Estudio comparativo de alternativas a Primefaces para integración con JSF en la construcción de un prototipo de vortal en el CIADTI

Autor:

Juan Guillermo Díaz Martínez

Director:

Luis Alberto Esteban Villamizar

Magister en Informática

Universidad de Pamplona

Facultad de Ingenierías y Arquitectura

Departamento de Ingenierías Electrónica, Eléctrica, Sistemas y Telecomunicaciones

Programa de Ingeniería de Sistemas

Pamplona, Norte de Santander – Colombia

2017

Resumen

En este documento se presenta un estudio comparativo de Frameworks o bibliotecas de componentes para JSF en el desarrollo de aplicaciones web la cual en este caso se aplicó a un prototipo de administrador vortal en el CIADTI. Se eligieron los tres mejores frameworks entre la gran variedad que existe en la web con los que se implementaron dos funcionalidades en el prototipo y luego se aplicaron cinco criterios con los que fueron evaluados. Los criterios fueron mantenibilidad y documentación, índices de interés, disponibilidad de componentes, usabilidad y funcionalidad. Finalmente de acuerdo a la puntuación total se seleccionó Primefaces como el mejor Framework, siendo el más completo para crear las interfaces de usuarios de las aplicaciones creadas con JavaServer Faces.

Tabla de Contenidos

Índice de Figuras.

Índice de Tablas.

Lista de Siglas.

- JSF - JavaServer Faces.
- MVC - Modelo Vista Controlador.
- API – Application Programming Interface o interfaz de programación de aplicaciones.
- ORM - Object Relational Mapping o mapeo de objeto relacional.
- HQL - Hibernate Query Languaje o Lenguaje de Consultas de Hibernate.
- POJO - Plain Old Java Objects u objeto java plano antiguo.
- EL - Expression Language o Lenguaje de expresiones.
- ACE - Componentes Avanzados de ICEfaces.
- VO - Value Object.
- RIA - Rich Internet Applications o Aplicaciones de Internet Ricas.
- CRUD - Create, Read, Update y Delete o crear, leer, actualizar y eliminar.
- DOM - Document Object Model.
- XML - eXtensible Markup Language o Lenguaje de Marcas Extensible.
- AJAX - Asynchronous JavaScript And XML o JavaScript asíncrono y XML.
- JSTL - JavaServer Pages Standard Tag Library o Librerías de etiquetas estándar de JSP.
- GNU, GPL - Licencia Pública General de GNU.

1 Introducción

Al iniciar un proyecto en el cual se desea desarrollar una aplicación web, se deben tener presente factores que van a determinar el crecimiento del mismo, como por ejemplo su estructura, aquí es donde entra en juego la importancia de usar un Framework, ya que estos proporcionan una estructura base que permite agilizar y optimizar estos procesos, reduciendo el tiempo y esfuerzo en el desarrollo de la aplicación. El desarrollo con Framework es menos propenso a sufrir errores, por ello es que actualmente la utilización de estas tecnologías es casi del 100%.

Las responsabilidades principales de un desarrollador web se dividen principalmente en dos: los desarrolladores *Front-End* que se especializan en sitios web y específicamente en su interfaz gráfica con las que el usuario final va interactuar, es por esto que sabe que es primordial la utilización que el usuario haga de ésta. Por otro lado, los *Back-End* se especializan en lo que sucede en el servidor y en el procesamiento de los datos que se envían al *Front-End*. Los desarrolladores *Front-End* son los que definen la estética de la aplicación, encargados de que los datos se muestren de manera atractiva para el usuario y de que la interacción que se realice sea muy llamativa.

El presente trabajo se encuentra organizado en cinco capítulos, donde el primero es introductorio a todo el trabajo en el cual se define desde el planteamiento del problema hasta la metodología de trabajo con la que se trabajó, el segundo capítulo se referenció toda la parte teórica necesaria para su normal desarrollo. Como parte de los resultados, en el tercer capítulo se realizó el comparativo del cual resultó una tecnología framework a recomendar, finalmente en el

capítulo cuarto se describieron las conclusiones de este estudio y en el quinto se relacionaron todas las referencias bibliográficas de las que se tuvo apoyo para el desarrollo del documento.

Para comenzar en este primer capítulo se va plantear el contenido del proyecto de tal forma que se desarrollará todo lo relacionado con la información correspondiente al problema, justificación y objetivos, brindando la explicación al lector del por qué la necesidad de desarrollar este proyecto de grado y el por qué, cómo, cuándo y dónde se llevará cabo el mismo.

1.1 Planteamiento del Problema.

Con el surgimiento de la Web (WWW) la cual ha tenido un crecimiento considerable en la actualidad han surgido gigantescos avances tecnológicos de desarrollo web por lo que la opción de adecuarse a las nuevas tecnologías es ya una necesidad. Compañías pequeñas, empresas grandes, instituciones, estados, gobiernos de distintos países, universidades, bibliotecas, están presentes hoy en día en Internet y debido al constante crecimiento en el ambiente de las tecnologías orientadas hacia la evolución del desarrollo web, se observa que la creación de sistemas de información es fundamental para casi cualquier organización. El problema que se maneja desarrollando de manera tradicional es disponer de un sistema con interfaces de usuario muy llamativas ya que estas son más que solo datos, se debe considerar también herramientas de desarrollo, código fuente, librerías, archivos de configuración, etc. Todo esto depende del equipo de trabajo o individuo que esté desarrollando el sistema de información, lo cual hace más complicado el proceso.

Para solventar la problemática se requiere la utilización de Frameworks en JavaServer Faces JSF debido a que se encargan de agilizar muchos procesos y hacer más eficiente el trabajo

del desarrollador. Estas herramientas de interfaces visuales generan grandes resultados en la interacción con el usuario, sin embargo existe una gran variedad de estas tecnologías en el desarrollo web que se ha convertido en un factor de riesgo o indecisión para los desarrolladores en la medida de que ocurra una incorrecta selección. Lo anterior implica probablemente el fracaso del proyecto o en el mejor de los casos puede ser un obstáculo para el futuro mantenimiento de la aplicación, por lo que el proceso de selección del Framework correcto para el tipo de sistema a desarrollar se convierte en un factor decisivo. Por tal motivo nace la necesidad de llevar a cabo este proyecto de estudio para determinar el Framework que mejor se adapte al desarrollo de una nueva interfaz del administrador Vortal en el CIADTI para que resulte más atractiva y de alguna forma más ágil en su uso como es el ejemplo del número de clics al trabajar en una página específica. De esta forma se pretende obtener interfaces de usuarios más óptimas teniendo en cuenta su funcionalidad, su fácil mantenimiento y amigabilidad con el usuario final.

1.2 Justificación.

La Universidad de Pamplona ofrece un software que permite integrar los procesos mediante una herramienta en entorno Web que desempeña la gestión, por medio del Campus Colaborativo, Campus Académico, Campus Administrativo, Campus Servicio, Campus Virtual y SNIES de la Institución con información completamente integrada y sistematizada en tiempo real.

El CIADTI ha desarrollado la infraestructura del software de la Universidad desde hace aproximadamente más de 15 años lo que ha permitido acumular una gran cantidad de código bajo la misma arquitectura, que ha funcionado adecuadamente, pero que por el ritmo de crecimiento en las tecnologías de desarrollo podría en un futuro llegar a ser difícil de mantener por lo que se debe ir adaptando a las nuevas tecnologías del desarrollo web.

En la actualidad se están realizando procesos de rediseño de interfaces de usuario para algunas de sus aplicaciones, lo que va dejando en evidencia que se están adaptando al uso de nuevas herramientas y es una oportunidad para incorporar un Framework de este lado (*Front-End*), adecuado y de fácil integración con lo que ya se cuenta actualmente. Es por esto que un estudio y análisis de Frameworks para el desarrollo de interfaces que van de la mano con el buen funcionamiento es el primer paso para una correcta selección y recomendación de la tecnología que resulte más completa en cuanto a la facilidad de integración a los desarrollos con los que se dispone actualmente.

1.3 Objetivos.

1.3.1 Objetivo General.

Realizar un comparativo de tecnologías Open Source alternativas a Primefaces que permita recomendar una de ellas para la construcción de un prototipo Vortal en el CIADTI.

1.3.2 Objetivos Específicos.

1. Hacer una revisión bibliográfica de Primefaces y otras tecnologías alternativas en el desarrollo Web.

2. Seleccionar 3 (Tres) de dichas tecnologías para realizar un comparativo de acuerdo a 5 (Cinco) criterios de evaluación según los enfoques del CIADTI.
3. Realizar mínimo 2 (Dos) funcionalidades para cada una de las tecnologías seleccionadas.
4. Identificar la mejor tecnología a partir del análisis previamente realizado.
5. Analizar los resultados obtenidos en el desarrollo del prototipo.

1.4 Materiales.

Para el desarrollo del proyecto se utilizaron diferentes materiales tecnológicos tangibles e intangibles como libros, software, equipos de cómputo con los cuales se pudo realizar el comparativo entre las tecnologías seleccionadas poniéndolas a prueba con las funcionalidades de un prototipo de software en el CIADTI para finalmente seleccionar la de mejores características. Los materiales utilizados en el desarrollo del proyecto se describen a continuación en las siguientes tablas.

Tabla 1: Materiales de Software

#	Nombre	Descripción
1	Administrador Vortal	Utilizado para analizar el funcionamiento actual e identificar posibles mejoras.
2	NetBeans 8.2	Utilizado para el desarrollo del prototipo.
3	JavaServer Faces	Framework utilizado para el desarrollo del prototipo.

4	Postgres 9.6	Utilizado como motor de Base de Datos para el prototipo.
5	pgAdmin 3	Utilizado como gestor de Base de Datos del prototipo.

Tabla 2: Materiales Hardware.

#	Nombre	Descripción
1	Equipo de Cómputo Portátil.	Utilizado para investigar, desarrollar el documento escrito y el prototipo.

Tabla 3: Materiales Talento Humano.

#	Nombre	Descripción
1	Personal de Desarrollo Tecnológico	Recopilación de información con respecto a funcionamiento de Base de Datos y aplicativo actual.

1.5 Modalidad de trabajo.

El presente trabajo es de modalidad pasantía cuyo fin principal es la investigación para la selección de las tres tecnologías más apropiadas para la transformación de interfaces de usuario del administrador Vortal en el CIADTI y la construcción de un prototipo del mismo, previamente realizado un análisis comparativo entre las tecnologías Frameworks web Open Source más acogidos de la actualidad en JSF en el lenguaje Java.

En el proyecto se definió un diagnóstico del estado actual del aplicativo administrador Vortal en el CIADTI, luego se identificaron los Frameworks más apropiados (tres) donde se desarrollaron las funcionalidades para que finalmente se eligiera el que cuenta con las mejores características de acuerdo a su evaluación, donde luego se pasó a la implementación definitiva del aplicativo.

El proyecto va dirigido principalmente al CIADTI de la Universidad de Pamplona sin embargo se debe tener en cuenta que el aplicativo administrador Vortal es utilizado por todos los estudiantes, administrativos y docentes quienes también son directamente beneficiados.

Las fuentes de investigación fueron virtuales, en las cuales se utilizaron archivos PDF y sitios oficiales de los Frameworks Open Source, todas estas fuentes confiables y oportunas para el proceso de investigación, además se contó con la información proporcionada por el grupo de desarrollo tecnológico en el CIADTI.

El principal instrumento para recolección de información además de los ya mencionados anteriormente, fue el seguimiento al funcionamiento del actual aplicativo administrador Vortal y algunos procesos que se llevan a cabo en el área de desarrollo tecnológico del CIADTI teniendo en cuenta la interacción con todos los usuarios bien sean estudiantes o administrativos, por lo que fue necesario para su implementación. Así se pudo ir documentando el proceso y adecuando con mejoras incluidas. Toda la información correspondiente al proceso de selección de tecnologías web para el desarrollo de interfaces de usuario Open Source más apropiado para la transformación del administrador Vortal en el CIADTI fue consultada e investigada en sus sitios oficiales. En este proyecto fue necesario el acceso a información de la base de datos de prueba

del CIADTI con la mayor responsabilidad requerida, ya que esta información es de su propiedad y no puede ser tomada para otros fines.

2 Marco Teórico.

Este capítulo contiene toda la información correspondiente a la parte teórica relacionada con los temas de interés desarrollados en este proyecto con el fin de establecer claridad a los lectores, además, muestra en forma cronológica investigaciones y estudios que se han realizado en el mundo tecnológico web con relación a Frameworks como Primefaces para la construcción de interfaces de usuario Open Source.

2.1 Ingeniería del Software.

La ingeniería del software es una disciplina de ingeniería preocupada por todos los aspectos de la producción de software desde las primeras etapas de especificación del sistema hasta el mantenimiento del sistema después de que éste se ha puesto en uso. Se preocupa de las teorías, métodos y herramientas para el desarrollo profesional de software. La ingeniería del software se preocupa del desarrollo de software rentable.

La ingeniería del software es la aplicación de un enfoque sistemático, disciplinado y cuantificable para el desarrollo, operación y mantenimiento del software (IEEE, 1990). El enfoque sistemático, disciplinado y cuantificable es con frecuencia calificado de modelo de proceso de software (en el sentido general) o de proceso de desarrollo de software (en el sentido específico). El proceso de desarrollo de software específico consiste en un conjunto particular de prácticas de desarrollo de software que son realizadas por el ingeniero de software en un orden predeterminado.

Los ingenieros adoptan un enfoque sistemático y organizacional para su trabajo. Cuando se habla de prácticas de desarrollo de software se hace referencia a un requisito empleado para

recomendar un enfoque disciplinado y uniforme del proceso de desarrollo de software, es decir, una actividad bien definida que contribuye a la satisfacción de los objetivos del proyecto; generalmente la salida de una práctica se convierte en la entrada de otra. Entre las prácticas de desarrollo de software se encuentran las siguientes (las prácticas pueden depender en base al proceso y la terminología asociada al mismo) (INTECO, 2009):

- Ingeniería de requisitos.
- Análisis de sistemas.
- Diseño/arquitectura a alto nivel.
- Diseño a bajo nivel.
- Codificación.
- Integración.
- Diseño y revisiones de código.
- Pruebas.
- Mantenimiento.
- Gestión de proyectos.
- Gestión de la configuración.

La mayoría de las disciplinas reconocen algunas prácticas como mejores prácticas. Una mejor práctica es una práctica que, a través de la experiencia e investigación, se ha probado que

lleva al resultado deseado fiablemente y se considera prudente y recomendable hacerla en una variedad de contextos.(INTECO, 2009)

2.2 Software Libre.

El «software libre» es una cuestión de libertad, no de precio. Para comprender este concepto, se debe pensar en la acepción de libre como en «libertad de expresión» y no como en «barra libre de cerveza». Con software libre se hace referencia a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. Se refiere a cuatro clases de libertad para los usuarios de software (Stallman, 2004):

- ***Libertad 0***: la libertad para ejecutar el programa sea cual sea su propósito.
- ***Libertad 1***: la libertad para estudiar el funcionamiento del programa y adaptarlo a sus necesidades —el acceso al código fuente es condición indispensable para esto.
- ***Libertad 2:*** la libertad para redistribuir copias y ayudar así al vecino.
- ***Libertad 3:*** la libertad para mejorar el programa y luego publicarlo para el bien de toda la comunidad —el acceso al código fuente es condición indispensable para esto.

Software libre es cualquier programa cuyos usuarios gocen de estas libertades. De modo que se debería ser libre de redistribuir copias con o sin modificaciones, de forma gratuita o cobrando por su distribución, a cualquiera y en cualquier lugar. Gozar de esta libertad significa, entre otras cosas, no tener que pedir permiso ni pagar para ello.

Asimismo, debería ser libre para introducir modificaciones y utilizarlas de forma privada, ya sea en el trabajo o en el tiempo libre, sin siquiera tener que mencionar su existencia. Si se

decide publicar estos cambios, no se debería estar obligado a notificarle a ninguna persona ni de ninguna forma en particular.

La libertad para utilizar un programa significa que cualquier individuo u organización podrá ejecutarlo desde cualquier sistema informático, con cualquier fin y sin la obligación de comunicárselo subsiguientemente ni al desarrollador ni a ninguna entidad en concreto.

La libertad para redistribuir copias supone incluir las formas binarias o ejecutables del programa y el código fuente tanto de las versiones modificadas como de las originales. La distribución de programas en formato ejecutable es necesaria para su adecuada instalación en sistemas operativos libres. No pasa nada si no se puede producir una forma ejecutable o binaria —dado que no todos los lenguajes pueden soportarlo—, pero todos deben tener la libertad para redistribuir tales formas si se encuentra el modo de hacerlo.

Para que las libertades 2 y 4 —la libertad para hacer cambios y para publicar las versiones mejoradas— adquieran significado, se debe disponer del código fuente del programa. Por consiguiente, la accesibilidad del código fuente es una condición necesaria para el software libre.

Para materializar estas libertades, deberán ser irrevocables siempre que no se comente ningún error; si el desarrollador del software pudiera revocar la licencia sin motivo, ese software dejaría de ser libre.

Sin embargo, ciertas normas sobre la distribución de software libre pueden parecer aceptables siempre que no planteen un conflicto con las libertades centrales.

De modo que se puede pagar o no por obtener copias de software libre, pero independientemente de la manera en que se obtenga, siempre se tendrá libertad para copiar, modificar e incluso vender estas copias.

El software libre no significa que sea «no comercial». Cualquier programa libre estará disponible para su uso, desarrollo y distribución comercial. El desarrollo comercial del software libre ha dejado de ser excepcional y de hecho ese software libre comercial es muy importante. (Stallman, 2004)

2.3 Open Source o Código Abierto.

Código abierto es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código. (GPSOS, 2017)

El código abierto no significa simplemente acceso al código fuente. Los términos de distribución del software de fuente abierta deben cumplir con los siguientes criterios (OpenSource Initiative, 2007):

- ***Libre redistribución***. La licencia no debe restringir a nadie vender o entregar el software como un componente de una distribución de software que contenga programas de distintas fuentes. La licencia no debe requerir ningún tipo de cuota por su venta.
- ***Código fuente***. El programa debe incluir el código fuente, y se debe permitir su distribución tanto como código fuente como compilado. Cuando de algún modo no se distribuya el código fuente junto con el producto, deberá proveerse un medio conocido para obtener el código fuente sin cargo, a través de Internet. El código fuente es la forma

preferida en la cual un programador modificará el programa. No se permite el código fuente deliberadamente confundido. Tampoco se permiten formatos intermedios, como la salida de un preprocesador, o de un traductor.

- ***Trabajos derivados***. La licencia debe permitir modificaciones y trabajos derivados, y debe permitir que estos se distribuyan bajo las mismas condiciones de la licencia del software original.
- ***Integridad del código fuente del autor***. La licencia puede restringir la distribución de código fuente modificado sólo si se permite la distribución de "patch files" con el código fuente con el propósito de modificar el programa en tiempo de construcción. La licencia debe permitir explícitamente la distribución de software construido en base a código fuente modificado. La licencia puede requerir que los trabajos derivados lleven un nombre o número de versión distintos a los del software original.
- ***No discriminar personas o grupos.*** La licencia no debe hacer discriminación de personas o grupos de personas.
- ***No discriminar campos de aplicación.*** La licencia no debe restringir el uso del programa en un campo específico de aplicación. Por ejemplo, no puede restringir su uso en negocios, o en investigación genética.
- ***Distribución de la licencia.*** Los derechos concedidos deben ser aplicados a todas las personas a quienes se redistribuya el programa, sin necesidad de obtener una licencia adicional.
- ***La licencia no debe ser específica a un producto.*** Los derechos aplicados a un programa no deben depender de la distribución particular de software de la que forma parte. Si el

programa es extraído de esa distribución y usado o distribuido dentro de las condiciones de la licencia del programa, todas las personas a las que el programa se redistribuya deben tener los mismos derechos que los concedidos en conjunción con la distribución original de software.

- ***La licencia no debe contaminar otro software.*** La licencia no debe imponer restricciones sobre otro software que es distribuido junto con él. Por ejemplo, la licencia no debe insistir en que todos los demás programas distribuidos en el mismo medio deben ser software open-source.
- ***La licencia debe ser tecnología neutral.*** Ninguna disposición de la licencia puede basarse en cualquier tecnología individual o estilo de interfaz.

Algunos ejemplos de este tipo de licencia son: GNU GPL, X Consortium, Artistic, MPL, entre otras. (OpenSource Initiative, 2007)

2.4 Modelo Vista Controlador (MVC).

El patrón Modelo-Vista-Controlador (MVC) surge con el objetivo de reducir el esfuerzo de programación, necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos, a partir de estandarizar el diseño de las aplicaciones. El patrón MVC es un paradigma que divide las partes que conforman una aplicación en el Modelo, las Vistas y los Controladores, permitiendo la implementación por separado de cada elemento, garantizando así la actualización y mantenimiento del software de forma sencilla y en un reducido espacio de tiempo. (Fernández & Yanette, 2012)

MVC es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

Se trata de un modelo muy maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones, y sobre multitud de lenguajes y plataformas de desarrollo.

• **El Modelo,** contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.

• **La Vista**, o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos de interacción con éste.

• **El Controlador**, que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.(U. Alicante, 2016)

2.5 Frameworks.

Un Framework es un término muy utilizado en la computación en general para referirse a un conjunto e bibliotecas, utilizadas para implementar la estructura estándar de una aplicación. Todo esto se realiza con el propósito de promover la reutilización de código, con el fin de ahorrarle trabajo al desarrollador al no tener que reescribir ese código para una nueva aplicación que desee crear. Existen diferentes Frameworks para diferentes propósitos, algunos orientados al desarrollo de aplicaciones web, otros para desarrollar aplicaciones multiplataforma, para un sistema operativo o lenguaje de programación en específico, entre otros.

Según Gamma, el Framework determina la arquitectura de una aplicación (Gamma, 1995). Este es un buen enfoque, ya que el Framework se encarga de definir la estructura general, sus particiones en clases y objetos, las responsabilidades clave, así como la colaboración entre dichas clases y objetos. Todos estos parámetros son definidos por el Framework, evitando que el usuario tenga que definirlos y se pueda enfocar en cosas específicas de su aplicación.

Un Framework ayuda a que se desarrolle una aplicación de una manera más rápida, ya que no se pierde tiempo en algunos detalles de diseño que muchas veces quitan más tiempo del que tomo construir en sí la lógica de la aplicación. (Anónimo, 2005)

2.5.1 Tipos de Framework Web.

Existen varios tipos de Frameworks Web, orientados a la interfaz de usuario, como Java Server Faces, orientados a la generación del modelo como Hibernate e Ibatis y otros que facilitan la generación del CRUD (métodos Create, Read, Update y Delete o crear, leer, actualizar y eliminar) de una aplicación.

La mayoría de Frameworks Web se encargan de ofrecer una capa de controladores de acuerdo con el patrón MVC, ofreciendo mecanismos para facilitar la integración con otras herramientas para la implementación de las capas de negocio y presentación. (Gutiérrez, 2007)

2.5.2 Características de los Frameworks Web.

En la **tabla 4** se pueden evidenciar algunas características que se pueden encontrar en casi todos los Frameworks.

Tabla 4: Características de frameworks

Características	Descripción
Abstracción de URLs y sesiones.	No es necesario manipular directamente las URLs ni las sesiones, el Framework ya se encarga de hacerlo.
Acceso a datos.	Incluyen las herramientas e interfaces necesarias para integrarse con herramientas de acceso a datos, en BBDD, XML, etc…
Controladores.	La mayoría de Frameworks implementa una serie de controladores para gestionar eventos, como una introducción de datos mediante un formulario o el acceso a una página. Estos controladores suelen ser fácilmente adaptables a las necesidades de un proyecto concreto.
Autentificación y control de acceso.	Incluyen mecanismos para la identificación de usuarios mediante login y password y permiten restringir el acceso a determinas páginas a determinados usuarios.
Internacionalización.	
Separación entre diseño y contenido.	

(Gutiérrez, 2007)

2.5.3 Framework MVC para la construcción del prototipo.

A continuación se va a mencionar de forma detallada el framework JSF sobre el cual se fundamentó este trabajo.

2.4.3.1 JSF.

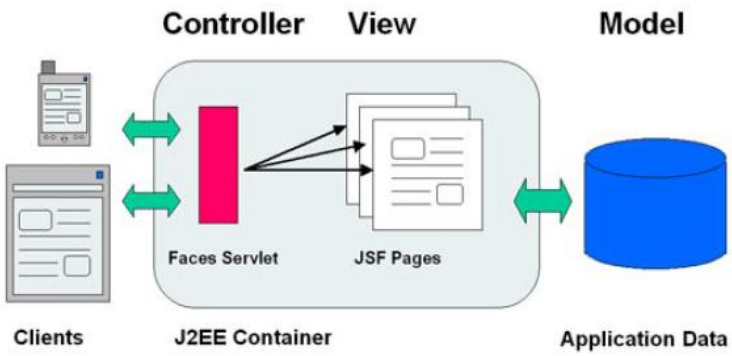
Figura 1: Logo de JSF

Fuente: <http://bit.ly/2hMum8j>

JavaServer Faces (JSF) es un Framework basado en el patrón MVC (Modelo Vista Controlador) para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE.

JSF pretende normalizar y estandarizar el desarrollo de aplicaciones web. Ofrece una clara separación entre el comportamiento y la presentación, lo que permite a cada miembro del equipo de desarrollo de una aplicación Web enfocarse en su parte del proceso de desarrollo, y proporciona un sencillo modelo de programación para enlazar todas las piezas.

Figura 2: Modelo de programación MVC



Fuente: (Teruelo, Mancha, Gallego, & Fernández, 2008)

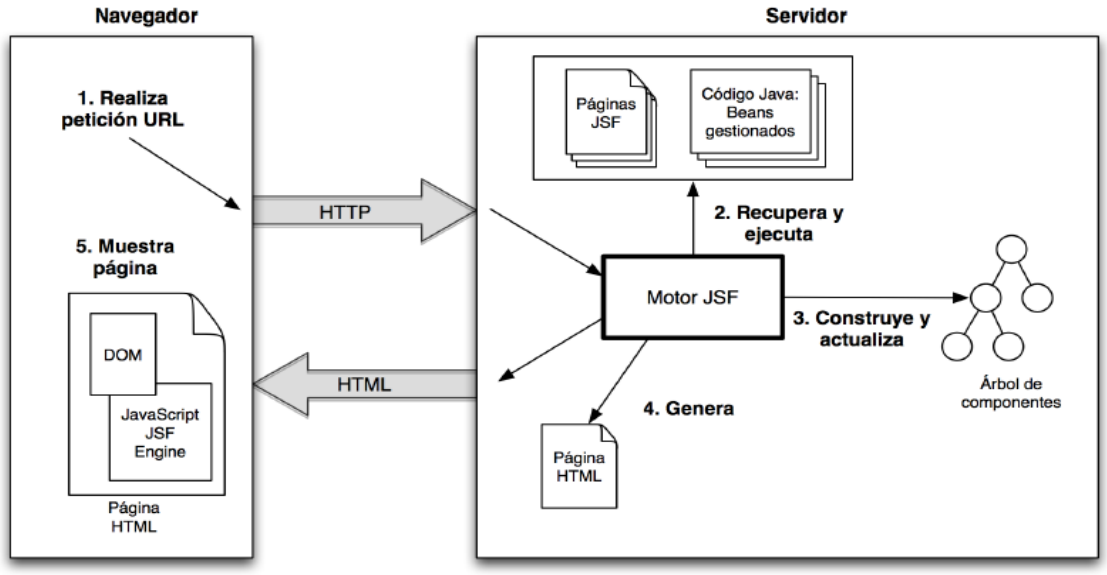
La tecnología JavaServer Faces surge como una solución a la separación entre la presentación y el comportamiento en una aplicación Web, de forma tal que las actividades de los autores de las páginas puedan separarse de las actividades de los desarrolladores de la lógica. Permite programar más rápidamente, ahorra trabajo, da consistencia al código, permite separar presentación de negocio y mejora la seguridad de tu aplicación. (Teruelo, Mancha, Gallego, & Fernández, 2008)

2.5.3.1.1 Frameworks RIA basados en el servidor.

Frente a las aplicaciones web tradicionales basadas en la navegación entre distintas páginas, en los últimos años se han popularizado las aplicaciones web de una única página que intentan simular las aplicaciones de escritorio. Son las denominadas RIA (Rich Internet Applications o Aplicaciones de Internet Ricas).

La definición de la interfaz de JSF 2 se realiza en forma de páginas XHTML con distintos tipos de etiquetas. Estas páginas se denominan páginas JSF. La siguiente figura muestra el funcionamiento de JSF para generar una página por primera vez.

Figura 3: Generación de una página JSF



Fuente: (Universidad de Alicante, 2012)

El navegador realiza una petición a una determinada URL en la que reside la página JSF que se quiere mostrar. En el servidor un servlet que se llama motor de JSF recibe la petición y construye un árbol de componentes a partir de la página JSF que se solicita.

Una vez que la página se muestra en el navegador, el usuario interactúa con ella (por ejemplo, pulsando en un botón, escribiendo texto en un campo o pinchando en una opción de un menú desplegable). En este momento es cuando se utiliza el enfoque de Ajax. (Universidad de Alicante, 2012)

2.5.3.1.2 Características de JSF.

JSF es un framework MVC (Modelo-Vista-Controlador) basado en el API de Servlets que proporciona un conjunto de componentes en forma de etiquetas definidas en páginas XHTML mediante el framework Facelets. Facelets se define en la especificación 2 de JSF como

un elemento fundamental de JSF que proporciona características de plantillas y de creación de componentes compuestos. Antes de la especificación actual se utilizaba JSP para componer las páginas JSF.

Para entender el funcionamiento de JSF es interesante compararlo con JSP. Se recuerda que una página JSP contiene código HTML con etiquetas especiales y código Java. La página se procesa en una pasada de arriba a abajo y se convierte en un servlet. Los elementos JSP se procesan en el orden en que aparecen y se transforman en código Java que se incluye en el servlet. Una vez realizada la conversión, las peticiones de los usuarios a la página provocan la ejecución del servlet. En JSF, la página HTML se genera como resultado de llamadas a métodos del árbol de componentes (en JSF se habla de realizar un render del componente). (Universidad de Alicante, 2012)

2.5.3.1.3 Evolución de JSF.

Especificaciones.

JavaServer Faces es el framework oficial de Java Enterprise para el desarrollo de interfaces de usuario avanzadas en aplicaciones web. La especificación de JSF ha ido evolucionando desde su lanzamiento en 2004 y se ha ido consolidando, introduciendo nuevas características y funcionalidades.

La especificación original de JavaServer Faces (1.0) se aprobó en marzo del 2004, con la Java Specification Request JSR 127. En esta especificación se define el funcionamiento básico de JSF, introduciéndose sus características principales: uso de beans gestionados el lenguaje JSF EL, componentes básicos y navegación entre vistas. (Universidad de Alicante, 2012)

La siguiente tabla muestra los distintos prefijos y espacios de nombres que se utilizan en las páginas JSF.

Figura 4: Prefijos y espacios de nombres utilizados en páginas JSF

Librería de etiquetas	Prefijo	URI
JSTL Core	c:	http://java.sun.com/jsp/jstl/core
JSTL Functions	fn:	http://java.sun.com/jsp/jstl/functions
JSF Facelets	ui:	http://java.sun.com/jsf/facelets
JSF HTML	h:	http://java.sun.com/jsf/htm
JSF Core	f:	http://java.sun.com/jsf/core

Fuente: (Universidad de Alicante, 2012)

2.5.3.1.4 Managed Bean.

Un apartado importante en el diseño de aplicaciones web. JSF usa beans para lograr esta separación. Las páginas JSF se refieren a las propiedades del *bean*, y la lógica de programa está contenida en el código de implementación del bean. Los beans son fundamentales para programar JSF. Una vez que un bean ha sido definido, puede ser accedido a través de etiquetas. JSF. (SGI-Consulting, 2010)

2.5.3.1.5 Ámbitos de los Beans.

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

Estos ámbitos normalmente mantienen beans y otros objetos que necesitan estar disponibles en diferentes componentes de una aplicación web.(SGI-Consulting, 2010)

Se encuentran anotados con "@ManagedBean" y seguido se tiene el alcance *(scope)* de sesión. El alcance puede estar de la siguiente forma:

- @RequestScoped → Está disponible en el mismo http request.
- @ViewScoped → Sólo disponible en la vista.
- @SessionScoped → Disponible en una sesión http de usuario.
- @ApplicationScoped → Disponible en el ciclo de vida de la aplicación, para todos los usuarios.

(Universidad de Alicante, 2012)

2.5.3.1.6 Manejo de Eventos.

Las aplicaciones de Web a menudo necesitan responder a los eventos del usuario, como los elementos seleccionados de un menú o dando un clic sobre un botón. Típicamente, se registran manejadores de eventos con componentes. JSF soportar tres clases de eventos:

- Eventos de cambio de valor.
- Eventos de acción.
- Eventos de fase.

(SGI-Consulting, 2010)

2.5.3.1.7 Árbol de Componentes.

JSF funciona en tres fases: primero genera un árbol de componentes (objetos Java) a partir de la vista JSF asociada a la petición, después activa el ciclo de vida de los componentes en el que se evalúan las expresiones EL y se procesan los eventos generados, y por último se renderizan los componentes resultantes.

En JSF los componentes se organizan en *vistas*. Cuando el framework recibe una petición, se construye una vista con los componentes relacionados con la petición.

Todos los componentes realizan tres tareas fundamentales:

- Validar los valores del componente.
- Manejar los eventos del componente.
- Renderizar el componente, normalmente generando código HTML.

(Universidad de Alicante, 2012)

2.5.3.1.8 Mensajes e Internacionalización.

Cuando se diseña una aplicación web se utiliza un idioma para generar la información, pero generalmente se quiere que pueda encontrarse expresada en algún otro idioma más.

La internacionalización es el proceso que permite diseñar una aplicación de tal forma que se encuentre adaptada a varios lenguajes y regiones sin cambios en la aplicación. (TIC-TEK, 2012)

2.5.4 Framework para la generación del modelo.

En esta sección se va a tratar el framework de persistencia de datos Hibernate con el cual se trabajó en el presente proyecto y del que se hablará en detalle.

2.5.4.1 Hibernate.

Figura 5: Logo de Hibernate

Fuente: <http://bit.ly/2imUCtH>

Antes de entrar en detalles de lo que es Hibernate se va mencionar un par de conceptos interesantes para comprender más la utilidad de esta importante herramienta de la cual se hizo uso para la generación del modelo.

2.5.4.1.1 Persistencia.

Las aplicaciones que se desarrollan hoy en día estructuran y guardan la información en sistemas de gestión de datos con el objetivo de que puedan ser reutilizados, bien desde la misma aplicación o desde otro proceso diferente. La capa de persistencia es la pieza que permite almacenar, recuperar, actualizar y eliminar el estado de los objetos que pueden ser persistentes en uno o más sistemas gestores de datos.

2.5.4.1.2 ORM.

Un ORM (Object Relational Mapping o mapeo de objeto relacional) es una capa que permite relacionar objetos con un modelo de datos relacional, de modo de ocultar todo el mecanismo de conexión al motor de base de datos y además no tener que escribir las sentencias SQL necesarias para hacer consultas y/o modificaciones a los registros de la base de datos. (Alvarez & Tejero, 2009)

Características.

- Definición de la correspondencia entre las clases y las tablas una sola vez (indicando que propiedad se corresponde con que columna, que clase con que tabla, etc…)
- Utilización de POJO's (**P**lain **O**ld **J**ava **O**bjects) en la aplicación y los hace persistentes con una sola instrucción: *orm.save(myObject).*
- Permite leer y escribir directamente en la Base Datos con VO (Value Object) (POJO).
- ORM es el middleware en la capa de persistencia que gestiona el modelo.
- Esto implica cierta penalización en el rendimiento.
- También hay sobrecarga en la gestión de los metadatos del mapeo, pero este coste es menor que el producido cuando se escribe a mano.

(Universidad de Murcia, 2011)

Para el funcionamiento de la aplicación que se desarrolle se necesitará crear y destruir sesiones todo el tiempo, quizá en cada petición. Puede ser útil pensar en una sesión como en una caché o colección de objetos cargados (a o desde una base de datos) relacionados con una única unidad de trabajo. Hibernate puede detectar cambios en los objetos pertenecientes a una unidad de trabajo.

- La interfaz ***SessionFactory*** permite obtener instancias *Session*.
- La interfaz ***Configuration*** se utiliza para configurar y "arrancar" Hibernate.
- La interfaz ***Query*** permite realizar peticiones a la base de datos y controla cómo se ejecuta dicha petición (query). Las peticiones se escriben en ***HQL*** (Hibernate Query Languaje o Lenguaje de Consultas de Hibernate) o en el dialecto SQL nativo de la

base de datos que se esté utilizando. Es importante tener en cuenta los objetos **Criteria** que se utilizan para crear y ejecutar consultas con objetos.

2.5.5 Frameworks JSF para la Vista.

A continuación se van a definir todos los frameworks de componentes para JSF elegidos que hacen parte del presente trabajo y también se van a mencionar algunos de los componentes más utilizados que incluyen.

2.5.5.1 PrimeFaces.

Figura 1: Logo de Primefaces

Fuente: <http://bit.ly/2yLX6Zi>

Primefaces es una librería de componentes visuales de código abierto para el conjunto Java Server Faces 2.0 en adelante desarrollada y mantenida por Prime Technology. Su objetivo principal es ofrecer un conjunto de componentes para facilitar la creación y diseño de aplicaciones web.

Los componentes de **Primefaces** cuentan con soporte nativo de Ajax, pero no se encuentra implícito, de tal manera que se tiene que especificar que componentes se deben actualizar al realizar una petición proporcionando así mayor control sobre los eventos. Cuenta

también con un módulo adicional para el desarrollo de aplicaciones web para dispositivos móviles con navegadores basados en WebKit.

Las principales características de PrimeFaces son:

- Soporte nativo de Ajax, incluyendo Push/Coment.G
- Kit para crear aplicaciones web móviles.
- Es compatible con otras librerías de componentes como Jboss RichFaces.
- Uso de JavaScript no intrusivo.
- Es un proyecto open source, activo y estable.

(Pech-May, Gomez-Rodriguez, Cruz-Diaz, & Lara-Jeronimo, 2010)

- Rico conjunto de componentes (HtmlEditor, Dialog, Autocompletar, Gráficos y muchos más).
- Skinning Framework con más de 35 temas incorporados y soporte para la herramienta de diseño de tema visual.
- Temas y diseños predefinidos.
- Amplia documentación.
- Comunidad de usuarios grande, dinámica y activa.

(primefaces.org, 2017)

2.5.5.2 ICEfaces.

Figura 2: Logo de ICEfaces

Fuente: <http://bit.ly/2z3Bs2h>

ICEfaces es un marco de desarrollo de aplicaciones de Internet enriquecidas (RIA) de código abierto para Java EE. ICEfaces funciona en plataformas que van desde computadoras de escritorio hasta teléfonos inteligentes y Apple a Android. Mejora la eficiencia del desarrollador de tal manera que reduce el tiempo de comercialización y los costos operativos.

ICEfaces es la forma más simple y rentable para que las empresas movilicen sus aplicaciones web Java EE. Las bibliotecas de componentes de ICEfaces aprovechan lo último en HTML5 así como las técnicas de diseño adaptativo / adaptativo para que una sola página se pueda ver de manera óptima en una amplia gama de dispositivos. Más allá de los componentes, Cloud Push ofrece la próxima generación de tecnologías push. Ahora, los usuarios pueden recibir alertas sobre las actualizaciones de las aplicaciones incluso cuando no están conectadas. (icesoft.org, 2017)

Los componentes de ACE (Componentes Avanzados de ICEfaces) utilizan una combinación de técnicas de representación basadas en el lado del servidor y en el cliente para proporcionar una experiencia de usuario rica y receptiva con requisitos de red y de procesamiento del servidor, reducidos.

Principales características (icesoft.org, 2017):

- Aprovecha los poderosos componentes de JavaScript de bibliotecas de terceros, como jQuery o PrimeFaces, mientras protege a los desarrolladores de aplicaciones ICEfaces de tener que aprender y/o usar JavaScript directamente.
- Admite una amplia funcionalidad del lado del cliente para mejorar la riqueza, la capacidad de respuesta y la escalabilidad de los componentes.
- Proporcionar un enfoque flexible y consistente para el tematizado y desvelado de UI en todos los componentes, en base a los temas de *jQuery ThemeRoller*.

Estos componentes funcionan en el nivel de infraestructura de ICEfaces y se pueden usar junto con cualquier componente JSF compatible en todas las plataformas (de escritorio o móviles). (icesoft.org, 2017)

2.5.5.3 RichFaces.

Figura 3: Logo de RichFaces

Fuente: <http://bit.ly/2isn7CH>

El proyecto RichFaces es un marco avanzado de componentes de interfaz de usuario para integrar fácilmente las capacidades de Ajax en aplicaciones comerciales utilizando JSF.

RichFaces 4 se basa en el soporte pionero de Ajax que comenzó con RichFaces 3 y está estandarizado en JSF 2. Además de ampliar estas capacidades de ajax, RichFaces también mejora otras áreas de JSF 2.

Este framework incluye (richfaces.jboss, 2016):

- Un conjunto completo de componentes habilitados para AJAX en dos bibliotecas.
  - a4j (ajax4jsf): controles AJAX centrados en la página.
  - rich: componentes autónomos y listos para usar.
- Validación del lado del cliente, ampliando JSR 303 Bean Validation hasta el navegador.
- Cola avanzada para satisfacer los requisitos de alto rendimiento de las aplicaciones empresariales del mundo real.
- Actualizaciones de componentes Push que incluyen integraciones de JavaMessaging Service (JMS) y varios mecanismos de transporte basados en el soporte del navegador.
- Cuenta con kit propio de desarrollo de componentes.
- Documentación completa que cubre las mejores prácticas de desarrollo y los detalles de los componentes.
- Detallados y automatizados testing facilities para los componentes, acciones, oyentes, y páginas.
- Amplia compatibilidad con navegadores cruzados.
- Comunidad grande y activa en su fundación.
- Validación de objetos para una fácil implementación de validación entre campos.

(richfaces.jboss, 2016)

2.5.5.4 OpenFaces.

Figura 4: Logo de OpenFaces

Fuente: <http://bit.ly/2j7flii>

OpenFaces es una biblioteca de código abierto de componentes JSF alimentados por AJAX, un marco Ajax y un marco de validación del lado del cliente. OpenFaces se basa en el conjunto de componentes JSF conocidos anteriormente como QuipuKit. Contiene una base de código totalmente revisada de QuipuKit e introduce muchos componentes y funciones.

OpenFaces se distribuye bajo un modelo de licencia dual. Significa que puede elegir entre usar la biblioteca bajo Licencia Pública General Reducida de GNU (LGPL) o comprar una licencia comercial.(openfaces.org, 2017)

2.5.5.5 OmniFaces

Figura 5: Logo de OmniFaces

Fuente: <http://bit.ly/2ywqhLN>

OmniFaces es una biblioteca de utilidades para JSF 2 que se enfoca en utilidades que facilitan las tareas diarias con JSF estándar. OmniFaces es una respuesta a los problemas

frecuentemente recurrentes encontrados durante las edades de desarrollo profesional de JSF y de las preguntas que se hacen en *Stack Overflow*.

OmniFaces está más orientado a "utilidades" que resuelven problemas prácticos cotidianos y soluciones para (pequeñas) deficiencias en JSF. Dichas utilidades y soluciones alternativas pueden basarse en componentes, pero OmniFaces no necesariamente se esfuerza por ser una "biblioteca de componentes" por sí misma. OmniFaces solo se puede usar junto con cualquier biblioteca de componentes. (omnifaces.org, 2017)

2.5.5.6 BootFaces.

Figura 6: Logo de BootFaces

Fuente: <http://bit.ly/2iyvxsh>

Es un framework JSF potente y liviano basado en *Bootstrap 3* y *jQuery UI* que le permite desarrollar aplicaciones empresariales de Front-End de forma rápida y sencilla.

BootsFaces intenta comenzar desde los pilares, centrándose primero en la estructura y el diseño de la página. Busca potentes características de diseño, aprovecha el *Bootstrap Grid System*. Las funciones de respuesta de BootsFaces ayudan a diseñar solo una versión del sitio web, que se adaptará a teléfonos móviles, tabletas pequeñas y grandes, y pantallas de escritorio. El sistema de compilación BootsFaces permite personalizar la apariencia de los componentes basados en Bootstrap y qué componentes incluir en la biblioteca. (bootsfaces.net, 2017)

2.5.5.7 ButterFaces.

Figura 7: Logo de ButterFaces

Fuente: <http://bit.ly/2zlZnbn>

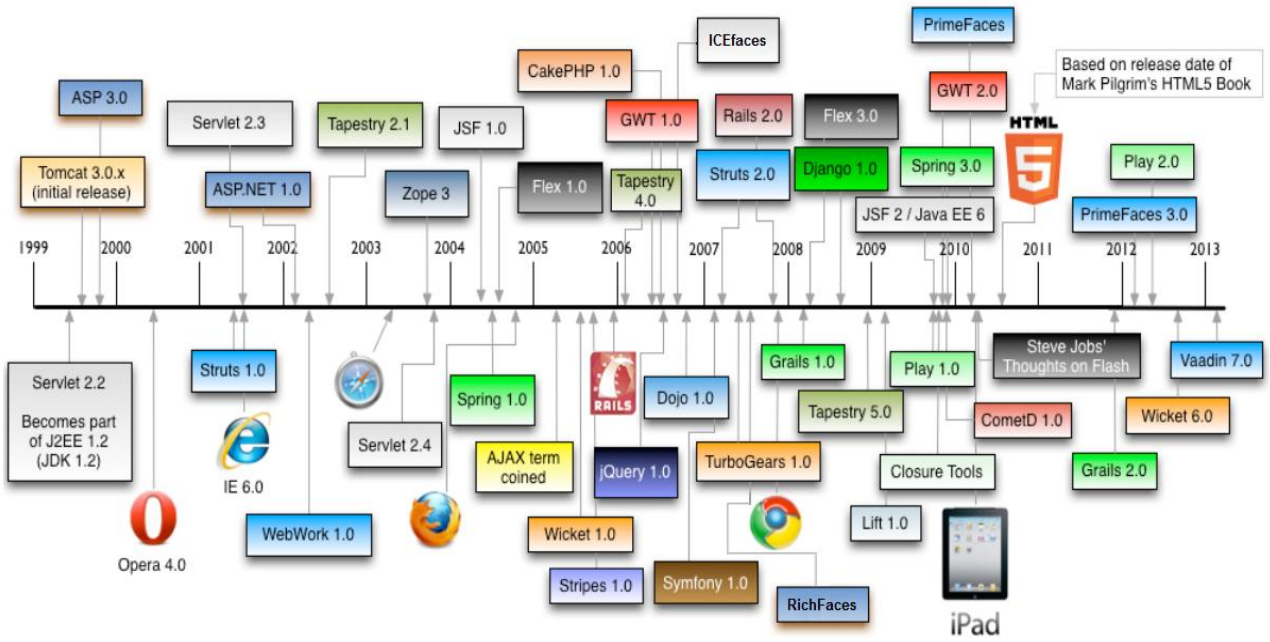
ButterFaces es un marco JSF liviano y sensible que combina las ventajas de *Bootstrap, jQuery* y *HTML 5* para desarrollar aplicaciones web rápidas, sencillas y modernas usando JSF 2. Arranca la mayoría de los componentes de entrada html y permite el uso de la mayoría de las características de html 5, por ejemplo, marcador de posición, tipo, patrón y similares.

Cada componente de entrada se amplía mediante una vista de solo lectura y tiene una etiqueta receptiva y atributos de información sobre herramientas que admiten *ajax* y validación. El diseño responsivo de Bootstraps y los complementos de jQuery ayudan a crear aplicaciones excelentes e inteligentes.(butterfaces.org, 2017)

2.6 Evolución histórica de Frameworks.

Existe una gran variedad de frameworks y librerías Open Source en la actualidad, la siguiente figura describe la evolución histórica de algunos de los más conocidos, entre los que se encuentran los frameworks utilizados en este trabajo.

Figura 8: Evolución en el tiempo de Frameworks Web



Fuente: Raible Designs – comparing jvm web frameworks [en línea] Imagen Editada <http://static.raibledesigns.com/repository/presentations/Comparing_JVM_Web_Frameworks_February2014.pdf>

En la anterior imagen se pueden observar muchos de los Frameworks que han sido mencionados en el presente trabajo comparativo. En general se ubican todos de acuerdo a su evolución desde los MVC tales como Struts, JSF, Spring, Symfony y Vaadin, hasta los que fueron construidos para interfaces de usuarios como ICEfaces, Richfaces y PrimeFaces, los cuales son tema del presente estudio.

2.7 Conclusiones del Capítulo.

El contenido de este capítulo deja como conclusión las ventajas y la importancia del uso de los Frameworks creados con el objetivo de ser eficientes al desarrollar aplicaciones de software bajo el patrón MVC. Entre ellos los que se encuentran para mayor facilidad en la implementación de software, es decir los que se utilizan en la parte de la generación del modelo

o los que facilitan el trabajo del lado del cliente los cuales entran a ser pieza fundamental a la hora de construir aplicaciones web como sistemas de software o sitios web empresariales Open Source en Java. Exponiendo una serie de características que los identifican a cada uno ya que todas estas son evoluciones que han surgido en el tiempo con el fin de brindar agilidad en el proceso de desarrollo y con esto mucha más facilidad en las actividades requeridas para el desarrollador para entonces así mejorar la experiencia del usuario al navegar resultando más llamativa teniendo en cuenta que se hace mayor enfoque en la lógica de negocio y aspecto visual.

Este capítulo es una base para la interpretación, estudio y análisis de acuerdo a las características y funcionamiento de cada uno de los Frameworks web en especial los que fueron creados para el desarrollo de las interfaces gráficas de usuarios ya previamente mencionados en el mundo de JSF. Analizando dichas características en cuantos a la madurez que han alcanzado los framework a nivel de componentes se puede determinar cuáles son los que en la capa de presentación (vistas) contienen mejor forma de combinación con los beans que son los que en definitiva le dan vida a estos y además cuenta con una gran variedad de propiedades.

Es fundamental entender que dependiendo del tipo de sistema de software o sitio web que se pretenda construir con JSF hay diferentes opciones en cuanto a Frameworks o bibliotecas de componentes Open Source que se pueden utilizar y depende únicamente del desarrollador seleccionar el más adecuado de modo que se adapte a sus necesidades. También se puede hacer uso de la integración de varios de estos Frameworks aprovechando lo mejor de cada uno en cuanto a características, funcionamiento y facilidad de uso.

3 Comparativo de Frameworks para Vistas.

El contenido de este capítulo hace referencia a la especificación del proceso de comparación de los 3 (Tres) Frameworks elegidos que actualmente se encuentran más completos y por tanto apropiados para su implementación con JSF y finalmente la selección de uno de ellos a partir de los 5 (Cinco) criterios de evaluación.

Para el propósito de este proyecto el cual es crear un prototipo del aplicativo Administrador Vortal en el CIADTI con el fin de hacer más llamativa e interactiva su vista o interfaces de usuarios pero sin dejar de lado el normal procesamiento de los datos e integración con la arquitectura actualmente existente, se hizo el siguiente estudio y análisis para determinar el Frameworks JSF Open Source más adecuado según características.

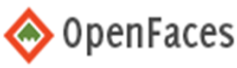
La forma en la que fueron evaluados cada uno de los criterios y de la misma manera el resultado final, está basado en una tesis de la Universidad Técnica del Norte de Ibarra Ecuador realizada en el 2011 cuyo título es “ESTUDIO COMPARATIVO DE FRAMEWORKS RIA PARA EL DESARROLLO DE APLICACIONES WEB CON JAVA SERVER FACES (JSF)”.

La siguiente calificación está basada en los componentes mínimos necesarios para el normal desarrollo de la aplicación como son los que muestran los datos en tablas, los que generan el menú, los que gestionan las pestañas, los que seleccionan datos de un menú desplegable y los del uso de Ajax.

Tabla 5: Descripción de calificación para la selección de 3 (Tres) Frameworks de JSF Open Source.

Calificación	Descripción
1	Sus componentes y características no son apropiadas para el propósito del proyecto.
2	Sus características son apropiadas pero su falta de componentes evidencia limitaciones considerables para el propósito del proyecto.
3	Sus características y conjunto de componentes son indicados para el normal desarrollo del proyecto.

Tabla 6: Asignación de calificaciones para los Frameworks JSF

Frameworks JSF Open Source		Calificación
	**Primefaces**	**3**
	**ICEfaces**	**3**
	**RichFaces**	**3**
	**OpenFaces**	**2**
	**BootFaces**	**2**
	**OmniFaces**	**2**

	**ButterFaces**	**1**

Los Frameworks o Bibliotecas de componentes de JSF que resultaron más apropiados para el desarrollo y transformación del administrador Vortal en el CIADTI según sus características y cantidad de componentes son Primefaces, ICEfaces y RichFaces por lo tanto en este capítulo se llevará a cabo un estudio más detallado de ellos a modo de comparación para seleccionar el mejor de los tres.

Para el comparativo se definieron los siguientes criterios:

- **Mantenibilidad y Documentación:** La mantenibilidad es el grado o facilidad de realizar cambios o modificaciones con el fin de corregir defectos, mejorar el rendimiento u otros atributos en el código del vortal al usar el respectivo framework y la documentación es todo el conjunto de documentos que se han ido generando al crear nuevas versiones del framework, por ejemplos las guía del usuario.
- **Índices de Interés:** Son representaciones estadísticas a través de gráficas obtenidas mediante el uso una herramienta online del motor de búsqueda de Google con el fin de visualizar la popularidad de los frameworks utilizados en cierto periodo de tiempo.
- **Disponibilidad de Componentes:** Es la cantidad de componentes del framework disponible en su sitio web con todos los recursos con los que cuenta para su correcto funcionamiento sin necesidad de utilizar los de un tercero.

- **Usabilidad:** Es la facilidad o capacidad con que los usuarios y desarrolladores puedan entender, aprender y usar el framework, bien sea a nivel de manejo de código o de estética de la interfaz de usuario.
- **Funcionalidad:** Es la capacidad del framework que junto con todos los elementos propios permite brindar el eficiente funcionamiento, satisfaciendo al usuario con el cumpliendo de sus objetivos.

A continuación se llevará a cabo la evaluación con cada uno de estos 5 criterios a cada uno de los 3 frameworks seleccionados.

3.1 Mantenibilidad y Documentación.

Este criterio busca evidenciar con qué cantidad de documentación cuenta el Framework para facilitar su aprendizaje de modo que sea más fácil su mantenimiento, además pretende mostrar qué tan frecuentes son las actualizaciones de versiones por parte de su organización desarrolladora de tal forma que también se podrá identificar la madurez del mismo.

3.1.1 Primefaces.

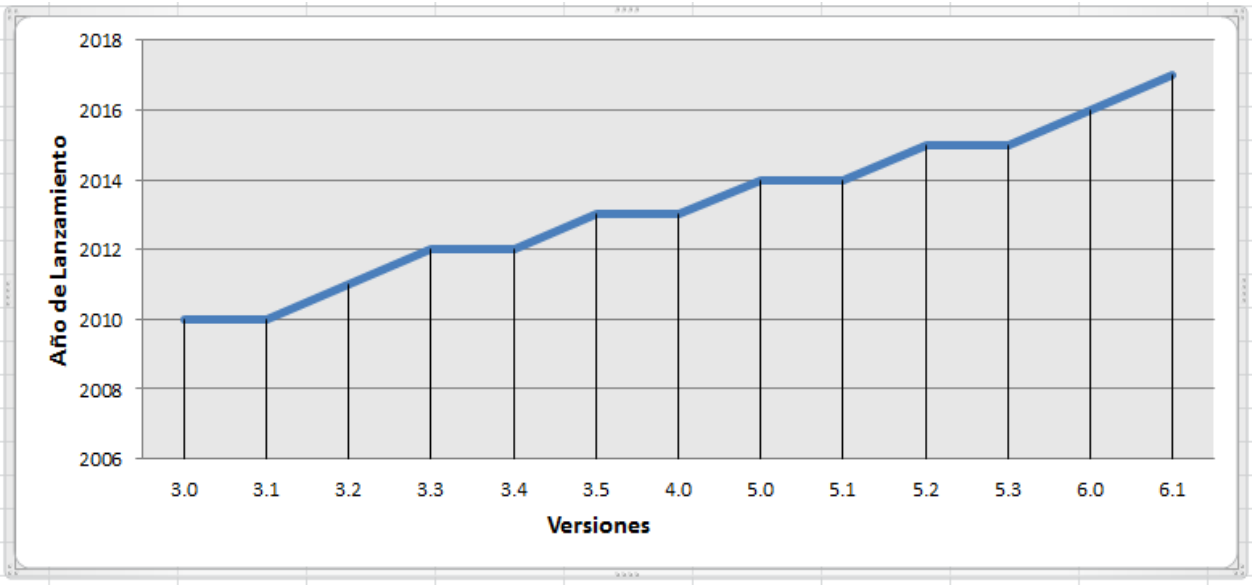
Para el análisis de mantenibilidad del Framework Primefaces se hizo una tabla describiendo todas las versiones a lo largo del tiempo desde que fue lanzado hasta la actualidad y luego una gráfica que muestre la variabilidad de tiempo teniendo en cuenta que para cada una se creó una guía muy completa para el usuario con el fin de demostrar el esfuerzo que se emplea para conservar y mejorar su funcionamiento. También cuenta con un soporte muy extenso de las componentes y clases en su Wiki y API Doc, adicionalmente dispone de una comunidad muy activa lo que de alguna manera contribuye a esta amplia documentación.

Tabla 7: Versiones desde el inicio de Primefaces hasta la actualidad

Versión	Guía del Usuario	Fecha de Lanzamiento
primefaces-3.0.jar	primefaces_user_guide_3_0.pdf	2010/04/10
primefaces-3.1.jar	primefaces_user_guide_3_1.pdf	2010/11/03
primefaces-3.2.jar	primefaces_user_guide_3_2.pdf	2011/07/05
primefaces-3.3.jar	primefaces_user_guide_3_3.pdf	2012/01/06
primefaces-3.4.jar	primefaces_user_guide_3_4.pdf	2012/09/03
primefaces-3.5.jar	primefaces_user_guide_3_5.pdf	2013/02/04
primefaces-4.0.jar	primefaces_user_guide_4_0.pdf	2013/10/02
primefaces-5.0.jar	primefaces_user_guide_5_0.pdf	2014/05/04
primefaces-5.1.jar	primefaces_user_guide_5_1.pdf	2014/10/06
primefaces-5.2.jar	primefaces_user_guide_5_2.pdf	2015/04/07
primefaces-5.3.jar	primefaces_user_guide_5_3.pdf	2015/10/18
primefaces-6.0.jar	primefaces_user_guide_6_0.pdf	2016/06/07
primefaces-6.1.jar **Versión Actual**	primefaces_user_guide_6_1.pdf	2017/04/19

Como se puede observar en la tabla anterior el trabajo de la comunidad desarrolladora de Primefaces es constante desde el año 2010, lo que hace de este framework el más joven de todos. La siguiente gráfica describe con mejor claridad el trabajo de mantenimiento del Framework.

Figura 9: Gráfica de mantenibilidad entre versiones de Primefaces



Fuente: Autor

3.1.2 ICEfaces.

Para ICEfaces se encuentra una documentación bastante extensa, al igual que Primefaces incluyes una Wiki para dar soporte a sus componentes, también contiene un conjunto de tutoriales y foros. Por otro lado resulta un poco molesto que sea necesario el registro para acceder a cualquiera de los recursos que se encuentran disponibles, además resulta un poco confuso al acceder al sitio de icefaces.org donde siempre aparece icesoft.org que ofrece otro tipo de tecnologías de desarrollo. Este framework es casi igual de antiguo que RichFaces desde el 2006-2007 y también ha alcanzado madurez gracias a su gran número de versiones, las cuales

seguidamente se mencionaran en una tabla junto a su fecha de lanzamiento a partir del 2010 periodo en el que se empezaron a actualizar agregando un grupo de componentes como es el que caso del uso de Ajax lo que generó mucha competencia junto a Primefaces y RichFaces.

Luego se mostrará una gráfica con la variabilidad entre las versiones.

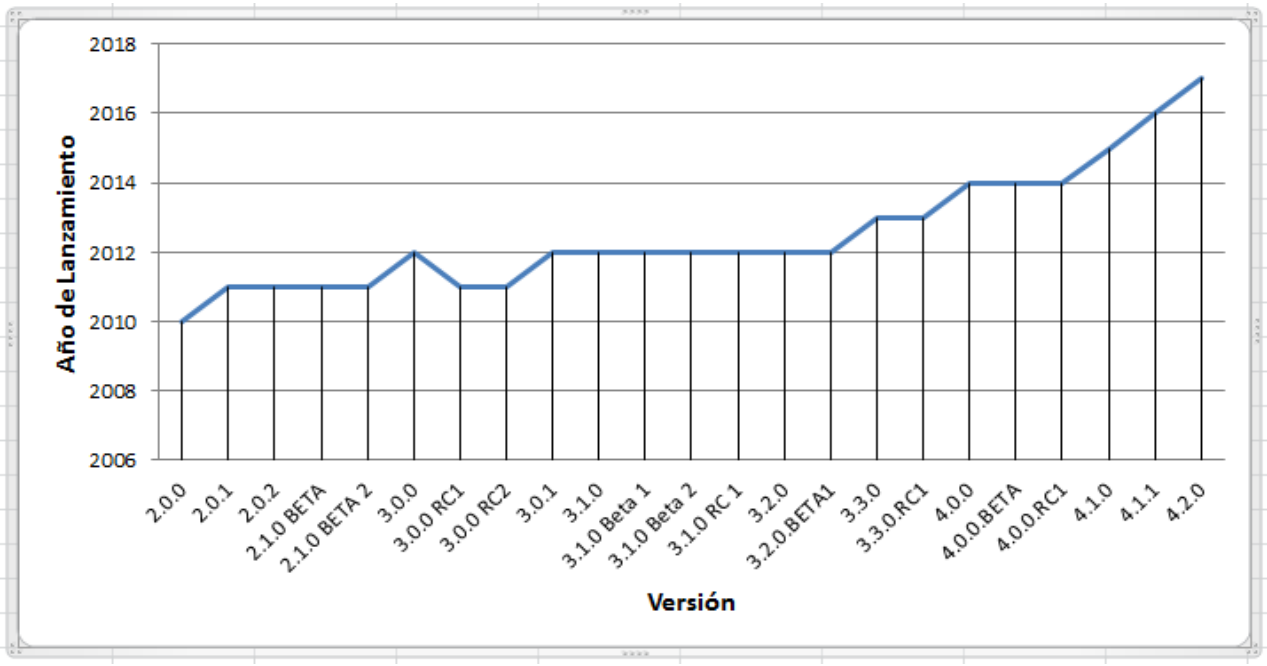
Tabla 8: Versiones de ICEfaces desde 2010 hasta la actualidad

Versión	Fecha Lanzamiento
ICEfaces 4.2.0	09/03/2017
ICEfaces 4.1.1	26/02/2016
ICEfaces 4.1.0	23/12/2015
ICEfaces 4.0.0	04/11/2014
ICEfaces 4.0.0.RC1	23/07/2014
ICEfaces 4.0.0.BETA	21/03/2014
ICEfaces 3.3.0	16/04/2013
ICEfaces 3.3.0.RC1	29/03/2013
ICEfaces 3.2.0	02/11/2012
ICEfaces 3.2.0.BETA1	27/09/2012
ICEfaces 3.1.0	24/07/2012

ICEfaces 3.1.0 RC 1	04/07/2012
ICEfaces 3.1.0 Beta 2	11/06/2012
ICEfaces 3.1.0 Beta 1	18/05/2012
ICEfaces 3.0.1	27/03/2012
ICEfaces 3.0.0	03/02/2012
ICEfaces 3.0.0 RC2	22/12/2011
ICEfaces 3.0.0 RC1	05/12/2011
ICEfaces 2.1.0 BETA 2	04/11/2011
ICEfaces 2.1.0 BETA	04/10/2011
ICEfaces 2.0.2	13/05/2011
ICEfaces 2.0.1	30/03/2011
ICEfaces 2.0.0	21/12/2010

Se puede observar en la tabla anterior que la cantidad de versiones de este framework es más numerosa que la de Primefaces por lo que se puede decir que las mejoras inicialmente eran mínimas ya que se encuentran muchas versiones BETA o RC a partir del año 2010. La siguiente gráfica describe con más detalle el trabajo de mantenimiento del framework ICEfaces.

Figura 10: Mantenibilidad entre versiones de ICEfaces



Fuente: Autor

3.1.3 RichFaces.

Este framework es maduro a pesar de la escasa documentación, tiene una comunidad bastante activa y ofrece una guía de usuario lanzamiento tras lanzamiento desde el 2007. Cuenta con muy pocos tutoriales en los que el desarrollador se pueda soportar para crear aplicaciones con el mismo, pero tiene disponible su Wiki y algunos foros. RichFaces alcanzó su fin de vida en Junio de 2016. En el siguiente cuadro se muestran algunas versiones teniendo en cuenta que la documentación no muestra en su totalidad la fecha de lanzamiento de cada una de las versiones.

Tabla 9: Versiones de RichFaces.

Versión
RichFaces 4.5.17.Final en 2016
RichFaces CDK 4.5.0

RichFaces 4.5.16
RichFaces 4.3.7
RichFaces 4.2.3
RichFaces 4.2.2
RichFaces 4.2.1
RichFaces 4.2.0
RichFaces 4.1.0
RichFaces 4.0.0
RichFaces 3.3.3
RichFaces 3.3.4 Binario
RichFaces 3.3.x GA (x:0-2) en 2009
RichFaces 3.3.2.SR1
RichFaces 3.3.1.GA
RichFaces 3.3.0.GA
RichFaces 3.2.x GA (x:0-2) en 2008
RichFaces 3.2.2.GA
RichFaces 3.2.1.GA
RichFaces 3.2.0.SR1
RichFaces 3.2.0.GA
RichFaces 3.1.x GA (x:0-6) entre 2007-2008
RichFaces 3.1.6.GA
RichFaces 3.1.5.GA
RichFaces 3.1.4.GA
RichFaces 3.1.3.GA

RichFaces 3.1.2.GA

En la tabla anterior se puede observar el contenido de información sobre versiones archivadas y descargas binarias. No se recomienda utilizar las versiones archivadas porque las versiones estables más nuevas tienen muchas correcciones de errores, características y actualizaciones que estas versiones no ofrecen. También será más difícil encontrar ayuda en la comunidad para estos lanzamientos.

3.1.4 Evaluación de mantenibilidad y documentación.

Tabla 10: Calificación de mantenibilidad y documentación para cada Framework

Criterio	Valor	Significado	Primefaces	ICEfaces	RichFaces
Mantenibilidad y Documentación	3	Muy Buena	X		
	2	Buena		X	
	1	Regular			X

3.2 Índices de Interés.

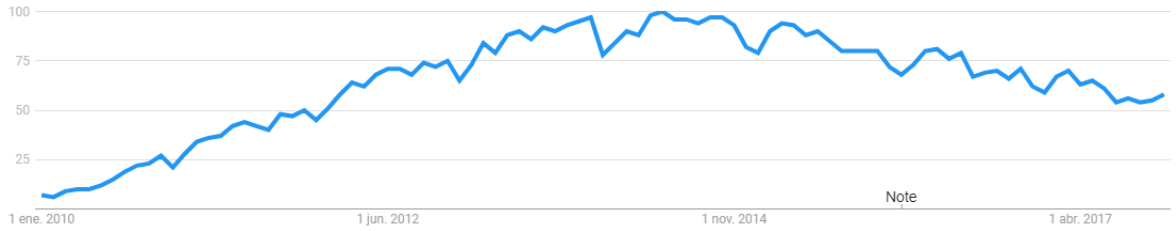
Este criterio pretende evidenciar la popularidad o qué tanto interés en un tema específico han tenido las personas que utilizan el buscador de Google. Este motor de búsqueda es el más utilizado a nivel mundial y nacional, posee diferentes herramientas que permiten facilitar información al usuario, una de estas es Google Trends que muestra a través de gráficas el comportamiento de la búsqueda en este caso de estudio de los frameworks Primefaces, ICEfaces y RichFaces en el tiempo.

3.2.1 Primefaces.

Para el caso de estudio se realizó la consulta con dos frases relativas a este framework a nivel mundial y nacional, las cuales son **Primefaces** o **JSF Primefaces** los resultados arrojados son los descritos a continuación.

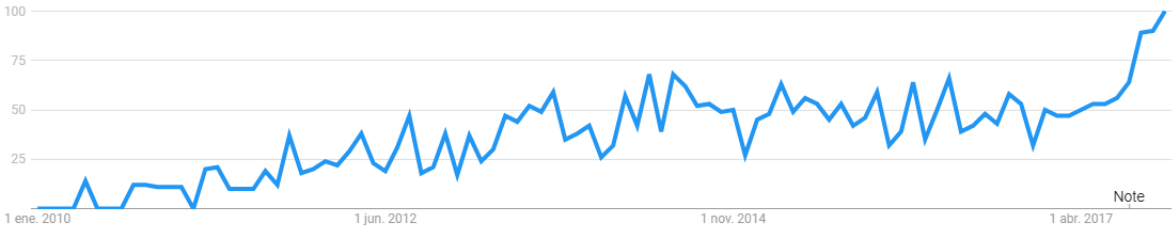
- ✓ **Búsqueda por la frase Primefaces o JSF Primefaces.**

Figura 11: Interés de Búsqueda Web por la frase Primefaces a nivel mundial Google Trends



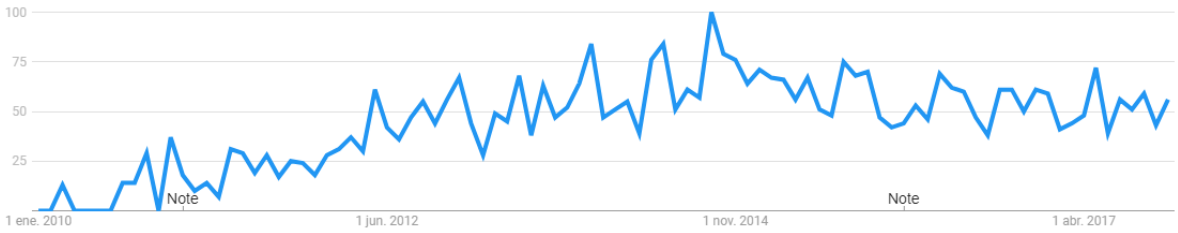
Fuente: Google Trends - < https://trends.google.es/trends/explore?cat=5&date=2010-01-01%202017-11-26&q=Primefaces> [citado el 27 de Noviembre de 2017]

Figura 12: Interés de Búsqueda YouTube por la frase Primefaces a nivel mundial Google Trends



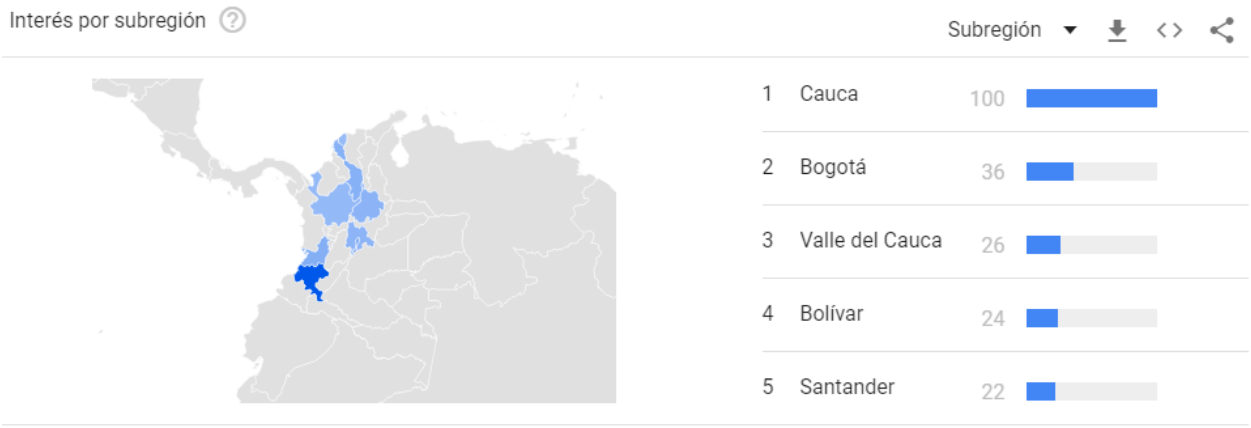
Fuente: Google Trends - < https://trends.google.es/trends/explore?cat=5&date=2010-01-01%202017-11-26&gprop=youtube&q=Primefaces > [citado el 27 de Noviembre de 2017]

Figura 13: Interés de Búsqueda Web por la frase Primefaces a nivel nacional Google Trends



Fuente: Google Trends - < https://trends.google.es/trends/explore?cat=5&date=2010-01-01%202017-11-26&geo=CO&q=Primefaces> [citado el 27 de Noviembre de 2017]

Figura 14: Interés de Búsqueda Web por la frase Primefaces a nivel regional Google Trends



Fuente: Google Trends - < https://trends.google.es/trends/explore?cat=5&date=2010-01-01%202017-11-26&geo=CO&q=Primefaces> [citado el 27 de Noviembre de 2017]

La gráfica arrojada por la primera búsqueda a nivel mundial muestra que entre el 2013 y 2015 aproximadamente existe un interés muy alto y constante referente a Primefaces. La siguiente filtrada por búsqueda de YouTube muestra un crecimiento constante desde que surgió hasta el presente lo que indica que los interesados se sienten más atraídos por los tutoriales.

En Colombia, es decir el resultado obtenido al buscar a nivel nacional se observó que la popularidad del framework fue creciendo desde su creación y se ha mantenido el interés desde el 2013 hasta el día de hoy y finalmente en la búsqueda a nivel regional se encuentra que en el departamento del Cauca se presentan mayores interesados en el tema, luego le sigue la ciudad de Bogotá. Todos estos resultados demuestran el gran crecimiento que ha tenido el uso de esta nueva tecnología para implementar las interfaces de JSF, lo que resultó una herramienta muy útil y fácil de usar.

3.2.2 ICEfaces.

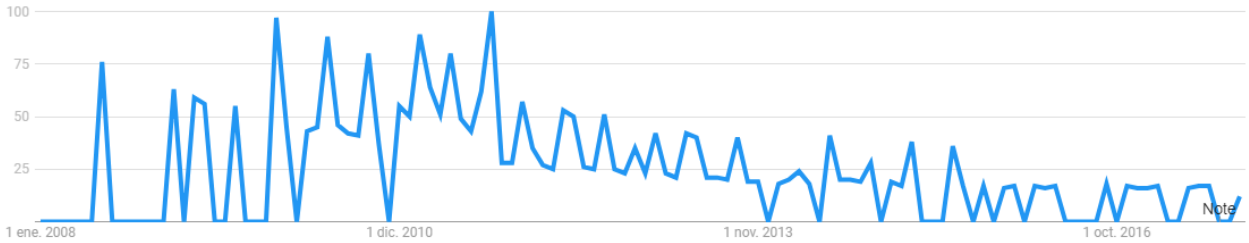
Para el caso de estudio de ICEfaces también se realizó la consulta con dos frases relativas a este framework a nivel mundial y nacional, las cuales son **ICEfaces** o **JSF ICEfaces** los resultados arrojados son los siguientes.

- ✓ **Búsqueda por la frase ICEfaces o JSF ICEfaces.**

Figura 15: Interés de Búsqueda Web por la frase ICEfaces a nivel mundial Google Trends

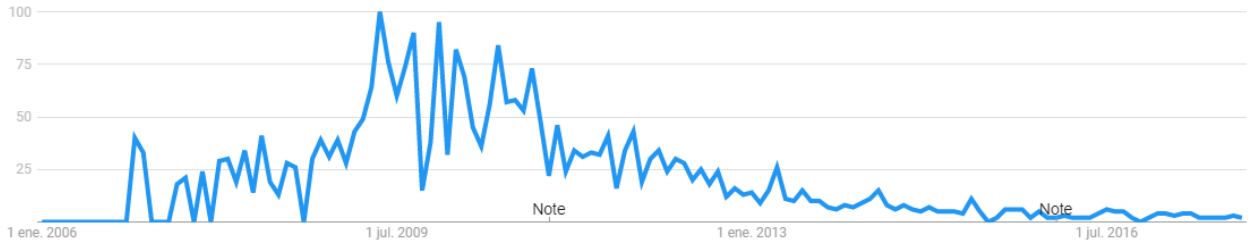
Fuente: Google Trends - < https://trends.google.es/trends/explore?cat=5&date=2006-01-01%202017-11-27&q=icefaces> [citado el 27 de Noviembre de 2017]

Figura 16: Interés de Búsqueda YouTube por la frase ICEfaces a nivel mundial Google Trends



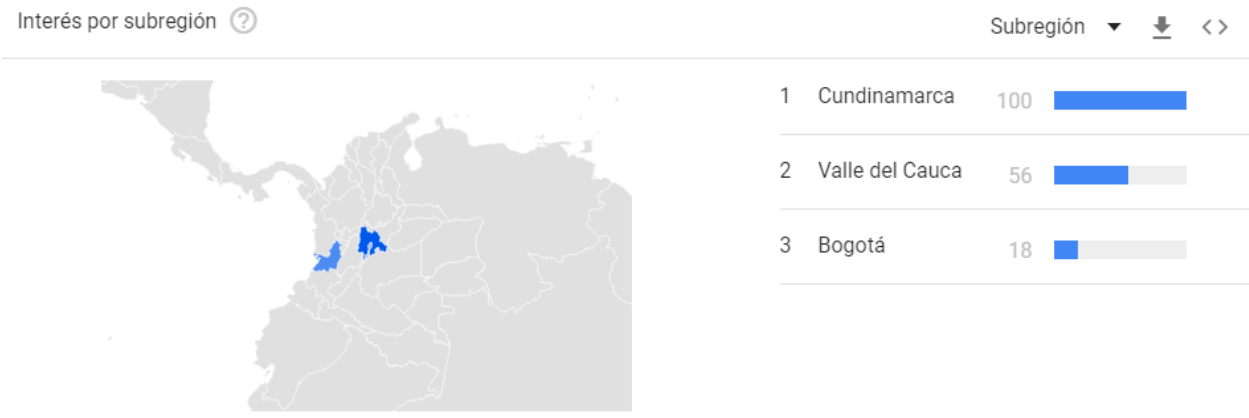
Fuente: Google Trends - < https://trends.google.es/trends/explore?cat=5&date=all_2008&gprop=youtube&q=icefaces> [citado el 27 de Noviembre de 2017]

Figura 17: Interés de Búsqueda Web por la frase ICEfaces a nivel nacional Google Trends



Fuente: Google Trends - < https://trends.google.es/trends/explore?cat=5&date=2006-01-01%202017-11-27&geo=CO&q=icefaces> [citado el 27 de Noviembre de 2017]

Figura 18: Interés de Búsqueda Web por la frase ICEfaces a nivel regional Google Trends



Fuente: Google Trends - < https://trends.google.es/trends/explore?cat=5&date=2006-01-01%202017-11-27&geo=CO&q=icefaces> [citado el 27 de Noviembre de 2017]

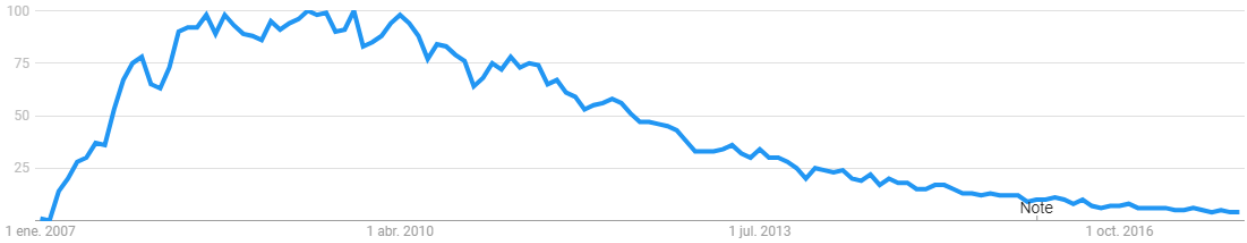
Los resultados obtenidos demuestran generalmente que el interés más alto que tuvo esta tecnología web fue aproximadamente entre 2009 y 2011, se puede observar que a partir de entonces ha venido disminuyendo considerablemente lo que deja en evidencia la ventaja que tiene Primefaces hasta este punto. El resultado obtenido al hacer la búsqueda a nivel nacional se observa que la mayor popularidad del framework ha tenido lugar en el departamento de Cundinamarca y como tal en la capital. Estos resultados demuestran la decadencia que ha tenido el uso de esta tecnología para trabajar las interfaces de JSF.

3.2.3 RichFaces.

Para esta tecnología de la misma forma se realizó la consulta con dos frases relativas a este framework a nivel mundial y nacional, las cuales son **RichFaces** o **JSF RichFaces** los resultados obtenidos son los siguientes.

- ✓ **Búsqueda por la frase RichFaces o JSF RichFaces.**

Figura 19: Interés de Búsqueda Web por la frase RichFaces a nivel mundial Google Trends



Fuente: Google Trends - < https://trends.google.es/trends/explore?cat=5&date=2007-01-01%202017-11-27&q=%2Fm%2F047fy_g> [citado el 27 de Noviembre de 2017]

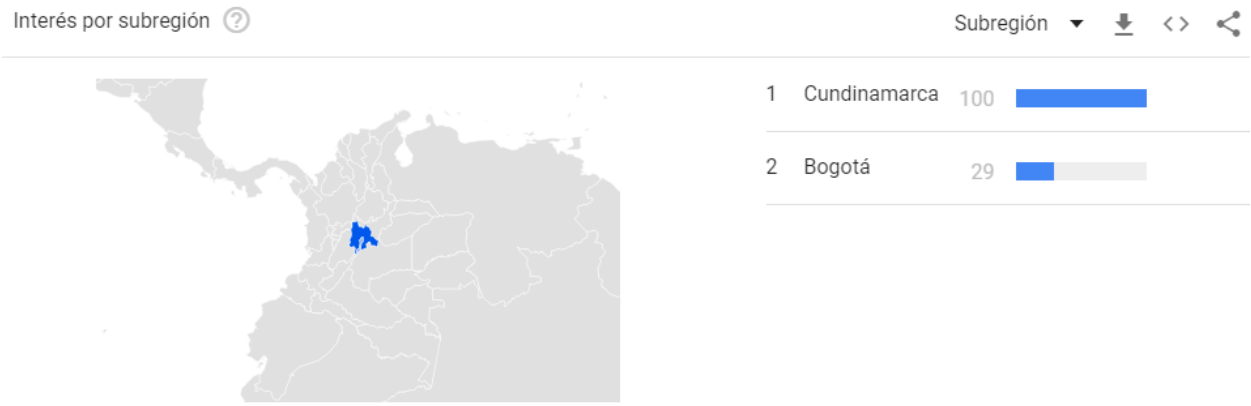
Figura 20: Interés de Búsqueda YouTube por la frase RichFaces a nivel mundial Google Trends

Fuente: Google Trends <https://trends.google.es/trends/explore?cat=5&date=all_2008&gprop=youtube&q=%2Fm%2F047fy_g> [citado el 27 de Noviembre de 2017]

Figura 21: Interés de Búsqueda Web por la frase RichFaces a nivel nacional Google Trends

Fuente: Google Trends < https://trends.google.es/trends/explore?cat=5&date=2008-01-01%202017-11-27&geo=CO&q=%2Fm%2F047fy_g> [citado el 27 de Noviembre de 2017]

Figura 22: Interés de Búsqueda Web por la frase RichFaces a nivel regional Google Trends



Fuente: Google Trends < https://trends.google.es/trends/explore?cat=5&date=2008-01-01%202017-11-27&geo=CO&q=%2Fm%2F047fy_g> [citado el 27 de Noviembre de 2017]

Las gráficas arrojadas al realizar las consultas referentes a este framework permiten observar que aproximadamente entre 2009 y 2010 fue en donde se produjo el más alto índice de interés por esta tecnología, a partir de este periodo de tiempo ha decrecido considerablemente hasta la actualidad, por ello también es válido deducir que Primefaces tiene una ventaja frente a esta biblioteca de componentes hasta este punto. La última gráfica referente a la búsqueda a nivel nacional demuestra que el interés en mayor proporción se encuentra en el departamento de Cundinamarca por lo que es muy similar al resultado obtenido con ICEfaces. Con estos resultados se deja en evidencia la decadencia que ha tenido el uso de esta tecnología en los últimos años.

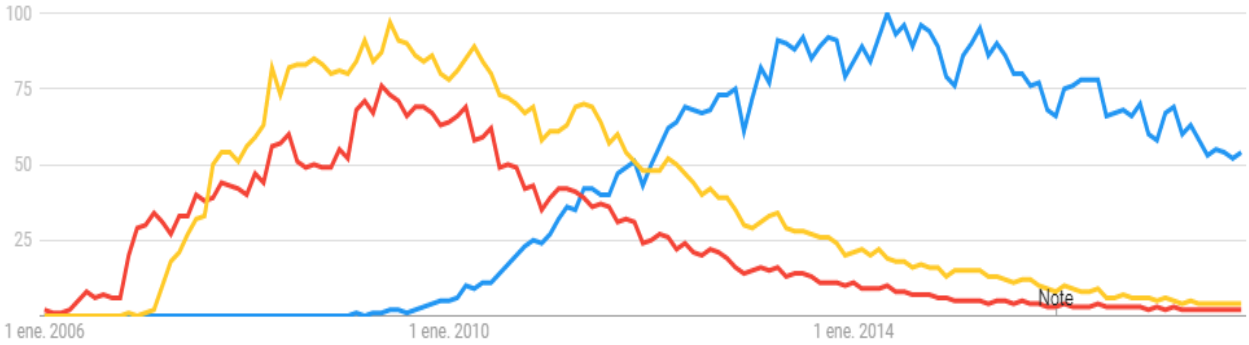
3.2.4 Evaluación de los Índices de Interés.

A continuación se observa la comparación a nivel mundial y nacional de los 3 frameworks utilizando la misma herramienta Google Trends donde se obtuvieron los siguientes resultados.

Figura 23: Colores para identificar cada framework al general la gráfica

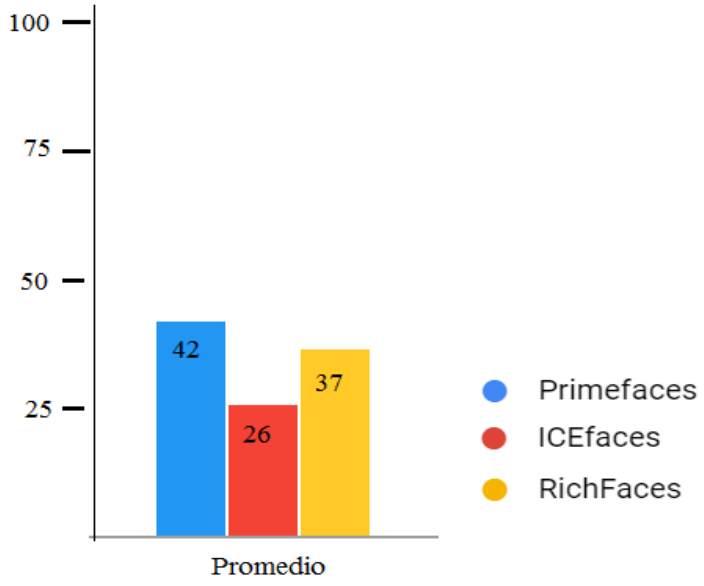
Fuente: Google Trends < https://trends.google.es/trends/explore?cat=5&date=2006-01-01%202017-11-27&q=Primefaces,ICEfaces,RichFaces> [citado el 27 de Noviembre de 2017]

Figura 24: Interés de Búsqueda Web para los 3 Frameworks a nivel mundial Google Trends



Fuente: Google Trends < https://trends.google.es/trends/explore?cat=5&date=2006-01-01%202017-11-27&q=Primefaces,ICEfaces,RichFaces> [citado el 27 de Noviembre de 2017]

Figura 25: Promedio general de Búsqueda Web para los 3 Frameworks a nivel mundial Google Trends

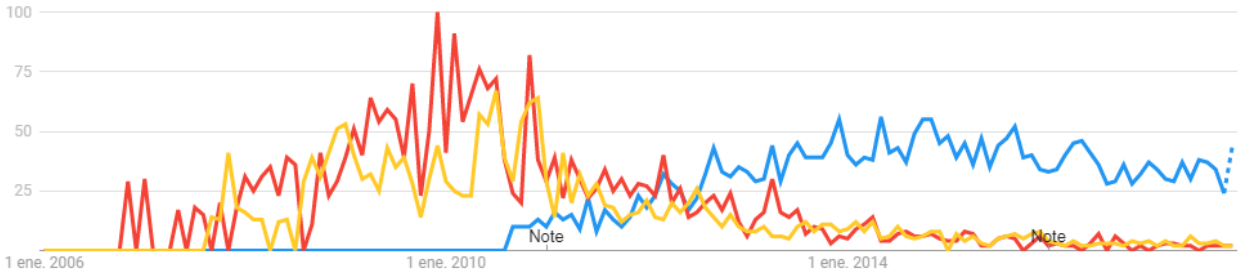


Fuente: Fuente: Google Trends < https://trends.google.es/trends/explore?cat=5&date=2006-01-01%202017-11-27&q=Primefaces,ICEfaces,RichFaces> [citado el 27 de Noviembre de 2017]

Figura 26: Colores para identificar cada framework al general la gráfica

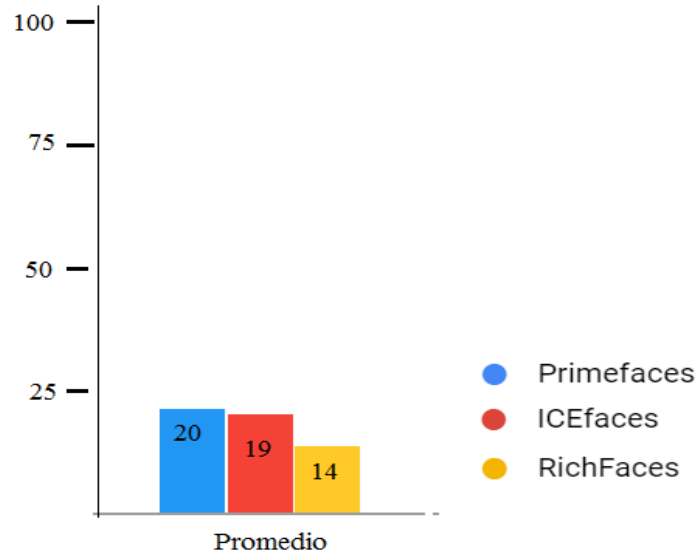
Fuente: Google Trends < https://trends.google.es/trends/explore?cat=5&date=2006-01-01%202017-11-27&geo=CO&q=Primefaces,ICEfaces,RichFaces> [citado el 27 de Noviembre de 2017]

Figura 27: Interés de Búsqueda Web para los 3 Frameworks a nivel nacional Google Trends



Fuente: Google Trends < https://trends.google.es/trends/explore?cat=5&date=2006-01-01%202017-11-27&geo=CO&q=Primefaces,ICEfaces,RichFaces> [citado el 27 de Noviembre de 2017]

Figura 28: Promedio general de Búsqueda Web para los 3 Frameworks a nivel nacional Google Trends



Fuente: Google Trends < https://trends.google.es/trends/explore?cat=5&date=2006-01-01%202017-11-27&geo=CO&q=Primefaces,ICEfaces,RichFaces> [citado el 27 de Noviembre de 2017]

Al analizar la gráfica del promedio obtenido a nivel mundial se puede deducir que Primefaces es actualmente la tecnología preferida para los usuarios o desarrolladores ya que las 2 restantes tuvieron sus mejores niveles de popularidad entre 2007 y 2012 aproximadamente, lo que demuestra que Primefaces es la más joven y por tanto la que más le interesa a los creadores de proyecto con JSF. Seguidamente se encuentra el promedio calculado a nivel nacional donde se puede visualizar que de igual forma Primefaces es la que más es consultada en la web aunque con un promedio muy cercano de ICEfaces, sin embargo en la gráfica de búsqueda web muestra que Primefaces es actualmente la tecnología con mayor popularidad.

Tabla 11: Calificación mediante índices de interés a partir del porcentaje de popularidad a nivel nacional de cada Framework

Criterio	Valor	Significado	Primefaces	ICEfaces	RichFaces
Índices de Interés	3	13,34% - 20%	X	X	X
	2	6,67% - 13,33 %			
	1	0% - 6,66%			

3.3 Disponibilidad de Componentes.

Este criterio está encargado de mostrar todos los componentes de cada uno de los frameworks sobre los cuales se basa este estudio, identificando los usados en la construcción de las vistas del prototipo de administrador de vortal, de igual forma se mostrará la o las etiquetas con la que deben ser usados dichos componente.

3.3.1 Primefaces.

Este framework es el que cuenta con mayor cantidad de componentes de los 3, posee un aproximado de 120 componentes que incluyen, además del conjunto estándar de componentes también otros extras como *HtmlEditors, Charts, date schedule* y un exportador de datos a *Excel, Pdf, Word* entre otros. Esta tecnología utiliza por debajo *jQuery* con sus interesantes widgets, plugins, temas y las interacciones de Ajax. En este marco de trabajo se evita el uso de otros JS/IU frameworks con el fin de tener una alta compatibilidad entre los componentes. También tiene alrededor de 36 temas integrados, además cuenta con las clases necesarias para que se puedan implementar los mismos. Seguidamente se van a mostrar los más usados de acuerdo a su utilidad.

Tabla 12: Componentes de Primefaces con sus etiquetas

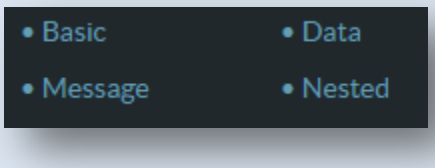
Componentes	Etiquetas
Ajax Core	• Basic • Event • Listener • Counter • Process • Validation • Dropdown • PartialSubmit • Selector • Search • Poll • Fragment • Status • RemoteCommand

Input	• AutoComplete • BooleanButton • Calendar • OneMenu • InputText • InputNumber • Listbox • ManyMenu • MultiSelectListbox • OneRadio • Spinner • Switch • Keyboard • ColorPicker • KeyFilter • TextEditor • Chips • InputTextarea • BooleanCheckbox • CheckboxMenu • Editor • InputGroup • InputMask • ManyButton • ManyCheckbox • OneButton • Signature • Slider • Password • Rating • Inplace • Knob • TriStateCheckbox
Button	• Button • CommandLink • SplitButton • CommandButton • Link

Data	• Carousel • DataGrid • DataList • DataScroller • DataTable • Diagram • PickList • OrderList • Organigram • GMap • DataExporter • Mindmap • Repeat • Ring • Schedule • TagCloud • Tree • TreeTable • HorizontalTree • Timeline
Panel	• Accordion • Dashboard • Fieldset • Grid CSS • Layout • NotificationBar • OutputPanel • Panel • PanelGrid • Ribbon • ScrollPanel • TabView • Toolbar • Wizard

Overlay	• ConfirmDialog • Dialog • LightBox • OverlayPanel • Tooltip • Sidebar
Menu	• Breadcrumb • ContextMenu • Dock • MegaMenu • Menu • Menubar • MenuButton • PanelMenu • SlideMenu • Stack • Steps • TabMenu • TieredMenu
Charts	• Area • Bar • Bubble • Donut • Line • Pie • MeterGauge • OHLC • Animate • Export • Interactive • Live • Static • Zoom • Combined • MultiAxis • Date • Responsive

Messages	• Growl • Messages
Multimedia	• Barcode • Compare • ContentFlow • Cropper • Galleria • GraphicImage • Media • PhotoCam • Switch
File	• Upload • Download
DragDrop	• Draggable • DataGrid • DataTable • Custom
Client Side Validation	• Basic • Bean • Custom • Event

Dialog Framework	

3.3.2 ICEfaces.

Esta biblioteca de componentes cuenta con cerca de 80 componentes, adicional a esto también se puede conectar los componentes ACE (Componentes Avanzados de ICEfaces) ya que estos son la actual generación de componentes open source de ICEfaces. Es considerado un framework que integra funcionalidad AJAX y permite a los desarrolladores Java EE crear aplicaciones RIA (Rich Internet Applications), por ello a continuación de la misma manera se mencionaran la mayoría de los componentes con los que normalmente se desarrollan sus aplicaciones.

Tabla 13: Componentes de ICEfaces con sus etiquetas

Componentes	Etiquetas

ICE Core	icecore:defaultAction icecore:focusManager icecore:idleMonitor icecore:jsEventListener icecore:loadBundle icecore:navigationNotifier icecore:push icecore:redirect icecore:refresh icecore:repeat
Ajax	ace:ajax ace:submitMonitor
Input	ace:autoCompleteEntry ace:colorEntry ace:dateTimeEntry ace:maskedEntry ace:richTextEntry ace:sliderEntry ace:textAreaEntry ace:textEntry

Buttons	ace:buttonGroup ace:checkboxButton ace:checkboxButtons ace:linkButton ace:menuButton ace:pushButton ace:radioButton ace:radioButtons
Selection	ace:autoCompleteEntry ace:comboBox ace:selectMenu ace:simpleSelectOneMenu ace:themeSelect
DataTable	ace:dataTable ace:cellEditor ace:columnGroup ace:dataExporter ace:rowExpansion ace:tableConfigPanel

Lists	ace:list ace:listControl ace:listExporter
Containers	ace:accordion ace:drawerPanel ace:panel ace:panelStack ace:splitPane ace:tabSet ace:tree
Dialogs	ace:confirmationDialog ace:dialog ace:notificationPanel ace:tooltip

Menús	ace:breadcrumbMenu ace:contextMenu ace:menu ace:menuBar ace:menuButton ace:menuSeparator ace:multiColumnSubmenu
Messages	ace:growlMessages ace:message ace:messages
File	ace:dynamicResource ace:fileEntry
Charts	ace:chart

Media	ace:audioPlayer ace:graphicImage ace:videoPlayer
Schedule	ace:schedule ace:scheduleExporter
Client Validators	ace:clientValidateRequired ace:clientValidateLength ace:clientValidateValueRange ace:clientValidateDecimal ace:clientValidatePattern

3.3.3 RichFaces.

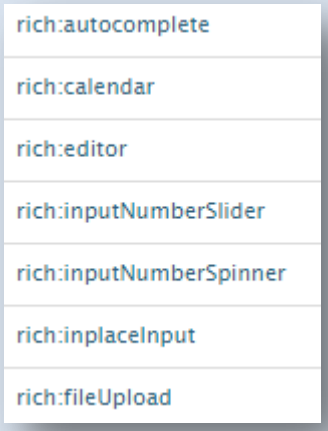
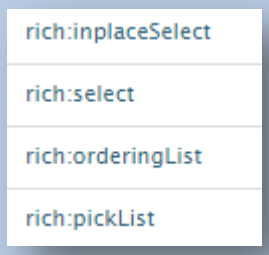
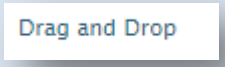
Este framework contiene cerca de 40 componentes ya que esta cantidad no ha crecido en los últimos años, por lo que lo deja muy por debajo de los otros 2, se integra con JSF mediante el empleo de 2 librerías de componentes, la primera *Ajax4jsf*, la cual permite integrar funcionalidad Ajax sin la necesidad de escribir código JavaScript, la segunda es la librería de interfaz de usuario *RichFacesUI*, permitiendo además una integración con otras librerías de componentes. A continuación se mencionaran los componentes con los que dispone.

Tabla 14: Componentes de RichFaces con sus etiquetas

Componentes	Etiquetas
Ajax Action	a4j:ajax a4j:commandButton a4j:commandLink a4j:actionListener a4j:jsFunction a4j:poll a4j:push a4j:param
Ajax Output/Containers	a4j:outputPanel a4j:status a4j:region a4j:mediaOutput a4j:log

Validation	Client Side Validation rich:graphValidator rich:message rich:messages rich:notify
Data Iteration	a4j:repeat rich:dataTable rich:extendedDataTable rich:collapsibleSubTable rich:dataScroller rich:list rich:dataGrid
Tree	rich:tree Tree Adaptors

Output/Panels	rich:chart rich:panel rich:togglePanel rich:tabPanel rich:collapsiblePanel rich:accordion rich:popupPanel rich:progressBar rich:tooltip
Menus	rich:panelMenu rich:toolbar rich:contextMenu rich:dropDownMenu

Inputs	
Selects	
Drag and Drop	

3.3.4 Evaluación de Disponibilidad de Componentes.

Previamente se pudo visualizar los componentes con los que dispone cada una de las bibliotecas, de acuerdo a la cantidad y de los que se necesitan para la construcción del prototipo de vortal, ahora se va a mostrar un comparativo general de los componentes con los que dispone

cada biblioteca respecto a las demás. Tanto ICEfaces como RichFaces proporcionan los siguientes controles de interfaz de componentes:

- Menús.
- Tablas de Datos.
- Selectores.
- Listas.
- Diálogos Modales.
- Pestañas.
- Árboles.
- Efectos visuales.
- Calendario.
- Ventanas emergentes.
- Paneles.
- Mensaje.
- Editor de texto enriquecido.
- Control drag and drop.
- Texto predictivo.
- Combos dinámicos anidados.
- Combo autocompletado.
- Barras de herramientas.
- Barras de progreso.
- Subir ficheros.
- Columnas de tablas ajustables en tamaño.
- Agrupar filas de tablas.
- Ordenar por columnas.
- Google Map.
- Visor de PDFs.

- Visor Flash.
- Visor QuickTime.
- ToopTip.

Los controles de interfaz que sólo proporciona ICEfaces y que no tiene RichFaces son:

- Selector de temas.
- Gráfico de barras.
- Gráfico de tortas.
- Estado de la red.
- Visor Multimedia.
- Descargar fichero.
- Exportar a XLS, CSV.
- Exportar a PDF.

Finalmente Primefaces además de disponer de los componentes anteriormente mencionados incorpora otros muchos adicionales dejando claro que con la gran variedad que contiene y de acuerdo con el fin con el que se vayan a utilizar lo hace claro ventajoso frente a los demás.

Tabla 15: Calificación mediante disponibilidad de componentes para cada Framework

Criterio	Valor	Significado	Primefaces	ICEfaces	RichFaces
Disponibilidad de Componentes	3	Alta	X		
	2	Media		X	
	1	Baja			X

3.4 Usabilidad.

Para este criterio se busca evaluar la mejora o facilidad de uso por medio de los frameworks a nivel de interfaz gráfica para los usuarios lo que vuelve más óptimo el proceso ya que el usuario requerirá de menos esfuerzo para realizar alguna tarea específica, por otro lado se analizará la usabilidad a nivel de código mencionando las ventajas y desventajas con las que cuenta cada uno. Por lo que se mostrarán qué tan completos se encuentran los componentes en lo relacionado a las clases o interfaces de las que disponen para su respectiva utilización y funcionamiento en el entorno de desarrollo del prototipo de administrador de vortal.

3.4.1 Primefaces.

Esta tecnología es la más completa de todas ya que permite al desarrollador aprender a utilizar sus componentes con mucha facilidad de igual forma estos resultan bastante agradables visualmente para el usuario por lo que la interfaz se vuelve mucho más llamativa. A continuación se visualiza la interfaz gráfica del inicio del vortal utilizando esta biblioteca de componentes.

Figura 29: Interfaz gráfica de inicio utilizando Primefaces

Fuente: Autor

Ahora se mencionarán los componentes más importantes en la construcción de la aplicación y básicamente las clases e interfaces que necesitan para su correcto funcionamiento.

- Uso del **PickList.**

Figura 30: Ejemplo del PickList de Primefaces

Fuente: Autor

Este componente representa dos listas una llamada *source* la cual permite mostrar cómo se visualiza en la imagen anterior los roles de un usuario seleccionado a la izquierda y la otra llamada *target* que de igual forma permite mostrar los roles asignados al usuario a la derecha. Esto es posible gracias a que Primefaces cuenta con una clase llamada *DualListModel* la cual se utiliza en el *Bean* o controlador de esta función llamado *RolDualListControl* de modo que debe ser llenado con estas dos listas después de sacar los roles del usuario que se encuentre seleccionado en ese momento.

- Uso de **RequestContext**.

```
RequestContext.getCurrentInstance().execute("mostrar()");
```

Esta es una clase abstracta que permite ser utilizada en el controlador para interactuar con dos funciones *JavaScript* que fueron creadas en la vista para mostrar y ocultar el *PickList* por medio de una propiedad ID contenida en un DIV.

```
RequestContext.getCurrentInstance().execute("PF('ConfirmDialogWidgetVar').show()");
```

Además permite ejecutar los diálogos de confirmación mediante el uso de una propiedad que la hace única frente a las demás bibliotecas la cual es el *widgetVar* referenciada dentro del controlador de la forma anterior.

- Uso de **panelMenu**.

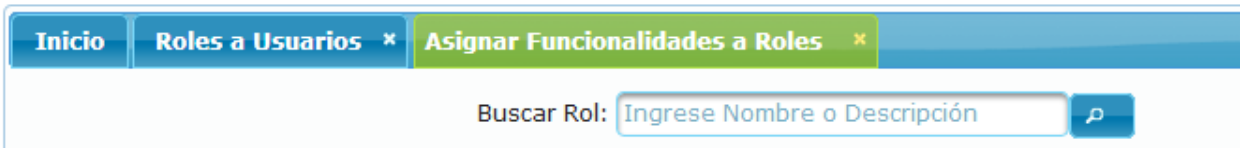
Figura 31: Ejemplo de panelMenu de Primefaces

Fuente: Autor

Este componente facilita la visualización del menú de funcionalidades del usuario logueado generadas recursivamente, siendo posible por la utilización de sus clases en el controlador llamadas: *MenuModel, Submenu, DefaultMenuItem, DefaultMenuModel,* y *DefaultSubMenu*.

- Uso de **tabView**.

Figura 32: Ejemplo de tabView de Primefaces



Fuente: Autor

Por medio de este componente se pueden gestionar el manejo de eventos referentes a las pestañas para cerrarlas (*TabCloseEvent*) o cambiar (*TabChangeEvent*) el enfoque las cuales se activan al dar clic en las funcionalidades generadas en el *panelMenu* y que finalmente se despliegan en el espacio del contenido.

3.4.2 ICEfaces.

Esta biblioteca de componentes para JSF contiene algunos elementos en común con Primefaces aunque sean llamados de forma diferente estos tienen la misma funcionalidad a nivel de interfaz. Hay que mencionar que visualmente es muy similar a los estilos del anterior haciendo referencia al tamaño de los componentes y de igual forma genera una interfaz de usuario muy agradable. A continuación se visualiza la interfaz gráfica del inicio del vortal utilizando esta biblioteca de componentes.

Figura 33: Interfaz gráfica de inicio utilizando ICEfaces

Fuente: Autor

Para la construcción de esta interfaz se tuvieron que hacer modificaciones ya que los componentes que utiliza Primefaces cuenta con una gran variedad de propiedades que vuelve la implementación más óptima, por ello hubo que utilizar uno de los tipos de *menú* de esta biblioteca que mejor se adaptó a la forma en la que originalmente está diseñada con el *panelMenu* de Primefaces. Cabe resaltar que cuando se ejecutó por primera vez el proyecto pidió *.jar* de Primefaces por lo que es válido decir que algunas versiones o esta última versión no son del todo independientes sino que pasan a requerir elementos de otra por la compatibilidad que existe con JSF.

Al modificar las etiquetas que definen los elementos se presenta una serie de complicaciones por lo que de alguna forma afecta el funcionamiento del *Back-End*, específicamente los controladores al no contar con las clases necesarias para su correcto funcionamiento de modo que por su compatibilidad con Primefaces y de acuerdo a la forma con la que se desarrolló el aplicativo se tuvo que recurrir al uso de las mismas con las que esta

trabajó. Para el funcionamiento del *PickList* en Primefaces, aquí se encuentra el componente llamado *List* que a diferencia del primero se debe especificar por aparte la llamada a cada lista lo que vuelve un poco más extenso el código ya que en Primefaces apenas es llamar al *DualListAE* el cual contiene las dos listas.

A nivel de código se resumirá la diferencia entre estas dos bibliotecas con un ejemplo muy básico del uso de los Dialog en la vista.

Figura 34: Ejemplo de dialog de Primefaces e ICEfaces

```
<p:dialog widgetVar="dialog">
    <h:outputText value="PrimeFaces Dialog!" />
</p:dialog>
<a href="#" onClick="dialog.show()">Abrir Dialog</a>

<ace:dialog rendered="#{bean.dialogOpen}">
    <h:outputText value="ICEfaces Dialog!" />
</ace:dialog>
<ace:linkButton action="#{bean.showDialog}">Abrir Dialog</ace:linkButton>
```

Fuente: Autor

Analizando estas líneas de código se puede decir que ICEfaces necesita una de ida y vuelta del lado del servidor y una propiedad de bean administrado, más un oyente de acción para abrir un diálogo. En cambio en PrimeFaces gracias a "*widgetVar*", se puede relacionar el componente a una variable del lado del cliente accesible en *jQuery*. Además, los componentes PrimeFaces manejan muchos de los casos comunes con menos código.

3.4.3 RichFaces.

Este framework o biblioteca de componentes se encuentra con un aspecto llamativo que puede que sea negativo en su interfaces de usuarios en cuanto al tamaño reducido que tienen tanto sus componentes como el tamaño de fuente que incorpora lo que puede llegar a ser una

desventaja para algunos usuarios. A continuación se visualiza la interfaz gráfica del inicio del vortal utilizando esta biblioteca de componentes.

Figura 35: Interfaz gráfica de inicio utilizando RichFaces

Fuente: Autor

Por otro lado haciendo referencia a la forma en la que se implementó el aplicativo a diferencia de ICEfaces y a pesar de que cuenta con menos componentes que este, contiene en su gran mayoría los componentes utilizados en Primefaces con el mismo nombre como es el ejemplo del *PickList* el cual tiene la estructura del código muy similar y el *panelMenu*, pero con el inconveniente de que no cuenta con las clases necesarias para su implementación por lo que fue necesario el uso de las propiedades de oyentes de acción para enviar los Id de las funcionalidades al método en el *Bean*.

3.4.4 Evaluación de Usabilidad.

Para la evaluación por medio de este criterio se tuvieron encuentra dos aspectos, la usabilidad de las interfaces de usuarios más enfocada a la apariencia o la estética y la segunda la

facilidad del uso del código para la construcción de la aplicación web en relación con los demás frameworks.

Tabla 16: Calificación mediante usabilidad para cada Framework

Criterio	Valor	Significado	Primefaces	ICEfaces	RichFaces
Usabilidad	3	Muy Buena	X		
	2	Buena		X	X
	1	Regular			

3.5 Funcionalidad.

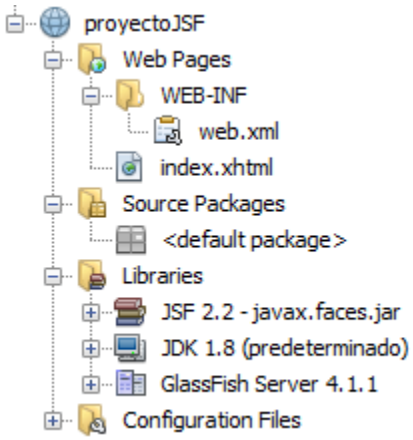
Uno de los aspectos más importantes que determinan la aceptación de un producto de software, es la funcionalidad. Este criterio tiene como propósito evaluar si los Frameworks alcanzan los objetivos para los que fueron diseñados, es decir, si brindan al usuario la utilidad necesaria para acabar con los problemas que ofrece resolver. A continuación se explica de una forma muy general la construcción y funcionamiento del prototipo de vortal, además la forma como se encuentra divido el patrón MVC.

3.5.1 Construcción del prototipo.

En la construcción del prototipo, inicialmente se puede decir que la estructura de una aplicación web en JSF es igual al de una aplicación web tradicional en java, se crea una carpeta con el nombre del proyecto por ejemplo *proyectoJSF* y dentro de esta carpeta se crea la subcarpeta Web Pages donde se contienen todas las subcarpetas y archivos necesarios para la vista de la aplicación y el paquete Source Packages donde se contienen todas las clases y sub paquetes necesarios para la lógica de la aplicación. La diferencia que posee una aplicación web

JSF radica en que se crea una archivo de configuración llamado *web*.xml dentro de la subcarpeta *WEB-INF* donde se establecen una serie de parámetros necesarios para el funcionamiento del Framework, como se puede observar en la figura es un archivo que contiene especificaciones basadas en etiquetas xml.

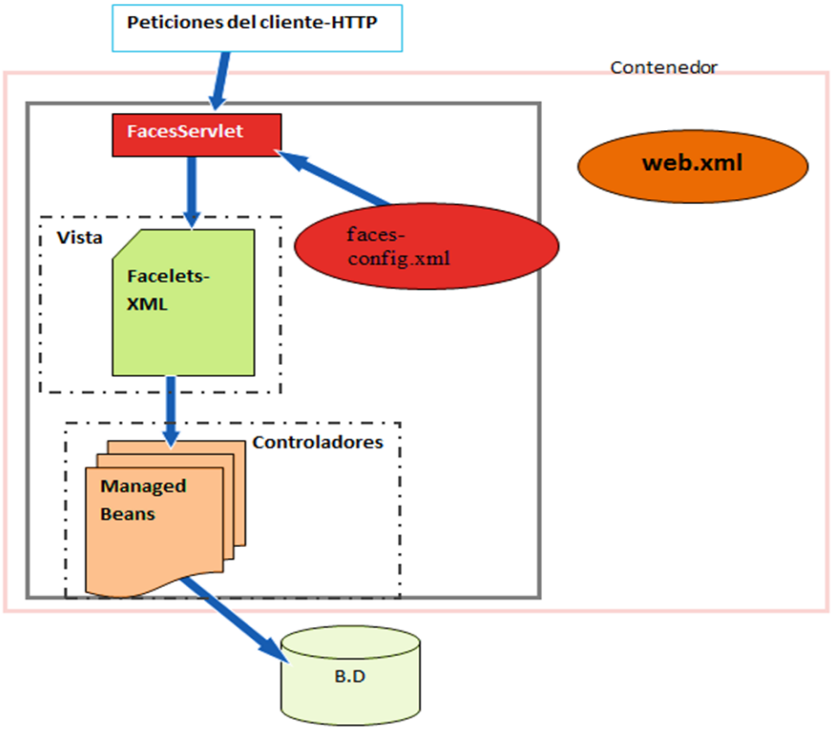
Figura 36: Estructura básica aplicación web JSF



Fuente: Autor

La siguiente figura muestra un bosquejo del modelo de funcionamiento de JSF y cada uno de los elementos fundamentales que intervienen en su ciclo de vida.

Figura 37: Modelo de funcionamiento JSF



Fuente: Autor *(Imagen Editada)*

El archivo de configuración *web.xml* es definido como un descriptor de despliegue para aplicaciones web basadas en J2EE y que contiene el control sobre cómo se debe desplegar la aplicación, estas definiciones son interpretadas por una herramienta de despliegue llamada desplegador que se encarga de dirigir la aplicación con los requisitos especificados.

Figura 38: Configuración del archivo web.xml en JSF

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                             http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
    <context-param>
        <param-name>javax.faces.PROJECT_STAGE</param-name>
        <param-value>Development</param-value>
    </context-param>
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>/faces/*</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>faces/index.xhtml</welcome-file>
    </welcome-file-list>
</web-app>
```

Fuente: Autor

Esta es la configuración del *web.xml* que la aplicación genera por defecto. Puede ser modificada por ejemplo para agregar los parámetros correspondientes a la asignación de un tema específico como es el caso de Primefaces, entre otras configuraciones que se pueden establecer.

Al analizar la estructura del archivo se puede observar la etiqueta *servlet* que sirve para establecer cuál será la interfaz de programación que se encargará de controlar las solicitudes hechas del lado del cliente y la respuesta que obtendrá.

En la etiqueta *servlet-mapping* se le especifica al *servlet* el patrón de todas las URLs (Uniform Resource Locator o Localizador Uniforme de Recursos) de la aplicación a desarrollar que lo podrá llamar.

La etiqueta *sesión-config* permite establecer por medio de la sub etiqueta *sesión-timeout* el tiempo que durará la sesión de un usuario inactiva. Por último la etiqueta *welcome-file-list* contiene una lista de archivos de bienvenida le la aplicación.

La etiqueta *servlet* especificada hace el llamado a la clase *FacesServlet*, la cual se encarga de recibir todas las peticiones HTTP e interactuar con la aplicación, es decir este actúa como intermediario controlando todo tipo de peticiones de servicios y asignando recursos dependiendo de las instrucciones definidas.

Por otro lado los *Facelets* son el lenguaje que utiliza actualmente Java Server Faces para el manejo de elementos en la vista mediante plantillas de estilos HTML.

En la generación de la página JSF por primera vez, el navegador realiza una petición de la determinada URL que desea cargar donde se encuentran los componentes JSF que se quieren mostrar, dicha petición es hecha al motor *FacesServlets* quien se ocupa de cargar el árbol de componentes según la solicitud.

En la construcción de la vista intervienen los *Beans* que ya fueron mencionados anteriormente y que son clases en java de respaldo que controla la interacción entre la interfaz de usuario y el modelo, además contiene propiedades y métodos oyentes de eventos que procesan a dichas propiedades.

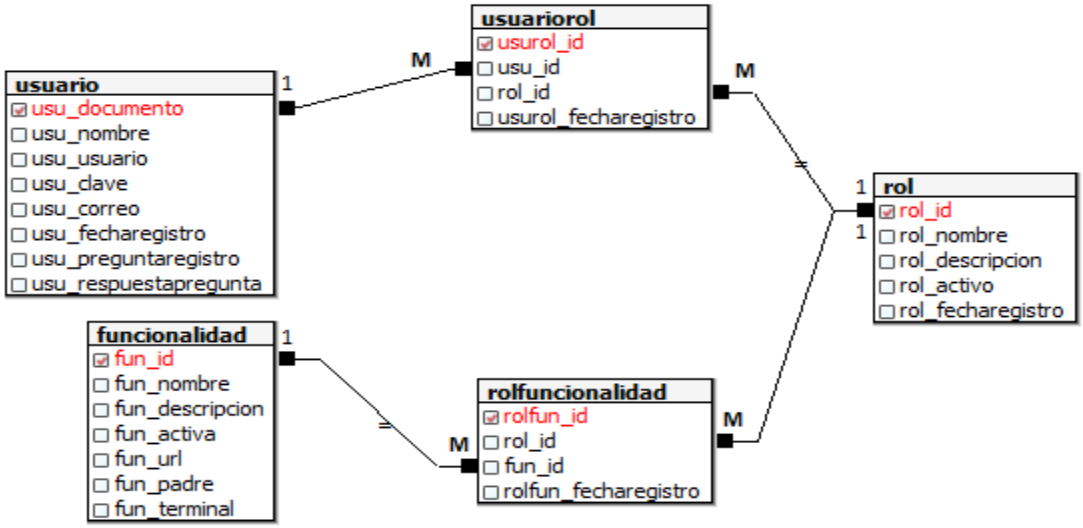
3.5.1.1 Modelo.

Para la construcción del prototipo de administrador de vortal fue necesario la creación de una base de datos con las tablas usuario, rol y funcionalidad con sus respectivas relaciones usuariorol y rolfuncionalidad. A partir de esta base de datos, se crean las funcionalidades con las que se van a trabajar en el prototipo, las cuales son:

1. **Login o logueo de usuarios.**
2. **Control Roles a Usuarios.**
   - Asignar Roles a Usuarios.
   - Eliminar Roles a Usuarios.
3. **Control Funcionalidades a Roles.**
   - Asignar Funcionalidades a Roles.
   - Eliminar Funcionalidades a Roles.

Ahora se mostrará el diseño del modelo entidad – relación.

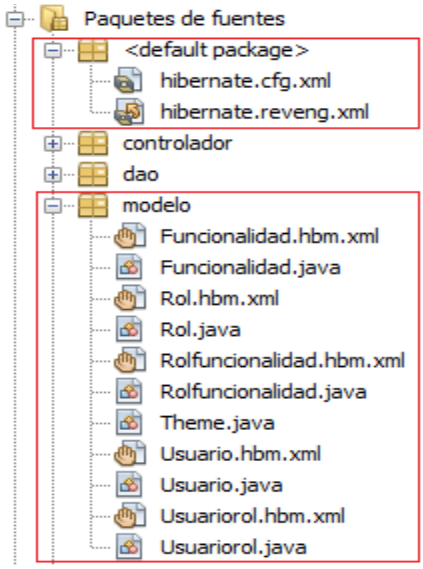
Figura 39: Modelo entidad relación de la base de datos



Fuente: Autor

La API de Hibernate transforma este modelo a uno orientado a objetos convirtiendo cada entidad en un objeto para ser accedidos desde los controladores.

Figura 40: Modelo objeto relacional



Fuente: Autor

Seguidamente se va a visualizar la estructura que tendría todos los archivos generados por hibernate *.hbm.xml* y el *.java*, como ejemplo se tomó el Usuariorol.

Figura 41: Archivo de mapeo de la entidad Usuariorol

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<!-- Generated 8/11/2017 03:24:53 PM by Hibernate Tools 4.3.1 -->
<hibernate-mapping>
    <class name="modelo.Usuariorol" table="usuariorol" schema="vortal" optimistic-lock="version">
        <id name="usurolId" type="int">
            <column name="usurol_id" />
            <generator class="assigned" />
        </id>
        <many-to-one name="rol" class="modelo.Rol" fetch="select">
            <column name="rol_id" not-null="true" />
        </many-to-one>
        <many-to-one name="usuario" class="modelo.Usuario" fetch="select">
            <column name="usu_id" length="24" not-null="true" />
        </many-to-one>
        <property name="usurolFecharegistro" type="timestamp">
            <column name="usurol_fecharegistro" length="22" not-null="true" />
        </property>
    </class>
</hibernate-mapping>
```

Fuente: Autor

Figura 42: La clase de mapeo Usuariorol

```
@Entity
@Table(name="usuariorol"
    ,schema="vortal"
)
public class Usuariorol  implements java.io.Serializable {


     private int usurolId;
     private Rol rol;
     private Usuario usuario;
     private Date usurolFecharegistro;

    public Usuariorol() {
    }

    public Usuariorol(int usurolId, Rol rol, Usuario usuario, Date usurolFecharegistro) {
       this.usurolId = usurolId;
       this.rol = rol;
       this.usuario = usuario;
       this.usurolFecharegistro = usurolFecharegistro;
    }

     @Id


    @Column(name="usurol_id", unique=true, nullable=false)
    public int getUsurolId() {
        return this.usurolId;
    }
    public void setUsurolId(int usurolId) {
        this.usurolId = usurolId;
    }

@ManyToOne(fetch=FetchType.LAZY)
    @JoinColumn(name="rol_id", nullable=false)
    public Rol getRol() {
        return this.rol;
    }

    public void setRol(Rol rol) {
        this.rol = rol;
    }

@ManyToOne(fetch=FetchType.LAZY)
    @JoinColumn(name="usu_id", nullable=false)
    public Usuario getUsuario() {
        return this.usuario;
    }

    public void setUsuario(Usuario usuario) {
        this.usuario = usuario;
    }

    @Temporal(TemporalType.TIMESTAMP)
    @Column(name="usurol_fecharegistro", nullable=false, length=22)
    public Date getUsurolFecharegistro() {
        return this.usurolFecharegistro;
    }

    public void setUsurolFecharegistro(Date usurolFecharegistro) {
        this.usurolFecharegistro = usurolFecharegistro;
    }}
```

Fuente: Autor

Así mismo se construyen cada una de las clases que representan las entidades de la base de datos de esta forma Hibernate permite persistir datos sencillamente desde los controladores.

Para el manejo de internacionalización en JSF el primer paso es crear el archivo de propiedades con los nombres de botones, ventanas, cuadros de diálogo, mensajes y elementos en general utilizados en las páginas del sitio web.

Figura 43: Archivo de configuración español_es.properties

```
#////////////////////////////////////////////////////////////////////////
#Mensajes de la Página de Login
#////////////////////////////////////////////////////////////////////////
loginTitulo                         =   Logueo Usuario Vortal
PanelTitulo                         =   Ingrese usuario y contraseña
OutputLabelUsuario                  =   Usuario
OutputLabelContrasena               =   Contraseña
CommandButtonIniciarSesion          =   Iniciar Sesión
#////////////////////////////////////////////////////////////////////////
#Mensajes de la Página adminVortal
#////////////////////////////////////////////////////////////////////////
adminVortalTitulo                   =   Administrador Vortal
```

Fuente: Autor

Luego se configura en el archivo *faces-config.xml* agregando las líneas.

Figura 44: Configuración de faces-config.xml

```
<application>
    <locale-config>
        <default-locale>es</default-locale>
        <supported-locale>en</supported-locale>
        <supported-locale>es_CO</supported-locale>
        <supported-locale>en_US</supported-locale>
    </locale-config>
    <resource-bundle>
        <base-name>util.espanol_es</base-name>
        <var>msg</var>
    </resource-bundle>
</application>
```

Fuente: Autor

Y por último se utiliza, haciendo referencia a cada etiqueta por su nombre en las páginas por medio de la variable *msg* declarada anteriormente de la siguiente forma.

Figura 45: Uso de la variable msg

```
<p:panel header="#{msg['PanelTitulo']}">
    <h:panelGrid columns="2" cellpadding="5">
        <h:outputLabel value="#{msg['OutputLabelUsuario']}"/>
        <p:inputText value="#{loginControl.usuario.usuUsuario}"
```

Fuente: Autor

Luego al visualizar la página en el *header* y *value* de cada componente aparecerá la frase o palabra que se definió.

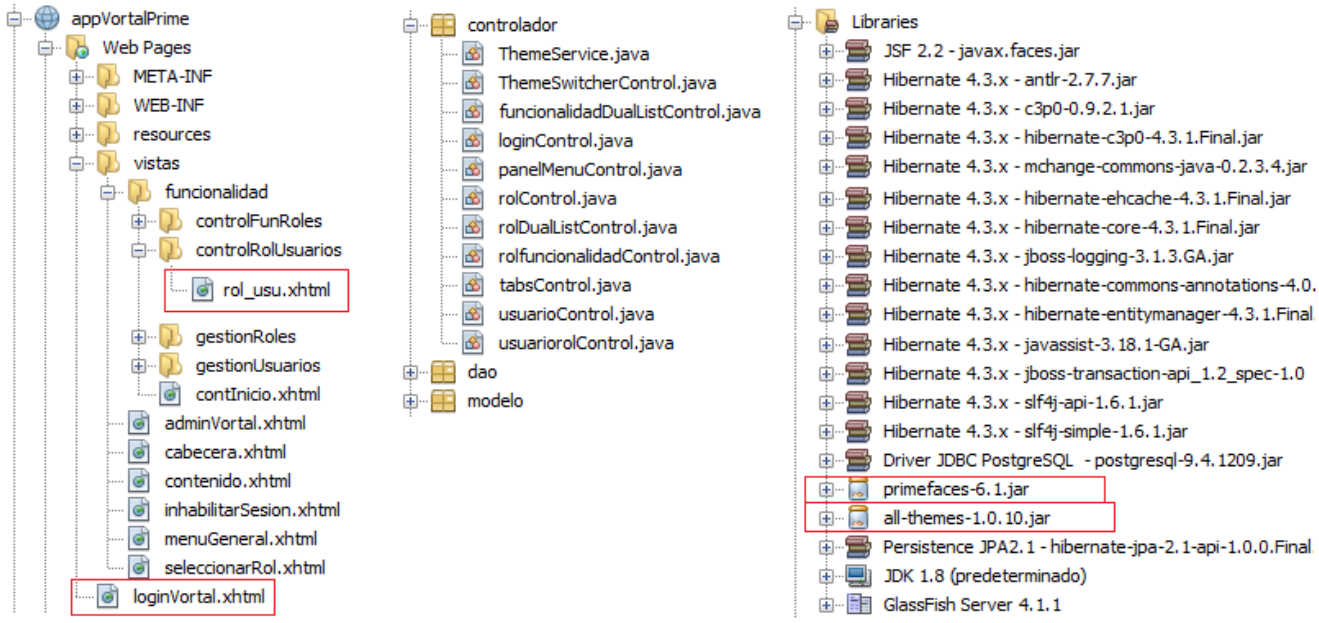
3.5.1.2 La Vista y el Controlador.

En esta sección el propósito es mostrar la forma en la que se desarrollaron las dos funcionalidades con cada uno de los tres frameworks, es decir la modificación se realizó en la parte de *Front-End* utilizando el mismo *Back-End*. De igual forma se mostrará la forma como interactúan las vistas con los controladores. A continuación se muestran las dos funcionalidades construidas por cada uno de los tres framework, las cuales son el login o logueo de usuarios y el control de roles de usuarios.

3.5.1.2.1 Primefaces.

Para trabajar con esta biblioteca de componentes fue necesario crear las siguientes carpetas, juntos con la biblioteca de *primefaces-6.1.jar* suficiente para trabajar con la totalidad del framework lo que lo hace ventajoso frente a los demás ya que aquellos necesitan de otras dependencias para su correcto funcionamiento y opcionalmente se agregó *.jar* de *all-themes.jar* para gestionar los temas de la aplicación.

Figura 46: Archivos necesarios para la construcción de la vista con Primefaces



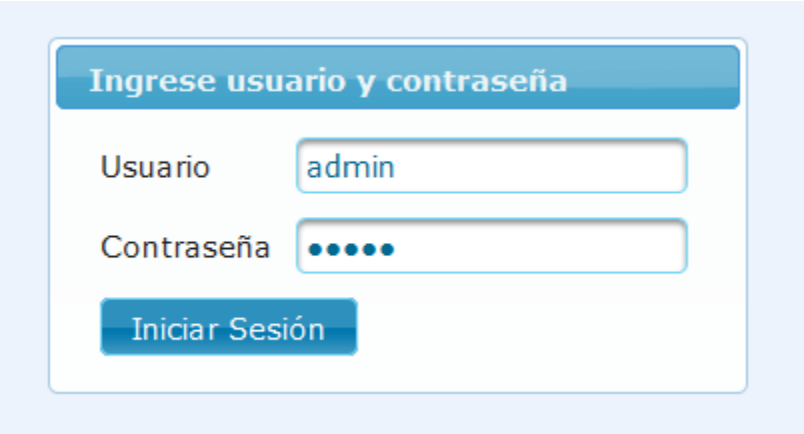
Fuente: Autor

A continuación se muestran las dos funcionalidades construidas mediante el uso de este framework, aclarando que el espacio de nombres está definido por el prefijo ***<p:...***

3.5.1.2.1.1 Funcionalidad de Login.

Al ejecutar la aplicación se encuentra la página de logueo de usuario de la siguiente forma.

Figura 47: Página de logueo de usuarios



Fuente: Autor

El usuario ingresa sus datos y presiona el botón iniciar sesión.

Figura 48: Código del formulario de logueo

```
<p:growl id="growl" showDetail="true" life="6000"/>
<h:form id="formLoginVortal">
    <p:panel header="#{msg['PanelTitulo']}">
        <h:panelGrid columns="2" cellpadding="5">
            <h:outputLabel value="#{msg['OutputLabelUsuario']}"/>
            <p:inputText value="#{loginControl.usuario.usuUsuario}"
                         required="true"
                         id="username"
                         requiredMessage="Debe Ingresar Nombre de Usuario"/>
            <h:outputLabel value="#{msg['OutputLabelContrasena']}"/>
            <p:inputText value="#{loginControl.usuario.usuClave}"
                        required="true"
                        id="password"
                        requiredMessage="Debe Ingresar su Contraseña"/>
            <f:facet name="footer">
                <p:commandButton id="loginButton"
                                 value="#{msg['CommandButtonIniciarSesion']}"
                                 update=":growl"
                                 process="@this, password, username"
                                 action="#{loginControl.autenticar()}"/>
            </f:facet>
        </h:panelGrid>
    </p:panel>
</h:form>
```

Fuente: Autor

Los datos son directamente pasados a un *Bean* de respaldo de la página de logueo llamado *loginControl*.

Figura 49: Bean de respaldo para la página de logueo (loginControl)

```
@Named(value = "loginControl")
@SessionScoped
public class loginControl implements Serializable {

    private Usuario usuario;

    public Usuario getUsuario() {
        return usuario;
    }

    public void setUsuario(Usuario usuario) {
        this.usuario = usuario;
    }

    public loginControl() {
        usuario = new Usuario();
    }
```

Fuente: Autor

Si el usuario presiona el botón *iniciar sesión* con los datos correctos. El *Bean* construye un nuevo objeto usuario con los datos proporcionados y, por medio del atributo value en los elementos *inputText* y password le pasa los valores utilizando lenguaje EL *(value="#{loginControl.usuario.usuUsuario}", #{loginControl.usuario.usuClave})* una vez obtenidos estos datos, el atributo *action* del elemento *commandButton* hace el llamado al método *autenticar* quien se encarga de crear la *sesión* al usuario y enviarlo a la página de selección de rol.

Figura 50: Código de construcción de la sesión en el método autenticar

```
ExternalContext externalContex = FacesContext.getCurrentInstance().getExternalContext();
externalContex.getSessionMap().put("usuario", usuario);
HttpSession sesion = (HttpSession) externalContex.getSession(false);
sesion.setAttribute("listaRolesDeUsuario", usuroles);
if (usuroles.size() == 1) {
    externalContex.redirect(externalContex.getRequestContextPath() + "/vistas/adminVortal.xhtml");
} else {
    externalContex.redirect(externalContex.getRequestContextPath() + "/vistas/seleccionarRol.xhtml");
}
```

Fuente: Autor

Si el usuario presiona el botón *iniciar sesión* con datos incorrectos. El método *autenticar* devuelve un error indicando que el usuario o contraseña son incorrectos.

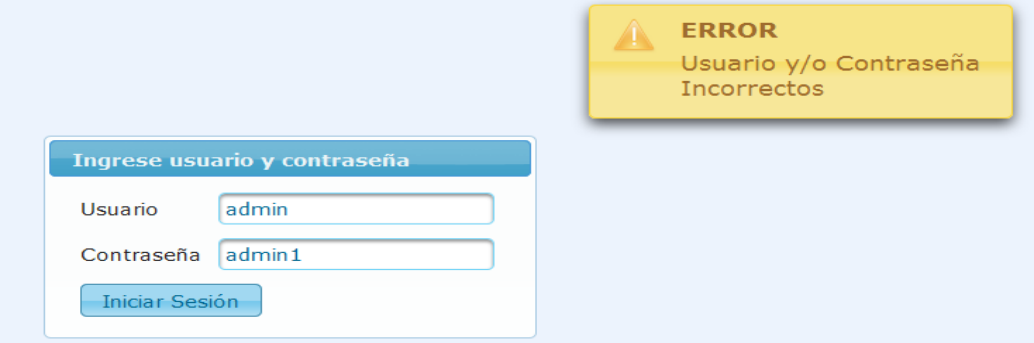
Figura 51: Devolución de error de logueo del método autenticar

```
FacesMessage msg = new FacesMessage(FacesMessage.SEVERITY_WARN, "ERROR",
        "No puede Iniciar Sesión, su Usuario No tiene Roles Asignados");
FacesContext.getCurrentInstance().addMessage(null, msg);
```

Fuente: Autor

En la página de logueo se recibe el mensaje por medio del elemento *growl* quien se encarga de hacerlo visible en pantalla.

Figura 52: Mensaje de error de logueo en pantalla



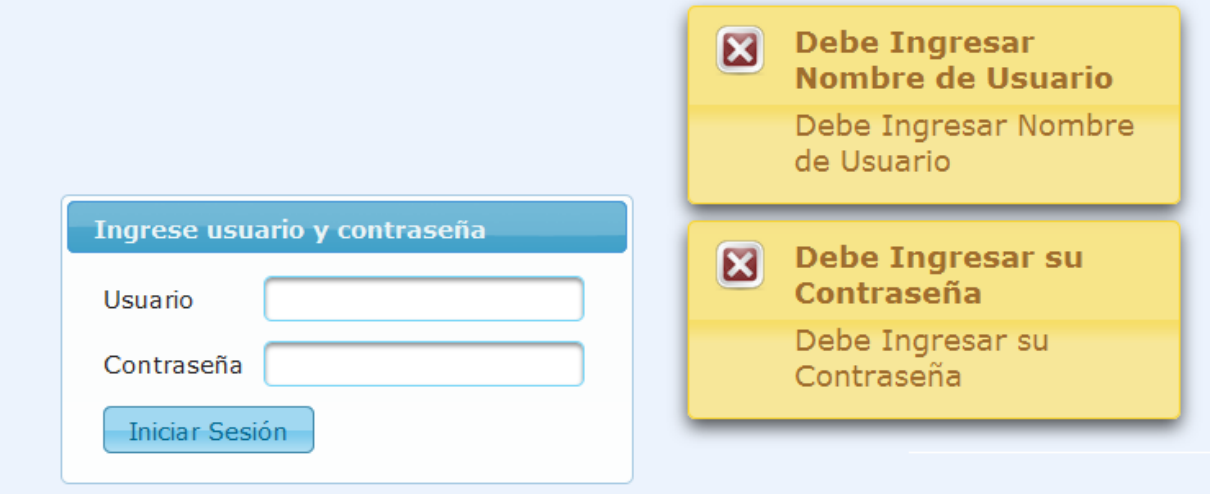
Fuente: Autor

En la validación de campos vacíos en logueo, al presionar el botón *iniciar sesión* se valida si los campos usuario y contraseña contienen información por medio del atributo *process*

del elemento *commandButton* donde se escriben los id de los campos que se desean verificar (*username* y *password*), en los elementos *inputText* y *password* se asignan dos atributos *required* con argumento *true* para saber que no deben ser enviados sin valor y *requiredMessage* donde se escribe el mensaje que se verá en pantalla en caso de querer enviar el elemento nulo.

En pantalla se puede observar que si se intenta enviar el formulario con los elementos vacíos muestra una alerta y no deja continuar con el proceso hasta llenarlos.

Figura 53: Mensajes de error por campos vacíos



Fuente: Autor

Después de loguearse correctamente el usuario es direccionado a la página *seleccionarRol* donde debe seleccionar uno de los roles según los que tenga asignados para poder continuar, este proceso es únicamente con el propósito de cargar las funcionalidades iniciales para el rol que ha sido seleccionado, el rol puede ser cambiado por el usuario después de estar adentro de la página administrador del Vortal.

Figura 54: Página de selección de rol del usuario logueado

Fuente: Autor

La página seleccionar rol consta de un cuadro de diálogo donde un elemento *selectOneMenu* es el encargado de mostrar los roles disponibles al usuario, un botón *continuar* quien se encarga de validar la selección de alguno de los roles disponibles y direccionar hacia la página de inicio del administrador Vortal y un botón *cancelar* que se encarga de volver a la página de logueo y eliminar la sesión.

Figura 55: Elemento selectOneMenu de la página seleccionarRol

```
<p:outputLabel for="selectOneMenuSeleccionarRol" value="Seleccione el Rol con el que Desea Ingresar: "/>
<p:selectOneMenu id="selectOneMenuSeleccionarRol" style="width:155px"
                 required="true" requiredMessage="Seleccione un Rol"
                 value="#{usuariorolControl.nombreRolSeleccionado}">

    <f:selectItem itemLabel="Seleccione un Rol" itemValue=""
                  noSelectionOption="true" />
    <f:selectItems var="usurol" value="#{usuariorolControl.rolesDelUsuario}"
                   itemLabel="#{usurol.rol.rolNombre}" itemValue="#{usurol.rol.rolNombre}"/>

    <p:ajax listener="#{usuariorolControl.actualizaRolSeleccionado}" event="change"/>
</p:selectOneMenu>
```

Fuente: Autor

El atributo *value* del elemento *selectOneMenu* se encarga de actualizar el valor de la variable *nombreRolSeleccionado* en el *Bean* de respaldo *usuariorolControl* la cual almacena el nombre del rol que se selecciona.

Figura 56: Declaración de variable nombreRolSeleccionado, objeto rolSeleccionado y lista de rolesDelUsuario en Bean usuariorolContol

```
private String nombreRolSeleccionado;
private Rol rolSeleccionado;
private List<Usuariorol> rolesDelUsuario;
private int asignados;
private int eliminados;

public String getNombreRolSeleccionado() {
    return nombreRolSeleccionado;
}

public void setNombreRolSeleccionado(String nombreRolSeleccionado) {
    this.nombreRolSeleccionado = nombreRolSeleccionado;
}
```

Fuente: Autor

Cada vez que se selecciona un ítem diferente de la lista de roles, el elemento *Ajax* llama al método *actualizaRolSeleccionado* el cual cambia el valor del objeto *rolSeleccionado.*

Figura 57: Método actualizaRolSeleccionado en Bean usuariorolControl

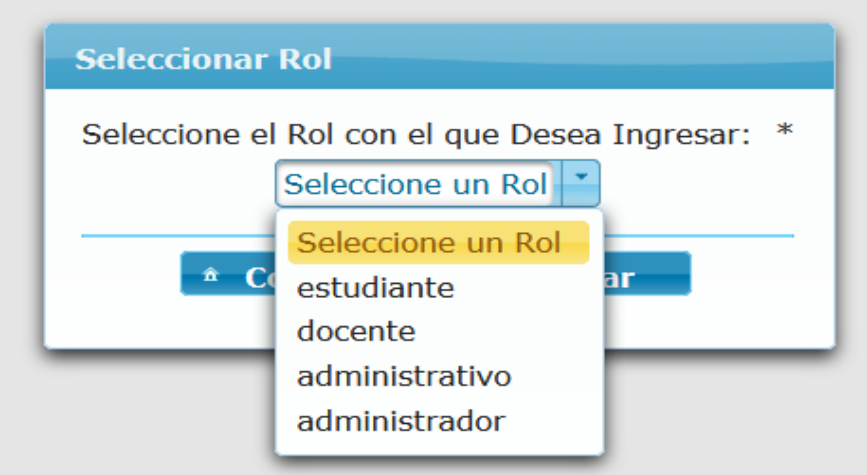
```
public void actualizaRolSeleccionado() {
    if (nombreRolSeleccionado != null) {
        if (rolesDelUsuario != null) {
            for (Usuariorol usurol : rolesDelUsuario) {
                if (usurol.getRol().getRolNombre().equals(nombreRolSeleccionado)) {
                    FacesContext facesContex = FacesContext.getCurrentInstance();
                    HttpSession sesion = (HttpSession) facesContex.getExternalContext().getSession(false);
                    rolSeleccionado = usurol.getRol();
                    sesion.setAttribute("rolSeleccionado", rolSeleccionado);
                }
            }
        } else {
            System.out.print("La Lista de Roles del Usuario llega Null ...");
        }
    } else {
        System.out.print("rolseleccionado Null ...");
    }
}
```

Fuente: Autor

En la vista el elemento *selectItems* tiene los atributos *value* que hace referencia a la lista de roles disponibles para el usuario, la cual en el *Bean* de respaldo es representado por la variable *rolesDelUsuario*; *itemLabel* que sirve para mostrar el nombre en pantalla por medio del objeto a

medida que recorre la lista y el atributo *itemValue* para asignar un valor a cada ítem de la lista, de tal forma que se puede observar.

Figura 58: Aspecto visual del elemento selectItems de la página seleccionarRol



Fuente: Autor

Al presionar el botón *continuar* verifica si hay un elemento seleccionado y si es así pasa a la vista de la página de inicio, de lo contrario muestra el mensaje de validación en la parte superior del panel.

Figura 59: El elemento commandButton para continuar en la página seleccionarRol

```
<p:commandButton value="Continuar" icon="ui-icon-arrowthick-1-e ui-icon-home"
                 id="btnContinuarSesion"
                 style="width: 150px; height: 30px; text-align: center; font-weight: bold"
                 process="@this,selectOneMenuSeleccionarRol"
                 update=":formSeleccionarRol:msgs"
                 action="#{usuariorolControl.cargarInicio()}"/>
```

Fuente: Autor

Figura 60: Mensaje de error de validación para selección de rol

Fuente: Autor

Figura 61: El elemento commandButton para cerrar en la página seleccionarRol

```
<p:commandButton value="Cerrar " icon="ui-icon-close"
                 id="btnCancelarContinuarSesion"
                 style="width: 150px; height: 30px; text-align: center; font-weight: bold"
                 process="@this"
                 actionListener="#{loginControl.cerrarSesion()}"/>
```

Fuente: Autor

Cabe destacar que si el usuario únicamente tiene asignado un rol el sistema pasa de la página de logueo hacia la página de inicio o adminVortal directamente y carga las funcionalidades de ese rol ya que no tiene sentido entrar a la página de selección de rol si solo hay uno.

Para la página de inicio se utilizaron los *facelets template* para poder diseñar la forma en la que quedó dividida la página de adminVortal, quedando así por secciones en la cual en la parte superior se encuentra la cabecera, a la izquierda el menú y a la derecha el espacio para el despliegue de todo el contenido de las funcionalidades con las que se vayan a trabajar. A continuación se puede entender mucho mejor.

Figura 62: Modelo de la página de inicio

```
<h:body>
    <ui:include src="inhabilitarSesion.xhtml"/>
    <div id="top" class="top">
        <ui:include src="cabecera.xhtml"/>
    </div>
    <div id="menu_contenido">
        <div id="left">
            <ui:include src="menuGeneral.xhtml"/>
        </div>
        <div id="content">
            <ui:include src="contenido.xhtml"/>
        </div>
    </div>
</h:body>
```

Fuente: Autor

Figura 63: Vista de página de inicio con Primefaces

Fuente: Autor

- **La cabecera**: Contiene el logo del administrador Vortal un mensaje de bienvenida para el usuario logueado y una barra que contiene un menú de selección para cambiar de rol igual al visto en la página *seleccionarRol* y un menú de opciones de usuario donde se cierra la sesión.

- **El menú**: Es construido por el *panelMenu* de funcionalidades recursivo. Por cada rol seleccionado las funcionalidades pueden variar y ser cargadas por medio de *Ajax*.
- **El contenido:** Es donde se abren las pestañas con el contenido necesario dependiendo de la funcionalidad seleccionada por el usuario.

El código que hace posible la visualización del menú de las funcionalidades tiene la siguiente estructura.

Figura 64: Código que pinta el menú de funcionalidades en pantalla.

```
<h:form id="formMenuGeneral">
    <p:outputPanel>
        <h1 style="color: lightgreen; background-color: dimgray" align="center" >Menú de Funcionalidades</h1>
        <p:panelMenu model="#{panelMenuControl.menuModel}" style="width:320px">
        </p:panelMenu>
    </p:outputPanel>
</h:form>
```

Fuente: Autor

El *Bean* de respaldo que permite llevar a cabo la lógica para la construcción del menú se llama *panelMenuControl* y tiene los métodos *sacaFuncRaiz(), crearMenu(), sacaFuncHijas(recibe la funcionalidad)* y el *menuRecursivo(recibe la funcionalidad y un submenu)*.

Figura 65: Bean de respaldo panelMenuControl

```
@Named(value = "panelMenuControl")
@SessionScoped
public class panelMenuControl implements Serializable {

    @Inject
    private tabsControl tabControl;

    private List<Rolfuncionalidad> listaFuncionalidades;
    private MenuModel menuModel;
```

Fuente: Autor

Figura 66: Método crearMenu recursivodel Bean de respaldo panelMenuControl

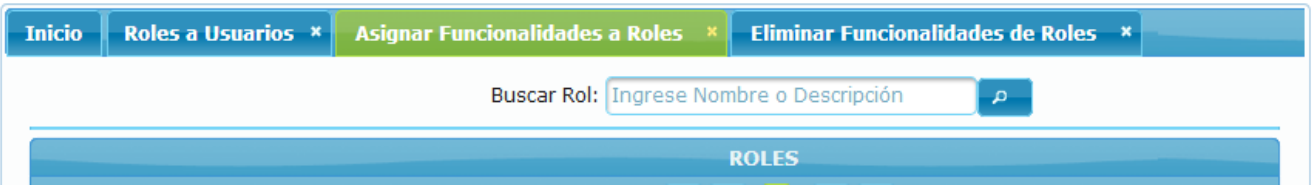
```
public void crearMenu() {
    FacesContext facesContex = FacesContext.getCurrentInstance();
    HttpSession sesion = (HttpSession) facesContex.getExternalContext().getSession(false);
    Rol rol = (Rol) sesion.getAttribute("rolSeleccionado");
    menuModel = new DefaultMenuModel();
    rolfuncionalidadDao rolfuncDao = new rolfuncionalidadDao();
    listaFuncionalidades = rolfuncDao.listarFuncionalidadesDelRol(rol);
    sesion.setAttribute("listaFuncionalidades",listaFuncionalidades);
    System.out.println("listaFuncionalidadesDelRol : " + listaFuncionalidades);

    System.out.println(" .......... crearMenu ");
    MenuModel nuevoMenuModel = new DefaultMenuModel();
    List<Funcionalidad> listFuncRaiz = sacaFuncRaiz();
    System.out.println(" .......... listFuncRaiz --> " + listFuncRaiz);
    for (Funcionalidad funcRaizTemp : listFuncRaiz) {
        if (funcRaizTemp.isFunTerminal()) {
            System.out.println(" .......... if ");
            DefaultMenuItem item = new DefaultMenuItem();
            item.setValue(funcRaizTemp.getFunNombre());
            System.out.println(" .......... funcRaizTemp.getFuncNombre() " + item.getValue());
            menuModel.addElement(item);
        } else {
            System.out.println(" .......... else ");
            DefaultSubMenu submenu = new DefaultSubMenu();
            submenu.setLabel(funcRaizTemp.getFunNombre());
            menuModel.addElement(submenu);
            menuRecursivo(funcRaizTemp, submenu);
        }
    }
}
```

Fuente: Autor

Con este método se llena el objeto *menuModel* el cual lleva todas las funcionalidades correspondientes al rol que inicialmente fue bajado de la sesión para así luego de ser llamado desde el componente sea pintado en la página.

Ahora para el manejo de contenido por pestañas es necesario que cuando se selecciona una funcionalidad se construye una pestaña con su nombre y contenido donde se llevan a cabo los procesos correspondientes a dicha funcionalidad.

Figura 67: Vista de contenidos por pestañas



Fuente: Autor

Cada vez que es seleccionada una funcionalidad terminal del menú se ejecuta un método llamado *agregarNewTab* del *Bean* de respaldo llamado *tabsControl*.

Figura 68: Método agregarNewTab del Bean de respaldo tabsControl

```
public void agregarNewTab(String funid) {
    System.out.println("agregarNewTab Id Fun: " + funid);
    Funcionalidad func = busFunc(funid);
    if (func != null) {
        System.out.println("Funcionalidad: " + func.getFunId());
        boolean titRep = false;
        if (!tabs.isEmpty()) {
            for (Funcionalidad tabTemp : tabs) {
                if (tabTemp.getFunId().equals(func.getFunId())) {
                    titRep = true;
                    activeIndex = tabs.indexOf(tabTemp) + 1;
                }
            }
        }
        if (!titRep) {
            tabs.add(func);
            activeIndex = tabs.size();
        } else {
            System.out.print("Repetido...  " + func.getFunId());
        }
    }
}
```

Fuente: Autor

La manera de pintar las pestañas en pantalla con sus respectivos nombres es el siguiente.

Figura 69: Código que permite visualizar las pestañas

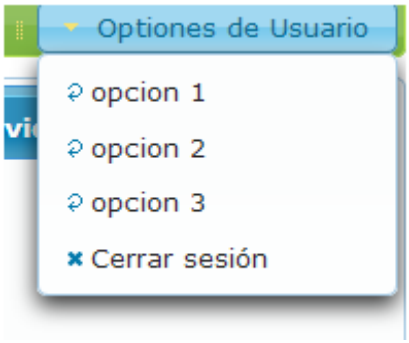
```
<h:form id="formPestanas">
    <p:tabView id="tabViewFunc" activeIndex="#{tabsControl.activeIndex}"
               cache="false" orientation="top" scrollable="true">
        <p:tab title="Inicio" closable="false" id="inicio" >
            <p:scrollPanel style="width: 890px;height: 720px;border: none;"
                           mode="native" >
                <ui:include src="funcionalidad/contInicio.xhtml"/>
            </p:scrollPanel>
        </p:tab>
        <c:forEach items="#{tabsControl.tabs}" var="tab">
            <p:tab id="tabFunc_#{tab.funId}" title="#{tab.funNombre}" closable="true">
                <ui:include src="/vistas/funcionalidad/#{tab.funUrl}.xhtml"/>
            </p:tab>
        </c:forEach>
        <p:ajax event="tabClose" listener="#{tabsControl.onTabCloseAction}"
                update=":formPestanas:tabViewFunc"/>
        <p:ajax event="tabChange" listener="#{tabsControl.onTabChangeAction}"/>
    </p:tabView>
</h:form>
```

Fuente: Autor

El elemento *forEach* itera sobre la lista de pestañas que se actualiza por medio de la variable *tabs* del *Bean* de respaldo *tabsControl* y pinta en pantalla cada *tab* con su respectivo título e incluye su contenido.

Para salir de la aplicación se ubica la opción en la parte superior derecha de la página de *adminVortal* o la página de inicio donde se encuentran una serie de opciones de usuario en la que la única que tiene funcionalidad es la de *cerrar sesión*.

Figura 70: Opción de cierre de sesión de la aplicación



Fuente: Autor

El código que permite visualizar la opción de cierre de sesión es el siguiente.

Figura 71: Código de opción de cierre de sesión

```
<p:menuButton value="Optiones de Usuario">
    <p:menuitem value="opcion 1" actionListener="#{toolbarView.save}" update="messages"
                icon="ui-icon-arrowrefresh-1-w" style="color: white"/>
    <p:menuitem value="opcion 2" actionListener="#{toolbarView.update}" update="messages"
                icon="ui-icon-arrowrefresh-1-w" style="color: white"/>
    <p:menuitem value="opcion 3" actionListener="#{toolbarView.delete}" ajax="false"
                icon="ui-icon-arrowrefresh-1-w" style="color: white"/>
    <p:menuitem value="Cerrar sesión" action="#{loginControl.cerrarSesion()}"
                icon="ui-icon-close" style="color: white"/>
</p:menuButton>
```

Fuente: Autor

Al presionar la opción *cerrar sesión* el atributo *action* del subelemento *menuitem* contenido en el elemento *menuButton* invoca al método *cerrarSesion* del *Bean* de respaldo *loginControl*, quien se encarga de invalidar la sesión creada por el usuario.

Figura 72: Método cerrar sesión del Bean de respaldo loginControl

```
public void cerrarSesion() {
    try {
        ExternalContext externalContex = FacesContext.getCurrentInstance().getExternalContext();
        HttpSession sesion = (HttpSession) externalContex.getSession(false);
        externalContex.getSessionMap().clear();
        sesion.invalidate();
        externalContex.invalidateSession();
        externalContex.redirect(externalContex.getRequestContextPath() + "/loginVortal.xhtml");
    } catch (Exception e) {

    }
}
```

Fuente: Autor

3.5.1.2.1.2 Funcionalidad de Control de Roles a Usuarios.

Esta funcionalidad permite asignar y eliminar roles a un usuario seleccionado de la lista mostrada por *dataTable* haciendo uso del componente *PcikList* mencionado anteriormente. Para ello se presiona en el menú en la funcionalidad donde dice *Roles a Usuarios*.

Figura 73: DataTable con la lista de usuarios

Fuente: Autor

Ahora se observa el diseño de la vista que genera este resultado.

Figura 74: Código del dataTable que lista los usuarios de la base de datos

```
<p:dataTable var="usu" value="#{usuarioControl.listaUsuarios}" rows="10" widgetVar="widgetDataTableUsuarios"
             paginator="true" id="dataTableUsuarios" rowKey="#{usu.usuDocumento}" style=""
             paginatorTemplate="{CurrentPageReport} {FirstPageLink} {PreviousPageLink} {PageLinks}
             {NextPageLink} {LastPageLink} {RowsPerPageDropdown}"
             rowsPerPageTemplate="5,10,15,20,25,30" emptyMessage="No hay datos para mostrar"
             selectionMode="single" selection="#{usuarioControl.usuarioSeleccionado}">
    <f:facet name="header">
        <p:outputPanel>
            <h:outputText value="USUARIOS" />
        </p:outputPanel>
    </f:facet>
    <p:column headerText="Documento"
              style="width: 10%;">
        <h:outputText value="#{usu.usuDocumento}" />
    </p:column>
    <p:column headerText="Nombre"
              style="width: 18%;">
        <h:outputText value="#{usu.usuNombre}" />
    </p:column>
    <p:column headerText="Usuario"
              style="width: 18%;">
        <h:outputText value="#{usu.usuUsuario}" />
    </p:column>
    <p:ajax  event="rowSelect"
             listener="#{rolDualListControl.rolesAsignarEliminarUsuario(usuarioControl.usuarioSeleccionado)}"
             update=":formPestanas:tabViewFunc:pgrid"/>
</p:dataTable>
```

Fuente: Autor

La propiedad *value="#{usuarioControl.listaUsuarios}"* de la etiqueta del componente *dataTable* es la que trae todos los datos de los usuarios desde el Bean *usuarioControl*.

Figura 75: Atributos del Bean usuarioControl

```
@Named(value = "usuarioControl")
@SessionScoped

public class usuarioControl implements Serializable {

    private List<Usuario> listaUsuarios;
    private Usuario usuarioSeleccionado;
    private String busqueda;
```

Fuente: Autor

En la parte inferior se encuentra un elemento *Ajax* que lo que hace básicamente es que a través de la propiedad del oyente mediante el método *rolesAsignarEliminarUsuario* en el *Bean* de *rolDualListControl* permite llenar el *dualListAE* para el usuario que se vaya seleccionando para así luego con una función JavaScript pueda ser mostrado y ocultado según se van seleccionando.

Figura 76: Contenido del método rolesAsignarEliminarUsuario para llenar el dualListAE

```
FacesContext facesContex = FacesContext.getCurrentInstance();
HttpSession sesion = (HttpSession) facesContex.getExternalContext().getSession(false);
sesion.setAttribute("rolesDisponibles", rolesDisponibles);
sesion.setAttribute("rolesAsignados", rolesAsignados);
dualListAE.setTarget(rolesAsignados);
dualListAE.setSource(rolesDisponibles);
RequestContext.getCurrentInstance().execute("mostrar()");
```

Fuente: Autor

Figura 77: Código del pickList llenado por el dualListAE

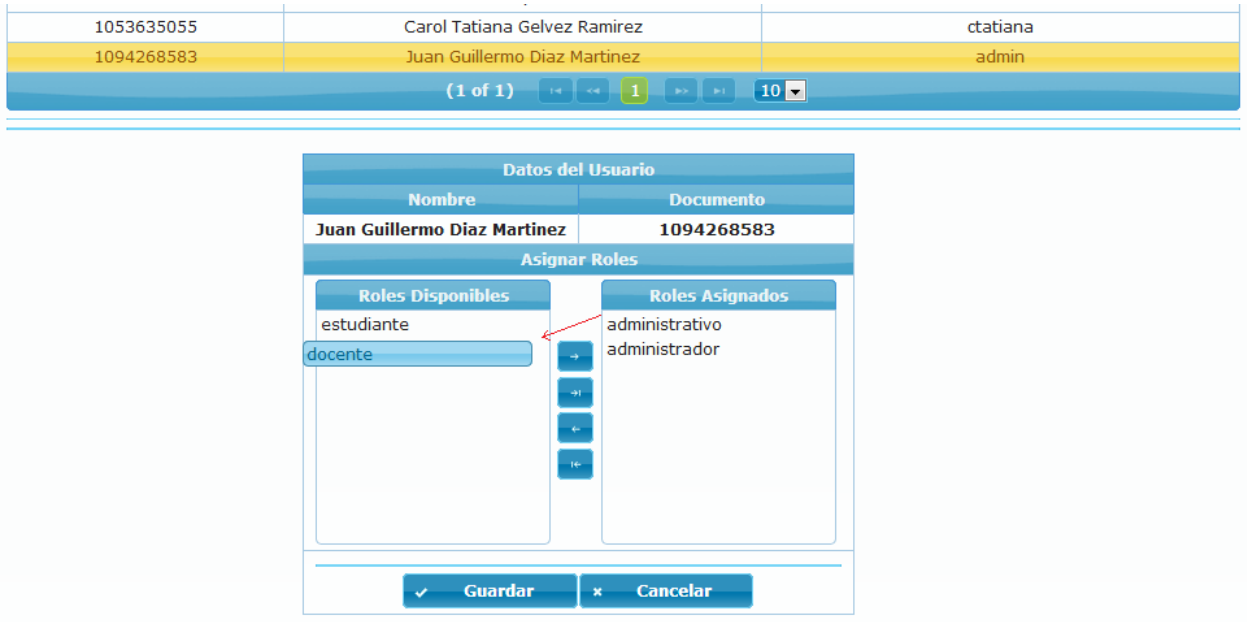
```
<p:row>
    <p:column style="font-weight: bold; text-align: center">#{usuarioControl.usuarioSeleccionado.usuNombre}</p:column>
    <p:column style="font-weight: bold; text-align: center">#{usuarioControl.usuarioSeleccionado.usuDocumento}</p:column>
</p:row>
<p:row>
    <p:column colspan="4" styleClass="ui-widget-header" style="text-align: center">Asignar Roles</p:column>
</p:row>
<p:row>
    <p:column colspan="2">
        <p:pickList id="plAsignarRoles" value="#{rolDualListControl.dualListAE}" var="rol"
                    itemLabel="#{rol.rolNombre}" itemValue="#{rol.rolNombre}">
            <f:facet name="sourceCaption">Roles Disponibles</f:facet>
            <f:facet name="targetCaption">Roles Asignados</f:facet>
        </p:pickList>
    </p:column>
</p:row>
```

Fuente: Autor

Después de haber seleccionado un usuario se muestran las listas con los roles que tiene asignados a la derecha y los que tiene disponibles para asignar a la izquierda, además se pueden realizar modificaciones bien sea arrastrando los roles o utilizando las flechas ubicadas en medio

de las dos listas de roles. Para que finalmente al dar clic en *Guardar* los cambios se registren directamente en la base de datos para el usuario que se encuentra seleccionado en ese momento.

Figura 78: Modificaciones de los roles del usuario Juan Guillermo Diaz



Fuente: Autor

El botón *Guardar* lleva al método *verificarRolesUsuarios* la lista de *source* que son los enviados en la izquierda y *target* los que van por la derecha, además se lleva el usuario seleccionado al que se le debe validar para realizar la respectiva operación. En caso de éxito en la operación se llama finalmente al método *asignarEliminarRolesUsuario* en el *Bean usuariorolControl* para terminar con la operación mostrando un mensaje de éxito.

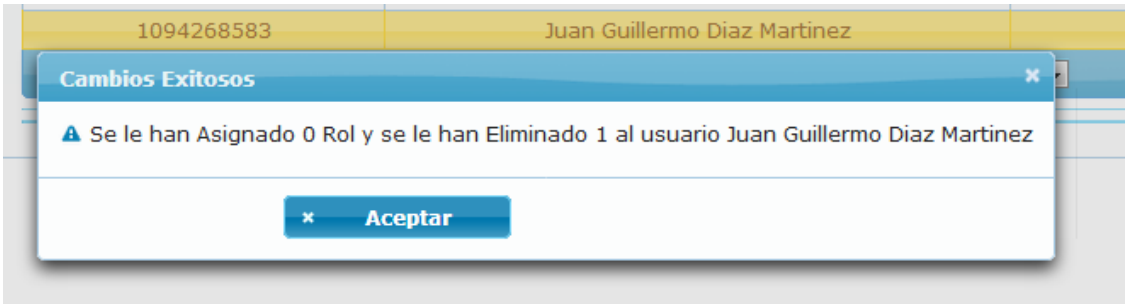
Figura 79: Botón Guardar para la verificación de las listas de roles del usuario

```
<p:commandButton id="btnAsignarRoles" value="Guardar" icon="ui-icon-check"
                 style="width: 150px; height: 30px; text-align: center; font-weight: bold"
                 actionListener="#{usuariorolControl.verificarRolesUsuario(rolDualListControl.dualListAE.source,
                                                                           rolDualListControl.dualListAE.target,
                                                                           usuarioControl.usuarioSeleccionado)}"/>
```

Fuente: Autor

Una vez guardados los cambios muestra un mensaje como el siguiente contabilizando los cambios de inserción y eliminación al usuario.

Figura 80: Diálogo de confirmación exitoso al guardar los cambios

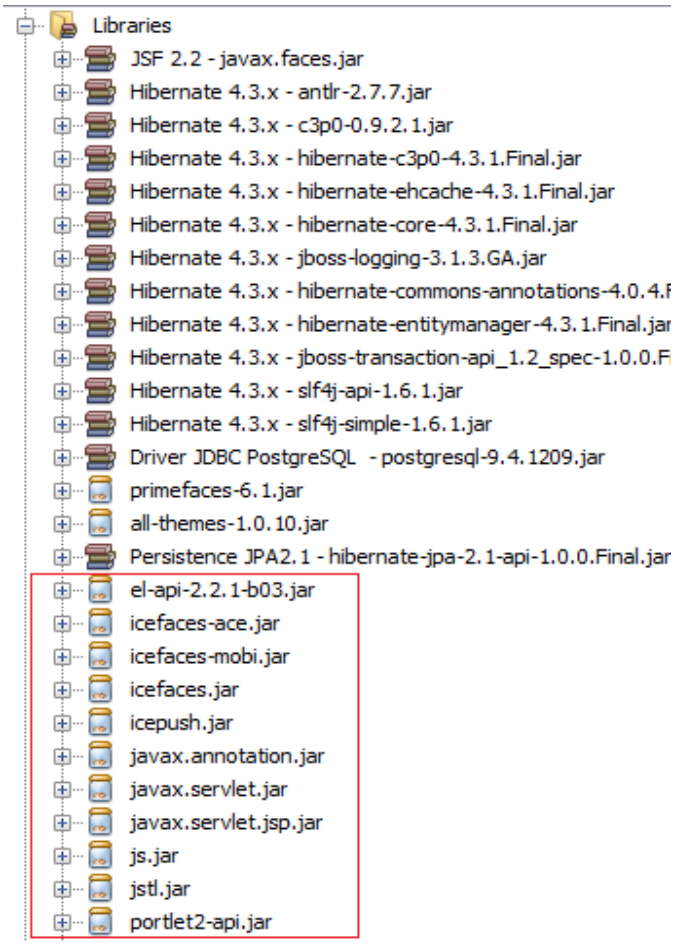


Fuente: Autor

3.5.1.2.2 ICEfaces.

Para el uso de esta biblioteca de componentes fue necesario agregar los siguientes archivos *.jar* de la versión 4.2.0 incluso también se tuvo que agregar las bibliotecas de Primefaces ya que al correr la aplicación requería de estas. Una desventaja notable que al igual que RichFaces requiere de varias dependencias para poder funcionar correctamente de modo que ya vienen contenidas en su paquete.

Figura 81: Librerías necesarias para trabajar con ICEfaces



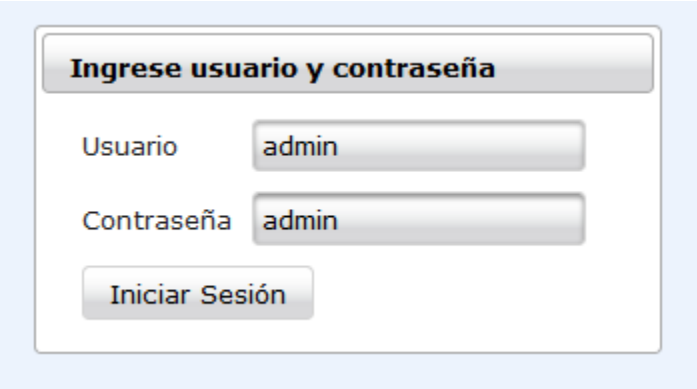
Fuente: Autor

A continuación se muestran las dos funcionalidades construidas mediante el uso de este framework, aclarando que el espacio de nombres está definido por los prefijos **<ace:...** e **<icecore:...**

3.5.1.2.2.1 Funcionalidad de Login.

Al ejecutar la aplicación con ICEfaces se encuentra la página de logueo de usuario de la siguiente forma.

Figura 82: Página de logueo de usuarios



Fuente: Autor

El usuario ingresa sus datos y presiona el botón iniciar sesión.

Figura 83: Código del formulario de logueo

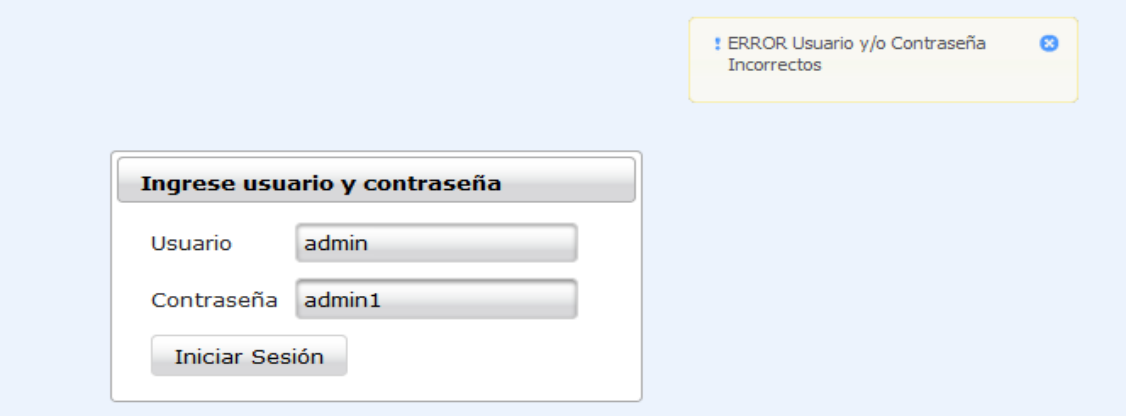
```
<ace:growlMessages id="growl" showDetail="true" />
<h:form id="formLoginVortal">
    <ace:panel header="#{msg['PanelTitulo']}">
        <h:panelGrid columns="2" cellpadding="5">
            <h:outputLabel value="#{msg['OutputLabelUsuario']}"/>
            <ace:textEntry value="#{loginControl.usuario.usuUsuario}"
                           required="true"
                           id="username"
                           requiredMessage="Debe Ingresar Nombre de Usuario"/>
            <h:outputLabel value="#{msg['OutputLabelContrasena']}"/>
            <ace:textEntry value="#{loginControl.usuario.usuClave}"
                          required="true"
                          id="password"
                          requiredMessage="Debe Ingresar su Contraseña"/>
            <f:facet name="footer">
                <p:commandButton id="loginButton"
                                 value="#{msg['CommandButtonIniciarSesion']}"
                                 update=":growl"
                                 process="@this, password, username"
                                 action="#{loginControl.autenticar()}"/>
            </f:facet>
        </h:panelGrid>
    </ace:panel>
</h:form>
```

Fuente: Autor

Ahora hay que tener presente que el trabajo de las vistas consiste en modificar solo la parte del *Front-End*, al igual que en Primefaces los datos son directamente pasados a un *Bean* de respaldo de la página de logueo llamado *loginControl* para realizar la autenticación.

Al ingresar usuario y/o contraseña incorrectos se visualiza el mensaje de la siguiente forma.

Figura 84: Mensaje de error de usuario y/o contraseña



Fuente: Autor

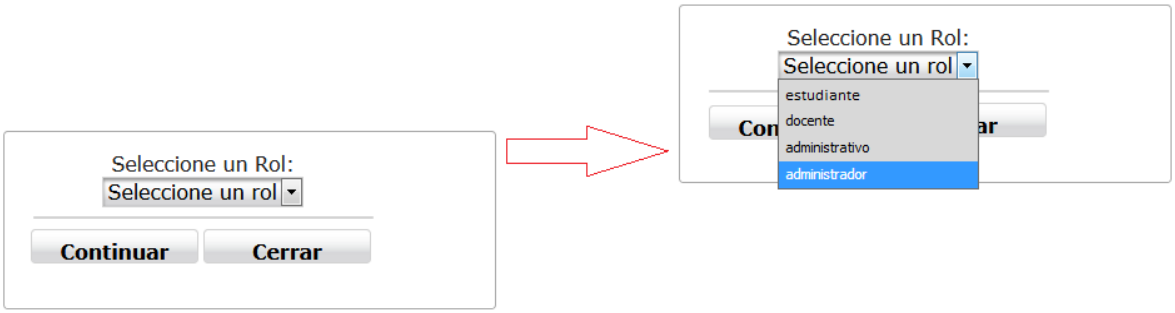
Para cuando los campos del formulario van vacíos se observa lo siguiente.

Figura 85: Pantalla de error campos vacíos

Fuente: Autor

Después de loguearse correctamente el usuario es direccionado a la página *seleccionarRol* donde debe seleccionar uno de los roles según los que tenga asignados para poder continuar.

Figura 86: Página de selección de rol



Fuente: Autor

Figura 87: Código de la página selección de rol

```
<ace:panel id="panel" style="text-align:center; width: 430px; align-content: center;">
    <h:panelGrid  columns="1" cellpadding="2" style="text-align: center;">
        <h:panelGroup>
            <ace:simpleSelectOneMenu id="selectProvince" value="#{usuariorolControl.nombreRolSeleccionado}"
                                     label="Seleccione un Rol:" labelPosition="top">
                <f:selectItem itemValue="" itemLabel="Seleccione un rol" />
                <f:selectItems var="usurol" value="#{usuariorolControl.rolesDelUsuario}"
                               itemLabel="#{usurol.rol.rolNombre}" itemValue="#{usurol.rol.rolNombre}"/>
                <ace:ajax listener="#{usuariorolControl.actualizaRolSeleccionado}" event="valueChange"/>
            </ace:simpleSelectOneMenu>

                <p:separator/>

                <ace:pushButton value="Continuar"
                                id="btnContinuarSesion"
                                style="width: 150px; height: 30px; text-align: center; font-weight: bold"
                                action="#{usuariorolControl.cargarInicio()}"/>
                <ace:pushButton value="Cerrar Sesion"
                                id="btnCancelarContinuarSesion"
                                style="width: 150px; height: 30px; text-align: center; font-weight: bold"
                                actionListener="#{loginControl.cerrarSesion()}"/>
        </h:panelGroup>
    </h:panelGrid>
</ace:panel>
```

Fuente: Autor

Ahora se mostrará la página de *adminVortal* o la página de inicio generada con este framework.

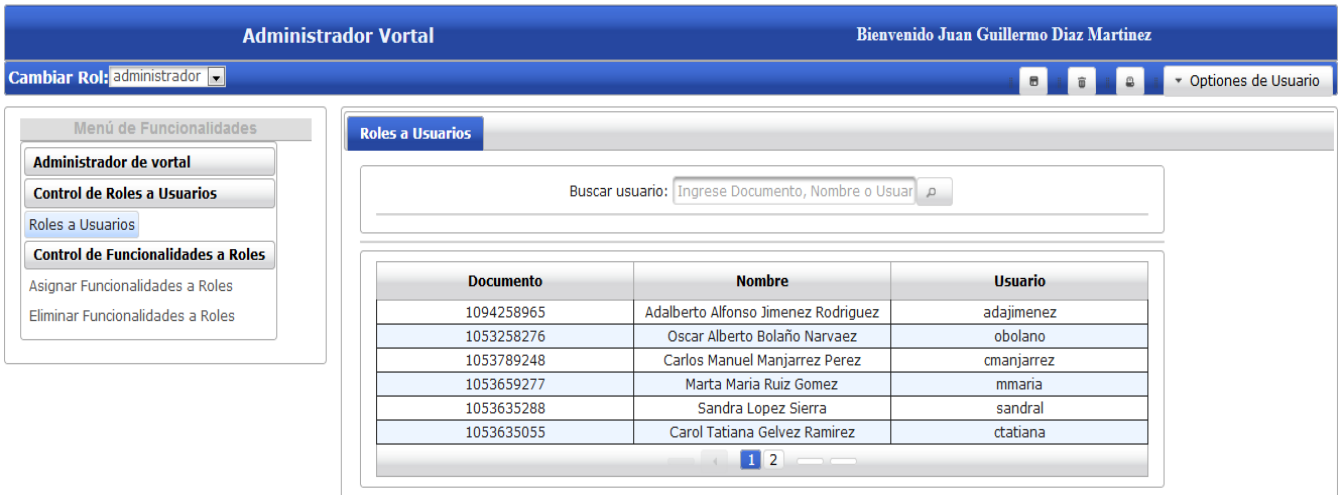
Figura 88: Vista de página de inicio con ICEfaces

Fuente: Autor

3.5.1.2.2.2 Funcionalidad de Control de Roles a Usuarios.

Utilizando los componentes *dataTable* y *list* de ICEfaces se desarrolló la funcionalidad muy similar a la que originalmente se implementó con Primefaces. Para ver su funcionamiento se presiona en el menú de funcionalidades donde dice *Roles a Usuarios.*

Figura 89: Página al seleccionar la funcionalidad Roles a Usuarios



Fuente: Autor

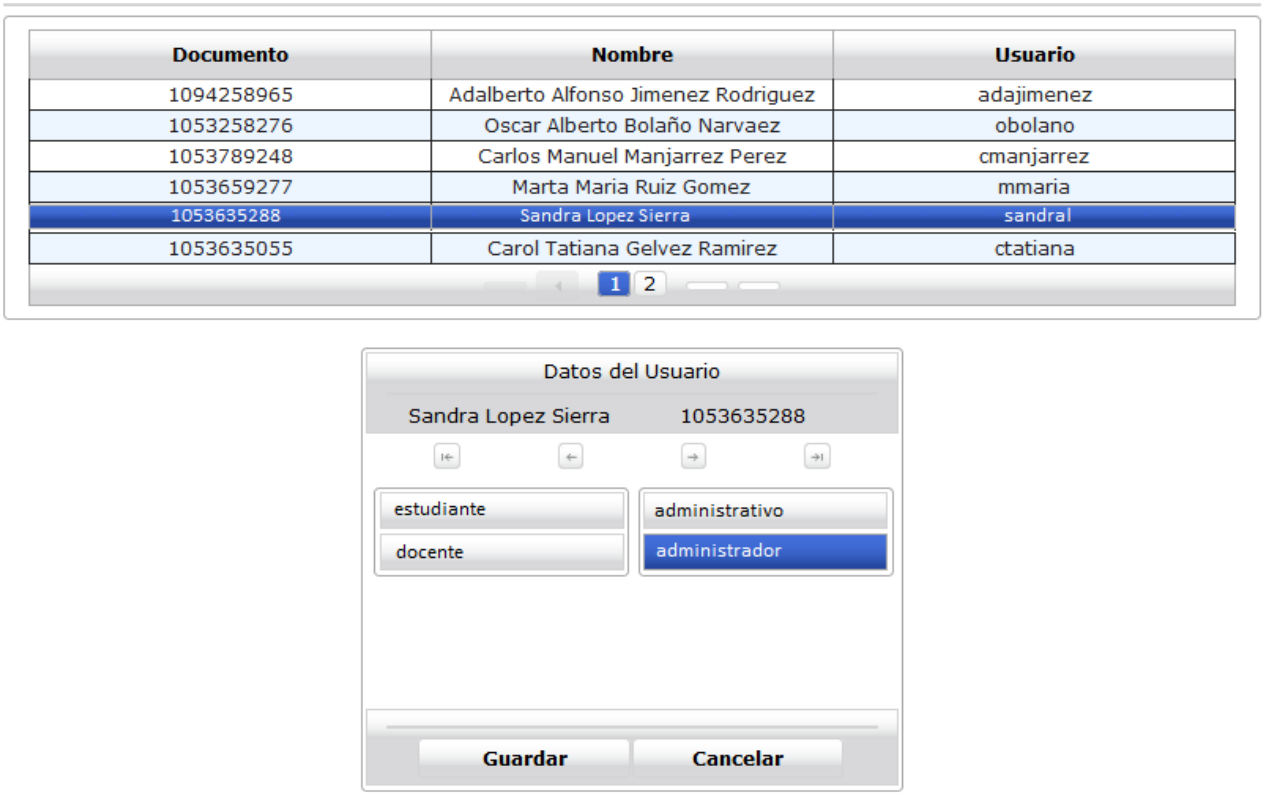
Figura 90: Componente dataTable que muestra la lista de usuarios

```
<ace:panel>
    <ace:dataTable id="selectionTable"
                   value="#{usuarioControl.listaUsuarios}" var="usud" rows="6"
                   selectionMode="#{usuarioControl.usuarioSeleccionado}"
                   paginator="true" paginatorPosition="bottom">

        <ace:ajax event="select" render="@this form:status" execute="@this" />
        <ace:ajax event="deselect" render="@this form:status" execute="@this" />

        <ace:column id="id">
            <f:facet name="header">
                Documento
            </f:facet>
            <h:outputText value="#{usud.usuDocumento}" />
        </ace:column>
        <ace:column id="name">
            <f:facet name="header">
                Nombre
            </f:facet>
            <h:outputText value="#{usud.usuNombre}" />
        </ace:column>
        <ace:column id="usuario">
            <f:facet name="header">
                Usuario
            </f:facet>
            <h:outputText value="#{usud.usuUsuario}"/>
        </ace:column>
    </ace:dataTable>
</ace:panel>
```

Fuente: Autor

Después de haber seleccionado un usuario se muestran en las listas con los roles en cada *list* de ICEfaces que se encuentran asignados a la derecha y los que tiene disponibles para asignar a la izquierda, además se pueden realizar modificaciones bien sea arrastrando los roles o utilizando las flechas ubicadas en la parte superior.

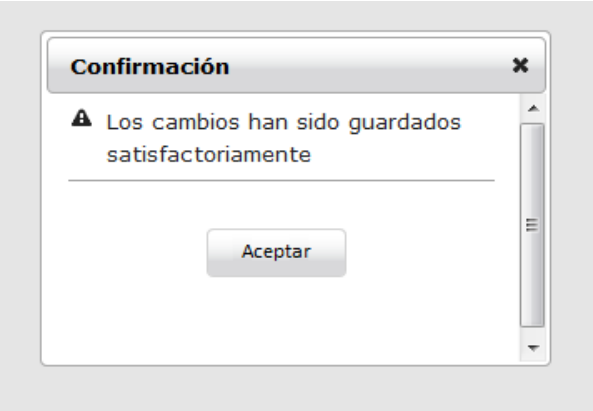
Figura 91: Modificaciones de los roles del usuario Sandra Lopez Sierra



Fuente: Autor

Una vez presionado el botón *Guardar* lleva al método que realiza toda la operación de la misma forma que el utilizado con Primefaces pero esta vez trabajando con listas de String, es decir con los nombres de los objetos de los roles de los usuarios. Si la operación es exitosa se visualiza el siguiente diálogo de confirmación.

Figura 92: Diálogo de confirmación exitoso al guardar los cambios

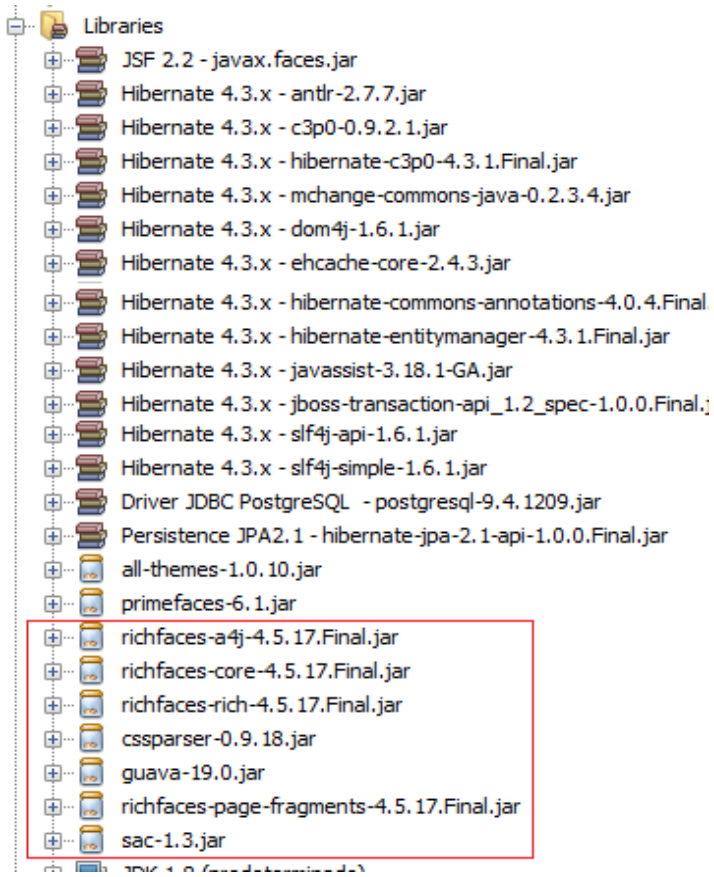


Fuente: Autor

3.5.1.2.3 RichFaces.

Para trabajar con esta biblioteca de componentes fue necesario agregar los siguientes paquetes *.jar* de la versión 4.5.17 al proyecto juntos con la biblioteca de *primefaces-6.1.jar* para utilizar algunos componentes que en el contexto de la construcción del prototipo de administrador de vortal son necesarios ya que es el framework con menos componentes de los tres aunque presenta una alta similitud con algunos elementos de Primefaces. De la misma forma que ICEfaces necesita de otras dependencias para su correcto funcionamiento.

Figura 93: Librerías necesarias para trabajar con RichFaces



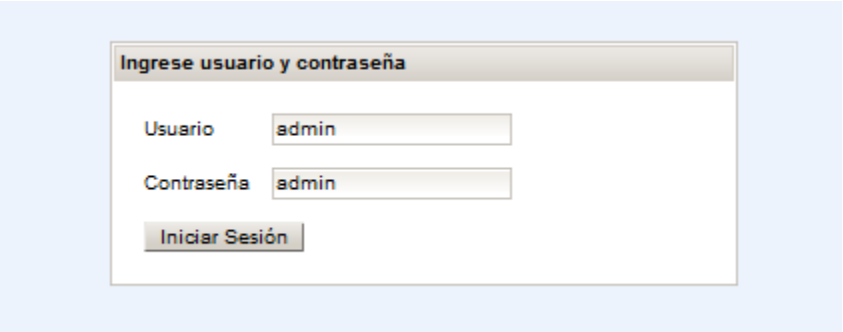
Fuente: Autor

A continuación se muestran las dos funcionalidades construidas mediante el uso de este framework, aclarando que el espacio de nombres está definido por los prefijos **<rich:...** y **<a4j:...**

3.5.1.2.3.1 Funcionalidad de Login.

Al ejecutar la aplicación con RichFaces se encuentra la página de logueo de usuario de la siguiente forma, observando la diferencia frente a los demás en tamaño de componentes y de fuentes.

Figura 94: Página de logueo de usuarios



Fuente: Autor

El usuario ingresa sus datos y presiona el botón iniciar sesión.

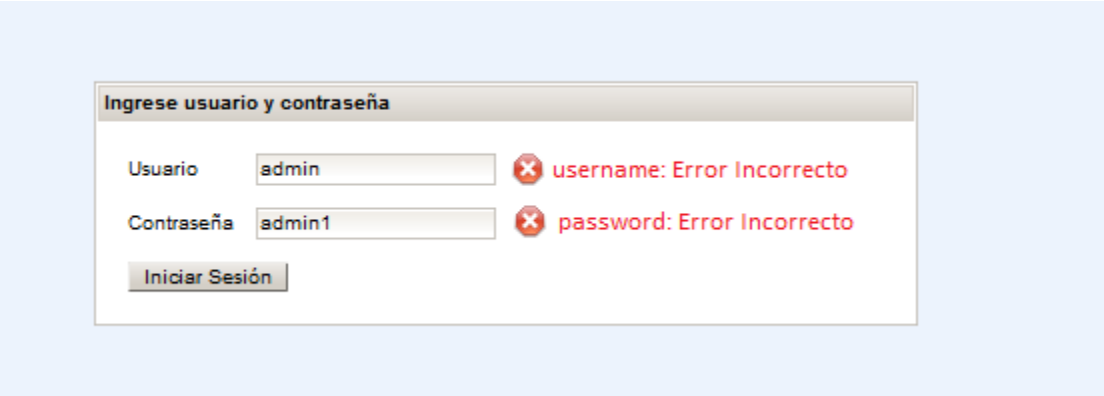
Figura 95: Código del formulario del Logueo

```
<h:form id="formLoginVortal">
    <rich:panel header="#{msg['PanelTitulo']}">
        <h:panelGrid columns="2" cellpadding="5">
            <h:outputLabel value="#{msg['OutputLabelUsuario']}"/>
            <h:inputText value="#{loginControl.usuario.usuUsuario}"
                         required="true"
                         id="username"
                         requiredMessage="Debe Ingresar Nombre de Usuario"/>
            <h:outputLabel value="#{msg['OutputLabelContrasena']}"/>
            <h:inputText value="#{loginControl.usuario.usuClave}"
                        required="true"
                        id="password"
                        requiredMessage="Debe Ingresar su Contraseña"/>
            <f:facet name="footer">
                <a4j:commandButton id="loginButton"
                                   value="#{msg['CommandButtonIniciarSesion']}"
                                   action="#{loginControl.autenticar()}"/>
            </f:facet>
        </h:panelGrid>
    </rich:panel>
</h:form>
```

Fuente: Autor

Al ingresar usuario y/o contraseña incorrectos se visualiza el mensaje de la siguiente forma.

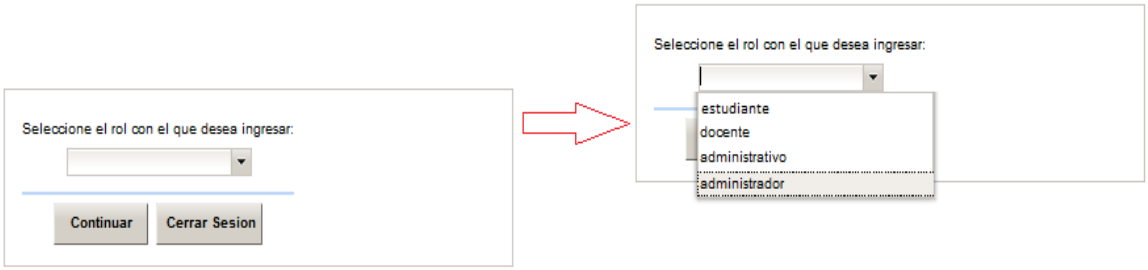
Figura 96: Mensaje de error de usuario y/o contraseña



Fuente: Autor

Después de loguearse correctamente el usuario es direccionado a la página *seleccionarRol* donde debe seleccionar uno de los roles según los que tenga asignados para poder continuar.

Figura 97: Página de selección de rol



Fuente: Autor

Figura 98: Código de la página selección de rol

```
<h:outputLabel for="selectOneMenuSeleccionarRol" value="Seleccione el rol con el que desea ingresar: "/>
<rich:select id="selectOneMenuSeleccionarRol" style="width:155px"
             required="true" requiredMessage="Seleccione un Rol"
             value="#{usuariorolControl.nombreRolSeleccionado}">

    <f:selectItem itemLabel="Seleccione un Rol" itemValue=""
                  noSelectionOption="true" />
    <f:selectItems var="usurol" value="#{usuariorolControl.rolesDelUsuario}"
                   itemLabel="#{usurol.rol.rolNombre}" itemValue="#{usurol.rol.rolNombre}"/>

    <a4j:ajax listener="#{usuariorolControl.actualizaRolSeleccionado}" event="change"/>
</rich:select>

<f:facet name="footer">
    <p:separator/>
    <a4j:commandButton value="Continuar"
                       id="btnContinuarSesion"
                       style="width: 80px; height: 30px; text-align: center; font-weight: bold"
                       action="#{usuariorolControl.cargarInicio()}"/> 
    <a4j:commandButton value="Cerrar Sesion"
                       id="btnCancelarContinuarSesion"
                       style="width: 90px; height: 30px; text-align: center; font-weight: bold"
                       actionListener="#{loginControl.cerrarSesion()}"/>
</f:facet>
```

Fuente: Autor

Ahora se mostrará la página de *adminVortal* o la página de inicio generada con este framework.

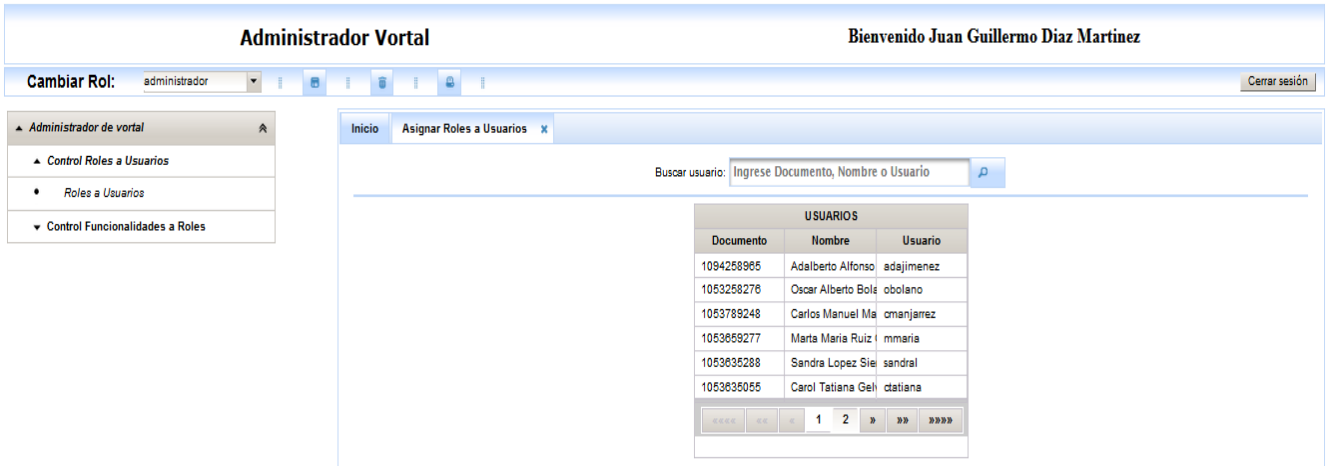
Figura 99: Vista de página de Inicio con RichFaces

Fuente: Autor

3.5.1.2.3.2 Funcionalidad de Control de Roles a Usuarios.

Utilizando los componentes *extendedDataTable* y el *pickList* de RichFaces se desarrolló la funcionalidad tan similar a la que se implementó con Primefaces. Para ver su funcionamiento se presiona en el *panelMenú* de funcionalidades donde dice *Roles a Usuarios*.

Figura 100: Página al seleccionar la funcionalidad Roles a Usuarios



Fuente: Autor

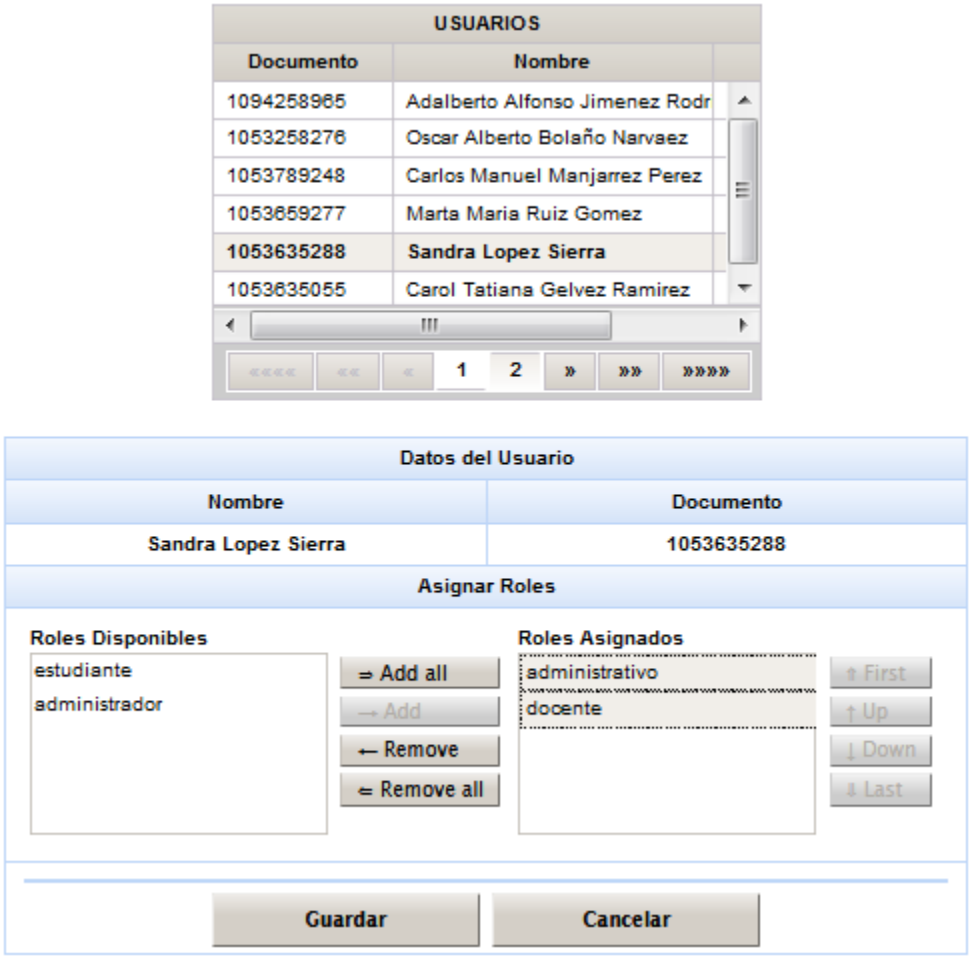
Figura 101: Componente extendedDataTable que muestra la lista de usuarios

```
<rich:extendedDataTable  var="usu" value="#{usuarioControl.listaUsuarios}" rows="6" id="dataTableUsuarios"
                         selectionMode="single" selection="#{usuarioControl.usuarioSeleccionado}"
                         style="height:220px; width:305px;">
    <f:facet name="header">
        <a4j:outputPanel>
            <h:outputText value="USUARIOS" />
        </a4j:outputPanel>
    </f:facet>
    <rich:column style="width: 200px;">
        <f:facet name="header">
            Documento
        </f:facet>
        <h:outputText value="#{usu.usuDocumento}" />
    </rich:column>
    <rich:column style="width: 250px;">
        <f:facet name="header">
            Nombre
        </f:facet>
        <h:outputText value="#{usu.usuNombre}" />
    </rich:column>
    <rich:column style="width: 150px;">
        <f:facet name="header">
            Usuario
        </f:facet>
        <h:outputText value="#{usu.usuUsuario}" />
    </rich:column>
    <a4j:ajax execute=":formPestanas:tabViewFunc:pgrid" event="selectionchange"
              listener="#{rolDualListControl.rolesAsignarEliminarUsuario(usuarioControl.usuarioSeleccionado)}"
              />
</rich:extendedDataTable>
```

Fuente: Autor

Al haber seleccionado un usuario se muestran en el *pickList* los roles disponibles y asignados como el *pickList* de Primefaces donde se ubican los asignados a la derecha y los que disponibles para asignar a la izquierda, por tanto también se pueden realizar modificaciones utilizando las flechas ubicadas entre las dos listas.

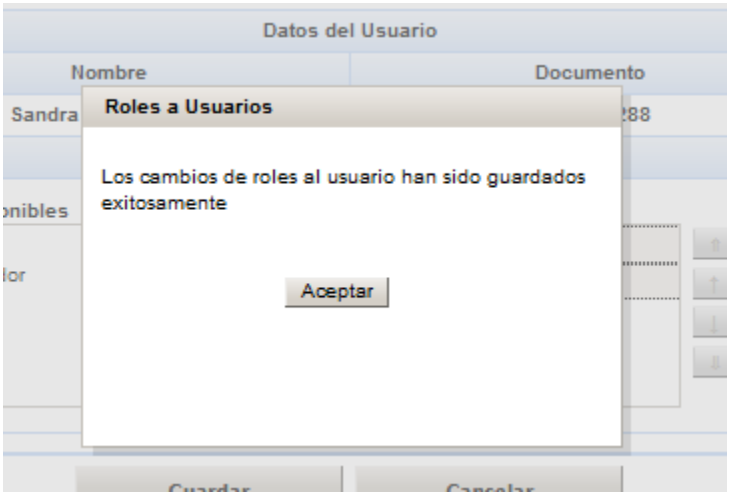
Figura 102: Modificaciones de los roles del usuario Sandra Lopez Sierra



Fuente: Autor

Una vez presionado el botón *Guardar* lleva al método que realiza toda la operación de la misma forma que el utilizado con Primefaces. Si la operación es exitosa se visualiza el siguiente diálogo de confirmación.

Figura 103: Diálogo de confirmación exitoso al guardar los cambios



Fuente: Autor

3.5.1.2.4 Evaluación de la funcionalidad.

Para la evaluación mediante de este criterio se tuvieron en cuenta todos los aspectos relacionados con la interacción entre componentes y *Beans* de respaldo o controladores, la manera en la que el framework utiliza sus propios elementos para su funcionamiento aun así pudiéndose utilizar los de un tercero como fue el caso de ICEfaces y un poco de RichFaces en relación a Primefaces que se soporta de sí mismo sin depender de ninguna otra biblioteca de componentes.

Tabla 17: Calificación mediante funcionalidad para cada Framework

Criterio	Valor	Significado	Primefaces	ICEfaces	RichFaces
Funcionalidad	3	Muy Bueno	X		X
	2	Bueno		X	
	1	Regular			

3.6 Selección del Framework de acuerdo a los criterios.

Para la selección del mejor Framework de acuerdo a la evaluación realizada con cada uno de los criterios se deben sumar los puntos asignados por cada evaluación. A continuación se observa la tabla con el puntaje total obtenido.

Tabla 18: Resultado de análisis comparativo

Criterio	Primefaces	ICEfaces	RichFaces
Mantenibilidad y Documentación	3	2	1
Índices de Interés	3	3	3
Disponibilidad de Componentes	3	2	1
Usabilidad	3	2	2
Funcionalidad	3	2	3
Puntaje Total	**15**	**11**	**10**

El resultado más importantes y que además era el propósito de este proyecto de investigación fue la selección del framework con mayor calificación luego de hacer el estudio y análisis según los criterios planteados, este proceso arrojó como mejor opción a Primefaces framework open source para JSF del cual se obtuvo la mayor puntuación, lo que indica que al momento de hacer el desarrollo del prototipo bajo los beneficios que ofrece dicho framework debían hacerse evidentes los potenciales que lo hicieron sobresalir por encima de los demás, y es exactamente lo que se evidenció, el framework Primefaces mostró gran facilidad al momento de utilizar sus componentes; como por ejemplo; la integración e interacción entre los elementos de la vista y las variables; y métodos de los *Beans* administrados; la integración de una lista de

temas predefinidos que incluso se pueden personalizar; la integración de clases Java para el funcionamiento de sus componentes; la construcción de aspectos visuales con marcos dinámicos, llamados *Facelets Templates*; validación de elementos y campos en la vista por medio de atributos específicos; integración con *Ajax* con el uso de tributos en botones y otros elementos para hacer llamado a métodos de *Beans* administrados; su rendimiento; entre otras características que hacen de Primefaces el framework recomendado para las construcciones de aplicaciones con interfaces de usuarios eficientes.

4 Conclusiones.

Se realizó la respectiva revisión bibliográfica con respecto a las alternativas frameworks web open source que mayor impacto tienen en el desarrollo de aplicaciones web en Java evidenciando y analizando sus características de funcionamiento por medio de los ejemplos prácticos que ofrecen en sus sitios web oficiales o artículos en general disponibles en la web y se encontró qué, todos ofrecen grande beneficios para el desarrollador en cuanto a facilidad de uso, pero los más apropiados para este proyecto de grado fueron Primefaces, ICEfaces y RichFaces.

Si bien es cierto que existe gran variedad de Frameworks para la implementación del lado del cliente o del *Front-End* disponibles para la construcción de aplicaciones en Java y se comprobó que no existe uno que se adapte a todo tipo de aplicativos, por ello la importancia de hacer un análisis comparativo para determinar el más apropiado según el tipo de aplicación que se pretende desarrollar y su alcance.

El proceso de validación de selección del Framework para las interfaces de usuarios más apropiado se hizo basados en la selección de alternativas de solución que utilizan diferentes Instituciones y Universidad al querer comparar tecnologías de desarrollo de las que por medio de puntuaciones en cada criterio se va evaluando a través análisis con respecto a los demás, teniendo en cuenta los fundamentos por los que se definen las puntuaciones.

Se desarrolló el prototipo de aplicación para el manejo de los procesos que actualmente lleva a cabo el aplicativo administrador Vortal en el CIADTI y se evaluó con algunas funcionalidades que pusieron a prueba su alcanzabilidad, haciendo más óptimos los procesos.

Con todo este proceso de estudio y análisis, la conclusión más significativa es el impacto que ha tenido la implementación de estas tecnologías en el CIADTI, es decir, el gran avance que las nuevas tecnologías han aportado al desarrollo de aplicaciones web con el fin de mejorar sus entornos a nivel de interfaces y la experiencia con los usuarios.

En definitiva la recomendación a modo personal es que se empiecen a transformar poco a poco las vistas de cada uno de los softwares de los que dispone actualmente el CIADTI haciendo uso tanto de estas tecnologías como de las que puedan ir surgiendo.

5 Fuentes Bibliográficas.

Alicante, U. de. (2016). Modelo Vista Controlador [MVC]. *Página Web*. Retrieved from https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html

Alvarez, J., & Tejero, -Uria. (2009). PFC: J2EE. Diseño e implementación de un Framework de Persistencia Consultor: Josep Maria Camps Riba. Retrieved from http://openaccess.uoc.edu/webapps/o2/bitstream/10609/761/1/00883tfc.pdf

Anónimo. (2005). Capítulo 2 : Marco Teórico. Retrieved from http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/sanchez_r_ma/capitulo2.pdf

bootsfaces.net. (2017). BootsFaces: el marco JSF de próxima generación basado en Bootstrap. Retrieved November 5, 2017, from https://www.bootsfaces.net/

butterfaces.org. (2017). ButterFaces: un marco JSF ligero y moderno. Retrieved November 6, 2017, from http://www.butterfaces.org/

Fernández, Y., & Yanette, D. (2012). Telemática revista digital de las tecnologías de la información y las telecomunicaciones. Retrieved from http://webcache.googleusercontent.com/search?q=cache:http://revistatelematica.cujae.edu.cu/index.php/tele/article/view/15

GPSOS. (2017). Definición de Open Source - ¿Que es el Open Source? Retrieved from https://www.gpsos.es/soluciones-open-source/definicion-de-open-source/

Gutiérrez, J. J. (2007). ¿Qué es un framework web? Retrieved from http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf

icesoft.org. (2017). ICEfaces JSF Framework Overview - ICEsoft Technologies. Retrieved October 31, 2017, from http://www.icesoft.org/java/projects/ICEfaces/overview.jsf

INTECO. (2009). CURSO DE INTRODUCCIÓN A LA INGENIERÍA DEL SOFTWARE Laboratorio Nacional de Calidad del Software NOTA DE EDICIÓN. Retrieved from https://jmpovedar.files.wordpress.com/2014/03/curso-de-introduccic3b3n-a-la-ingenieria-del-software.pdf

omnifaces.org. (2017). Muestra de OmniFaces. Retrieved November 6, 2017, from http://showcase.omnifaces.org/

openfaces.org. (2017). OpenFaces. Retrieved November 5, 2017, from http://www.openfaces.org/

OpenSource Initiative. (2007). The Open Source Definition | Open Source Initiative. Retrieved

from https://opensource.org/osd

Pech-May, F., Gomez-Rodriguez, M. A., Cruz-Diaz, L. A. de la, & Lara-Jeronimo, S. U. (2010). Desarrollo de Aplicaciones web con JPA, EJB, JSF y PrimeFaces. *Instituto Tecnológico Superior de Los Ríos. México.* Retrieved from http://www.tamps.cinvestav.mx/~fpech/sd/files/paper001.pdf

primefaces.org. (2017). Documentación | PrimeFaces. Retrieved October 25, 2017, from https://www.primefaces.org/documentation/

SGI-Consulting. (2010). Tutorial de JavaServer Faces. Retrieved from http://www.sgi-consulting.com/sgi2/jsf/JSF.pdf

Stallman, R. M. (2004). Software libre para una sociedad libre, 232.

Stallman Richard. (2016). gnu.org. *Página Web*. Retrieved from https://www.gnu.org/philosophy/open-source-misses-the-point.html

Teruelo, S. Á., Mancha, M. Á. B., Gallego, C. R., & Fernández, A. M. S. (2008). Java Server Faces (JSF). *Departamento de Ingeniería Telemática Universidad Carlos III de Madrid*. Retrieved from http://www.it.uc3m.es/spickin/docencia/comsoft/presentations/spanish/07-08/JSF.pdf

TIC-TEK. (2012). Internacionalización con JSF |. Retrieved October 9, 2017, from http://tic-tek.es/internacionalizacion-con-jsf/

Universidad de Alicante. (2012). Componentes de presentación. Retrieved from http://www.jtech.ua.es/j2ee/publico/jsf-2012-13/wholesite.pdf

Universidad de Murcia. (2011). HIBERNATE Aplicaciones Distribuidas. Retrieved from https://aulavirtual.um.es/access/content/group/3871_G_2011_N_N/Teoria/T5A - Hibernate.pdf