se lo realizara en un Servidor Linux CentOS que se encuentra en los Laboratorios Informáticos de la Facultad de Ingeniería en Sistemas Computacionales.

Instalación de Java SE 6

JDK (Java Development Kit) es un conjunto de herramientas (programas y librerías) que permiten desarrollar (compilar, ejecutar, generar documentación, etc.) programas en lenguaje Java.

Descargar Java SE Development Kit (JDK) de la página http://www.oracle.com/technetwork/java/javase/downloads/index.html. En esta guía se está utilizando dk-6u24-linux-x64-rpm.bin

```
[root@centos ~]# chmod +x jdk-6u24-linux-x64-rpm.bin
[root@centos ~]# ./jdk-6u24-linux-x64-rpm.bin
```

Comprobar que la instalación del JDK está trabajando correctamente, esto debe resultar en un mensaje que muestra la versión de Java recientemente instalada.

```
[root@centos ~]# java -version
java version "1.6.0_24"
Java(TM) SE Runtime Environment (build 1.6.0_24-b07)
Java HotSpot(TM) 64-Bit Server VM (build 19.1-b02, mixed mode)
```

Exportar la variable JAVA_HOME:

```
export JAVA_HOME="/usr/java/jdk1.6.0_224"
```

Instalar PostgreSQL en CentOS

PostgreSQL es un sistema de base de datos objeto-relacional de gestión (ORDBMS) disponible para muchas plataformas, incluyendo Linux, FreeBSD, Solaris, de Windows y Mac OS X. Esta publicado bajo la licencia de PostgreSQL Licence. PostgreSQL es un avanzado sistema de bases de datos relacionales

OpenSource. PostgreSQL se caracteriza por ser un sistema estable, de alto rendimiento, gran flexibilidad ya que funcionar la mayoría de los sistemas Unix, además tiene características que permiten extender fácilmente el sistema. PostgreSQL puede ser integrada al ambiente Windows permitiendo de esta manera a los desarrolladores, generar nuevas aplicaciones o mantener las ya existentes.

Instalación con YUM

PostgreSQL puede ser instalado utilizando RPM (binary) o SRPM (source), gestionado por YUM. Este está disponible para las siguientes distribuciones de Linux (versiones de 32 - 64-bit): CentOS (versions 5 and up).

Configurar YUM repository

On CentOS: /etc/yum.repos.d/CentOS-Base.repo, [base] and [updates] sections

Descargar e instalar PGDG RPM file

PGDG es la versión para cada distribution/architecture/database de PostgreSQL. Buscar en http://www.pgrpms.org y encontrar el RPM correcto. Por ejemplo, para instalar PostgreSQL 9.0 en CentOS 5.5 32-bit:

wget http://www.pgrpms.org/9.0/redhat/rhel-5-i386/pgdg-centos-9.0-2.noarch.rpm

Instalación RPM:

rpm -ivh pgdg-centos-9.0-2.noarch.rpm

Install PostgreSQL

Lista de packages disponibles:

yum list postgres*

Instalar PostgreSQL 9.0 server:

yum install postgresql90-server

Comandos Post-installation

Después de instalar los packages, la base de datos tiene que ser inicializado y configurado.

Initialize

Comando para inicializar la base de datos en PGDATA:

service <*name*> initdb

Startup

Comando para que inicie automáticamente al reiniciar:

chkconfig <*name*> on

Control service

Comando para controlar el servicio de la base de datos:

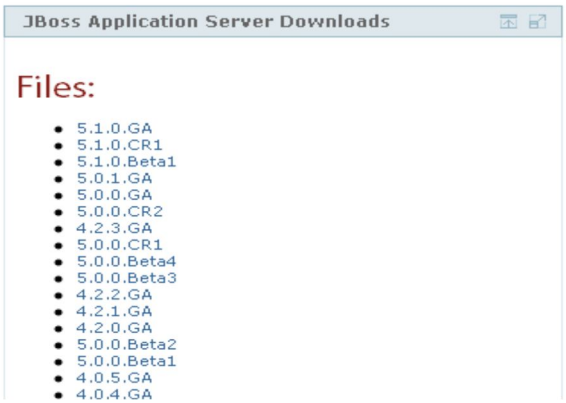
service <name> <command> cuando <command> puede ser:

- *start* : start the database
- *stop* : stop the database
- *restart* : stop/start the database; used to read changes to core configuration files
- *reload* : reload pg_hba.conf file while keeping database running

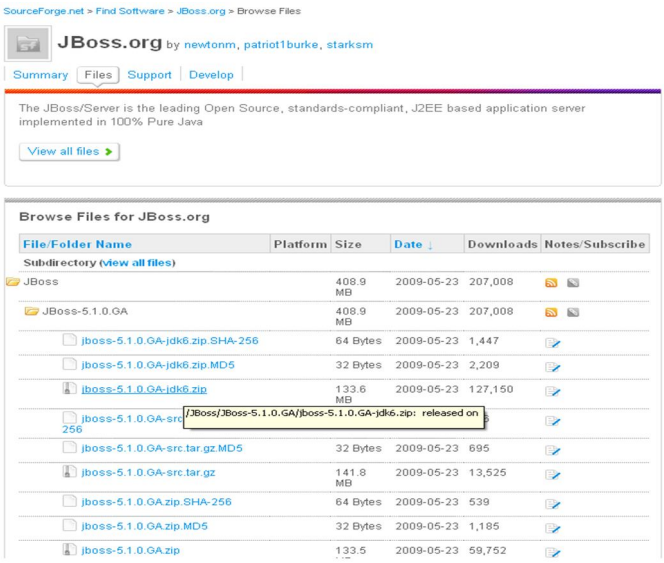
Instalacion de JBoss Application Server

JBoss es un servidor de aplicaciones JEE de código abierto implementado en Java. Es una plataforma para desarrollar y ejecutar aplicaciones empresariales java. Provee todas las funcionalidades de JEE5.

Descargar JBoss Application Server 5.1 de la página http://www.jboss.org/jbossas/ y seleccionar la última versión GA (General Availability) es una versión estable.



Descargar el zip y descomprirlo

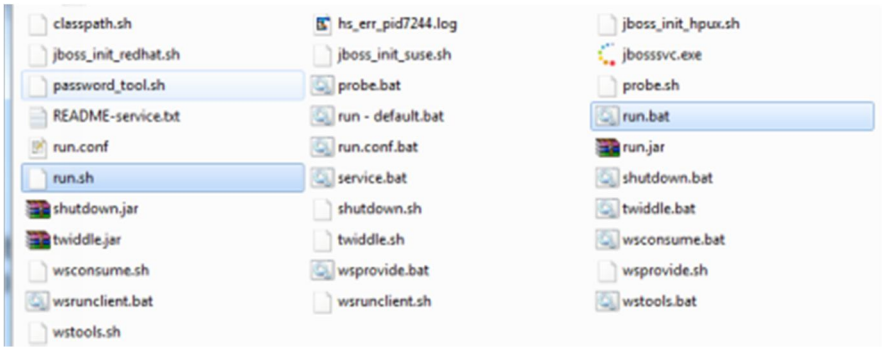


Arrancar el servidor

cd jboss-5.1.0.GA/bin ejecutar el archivo run

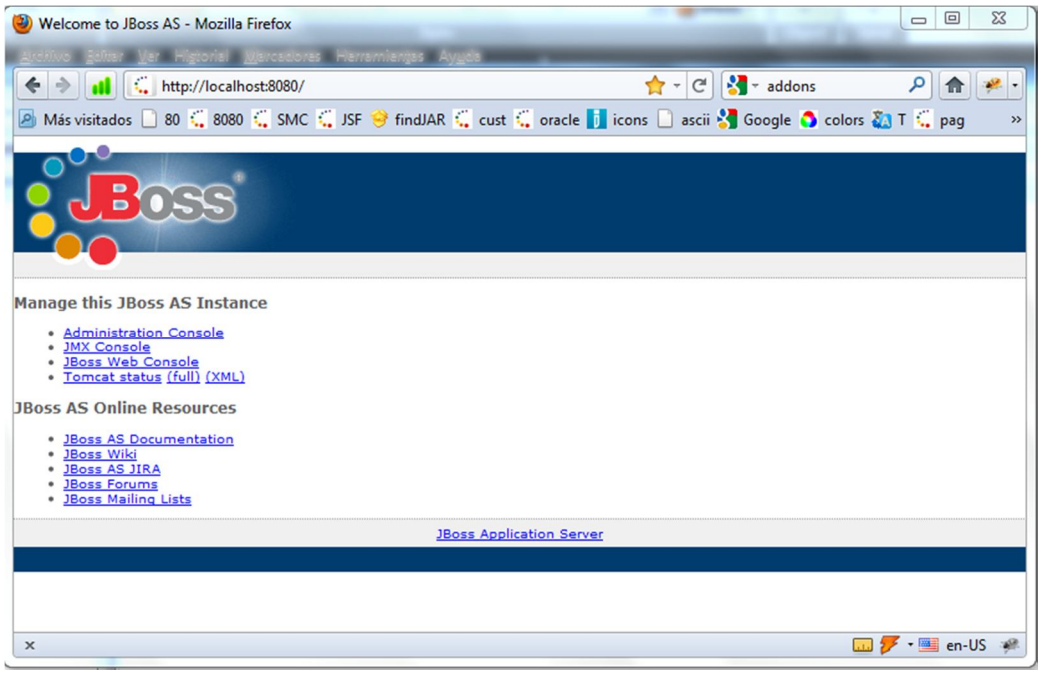
./run.bat

./run.sh



Por defecto JBoss solo escucha en localhost

./run.sh –b 0.0.0.0



Estructura de directorios

```
jboss/
    /bin
    /client
    /docs
    /lib
    /server
```

Distintos tipos de configuraciones Predefinidas

```
jboss/server/
    /all
    /default
    /minimal
    /production
```

Seleccionar la configuración que se desea arrancar:

./run.sh -c default

Directorios en cada configuración:

```
jboss/server/default/
    /conf
    /data
    /deploy
    /lib
    /log
    /tmp
    /work
```

Desplegar aplicaciones

prjSMCEAR.ear:

El archivo prjSMCEAR.ear (Enterprise Archive) contiene todos los recursos de la aplicación: prjSMCModel, prjSMCWeb.

prjSMCModel.jar:

El archivo prjSMCWeb.jar (Java Archive) contiene todos los recursos de la capa de lógica de negocio.

prjSMCWeb.war:

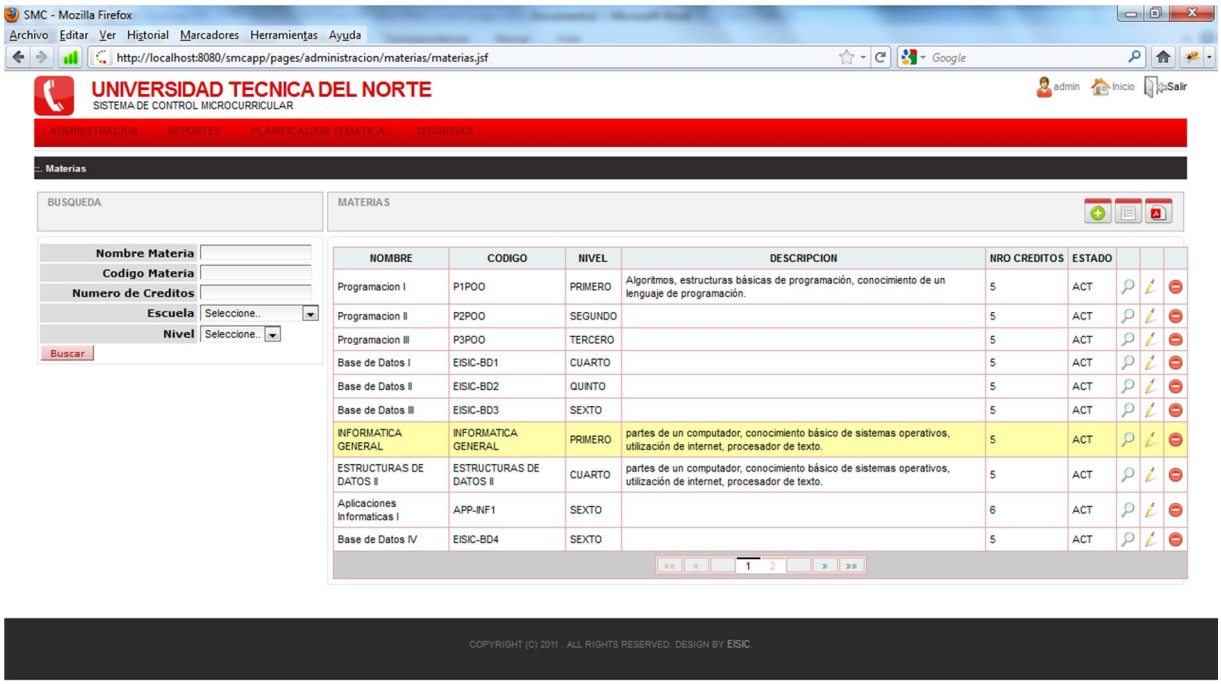
El archivo prjSMCWeb.war (Web Application Archive) contiene todos los recursos de la aplicación web.

smc-ds.xml

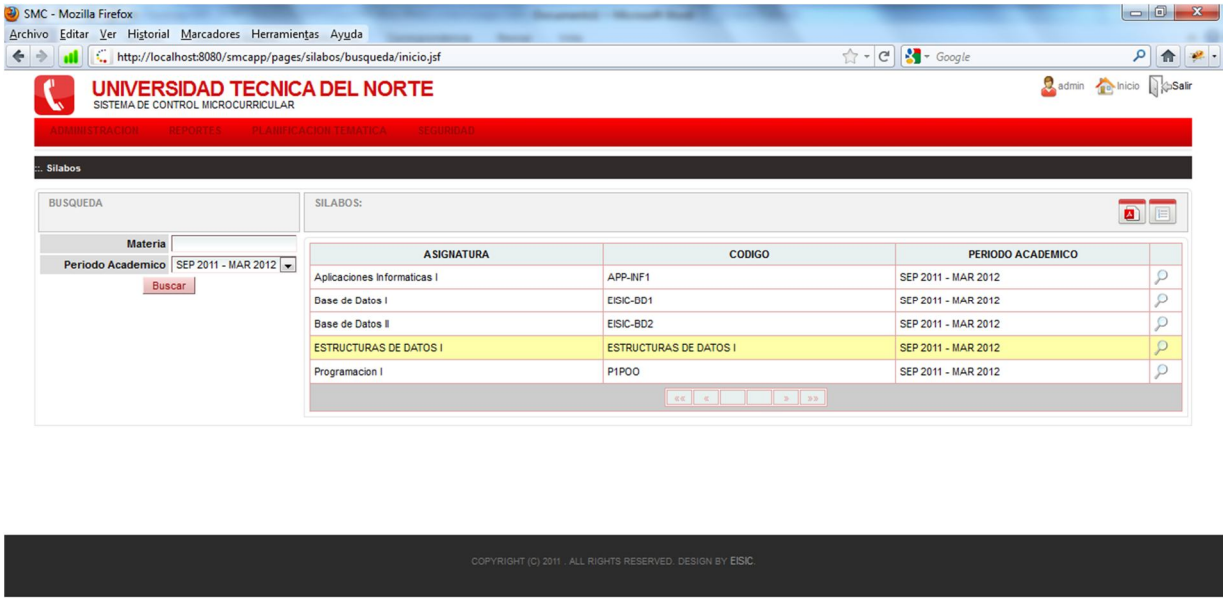
El archivo smc-ds.xml es un DataSource parte de la especificación JDBC y puede verse como una fábrica de conexiones para poder trabajar con datos de una base de datos. Permite al contenedor ocultar el poll de conexiones y el manejo de transacciones del código de aplicación.Pantallas Principales del Sistema Microcurricular

Pantalla de administración de Materias

Pantalla de Planificación de Unidades Temáticas



Pantalla de administración de Usuarios



ANEXO 3

En el siguiente link encontraremos los manuales de usuario del Sistema

MANUAL DE USUARIO

MANUAL TÉCNICO