	UNIVERSIDAD DON BOSCO **FACULTAD DE ESTUDIOS TECNOLÓGICOS** **COORDINACIÓN DE COMPUTACIÓN**
Ciclo II	**Desarrollo de aplicaciones con Web Frameworks** **Guía de Laboratorio No. 04** **"Introducción a JSF 2"**

I. Objetivos

Que el alumno comprenda el uso del Framework Java Server Faces (JSF)

Que el alumno configure JSF 2 y el use sus Tags.

II. INTRODUCCIÓN

JSF (Java Server Faces) es un framework de desarrollo basado en el patrón MVC (Modelo Vista Controlador). Al igual que Struts, JSF pretende normalizar y estandarizar el desarrollo de aplicaciones web. Hay que tener en cuenta JSF es posterior a Struts y por lo tanto se ha nutrido de la experiencia de éste. De hecho el creador de Struts Craig R. McClanahan) también es líder de la especificación de JSF.

JSF incluye:

- Un conjunto de APIs para representar componentes de una interfaz de usuario y administrar su estado, manejar eventos, validar entrada, definir un esquema de navegación de las páginas y dar soporte para internacionalización y accesibilidad.
- Un conjunto por defecto de componentes para la interfaz de usuario.
- Dos bibliotecas de etiquetas personalizadas para JavaServer Pages que permiten expresar una interfaz JavaServer Faces dentro de una página JSP.
- Un modelo de eventos en el lado del servidor.
- Administración de estados.

La especificación de JSF fue desarrollada por la Java Community Process

Versiones de JSF:

- JSF 1.0 (11-03-2004)- lanzamiento inicial de las especificaciones de JSF
- JSF 1.1 (27-05-2004) Lanzamiento que solucionaba errores. Sin cambios en las especificaciones ni en el renderkit de HTML..
- JSF 1.2 (11-05-2006) lanzamiento con mejoras y corrección de errores.
- JSF 2.0 (12-08-2009) último lanzamiento.

Las principales implementaciones de JSF son:

- JSF Reference Implementation de Sun Microsystems.
- MyFaces proyecto de Apache Software Foundation.
- Rich Faces, de Jboss. Trae componentes adicionales para crear aplicaciones más "ricas"
- ICEfaces Contiene diversos componentes para interfaces de usuarios más enriquecidas, tales como editores de texto enriquecidos, reproductores de multimedia, entre otros.
- jQuery4jsf Contiene diversos componentes sobre la base de uno de los más populares framework javascript jQuery.

III. MATERIALES Y EQUIPO

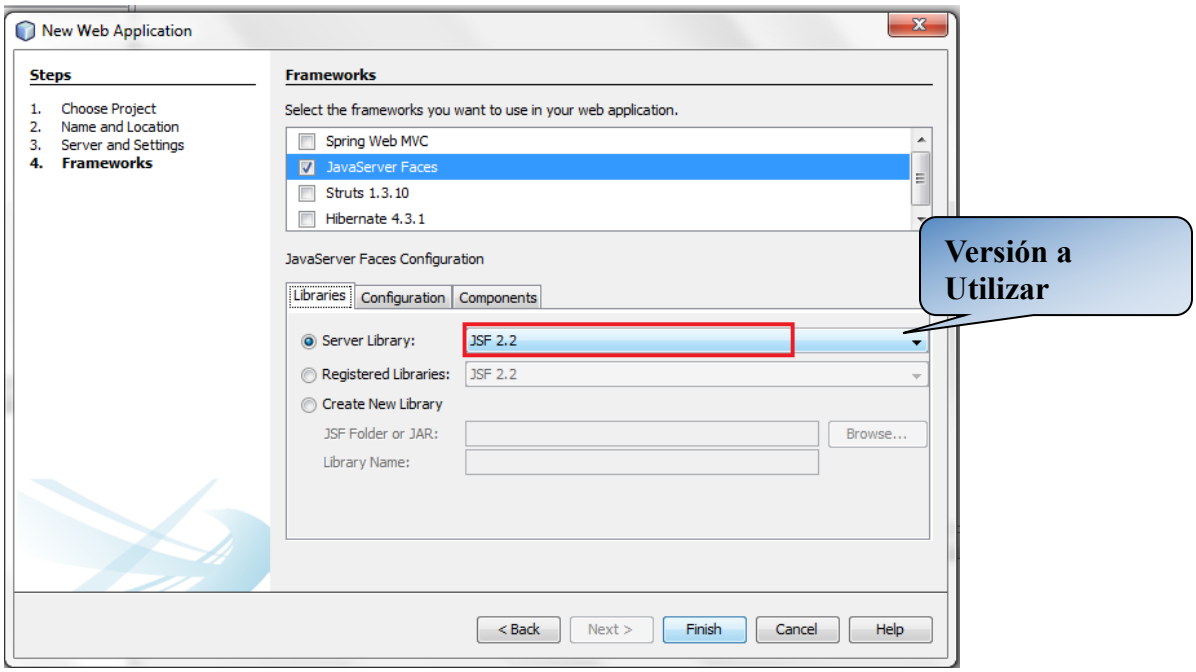
No.	Requerimiento	Cantidad
1	Guía de Laboratorio #3	1
2	Una maquina que tenga instalado NetBeans 8 o Superior	1

IV. PROCEDIMIENTO

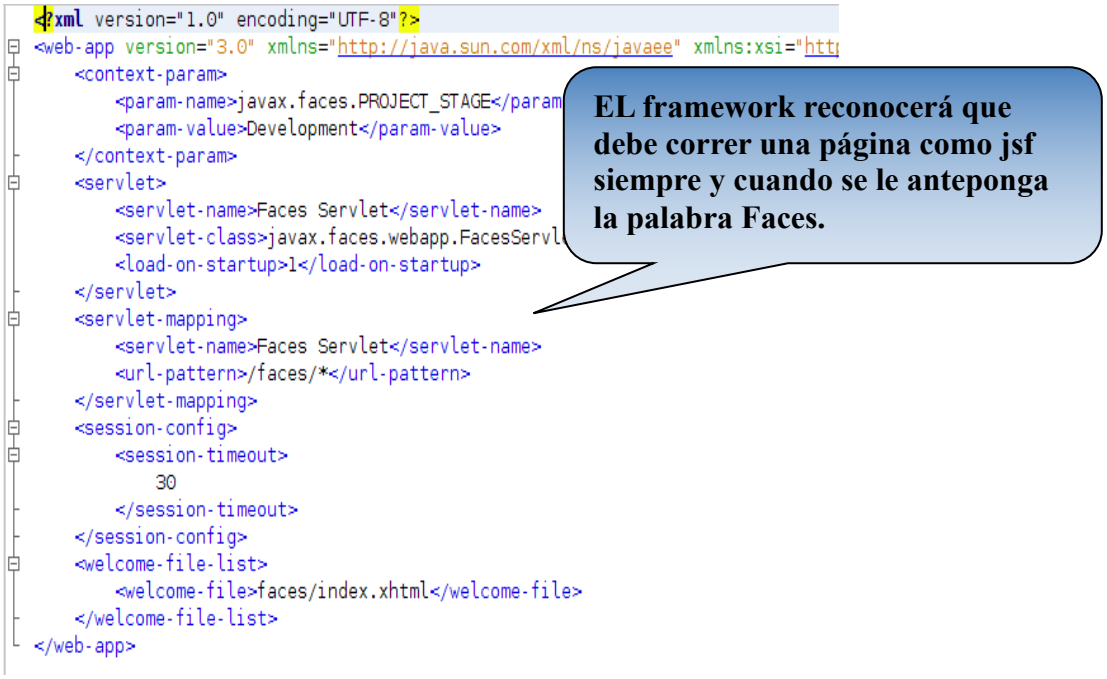
Parte I: Creando un login simple.

1. Para nuestro primer ejemplo que será un login necesitaremos crear un proyecto web

llamado **“JSFBasic”**, seleccionar el framework **“Java Server Faces”** utilizando la versión 2.0 o superior.



Configuración del archivo web.xml. En este archivo se puede observar cómo FacesServlet es un Servlet centralizado que se encargará de la ejecución directa de las paginas JSF, siempre y cuando se le anteponga la palabra “**faces**”.



2. Modificar el archivo **“index.xhtml”** en este punto se hace referencia a un objeto

llamado “login”. Este objeto es un ManagedBean y será creado en pasos posteriores.

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xh
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      >
    <h:head>
        <title>Index JSF</title>
    </h:head>
    <h:body>
        <h:form>
             Ingrese su nombre de usuario: <h:inputText value="#{login.nombre}" /><br/>
             Ingrese su contraseña: <h:inputSecret value="#{login.password}" /><br/>
        <h:commandButton value="Ingresar" action="login1"/>
        </h:form>
    </h:body>
</html>
```

Nota: JSF separa la capa de presentación de la lógica de negocio. Para que nuestras páginas JSF puedan acceder a esta lógica, utilizamos **Managed Beans**.

¿Cómo declaramos un bean?

Desde el fichero faces-config.xml o con anotaciones.

- Utilizando el fichero faces-config.xml

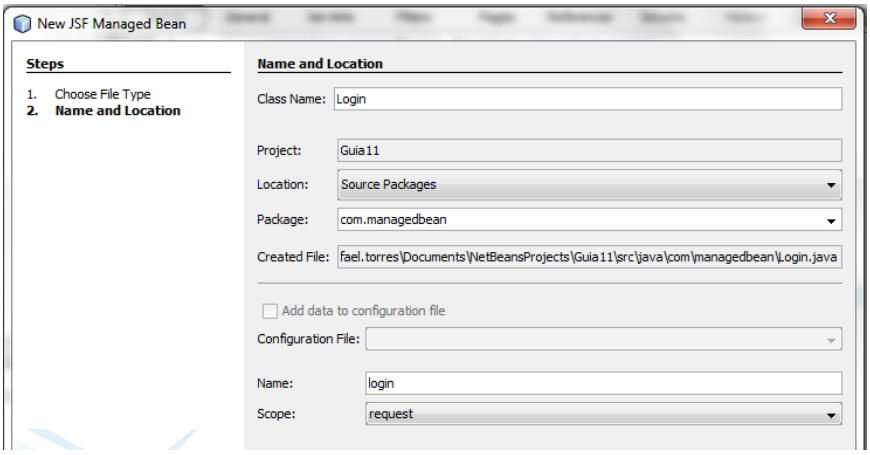
```
<managed-bean>
<managed-bean-name>userBean</managed-beanname>
<managed-bean-class>
com.examples.UserBean</managed-bean-class>
<managed-bean-scope>session</managed-bean-scope> </managed-bean>
```

- Utilizando anotaciones (la manera preferida por los desarrolladores)

```
package com.examples;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
@ManagedBean
@SessionScoped
public class UserBean {...}
```

3. Crear un paquete llamado **com.managedbean**, el cual contendrá los ManagedBean a utilizar en esta práctica.

4. Crear el ManageBean llamado "**Login**" y que contendrá las propiedades **nombre** y **password**, para ello dar click derecho sobre **Web Pages->New->Other...->JavaServer Faces->JSF Managed Bean.**



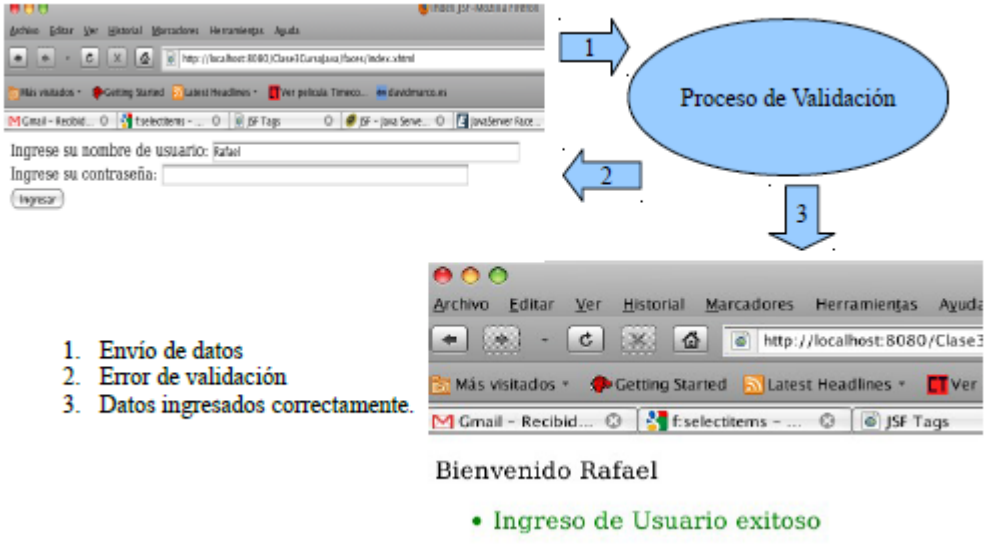
5. El Managed Bean deberá tener el código que se muestra en la siguiente imagen. Éste utiliza anotaciones para declarar que la clase es un Managedbean y a la vez declara que es un objeto de Sesión, las anotaciones facilitan mucho el trabajo ya que en otros framework como Strust todas estas definiciones quedaban en archivos xml.

```
@ManagedBean
@SessionScoped
public class Login {
    private String nombre;
    private String password;

    /** Creates a new instance of Login */
    public Login() {
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

}
```

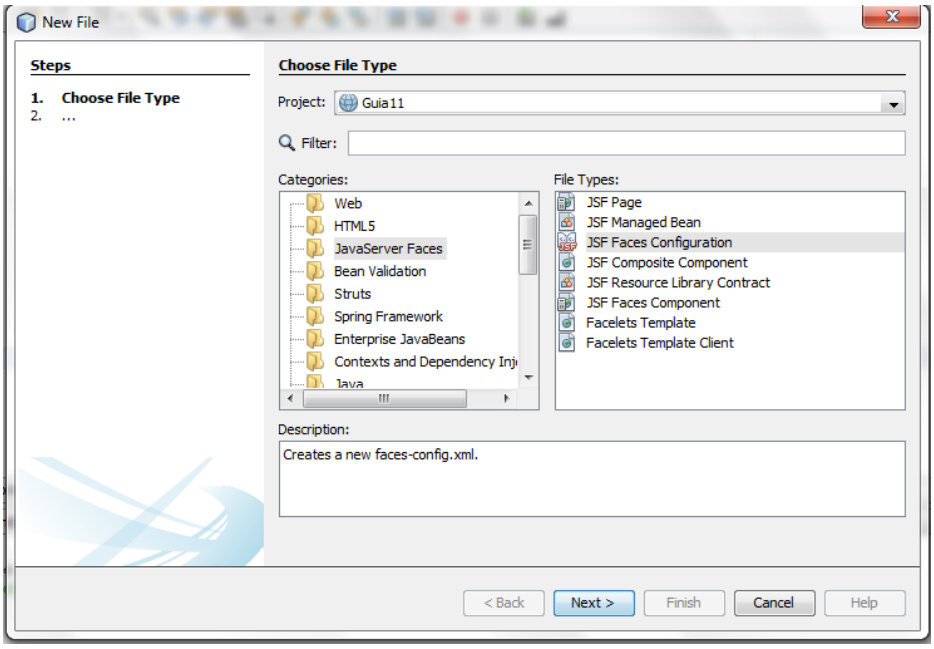
6. Crear una página llamada **“bienvenido.xhmtl”** la cual será la página a mostrar al dar click en el botón de **ingresar**.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
To change this template, choose Tools | Templates
and open the template in the editor.
-->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      >
    <h:head>
        <title>Bienvenido</title>
    </h:head>
    <h:body>
        <p>
            Bienvenido <h:outputText value="#{login.nombre}"/>
        </p>
    </h:body>
</html>
```

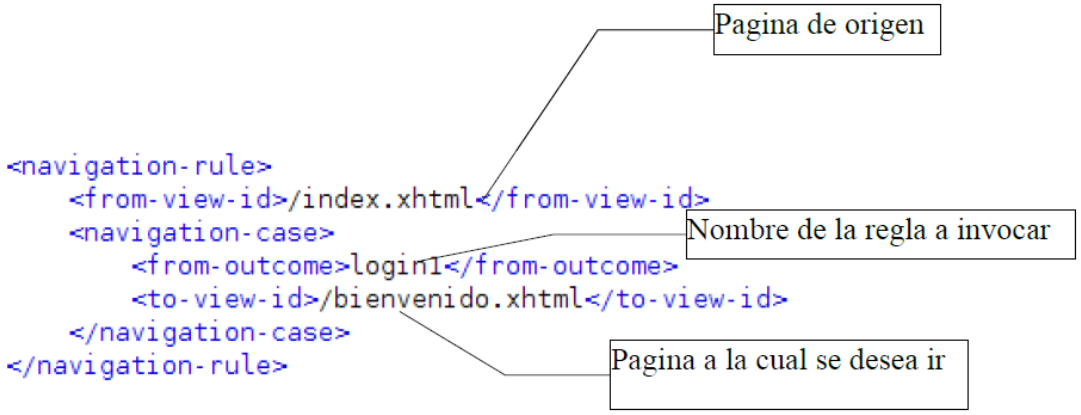
El resultado a ser ejecutado en el navegador es el siguiente



7. Como siguiente punto se definirán las reglas de navegación, para ello nos vamos al archivo faces-config.xml, por estar en la versión 2.0 o superior este archivo no aparece por defecto, sin embargo es posible agregarlo. Dar click derecho sobre **WEB-INF->New->Other...->JavaServer Faces->JSF Faces Configuration.**



8. Agregar la siguiente regla de navegación en el archivo **"faces-config.xml"**



Cuando se da click sobre el **commandButton** el cual tiene definido como action el nombre de la regla de navegación a invocar. Al colocar el outcome "**login1**" redireccionará a la página **bienvenido.xhml**.

<h:commandButton value="Ingresar" action="login1"/>

Por estar utilizando la versión 2.0 o superior podríamos incluso haber omitido la regla de navegación dentro del **faces-config.xml** y si luego de dar click en el botón Ingresar se quisiera dirigir a la pagina **bienvenido.xhml** bastaría con poner dentro del **action** el nombre de la pagina, El tag incluso podría quedar de la siguiente manera:

<h:commandButton value="Ingresar" action="bienvenido"/>

Parte II: Creando un logueo un poco más complicado

Para esta parte se creará un logueo simple en el cual la validación de usuario y password se encuentre dentro del **ManagedBean**. Debemos modificar el archivo **index.xhtml** para que se visualice de la siguiente manera:

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      >
    <h:head>
        <title>Index JSF</title>
    </h:head>
    <h:body>
        <h:form>
             Ingrese su nombre de usuario: <h:inputText value="#{login.nombre}"/><br/>
             Ingrese su contraseña: <h:inputSecret value="#{login.password}"/><br/>
             <h:commandButton value="Ingresar" action="#{login.validar}"/>
        </h:form>
         <h:messages errorStyle="color: red" infoStyle="color: green"/>
    </h:body>
</html>
```

Como se puede observar las modificaciones consisten en llamar un método dentro del **ManagedBean** para realizar la validación, esto se visualiza dentro del tag **commandButton** en su etiqueta action, la cual queda de la siguiente manera **action="#{login.validar}"** y el siguiente cambio consiste en agregar la etiqueta **<h:messages />**. El funcionamiento es mostrar todos los mensajes que genere la aplicación.

Ahora deberemos crear un nuevo paquete llamado "**com.util**" y dentro de él una clase llamada "**JsfUtil**" la cual contendrá el siguiente código.

```java
package com.util;

import javax.faces.application.FacesMessage;
import javax.faces.context.FacesContext;

/**
 *
 * @author rafael
 */
public class JsfUtil {


    public static void addErrorMessage(String msg) {
        FacesMessage facesMsg = new FacesMessage(FacesMessage.SEVERITY_ERROR, msg, msg);
        FacesContext.getCurrentInstance().addMessage(null, facesMsg);
    }

    public static void addSuccessMessage(String msg) {
        FacesMessage facesMsg = new FacesMessage(FacesMessage.SEVERITY_INFO, msg, msg);
        FacesContext.getCurrentInstance().addMessage(null, facesMsg);
    }
}
```

Para poder invocar los mensajes de error que se encuentran dentro de un archivo de tipo **properties** deberemos de configurar el archivo **faces-config.xml**, para que el proyecto comprenda dónde se encuentra ubicado el archivo que será de lectura para los mensajes, la configuración se muestra a continuación:

```xml
<faces-config version="2.0"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java
  <application>
      <resource-bundle>
          <base-name>/Bundle</base-name>
          <var>bundle</var>
      </resource-bundle>
  </application>

    <navigation-rule>
        <from-view-id>/index.xhtml</from-view-id>
        <navigation-case>
            <from-outcome>login1</from-outcome>
            <to-view-id>/bienvenido.xhtml</to-view-id>
        </navigation-case>
    </navigation-rule>
</faces-config>
```

Crear un archivo de tipo **properties** llamado **Bundle** que contendrá las llaves que se utilizarán.

```
LoginCorrecto=Ingreso de Usuario exitoso
LoginError=Usuario o password incorrecto
```

Como siguiente paso procederemos a crear el método validar dentro del ManagedBean Login, el cual quedara de la siguiente manera.

```
public String validar(){
    if(getNombre().equals("Rafael") && getPassword().equals("123456")){
        JsfUtil.addSuccessMessage(ResourceBundle.getBundle("/Bundle").getString("LoginCorrecto"));
        return "bienvenido";
    }
    else{
        JsfUtil.addErrorMessage(ResourceBundle.getBundle("/Bundle").getString("LoginError"));
        return null;
    }
}
```

Observe el tipo de dato a devolver será **String**, esto se debe a que en el **return** debemos agregar el nombre de una regla de navegación creada con anterioridad ó simplemente (si queremos navegar a una página dentro de la misma carpeta) el nombre de la página. Otro detalle importante es que utilizamos la clase **JsfUtil** para definir el tipo de mensaje a utilizar.

Como punto final para esta parte debemos de modificar la página **bienvenido.xhml** quedando de la siguiente manera.

```
To change this template, choose Tools | Templates
and open the template in the editor.
-->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      >
    <h:head>
        <title>Bienvenido</title>
    </h:head>
    <h:body>
        <p>
            Bienvenido <h:outputText value="#{login.nombre}"/>
             <h:messages errorStyle="color: red" infoStyle="color: green"/>
        </p>
    </h:body>
</html>
```

El resultado en el navegador se debe ver de la siguiente manera

1. Envío de datos
2. Error de validación
3. Datos ingresados correctamente.

Parte II: Validación de campos

Ahora validaremos los campos de entrada como requeridos y definiremos un mensaje personalizado.

Modificar el archivo **index.xhtml** este deberá de quedar tal y como se muestra a continuación:

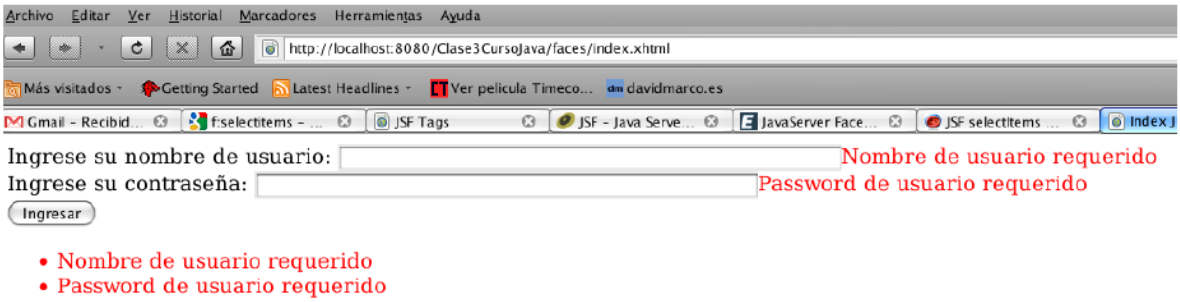
```
'xml version='1.0' encoding='UTF-8' ?>
 DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
itml xmlns="http://www.w3.org/1999/xhtml"
     xmlns:h='http://java.sun.com/jsf/html"
     xmlns:f='http://java.sun.com/jsf/core"
     >
   <h:head>
      <title>Index JSF</title>
   </h:head>
   <h:body>
      <h:form>
          Ingrese su nombre de usuario:
          <h:inputText id="nombre' value="#{login.nombre}' required='true" requiredMessage='#{bundle.NombreRequerido}"/>
          <h:message for="nombre" errorStyle='color: red'/>
          <br/>
          Ingrese su contraseña:
          <h:inputSecret id="password" value='#{login.password}" required="true' requiredMessage="#{bundle.PasswordRequerido}'/>
          <h:message for="password' errorStyle="color: red"/>
          <br/>
           <h:commandButton value='Ingresar" action="#{login.validar}'/>
      </h:form>
       <h:messages errorStyle="color: red" infoStyle="color: green"/>
   </h:body>
/html>
```

Agregar dentro del archivo **Bundle.properties** las siguientes llaves

NombreRequerido=Nombre de usuario requerido

PasswordRequerido=Password de usuario requerido

El resultado al correr en el navegador será similar al siguiente



Parte II: Tags de JSF

Crear una nueva página **xhtml** llamada **Form.xhml** la cual contendrá el siguiente código:

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
   <h:head>
      <title>Facelet Title</title>
   </h:head>
   <h:body>
     <h2> Etiquetas html jsf </h2>
   <h:form>


   <h:outputText value="#{bundle.nombre}"/>
   <h:inputText value="#{etiquetas.nombre}"/>
   <br/>

   <h:outputText value="#{bundle.apellidos}"/>
   <h:inputText value="#{etiquetas.apellidos}" maxlength="30" size="30" />
   <br/>

   <h:outputText value="#{bundle.password}"/>
   <h:inputSecret value="#{etiquetas.password}" maxlength="8" size="8"/>
   <br/>

   <h:outputText value="#{bundle.comentario}"/>
   <h:inputTextarea rows="3" cols="10" value="#{etiquetas.comentario}"/>
```

```
    <br/>

    <h:outputText value="#{bundle.fumador}"/>
    <h:selectBooleanCheckbox value="#{etiquetas.fumador}" />
    <br/>

    <h:outputText value="#{bundle.profesion}"/>

    <h:selectOneMenu value="#{etiquetas.profesion}">
        <f:selectItem itemValue="estudiante" itemLabel="Estudiante"/>
        <f:selectItem itemValue="programador" itemLabel="Programador"/>
        <f:selectItem itemValue="analista" itemLabel="Analista"/>
        <f:selectItem itemValue="diseñador" itemLabel="Diseñador"/>
    </h:selectOneMenu>
    <br/>

    <h:outputText value="#{bundle.aficiones}"/>
    <h:selectManyCheckbox value="#{etiquetas.aficiones}">
        <f:selectItem itemValue="leer" itemLabel="Leer"/>
        <f:selectItem itemValue="cine" itemLabel="Cine"/>
        <f:selectItem itemValue="musica" itemLabel="Música"/>
    </h:selectManyCheckbox>
    <br/>


    <h:outputText value="#{bundle.lenguajes}"/>

    <h:selectManyListbox value="#{etiquetas.lenguajes}" >
        <f:selectItem itemValue="c" itemLabel="C"/>
        <f:selectItem itemValue="java" itemLabel="Java"/>
        <f:selectItem itemValue="struts" itemLabel="Struts"/>
        <f:selectItem itemValue="jsf" itemLabel="Jsf"/>
    </h:selectManyListbox>
    <br/>

    <h:commandButton action="resultado"  styleClass="boton" value="enviar"/>
    </h:form>

    </h:body>
</html>
```

Crear un ManagedBean llamado Etiquetas y modificarlo para que quede de la siguiente manera.

```
/*
 * To change this template, choose Tools | Templates
```

```
 * and open the template in the editor.
 */

package com.managedbean;

import java.util.ArrayList;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;

/**
 *
 * @author rafael
 */
@ManagedBean
@SessionScoped
public class Etiquetas {
       private String nombre;
          private String apellidos;
          private String password;
          private String comentario;
          private Boolean fumador;
          private ArrayList <String> aficiones;
          private String profesion;
       private ArrayList <String> lenguajes;

   /** Creates a new instance of Etiquetas */
   public Etiquetas() {
   }

   public ArrayList<String> getAficiones() {
      return aficiones;
   }

   public void setAficiones(ArrayList<String> aficiones) {
      this.aficiones = aficiones;
   }

   public String getApellidos() {
      return apellidos;
   }

   public void setApellidos(String apellidos) {
      this.apellidos = apellidos;
   }

   public String getComentario() {
      return comentario;
```

```
    }

    public void setComentario(String comentario) {
        this.comentario = comentario;
    }

    public Boolean getFumador() {
        return fumador;
    }

    public void setFumador(Boolean fumador) {
        this.fumador = fumador;
    }

    public ArrayList<String> getLenguajes() {
        return lenguajes;
    }

    public void setLenguajes(ArrayList<String> lenguajes) {
        this.lenguajes = lenguajes;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getProfesion() {
        return profesion;
    }

    public void setProfesion(String profesion) {
        this.profesion = profesion;
    }

}
```

Como último punto agregar las llaves al archivo Bundle.

```
Titulo=Una aplicacion sencilla Java Server Faces
TextoUsuario=Nombre
TextoPassword=Clave
TetoBotonActionLogin=Ingresar
nombre=Introduce tu nombre
apellidos=Introduce tus apellidos
password=Introduce tu contraseña
comentario=Introduce tu comentario
fumador=Eres fumador, activa la casilla en caso de confirmación
profesion=Selecciona tu profesión
aficiones=Selecciona tus aficiones favoritos
lenguajes=Selecciona los lenguajes que conoces
resultado=Datos introducidos en el formulario
```

Proceder a ejecutar la página xthml y ver el resultado.

V. DISCUSIÓN DE RESULTADOS

Crear un Formulario de ingreso de un estudiante, en el cual deberá pedir los siguientes datos:

- Identificación (DUI)
- Nombre Completo
- Dirección
- Carnet UDB

Con la información solicitada conectar con una base de datos y guardar la información utilizando JDBC. Genere un DAO para ello.

Investigue qué son los datatables de JSF. Con la información registrada en la base de datos, genere un ArrayList de estudiantes para ser iterado con él. Recuerde seguir usando el DAO anterior.

Investigue en qué consiste la validación **f:validateRegex y realice la validación de DUI y**

carnet UDB.

Investigue cómo realizar la internacionalización en la aplicación de ejemplo. Actualmente sólo se encuentra en español, incluya el idioma inglés.

HOJA DE EVALUACIÓN

Hoja de cotejo: 4

Alumno:	Carnet:
Docente:	Fecha:
Título de la guía:	No.:

Actividad a evaluar	Criterio a evaluar	Cumplió		Puntaje
		SI	NO	
Discusión de resultados	Realizó los ejemplos de guía de práctica (40%)			
	Presentó todos los problemas resueltos (20%)			
	Funcionan todos correctamente y sin errores (30%)			
	Envió la carpeta comprimida y organizada adecuadamente en subcarpetas de acuerdo al tipo de recurso (10%)			
	PROMEDIO:			
Investigación complementaria	Envió la investigación complementaria en la fecha indicada (20%)			
	Resolvió todos los ejercicios planteados en la investigación (40%)			
	Funcionaron correctamente y sin ningún mensaje de error a nivel de consola o ejecución (4 0%)			
	PROMEDIO:			