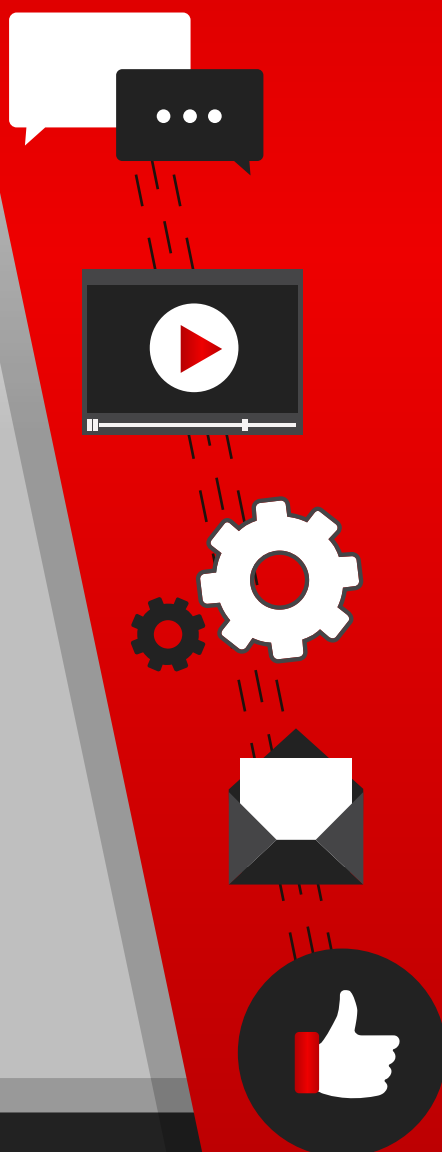


DESARROLLO DE
APLICACIONES WEB EN



FAVA - Formación en Ambientes Virtuales de Aprendizaje

SENA - Servicio Nacional de Aprendizaje.

ESTRUCTURA DE CONTENIDOS

Pág.

1. Introducción a Java Platform EE.	5
1.1 Java Server Faces o JSF.	5
1.1.1 Faces servlet.	6
1.1.2. Páginas y componentes.	6
1.1.3 Facelets.	7
1.1.4 Backing beans	7
1.1.5 Lenguaje de expresiones o EL.	7
1.1.6. Servidor de aplicaciones	8
2. Plan para el desarrollo de la aplicación.	9
2.1 Creación de la base datos.	9
3. Creación del proyecto en Netbeans 8.2	12
4. Creación de la capa de persistencia o JPA (Java Persistence API).	16
5. Creación de los facelets.	23
6. Maquetación de la aplicación.	30
6.1 Conceptos de maquetación con JSF.	31
6.2 Aplicación de estilos CSS.	37
6.2.1. Se incluirá un banner en la sección “encabezado”.	38
6.2.2. Aplicación de estilos para la sección de contenido.	39
6.2.3. Aplicación de estilos para la sección del menú.	39
7. Desarrollo de los casos de uso.	41
7.1. Caso de uso “Asignar Cita”.	41
7.1.1. Ajustar el nombre de los campos.	43
7.1.2. Retirar el campo Número de cita.	46
7.1.3. Eliminar las opciones de navegación del formulario.	47
7.1.4. Ajustar la clase Paciente.	48
7.1.5. Ajustar la clase Medico.	48
7.1.6. Ajustar la clase Consultorios.	49
7.2. Caso de uso “Consultar Citas”.	50
7.2.1 Cambiar los nombres de las columnas.	50
7.2.2 Dejar solamente la opción de eliminar cita.	52
7.2.3 Eliminar los enlaces del formulario.	53
8. Construcción del instalador de la aplicación.	54
8.1 Creación del instalador usando Netbeans.	54
Glosario	57
Bibliografía	58
Control del documento	59

Desarrollo de aplicaciones web JAVA

Introducción



El desarrollo de aplicaciones en Java ha evolucionado desde la introducción de la especificación Java SE o Standard Edition. Originalmente Java permitía el desarrollo de aplicaciones cliente-servidor que se ejecutaban en una máquina virtual y además se podían portar a otros sistemas operativos en los cuales estuviera disponible una máquina virtual Java como Windows, Linux, MacOS entre otros.

Debido al auge de internet y a la gran comunidad de desarrolladores Java que requerían hacer aplicaciones para la web se hizo necesaria la introducción de una serie de herramientas o tecnologías conocidas en un principio como J2EE y actualmente como Java Platform Enterprise Edition.

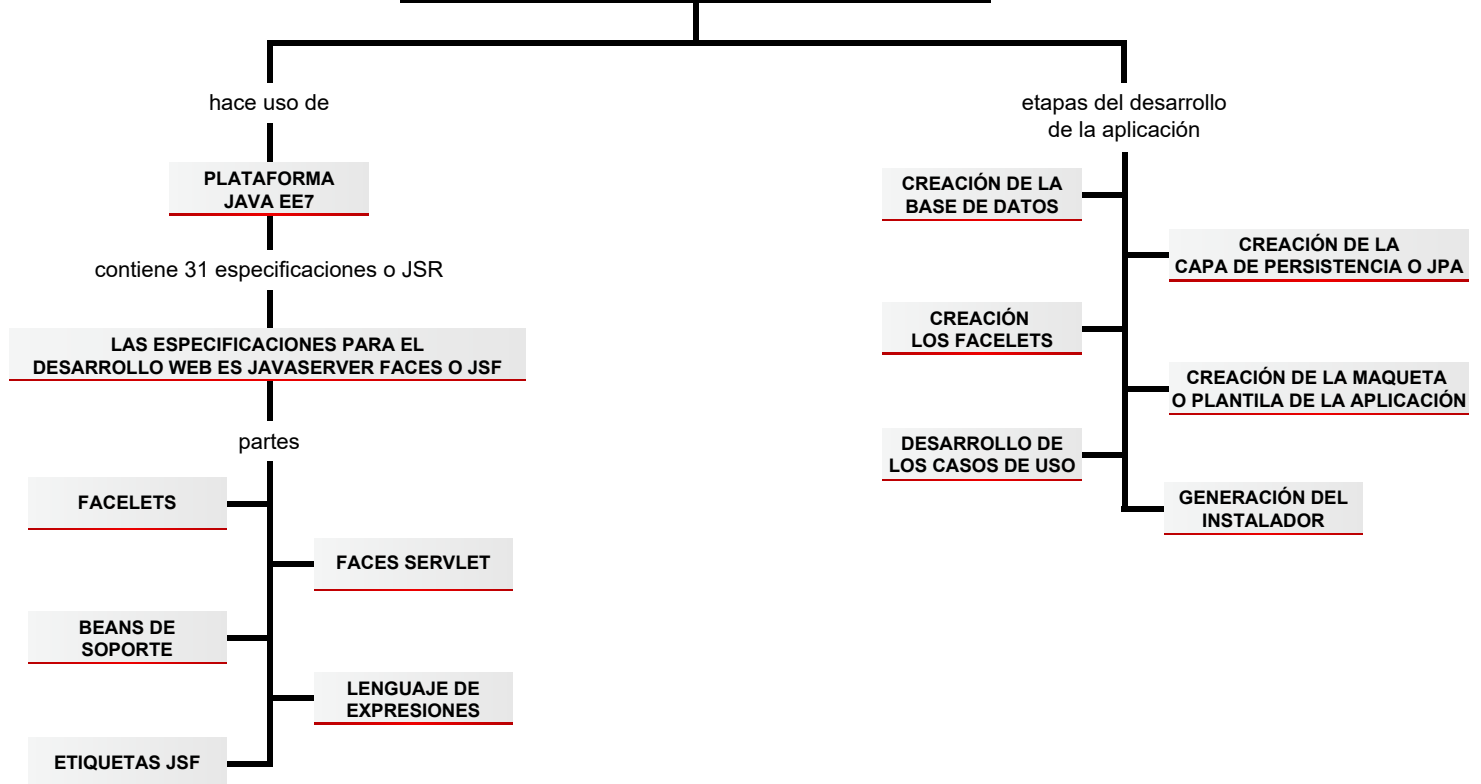
La plataforma Java Platform EE introduce una serie de especificaciones para la construcción de aplicaciones web de n capas que se ejecutan en un servidor de aplicaciones.

En este recurso se desarrollará una aplicación desde cero usando dos de las especificaciones introducidas en Java Platform EE a saber: JavaServer Faces o JSF y la tecnología de Facelets. También se usarán conceptos ya vistos en el programa ADSI como son JDBC, diseño web, SQL entre otros.

La plataforma Java Platform EE contiene más especificaciones que permiten realizar aplicaciones web empresariales como Enterprise Java Beans, Web services, entre otros, que no se abordarán en este recurso. El aprendiz podrá hacer uso de la bibliografía y los recursos complementarios para profundizar en el tema.

Es importante que el aprendiz haya comprendido los contenidos de los objetos de aprendizaje sobre diseño web y bases de datos ya que este recurso requiere de esos conocimientos. También debe tener claridad sobre el patrón de diseño Modelo Vista Controlador o MVC introducido en recursos anteriores.

Mapa de contenido



Desarrollo de contenidos

1. Introducción a Java Platform EE.

La plataforma Java Platform EE, originalmente J2EE, nació como respuesta al creciente número de tecnologías que las empresas debían apropiarse para dar soluciones a sus necesidades de negocio. Su primera versión fue lanzada en 1999 como la punta de lanza de diez especificaciones JSR o Java Specification Request. Desde entonces su evolución se puede resumir en la figura 1.1.

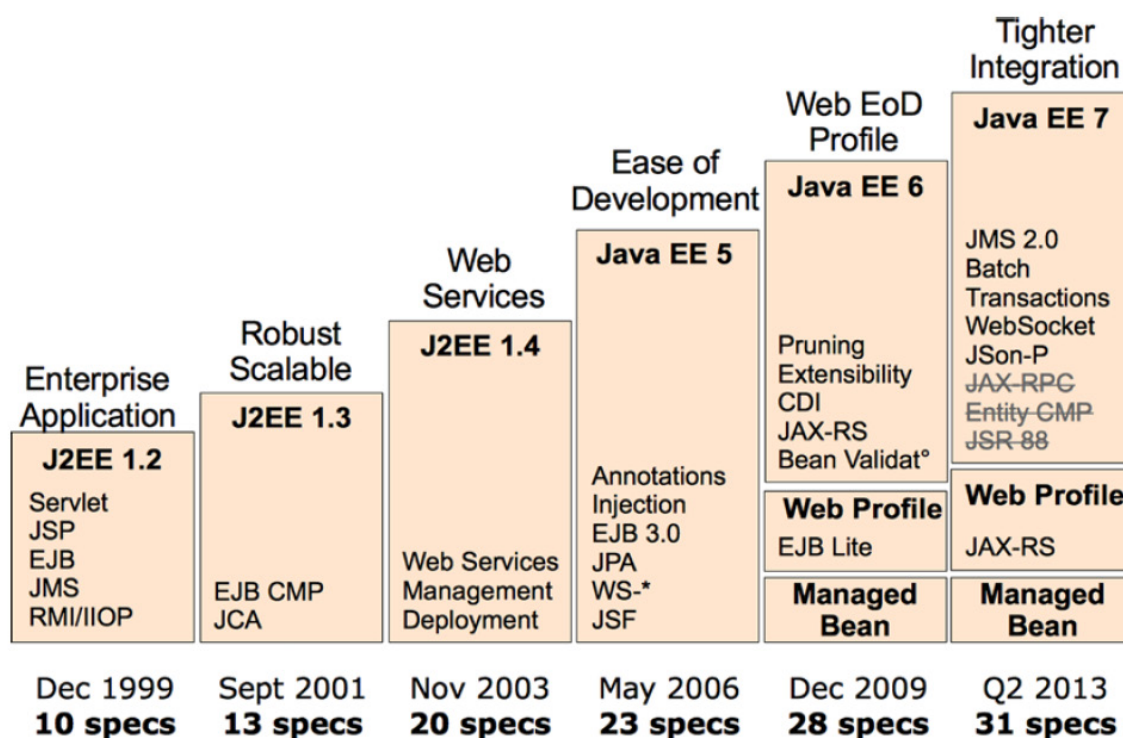


Figura 1.1 Evolución de la plataforma Java EE
Tomado de (Goncalves, 2012)

1.1 Java Server Faces o JSF.

JSF es un framework que permite el desarrollo de aplicaciones web usando el patrón Modelo-Vista-Controlador. Las aplicaciones desarrolladas con JSF interceptan las llamadas al servidor HTTP a través de los Faces servlets y producen código HTML que se despliega en los navegadores.

Lo anterior significa que JSF proporciona las herramientas para la realización de aplicaciones web dinámicas y además suministra las herramientas para la programación tanto del lado del servidor como del lado del cliente.

La arquitectura de una aplicación JSF es la siguiente:

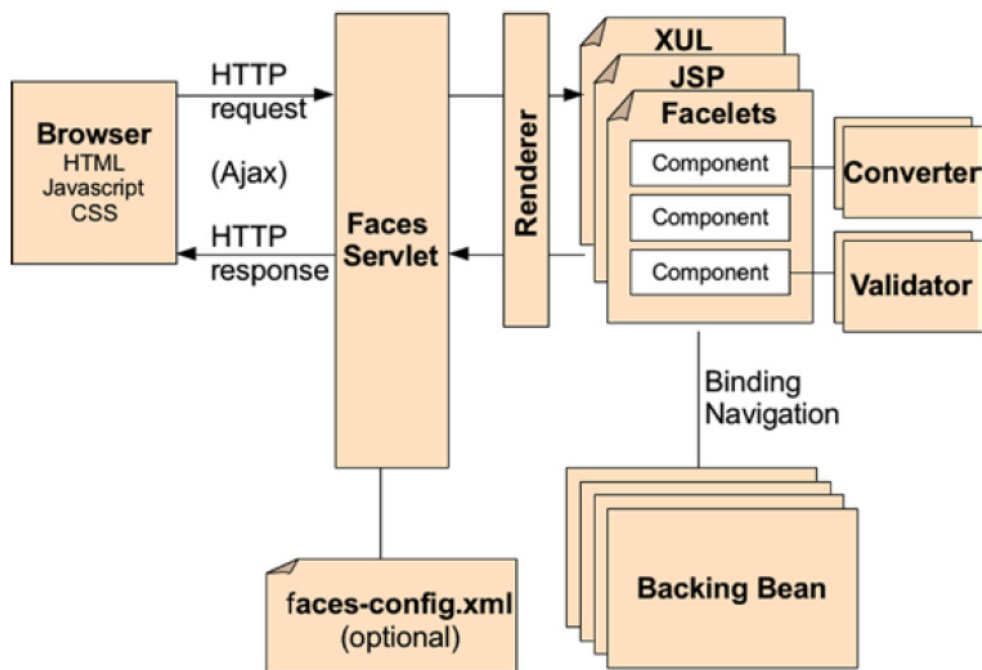


Figura 1.2. Arquitectura de una aplicación web JSF
Tomado de (Goncalves, 2012)

En general una aplicación JSF contiene lo siguiente (Geary, 2010):

1. Un conjunto de componentes para desarrollo de la interfaz o UI.
2. Un modelo de programación orientado a eventos.
3. Un modelo de componentes que permite a otros programadores suministrar código para las aplicaciones.

1.1.1 Faces servlet.

Este servlet (servicio que se ejecuta en el servidor de aplicaciones) se encarga de procesar el ciclo de vida de las solicitudes que envían los clientes de las aplicaciones. Se puede decir que es el motor que transforma las entradas y construye las salidas con base en los demás componentes de una aplicación JSF.

1.1.2. Páginas y componentes.

Como el framework JSF tiene que enviar respuesta a los navegadores de los clientes requiere de tecnologías para el dibujo de páginas o PDL (Page Description Language). Las aplicaciones pueden usar las tecnologías disponibles como son HTML y CSS. Pero el método preferido son los facelets.

1.1.3 Facelets.

Los facelets son páginas formadas por un árbol de componentes que suministran la funcionalidad para comunicarse con el usuario final. JSF tiene un conjunto de componentes que permiten al programador desarrollar sus páginas.

Los facelets son además páginas XHTML que usan, entre otros, las etiquetas provistas por el framework JSF (xmlns:h="http://xmlns.jcp.org/jsf/html").

Los componentes provistos por JSF pueden ser no visuales como la etiqueta <h:head> <h:body> o visuales como la etiqueta <h:outputLabel>. Los facelets pueden usar varios PDL como son HTML, CSS, JSF Tags.

En este recurso se utiliza la estrategia de aprendizaje basado en ejemplos. A lo largo del mismo se introducirá código que será posteriormente analizado.

Para una completa referencia del lenguaje EL y el conjunto de etiquetas suministradas por JSF se recomienda seguir los recursos que se presentan en la bibliografía de este recurso.

1.1.4 Backing beans.

Los beans de soporte o “backing beans” es una clase java especializada que sincroniza valores entre componentes, procesa la lógica del negocio y maneja la navegación entre páginas. Se pueden ver los beans como los controladores del modelo MVC.

1.1.5 Lenguaje de expresiones o EL.

Es el enlace entre la capa de presentación representada por los facelets y la capa de controladores suministrada por los beans de soporte o backing beans.

A través del lenguaje de expresiones las páginas pueden consultar los valores y atributos de las clases del modelo de datos y demás elementos almacenados.

La sintaxis básica del lenguaje de expresiones es:

```
{expresión}
```

Las sentencias del tipo `{expresión}` son interpretadas al momento de ejecución de las páginas. Las expresiones pueden usar la mayoría de los operadores de Java como son: aritméticos, relacionales, lógicos, entre otros.

A continuación un ejemplo del uso de lenguaje de expresiones:

```
#{productos.precio}
```

El anterior código traerá el atributo “precio” de la clase “productos” y lo asignará al control que defina el programador.

1.1.6. Servidor de aplicaciones.

Un elemento importante de la plataforma Java EE es el servidor de aplicaciones el cual debe estar habilitado para soportar las especificaciones de la misma.

El servidor usado para probar las especificaciones JSR y que cumple con las mismas es GlassFish. Al momento de escribir el recurso está en la versión 5. Existen otros servidores de aplicaciones para la plataforma Java EE como son: Jboss, WebLogic, Websphere, entre otros.

Para este recurso se usará la versión de GlassFish que viene integrada y manejada por Netbeans 8.2.

También se pueden desarrollar aplicaciones web Java con el servidor de aplicaciones Tomcat el cual es mantenido por la fundación Apache. Sin embargo Apache Tomcat es básicamente un servidor de servlets y no implementa las demás JSR de la plataforma como sí lo hace GlassFish.

2. Plan para el desarrollo de la aplicación.

El presente recurso tiene como objetivo llevar a cabo el caso de uso de Gestión Odontológica de acuerdo al siguiente diagrama:

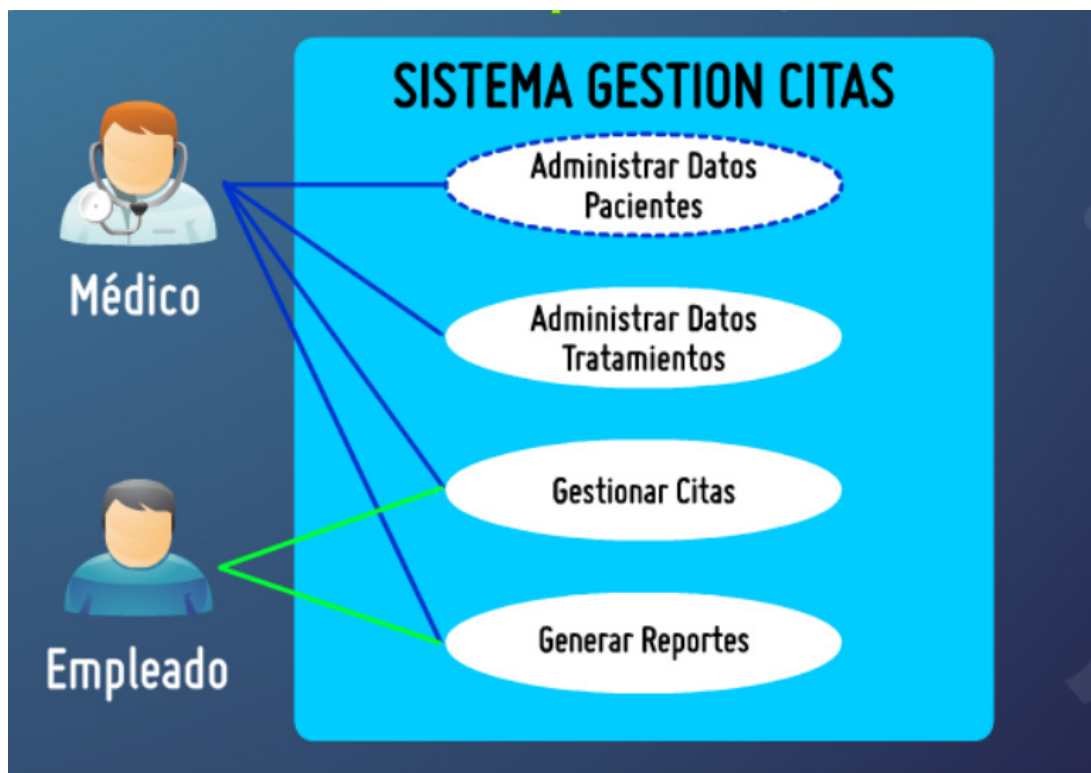


Figura 2.1 Caso de uso a desarrollar.

Los pasos que se darán para desarrollar la aplicación son:

- 1) Crear el proyecto en Netbeans.
- 2) Generar la capa de persistencia o clases de la base de datos.
- 3) Generar los facelets.
- 4) Realizar la maquetación de la aplicación.
- 5) Hacer ajustes al código generado para elaborar los casos de uso.

Para el desarrollo

2.1 Creación de la base datos.

Para este recurso se usará la base de datos diseñada en los objetos de aprendizaje del motor MySQL.

El script de creación de la base es el siguiente:

```
CREATE DATABASE citas ;
USE citas ;
CREATE TABLE `Consultorios` (
  `ConNumero` int(3) NOT NULL,
  `ConNombre` varchar(50) NOT NULL,
  PRIMARY KEY (`ConNumero`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `Medicos` (
  `MedIdentificacion` char(10) NOT NULL,
  `MedNombres` varchar(50) NOT NULL,
  `MedApellidos` varchar(50) NOT NULL,
  PRIMARY KEY (`MedIdentificacion`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `Pacientes` (
  `PacIdentificacion` char(10) NOT NULL,
  `PacNombres` varchar(50) NOT NULL,
  `PacApellidos` varchar(50) DEFAULT NULL,
  `PacFechaNacimiento` date NOT NULL,
  `PacSexo` enum('M','F') NOT NULL,
  PRIMARY KEY (`PacIdentificacion`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `Tratamientos` (
  `TraNumero` int(11) NOT NULL AUTO_INCREMENT,
  `TraFechaAsignado` date NOT NULL,
  `TraDescripcion` text NOT NULL,
  `TraFechaInicio` date NOT NULL,
  `TraFechaFin` date NOT NULL,
  `TraObservaciones` text NOT NULL,
  `TraPaciente` char(10) NOT NULL,
  PRIMARY KEY (`TraNumero`),
  KEY `TraPaciente` (`TraPaciente`),
  CONSTRAINT `Tratamientos_ibfk_1` FOREIGN KEY (`TraPaciente`) REFERENCES `Pacientes` (`PacIdentificacion`)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=latin1;

CREATE TABLE `citas` (
  `CitNumero` int(11) NOT NULL AUTO_INCREMENT,
  `CitFecha` date NOT NULL,
  `CitHora` varchar(10) NOT NULL,
  `CitPaciente` char(10) NOT NULL,
  `CitMedico` char(10) NOT NULL,
  `CitConsultorio` int(3) NOT NULL,
  `CitEstado` enum('Asignada','Cumplida','Solicitada','Cancelada') NOT NULL DEFAULT 'Asignada',
  PRIMARY KEY (`CitNumero`),
  KEY `CitPaciente` (`CitPaciente`),
  KEY `CitMedico` (`CitMedico`),
  KEY `CitConsultorio` (`CitConsultorio`),
  CONSTRAINT `citas_ibfk_1` FOREIGN KEY (`CitPaciente`) REFERENCES `Pacientes` (`PacIdentificacion`),
  CONSTRAINT `citas_ibfk_2` FOREIGN KEY (`CitMedico`) REFERENCES `Medicos` (`MedIdentificacion`),
  CONSTRAINT `citas_ibfk_3` FOREIGN KEY (`CitConsultorio`) REFERENCES `Consultorios` (`ConNumero`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

El modelo relacional de la base de datos es el siguiente:

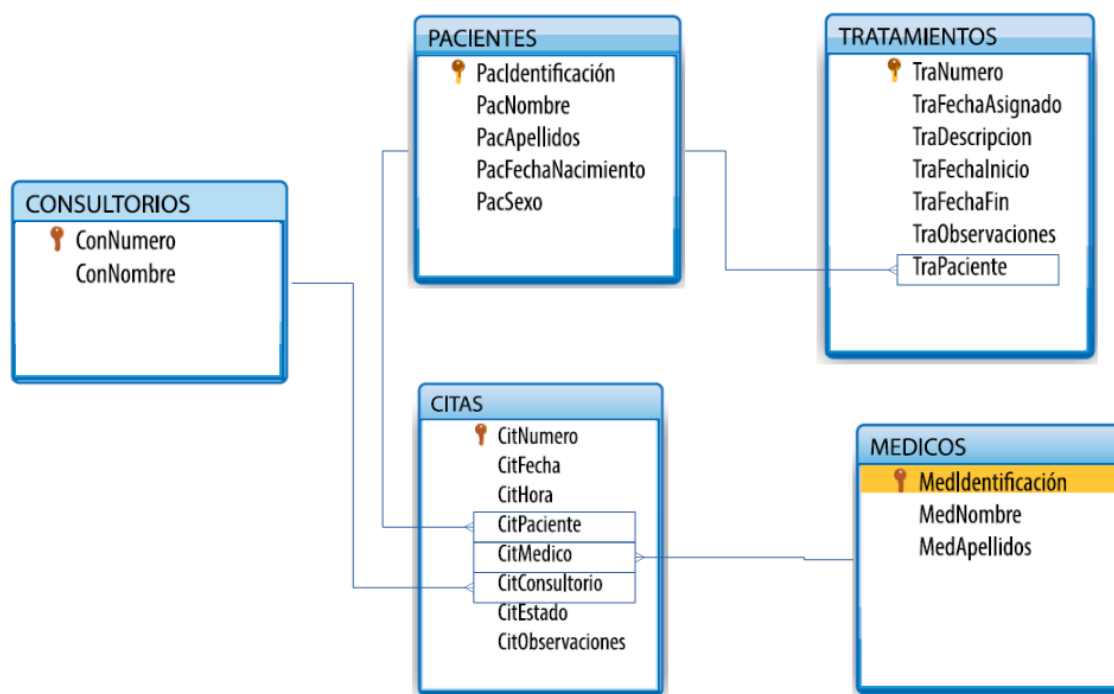


Figura 2.2 Modelo relacional de la base de datos.

3. Creación del proyecto en Netbeans 8.2

Para crear el proyecto se ingresa a Netbeans y se siguen los siguientes pasos:

1) Seleccionar: “File→New project:

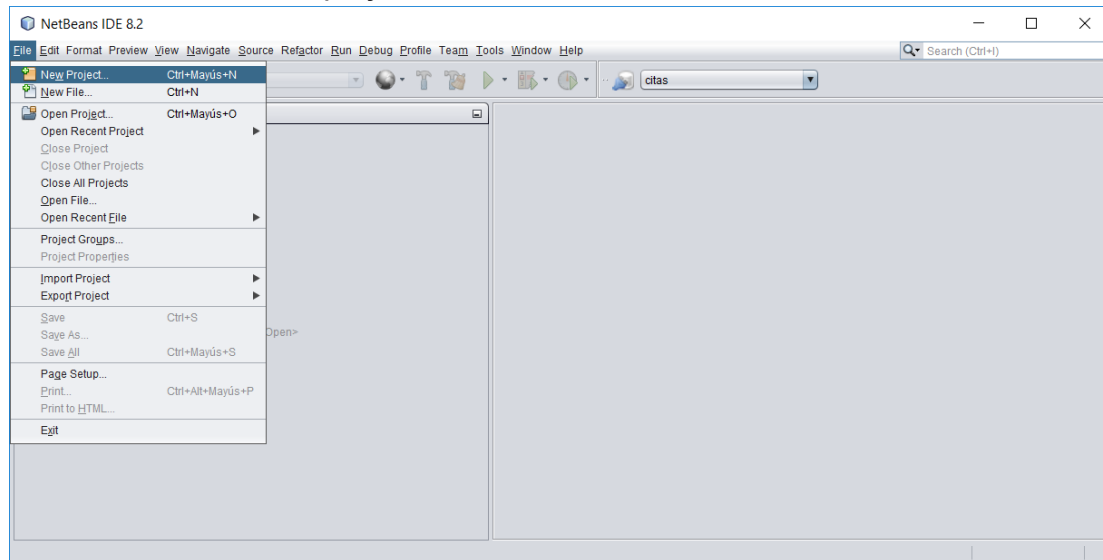


Figura 3.1. Creación del proyecto.

2) Seleccionar “Java web” → “Web application”:

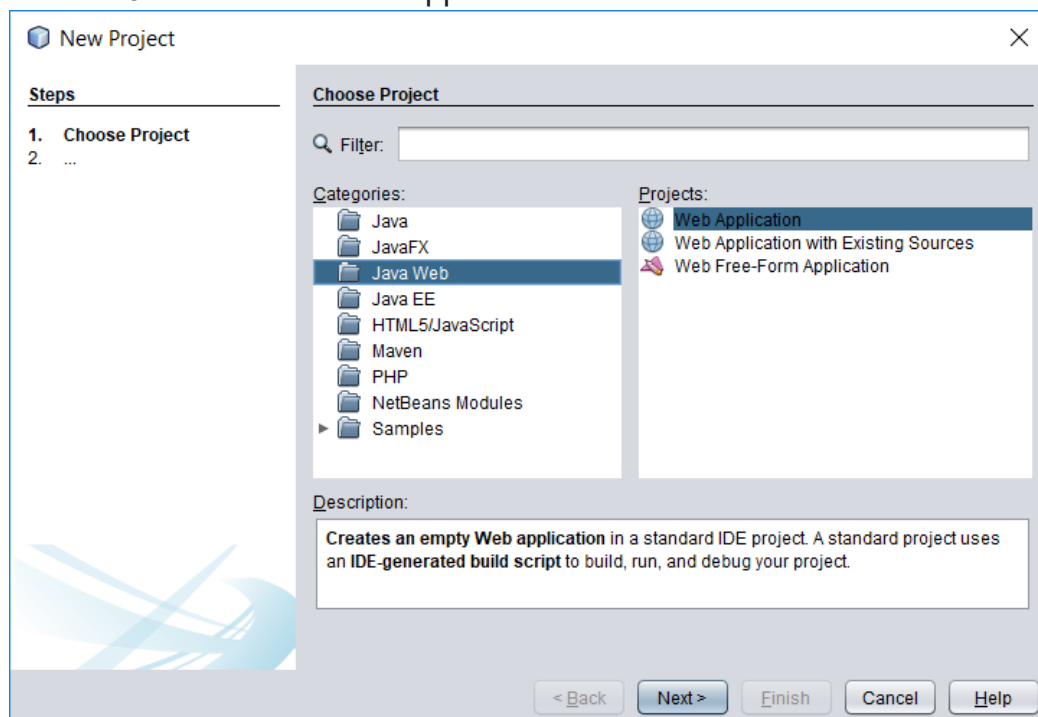


Figura 3.2. Tipo de proyecto Java Web.

3) Ingresar el nombre de la aplicación: “GestionOdontologica_v1”:

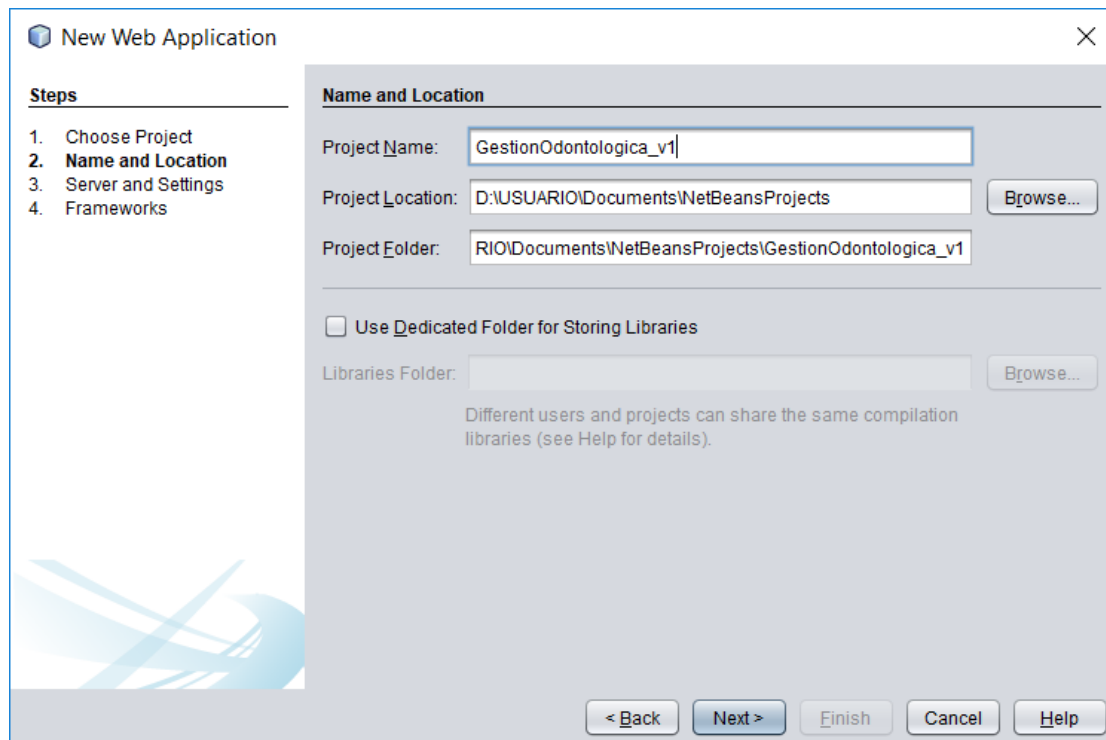


Figura 3.3. Ingreso del nombre del proyecto.

4) Seleccionar el tipo de servidor y versión de la Java Platform EE:

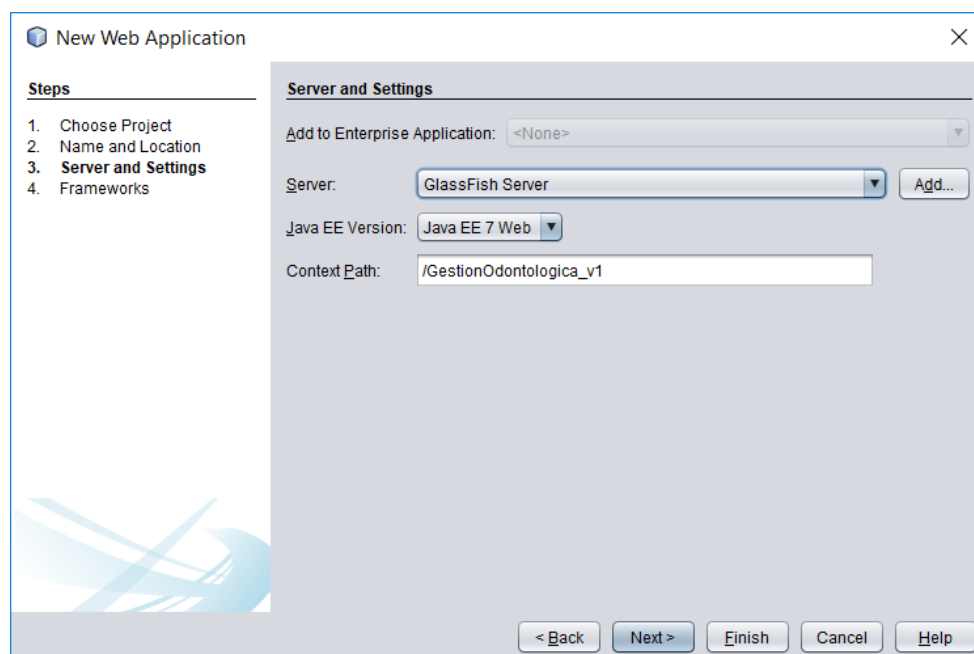


Figura 3.4 Selección del servidor Java EE

Para el presente recurso se usará el servidor Java EE GlassFish. Si no aparece dentro de las opciones del Netbean se debe oprimir el botón “Add” para agregarlo.

La versión de Java Platform será la 7.

5) Selección del framework MVC.

En este recurso se usará el framework JSF o JavaServer Faces en la versión 2.2.

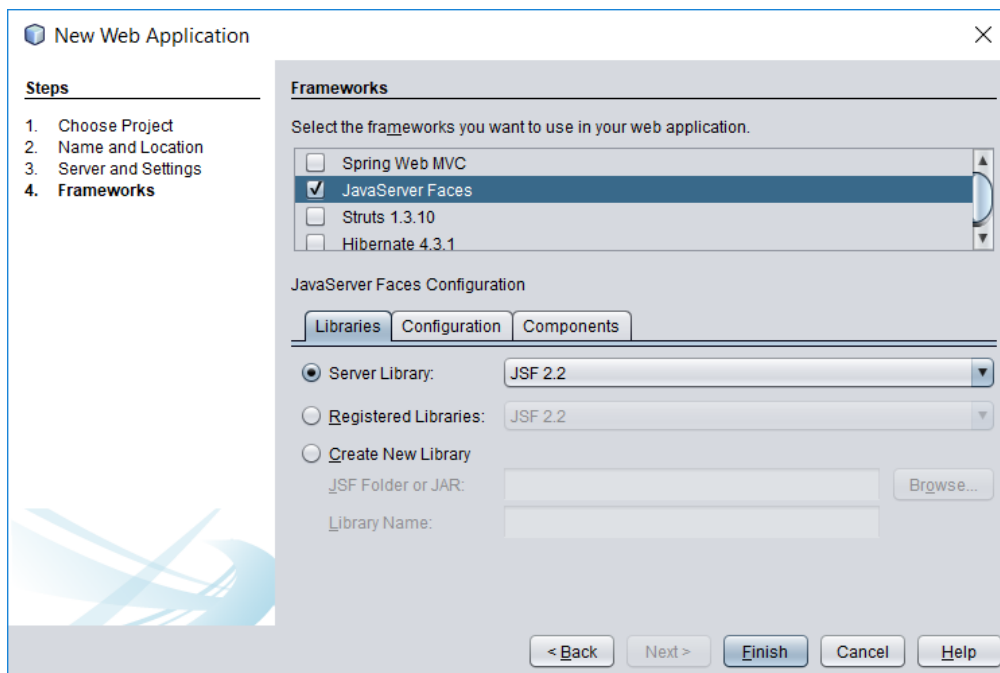


Figura 3.5 Selección del framework

Por último se oprime el botón “Finish” y la aplicación genera el proyecto como se muestra a continuación:

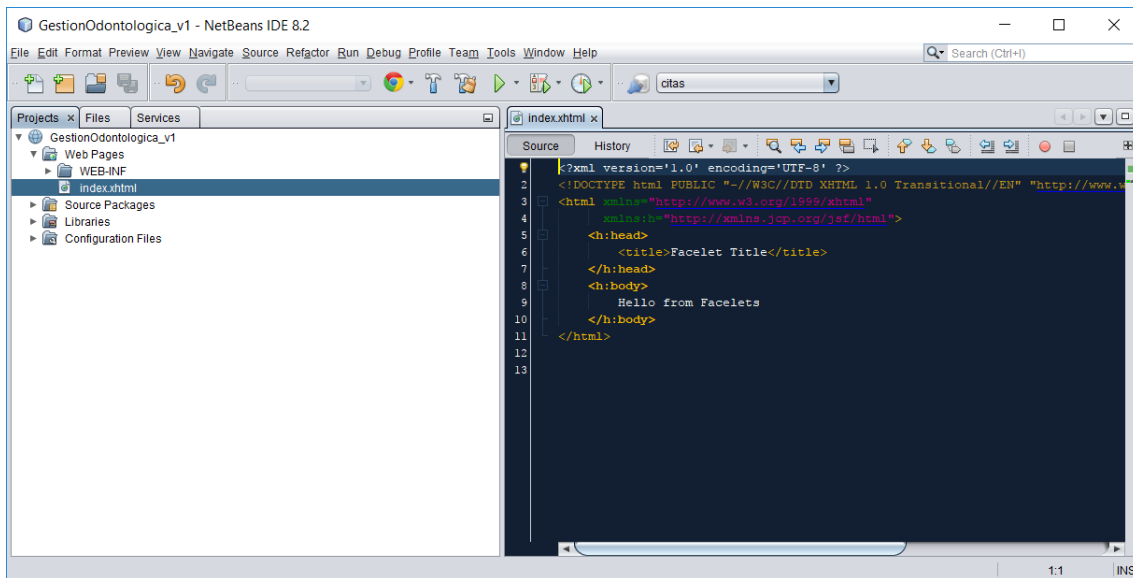


Figura 3.6. Proyecto inicial

4. Creación de la capa de persistencia o JPA (Java Persistence API).

Una vez creado el proyecto se procede a crear las clases que permitirán interactuar directamente con la base de datos. Estas clases son denominadas clases de entidad o “entity classes” y en su conjunto forman la capa de persistencia de la aplicación. Se pueden generar de forma automática usando Netbeans.

Para lo anterior se oprime el botón derecho del ratón sobre el nodo del proyecto: “Web Pages”. Luego se da click sobre “New”→”Entity Classes from Database”.

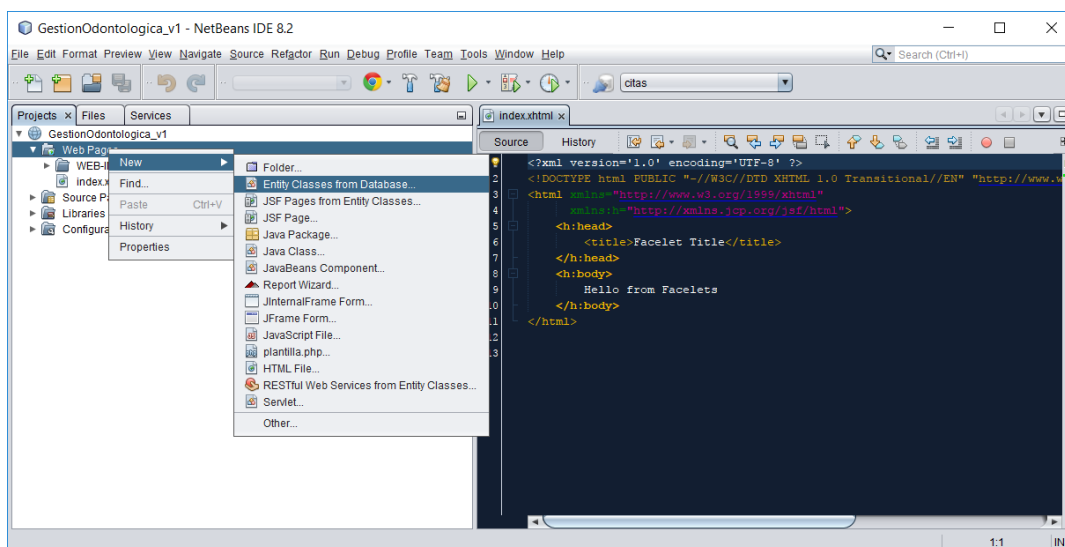


Figura 4.1. Creación de las clases de la base de datos.

Si no aparece la opción “Entity Classes from Database” en la ventana se debe oprimir la opción “Other” que está al final. Aparecerá lo siguiente en pantalla:

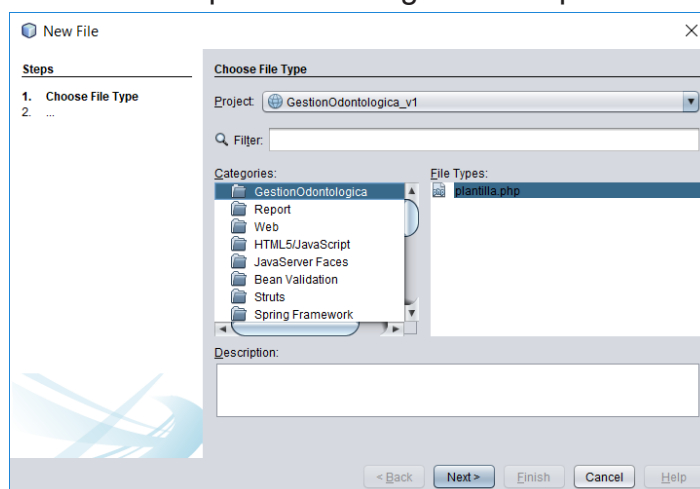


Figura 4.1a. Ubicación de la opción de creación de la JPA

Luego se da el criterio de búsqueda: “Entity” así:

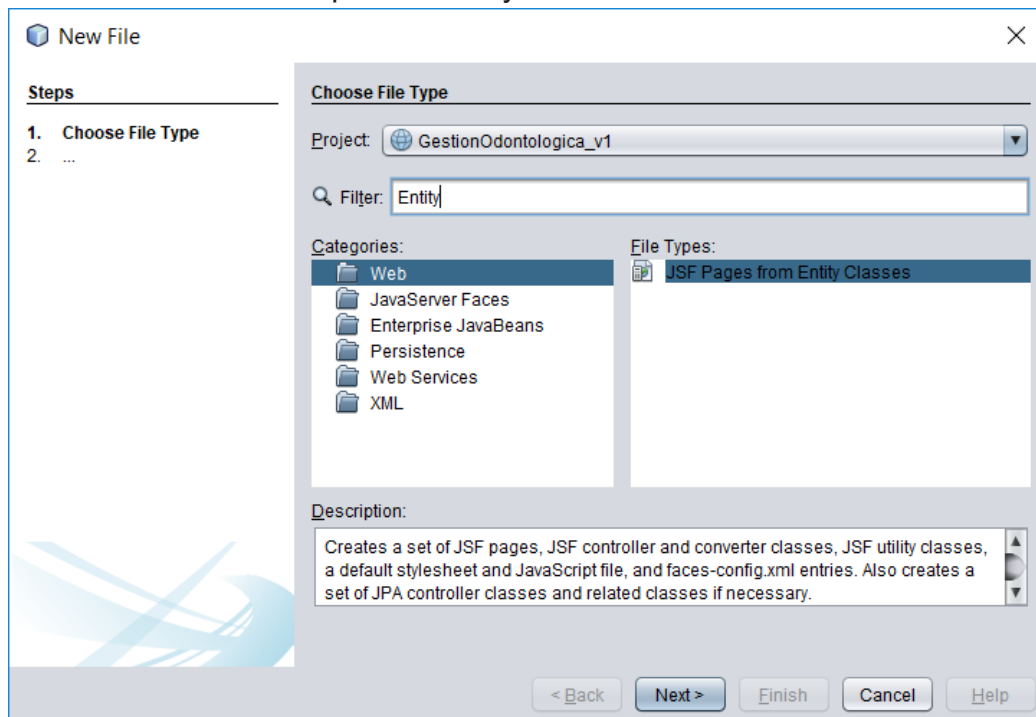


Figura 4.1b. Ubicación de la opción de creación de la JPA

Una vez seleccionada la opción aparece lo siguiente:

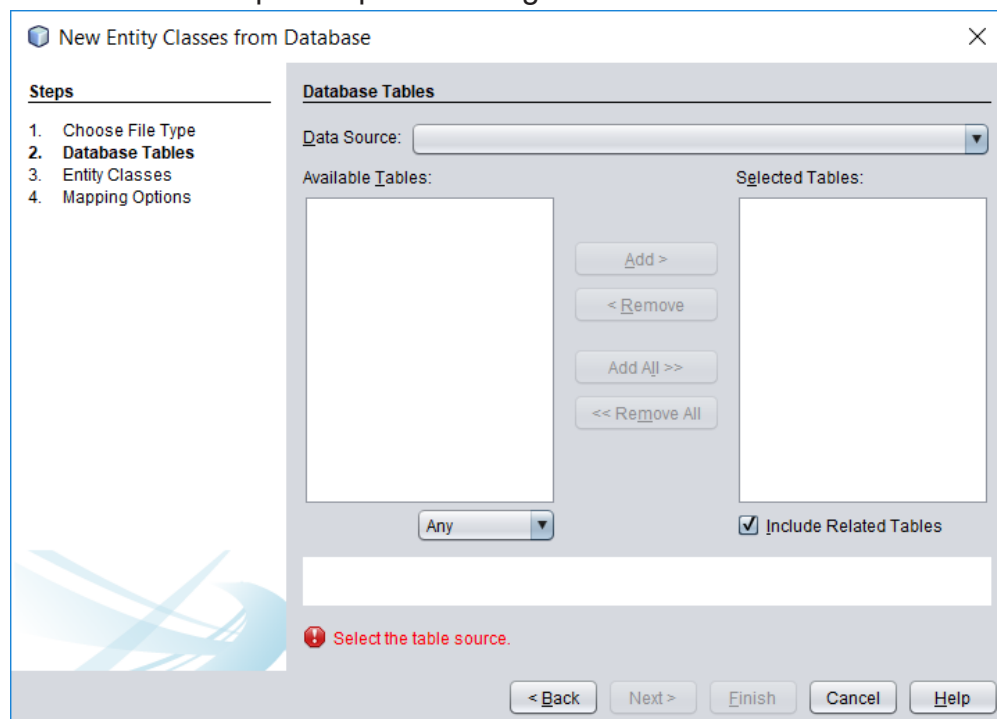


Figura 4.2. Creación de las clases de la base de datos.

En el campo de selección: “Data Source” se selecciona “New Data Source”.

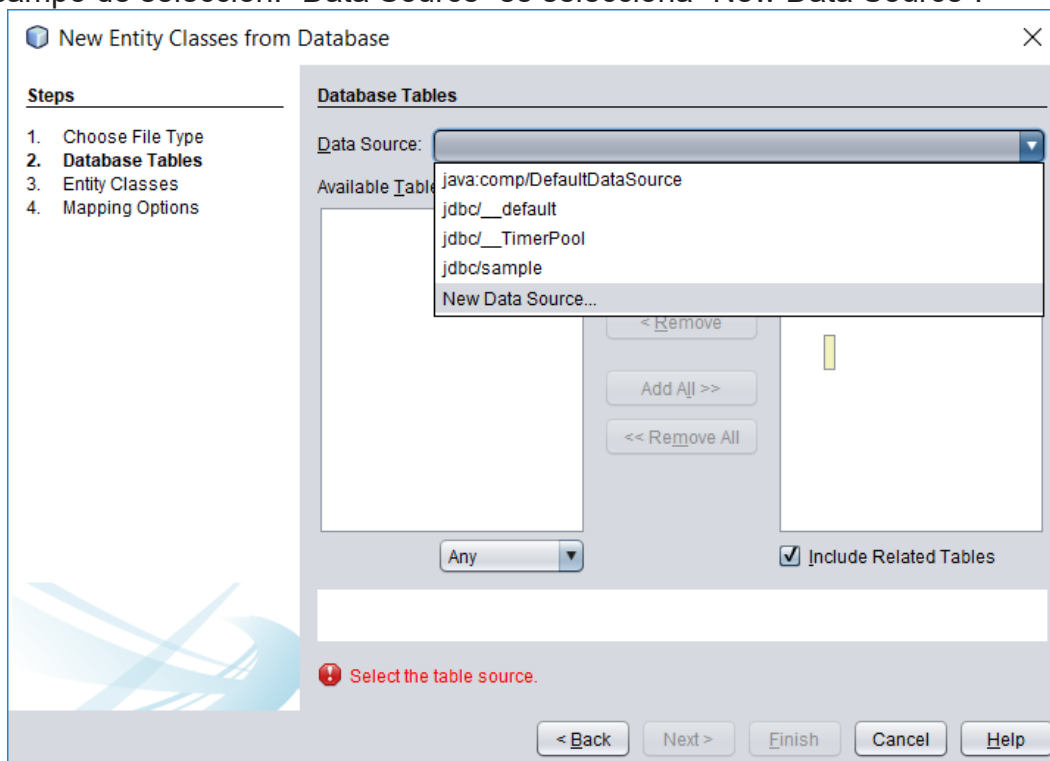


Figura 4.3. Creación de la capa de persistencia

Una vez seleccionada aparece lo siguiente en pantalla:

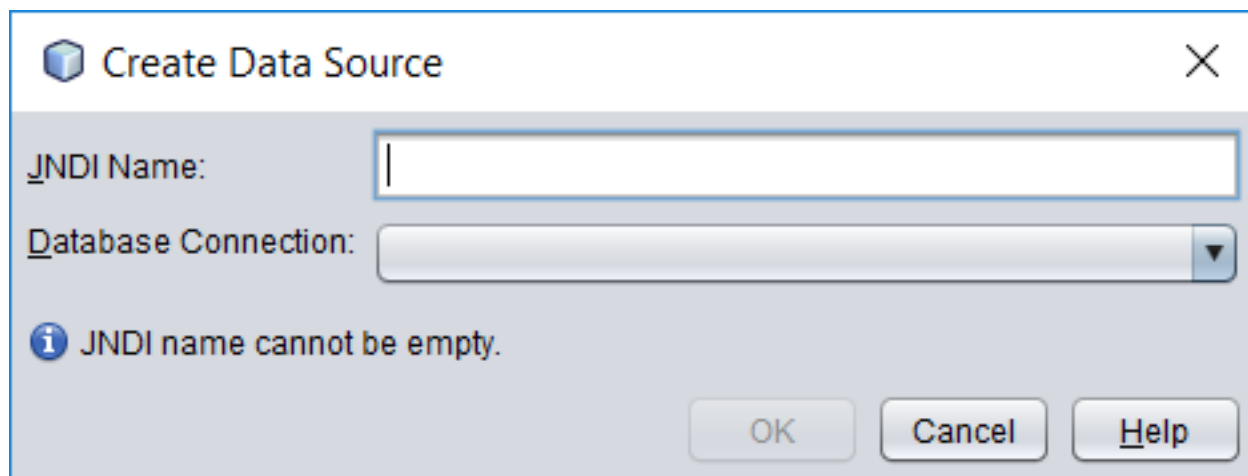


Figura 4.4. Creación de la capa de persistencia

Se debe introducir el texto: “jndi/jdbc” y a continuación seleccionar la base de datos del proyecto. En este caso seleccionamos la base de datos “citas” que se realizó en el motor Mysql.

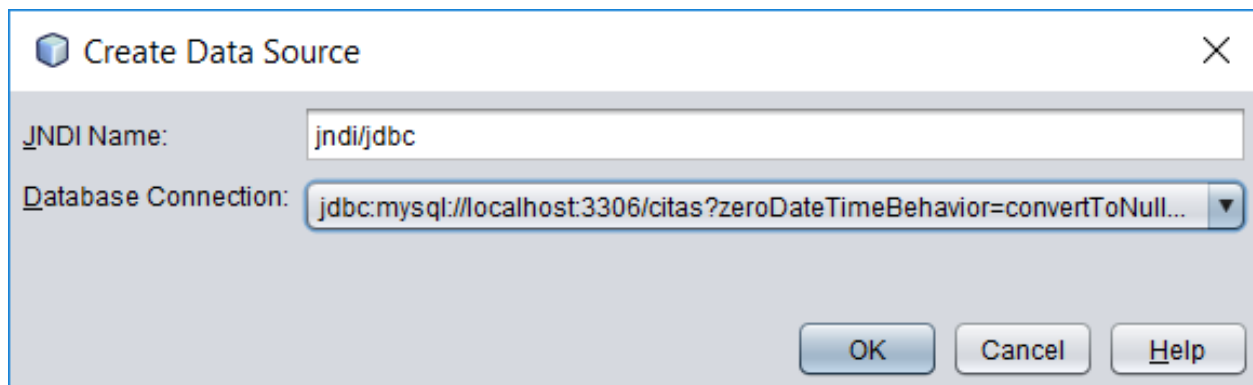


Figura 4.5. Creación de la capa de persistencia

Una vez ingresada la conexión de la base de datos se oprime el botón “Ok”. Aparecerá lo siguiente en pantalla:

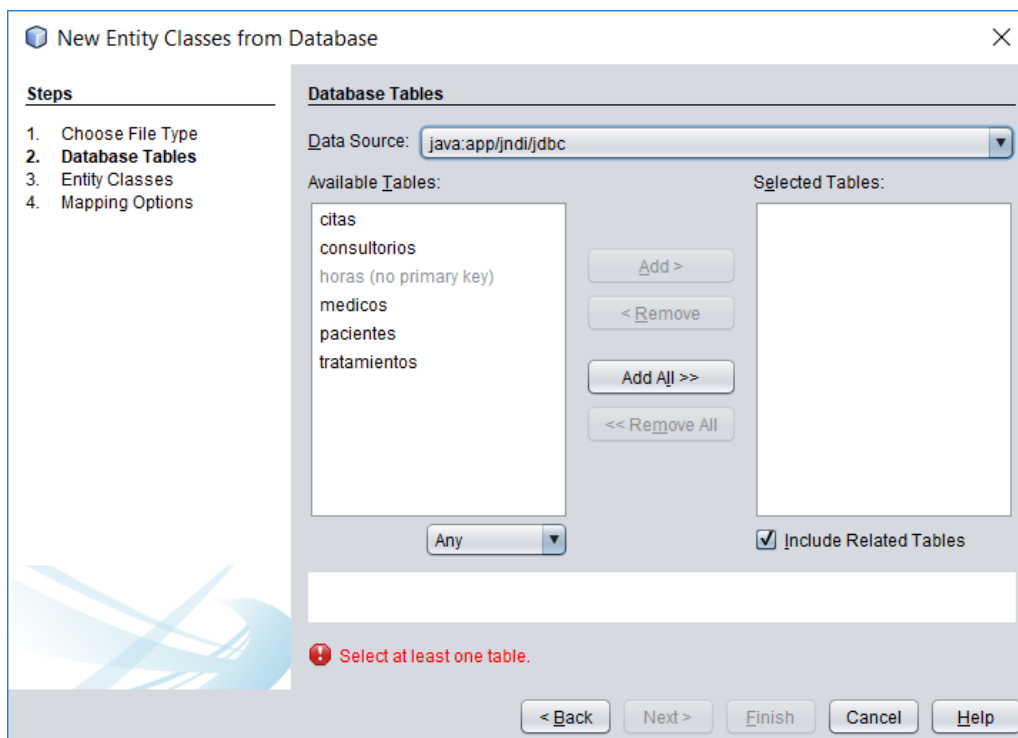


Figura 4.6. Creación de la capa de persistencia

La ventana muestra las tablas que se fueron creadas en la base de datos. Todas están habilitadas excepto aquellas que no tienen una llave primaria como es el caso de la tabla “horas”.

Se debe oprimir el botón “Add All”. Luego aparece lo siguiente en pantalla:

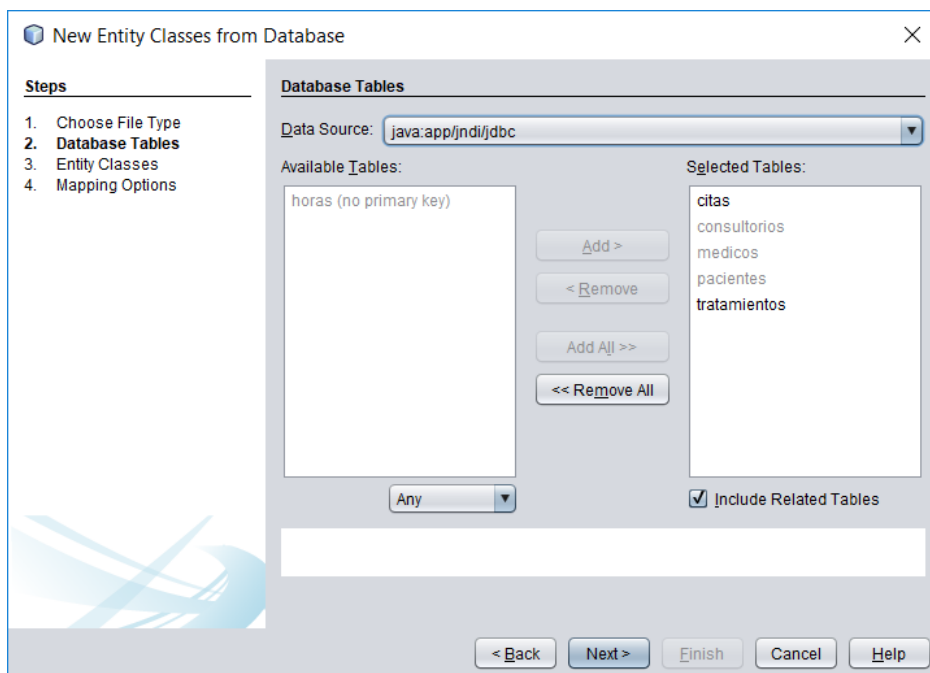


Figura 4.7 Creación de la capa de persistencia

Se debe dar click en el botón “Next”. Aparecerá lo siguiente en pantalla:

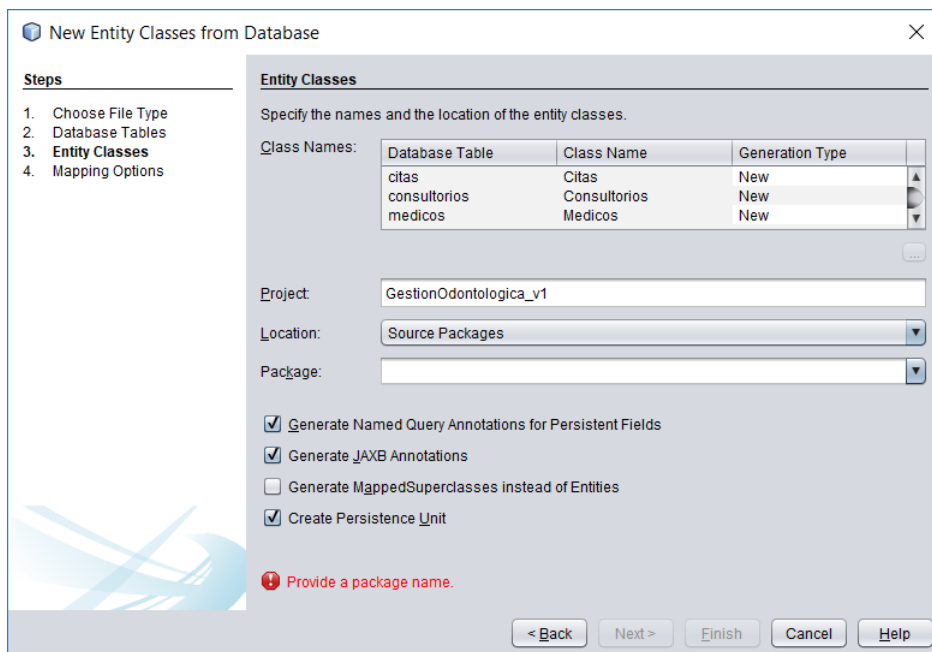


Figura 4.8. Creación de la capa de persistencia

Se debe ingresar “jpa.entities” para el nombre del paquete (Package). Este nombre es un estándar comúnmente usado. Se puede usar otro de acuerdo al estándar que usen en el equipo de desarrollo del aprendiz.

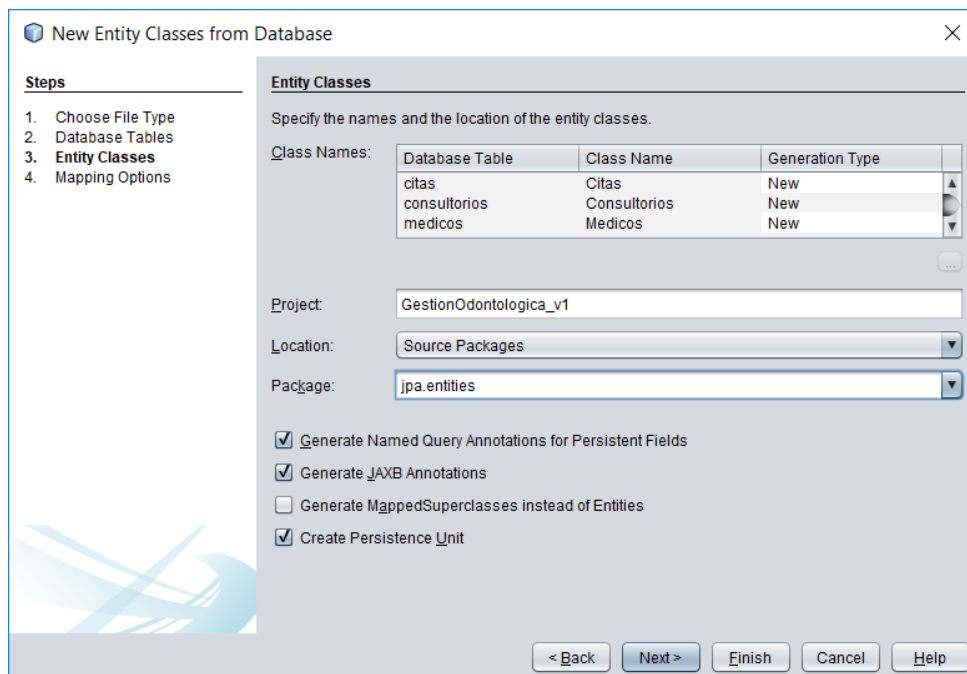


Figura 4.9. Creación de la capa de persistencia

Se oprime el botón “Next”. Aparece lo siguiente en pantalla:

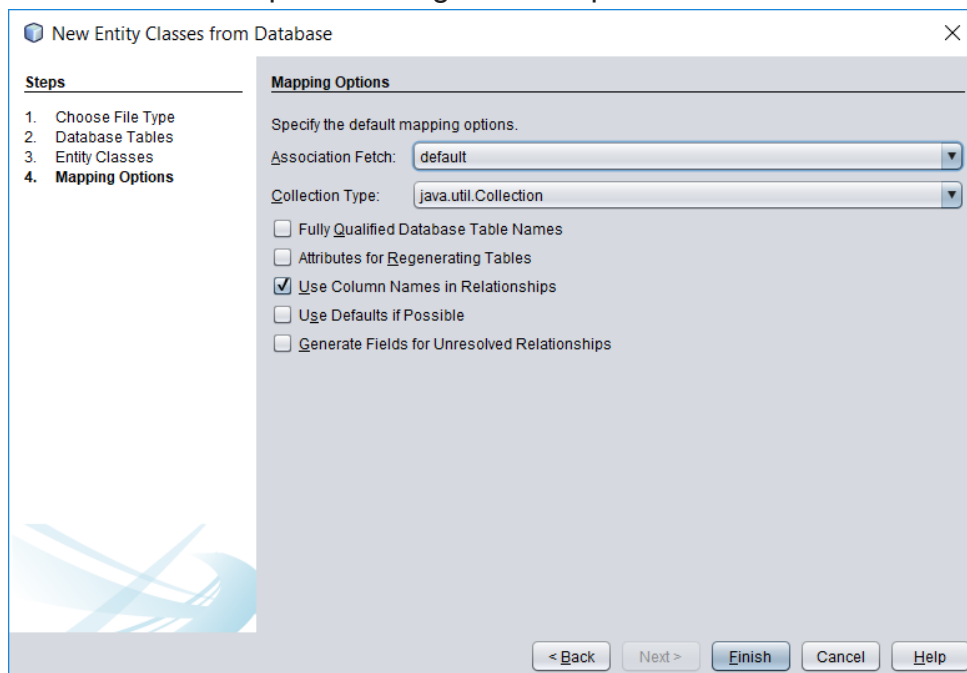


Figura 4.10. Creación de la capa de persistencia

Esta ventana se deja sin cambios. Por último se oprime el botón “Finish”.

En el proyecto aparecerán cinco nuevas clases en el nodo de “Source Packages” como se muestra a continuación:

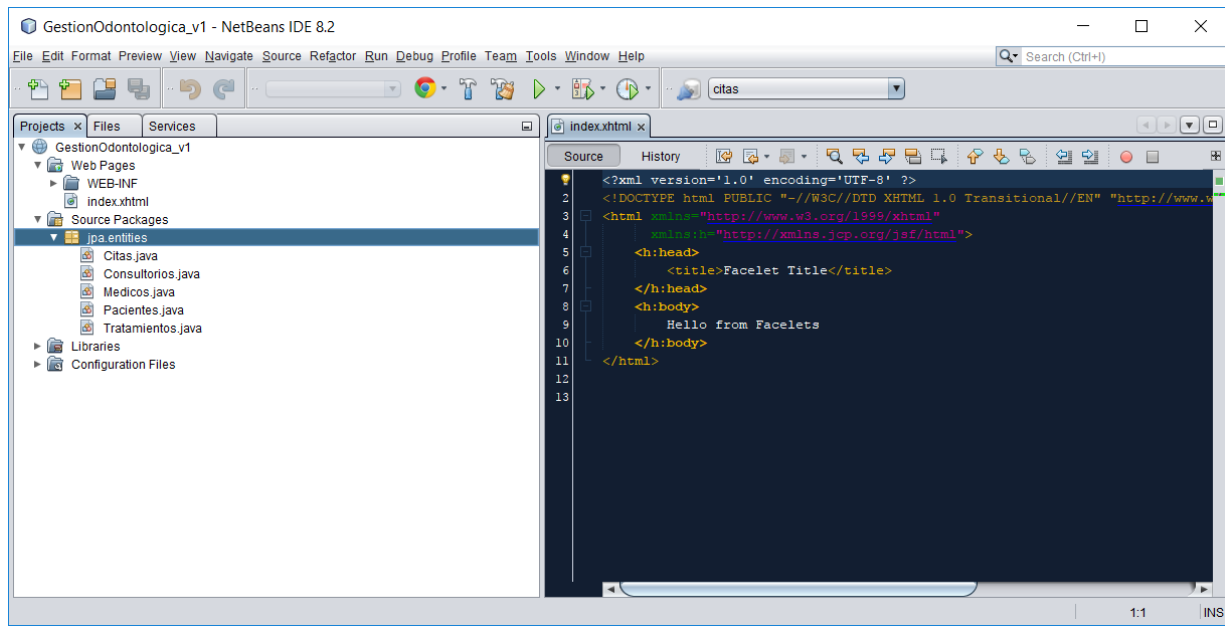


Figura 4.11 Creación de la capa de persistencia

5. Creación de los facelets.

Una vez creada la capa de persistencia se procede a generar los facelets con el apoyo de Netbeans.

Para los anterior se oprime click derecho sobre el nodo “Web Pages” y a continuación se selecciona “New” → “JSP Pages from Entity classes”.

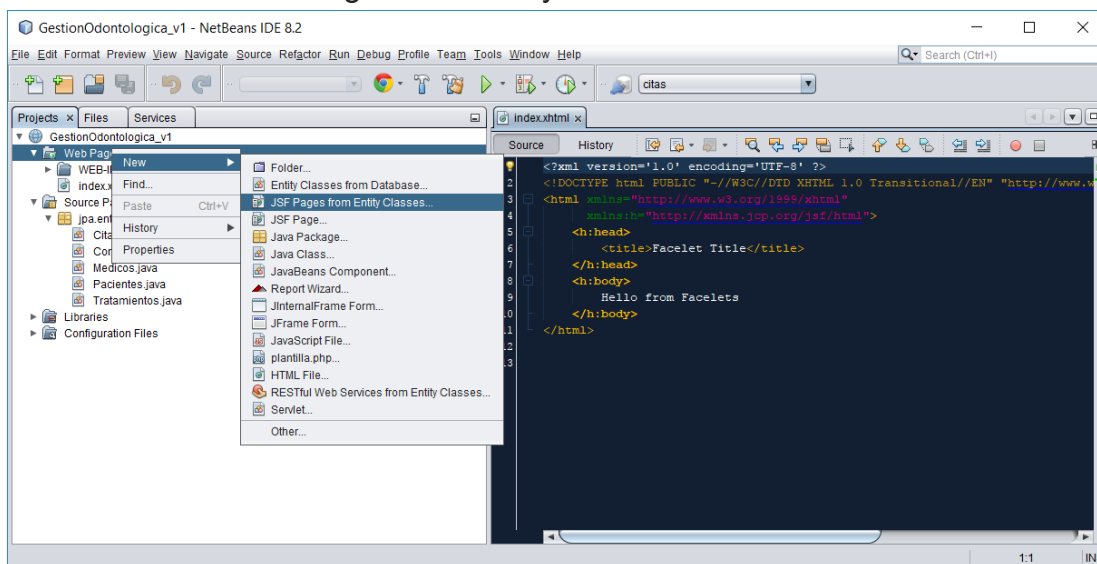


Figura 5.1 Creación de los facelets paso 1

Una vez seleccionada la opción aparece lo siguiente en pantalla:

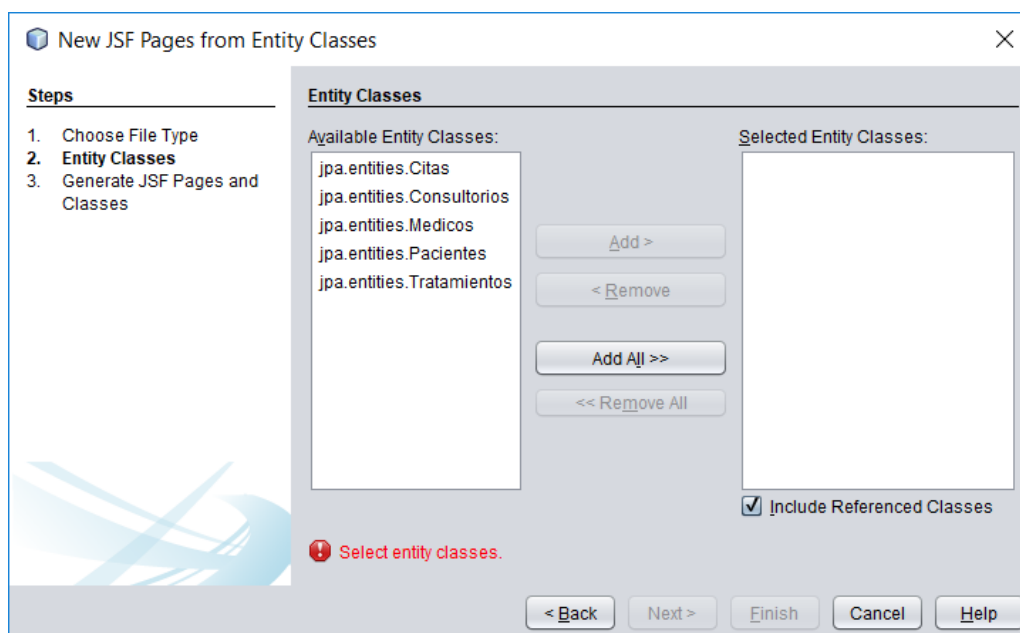


Figura 5.2 Creación de los facelets paso 2

Se oprime el botón “Add All >>” para seleccionar todas las entidades.

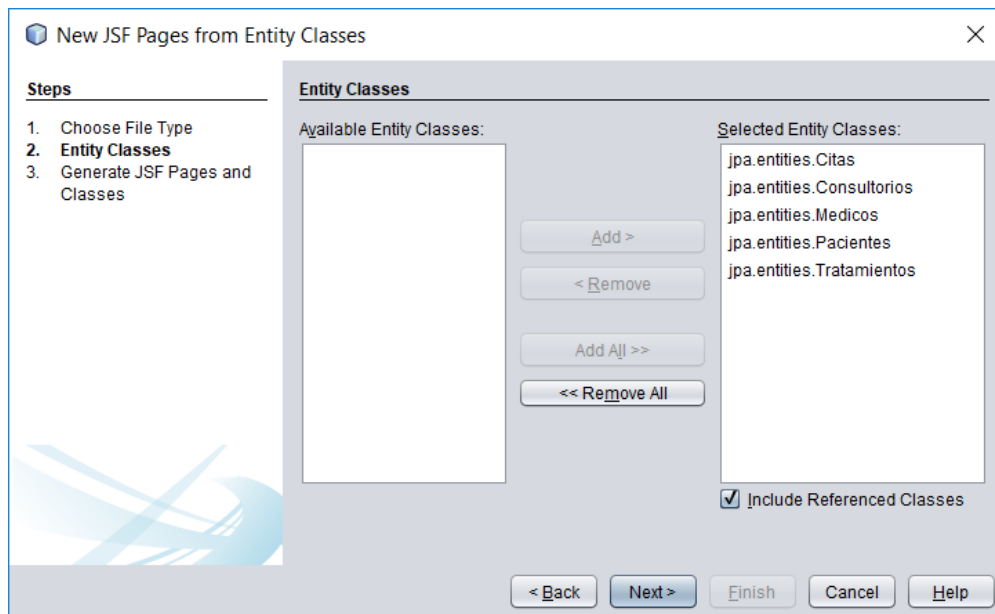


Figura 5.3 Creación de los facelets paso 3

Se oprime el botón “Next” para continuar. Aparece lo siguiente en pantalla:

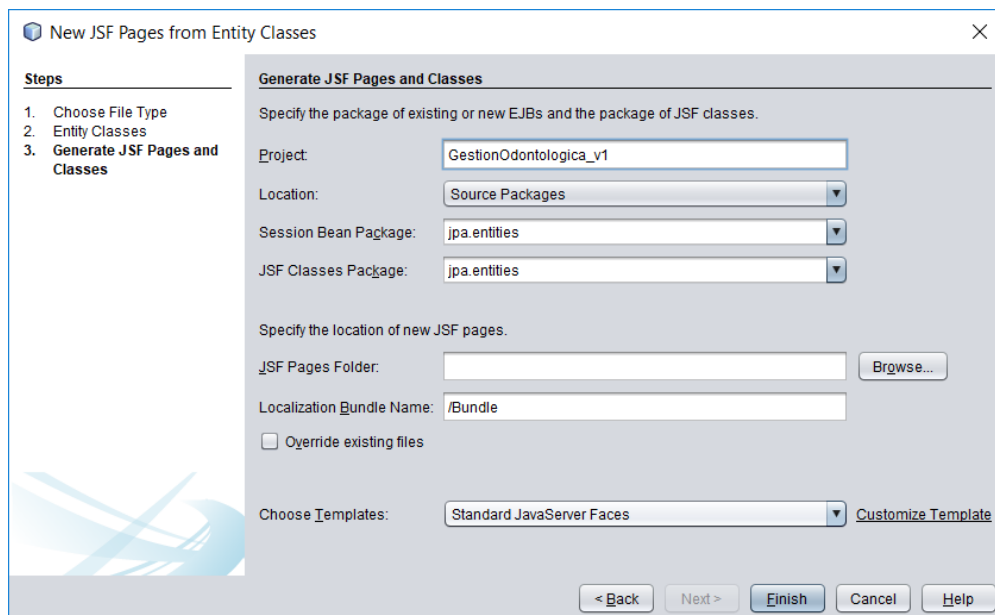


Figura 5.4 Creación de los facelets paso 4

En la opción “Session Bean Package” se debe ingresar “jpa.sessions”.

En la opción “JSF Classes Package” se debe ingresar “jsf”.

En la opción “Localization Bundle Name” se debe ingresar “/resources/Bundle”.

Por último se da click sobre el botón “Finish”.

El proyecto ahora tiene la siguiente presentación:

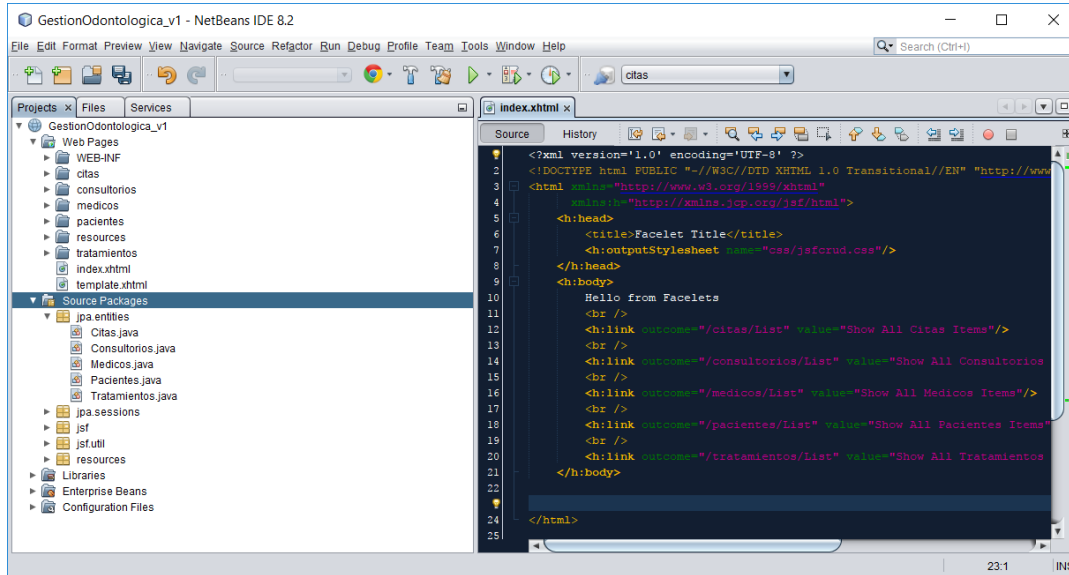


Figura 5.5 Creación de los facelets paso 5

Se puede observar que se crearon las carpetas: citas, consultorios, médicos, pacientes y tratamientos.

En cada carpeta se crearon los archivos: Create.xhtml, Edit.xhtml, List.xhtml y View.xhtml como se muestra a continuación.

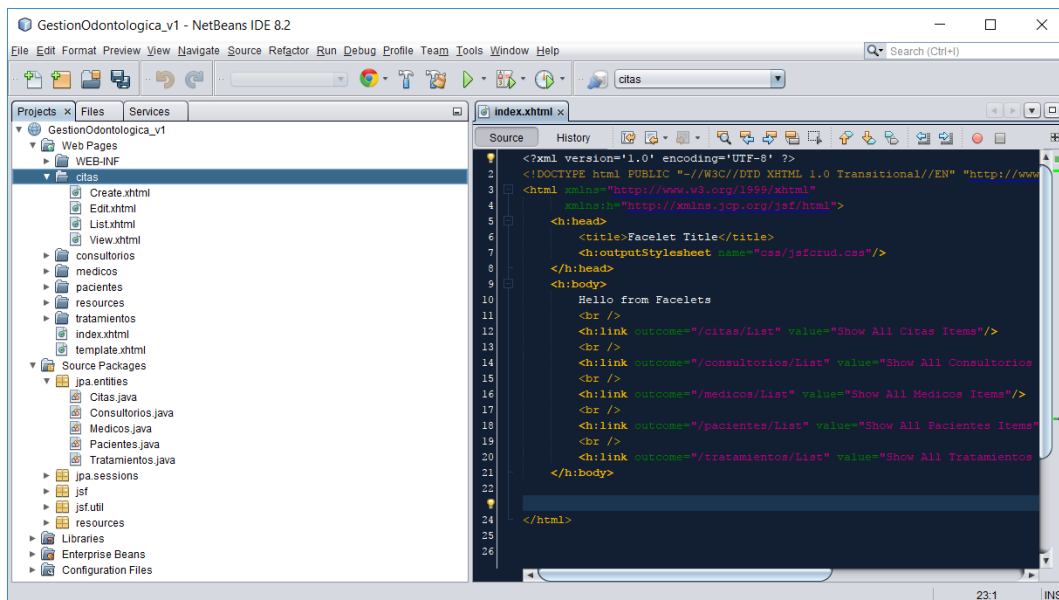


Figura 5.6 Creación de los facelets paso 6

En este momento se puede hacer una prueba del código generado. Para lo anterior se oprime el botón de ejecutar que se encuentra en la barra de herramientas. Una vez se oprime aparece lo siguiente en el navegador por defecto del sistema:

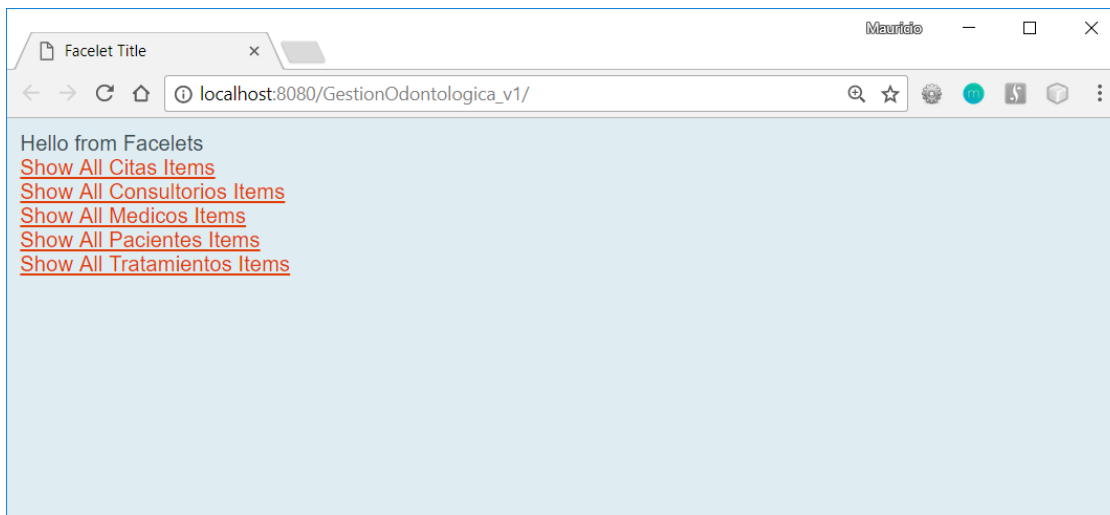


Figura 5.7 Creación de los facelets paso 7

Se puede observar que el servidor GlassFish utiliza el puerto 8080 para dibujar las páginas web.

En este momento se tiene una aplicación web operativa realizada usando la especificación de la Java Platform Enterprise llamada JavaServer Faces o JSF.

También se observa que al dar click sobre cualquiera de las opciones el sistema muestra los registros usando el facelet “List.xhtml” que corresponda. En el caso de los pacientes aparece lo siguiente en pantalla:

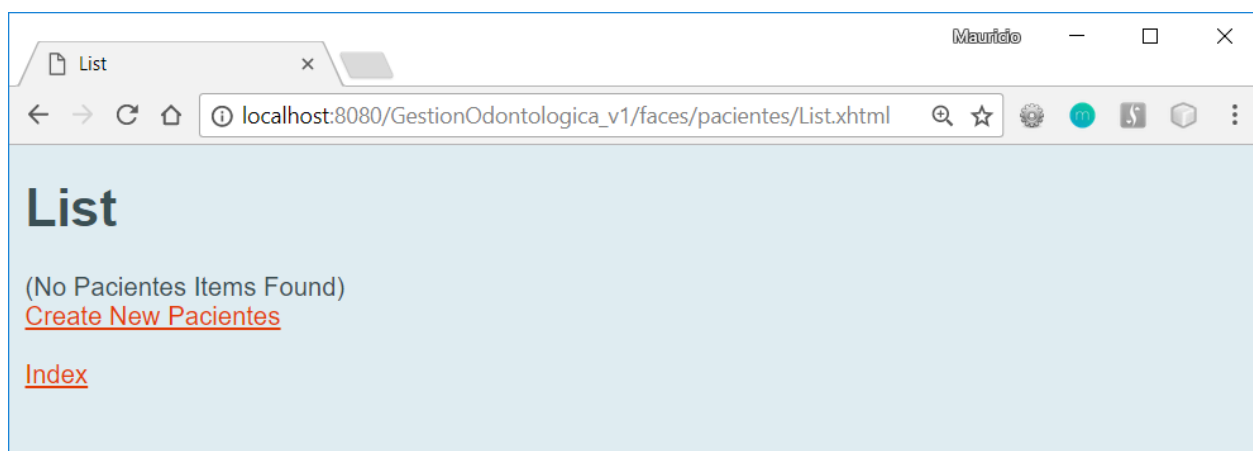


Figura 5.8 Creación de los facelets paso 8

Para crear un nuevo paciente se da click sobre el botón “Create New Paciente”. Aparece lo siguiente en pantalla:

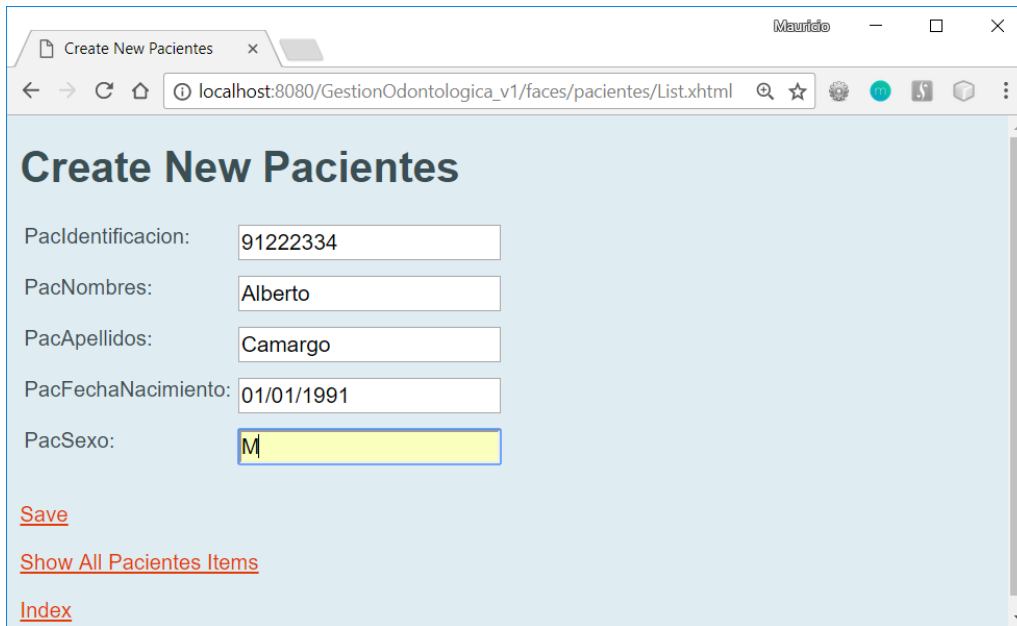


Figura 5.9 Facelet para creación de pacientes

Una vez se oprime el enlace “Save” aparece lo siguiente:

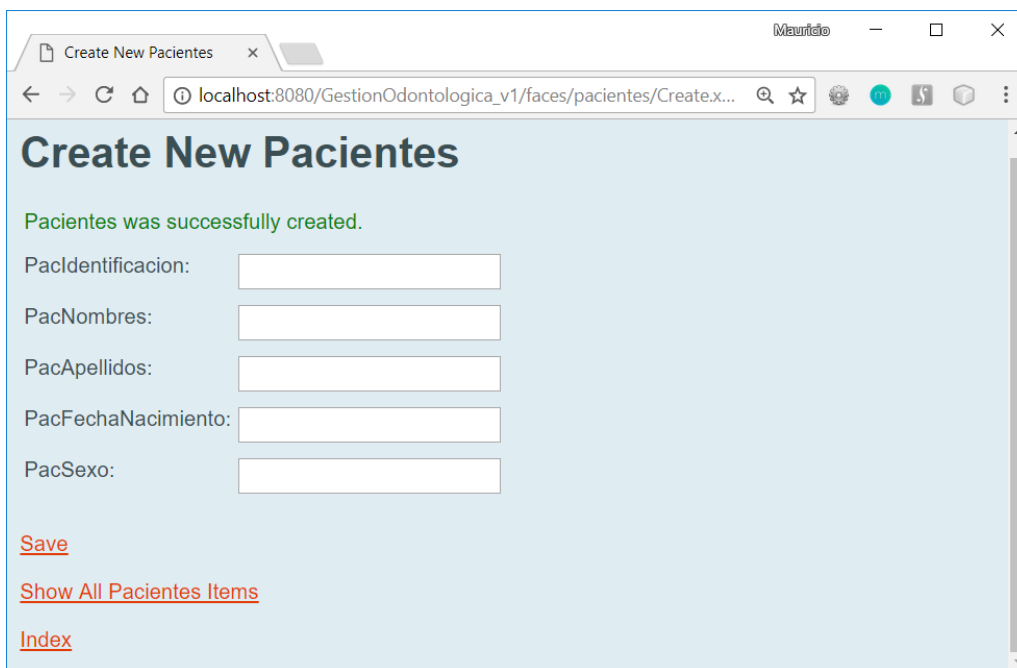
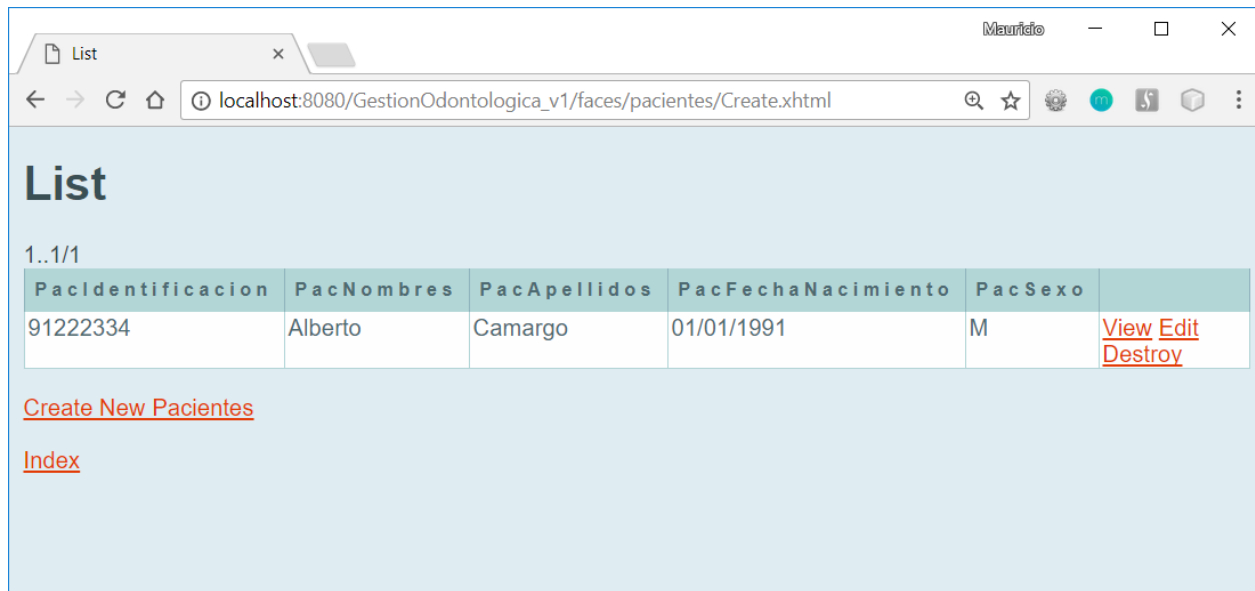


Figura 5.10 Facelet para creación de pacientes paso 2

Si se oprime el vínculo “Show all Pacientes Items” aparece lo siguiente:



PacIdentificacion	PacNombres	PacApellidos	PacFechaNacimiento	PacSexo	
91222334	Alberto	Camargo	01/01/1991	M	View Edit Destroy

[Create New Pacientes](#)

[Index](#)

Figura 5.11. Facelet para listar los pacientes.

Netbeans también ha generado un archivo index.xhtml que contiene los enlaces a las entidades generadas como aparece a continuación:

```
<!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://xmlns.jcp.org/jsf/html">
  <h:head>
    <title>Facelet Title</title>
    <h:outputStylesheet name="css/jsfcrud.css"/>
  </h:head>
  <h:body>
    Hello from Facelets
    <br />
    <h:link outcome="/citas/List" value="Show All Citas Items"/>
    <br />
    <h:link outcome="/consultorios/List" value="Show All Consultorios Items"/>
    <br />
    <h:link outcome="/medicos/List" value="Show All Medicos Items"/>
    <br />
    <h:link outcome="/pacientes/List" value="Show All Pacientes Items"/>
    <br />
    <h:link outcome="/tratamientos/List" value="Show All Tratamientos Items"/>
  </h:body>
</html>
```

```

1 <?xml version='1.0' encoding='UTF-8' ?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml"
4     xmlns:hc="http://xmlns.jcp.org/jsf/html">
5   <h:head>
6     <title>Facelet Title</title>
7     <h:outputStylesheet name="css/jsfcrud.css"/>
8   </h:head>
9   <h:body>
10     Hello from Facelets
11     <br />
12     <h:link outcome="/citas/List" value="Show All Citas Items"/>
13     <br />
14     <h:link outcome="/consultorios/List" value="Show All Consultorios Items"/>
15     <br />
16     <h:link outcome="/medicos/List" value="Show All Medicos Items"/>
17     <br />
18     <h:link outcome="/pacientes/List" value="Show All Pacientes Items"/>
19     <br />
20     <h:link outcome="/tratamientos/List" value="Show All Tratamientos Items"/>
21   </h:body>
22
23 </html>
24

```

Figura 5.12. Archivo index.xhtml generado.

Todas las opciones descritas anteriormente fueron generadas automáticamente. A continuación se procederá a realizar las adaptaciones y ajustes requeridos para resolver el caso de uso del Sistema de Gestión Odontológica.

6. Maquetación de la aplicación.

Para este recurso se usará el siguiente esquema de maquetación:

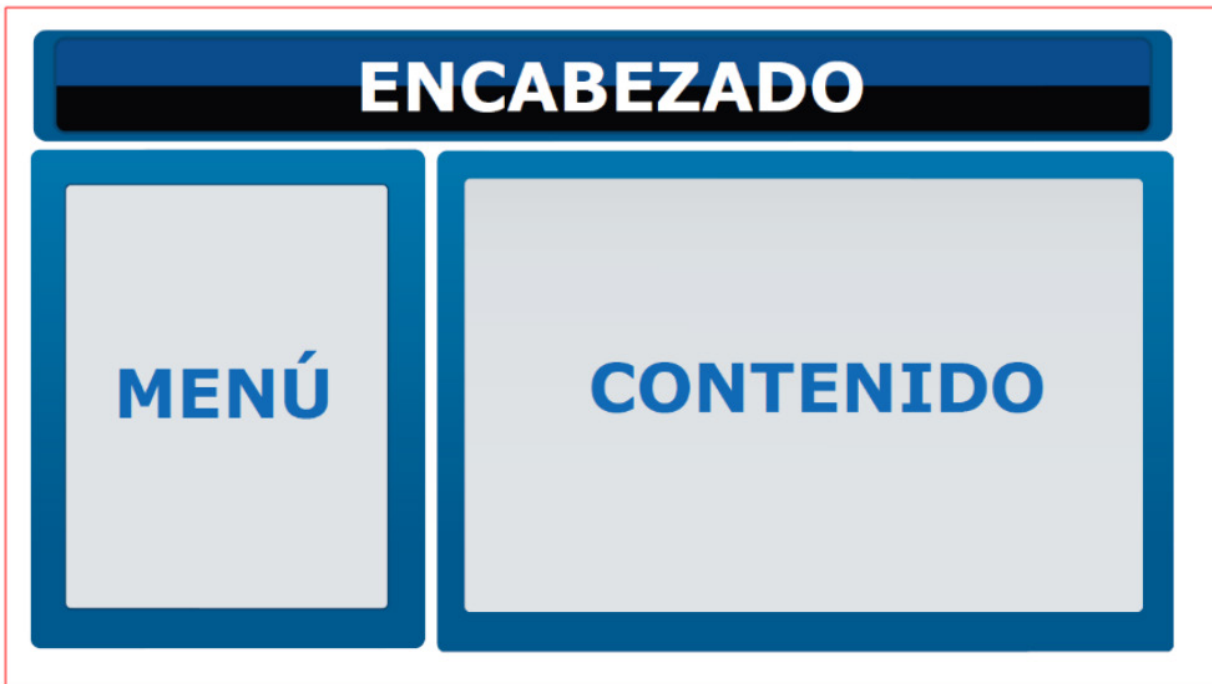


Figura 6.1 Modelo de maqueta a utilizar.

Para la maquetación se dispone de las siguientes herramientas:

- Estilos CSS.
- Estructura de plantillas de JavaServer Faces.
- Elementos HTML.

6.1 Conceptos de maquetación con JSF.

El generador de código de Netbeans provee el archivo template.xhtml el cual fue aplicado a todos los archivos.xhtml que se generaron como se muestra a continuación:

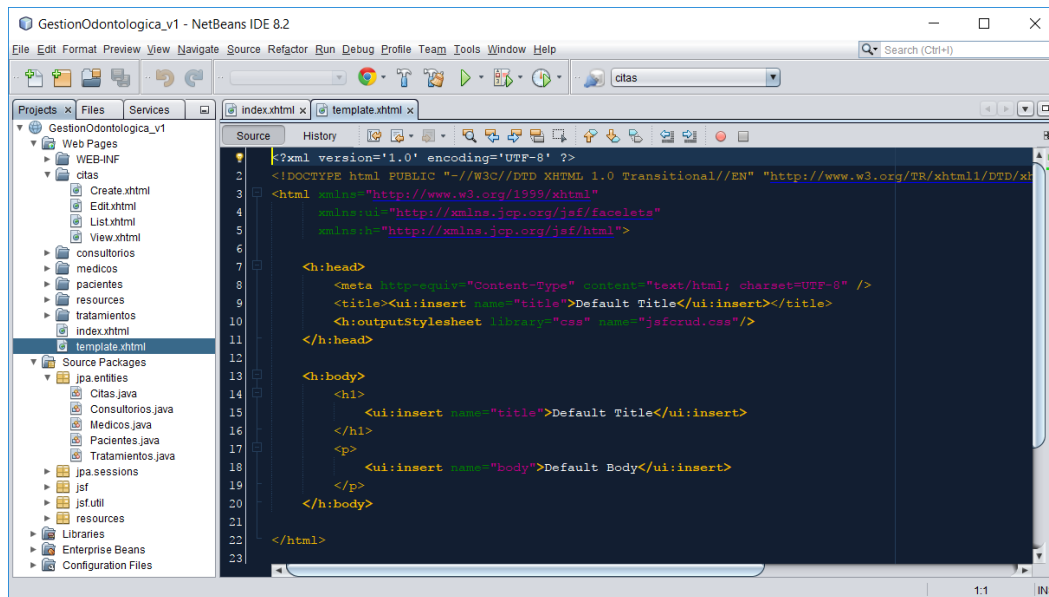


Figura 6.2 Template o plantilla generada por Netbeans.

Esta plantilla será la que aplicará la estructura a todas las vistas del proyecto. El código en esta etapa del proyecto es el siguiente:

```
<?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:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:h="http://xmlns.jcp.org/jsf/html">

  <h:head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title><ui:insert name="title">Default Title</ui:insert></title>
    <h:outputStylesheet library="css" name="jsforud.css"/>
  </h:head>

  <h:body>
    <h1>
      <ui:insert name="title">Default Title</ui:insert>
    </h1>
    <p>
      <ui:insert name="body">Default Body</ui:insert>
    </p>
  </h:body>

</html>
```

A esta plantilla se le realizarán varias modificaciones hasta llegar a la maqueta o modelo planteado.

A continuación se insertará el código para que la plantilla contenga un encabezado y una sección de contenido. Para lo anterior se usarán dos etiquetas `<div>` así:

```
<?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:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:h="http://xmlns.jcp.org/jsf/html">

  <h:head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title><ui:insert name="title">Default Title</ui:insert></title>
    <h:outputStylesheet library="css" name="jsfcrud.css"/>
  </h:head>

  <h:body>
    <div id="encabezado">
      <h1><ui:insert name="header">Sistema de Gestión Odontológica</ui:insert></h1>
    </div>

    <h1>
      <!--ui:insert name="title">Default Title</ui:insert-->
    </h1>
    <div id="contenido">
      <ui:insert name="body">Default Body</ui:insert>
    </div>
  </h:body>
</html>
```

A continuación se procede a incluir el menú. Para lo anterior se hará uso de la etiqueta de JSF llamada `"h:commandLink"` que es similar a una etiqueta `<a>` en html. La estructura es la siguiente:

```
<h:commandLink action="ruta_en_jsf">Nombre</h:commandLink>
```

Donde:

"action" es la ruta donde se dirigirá el enlace.

Antes de la creación de menú se deben crear las rutas. Una ruta es la asociación entre una etiqueta y un archivo xhtml. La configuración de la rutas se realiza en el archivo “faces-config.xml” ubicado como se muestra a continuación:

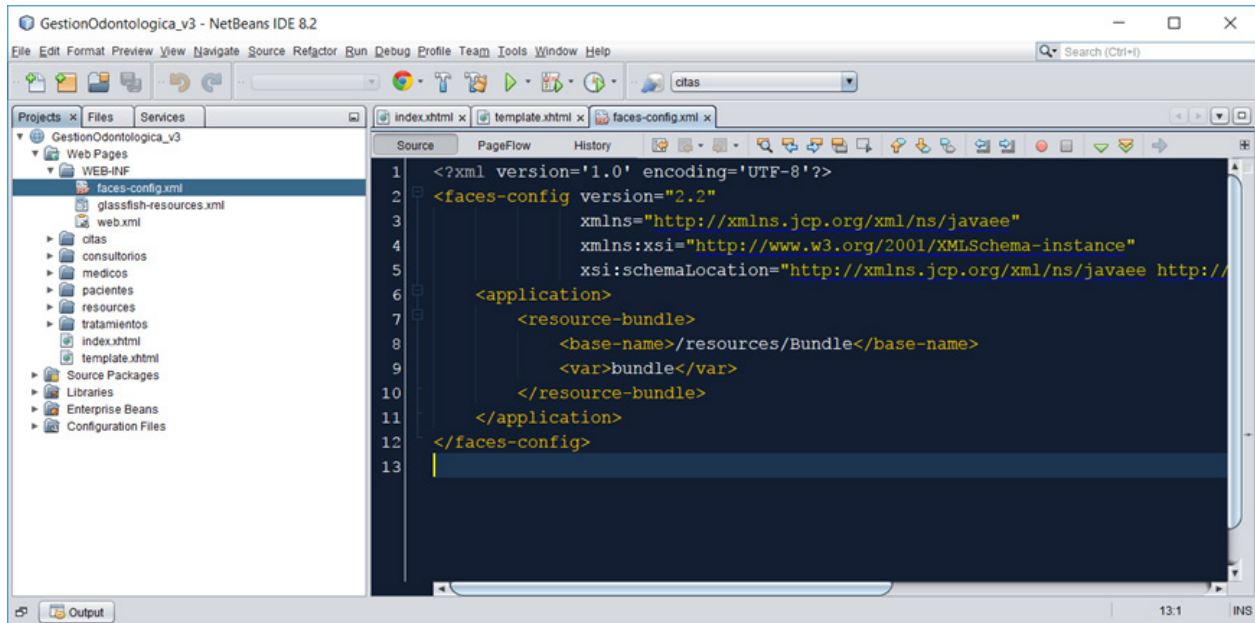


Figura 6.3 Archivo de configuración faces-config.xml

Para este recurso se incluirán las siguientes rutas:

```

<navigation-rule>

    <from-view-id>*</from-view-id>
    <navigation-case>
        <from-outcome>inicio</from-outcome>
        <to-view-id>/index.xhtml</to-view-id>
    </navigation-case>
    <navigation-case>
        <from-outcome>asignar_cita</from-outcome>
        <to-view-id>/ citas/Create.xhtml</to-view-id>
    </navigation-case>
    <navigation-case>
        <from-outcome>consultar_cita</from-outcome>
        <to-view-id>/ citas/List.xhtml</to-view-id>
    </navigation-case>

</navigation-rule>

```

La estructura de las rutas es la siguiente:

```

<navigation-case>
    <from-outcome>nombre_de_la_ruta</from-outcome>
    <to-view-id>ubicacion_del_archivo_xhtml</to-view-id>
</navigation-case>

```

Para el menú se utilizará la etiqueta html <nav>. Una vez incluidos los cambios, el archivo de plantilla queda 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">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:h="http://xmlns.jcp.org/jsf/html">

  <h:head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title><ui:insert name="title">Default Title</ui:insert></title>
    <h:outputStylesheet library="css" name="jsforud.css"/>
  </h:head>

  <h:body>
    <div id="encabezado">
      <h1><ui:insert name="header">Sistema de Gestión Odontológica</ui:insert></h1>
    </div>

    <nav id="menu">
      <h:form>
        <ul>
          <li><h:commandLink action="inicio">Inicio</h:commandLink></li>
          <li><h:commandLink action="asignar_cita">Asignar Cita</h:commandLink></li>
          <li><h:commandLink action="consultar_cita">Consultar Cita</h:commandLink></li>
        </ul>
      </h:form>
    </nav>

    <div id="contenido">
      <ui:insert name="body">Default Body</ui:insert>
    </div>
  </h:body>
</html>
```

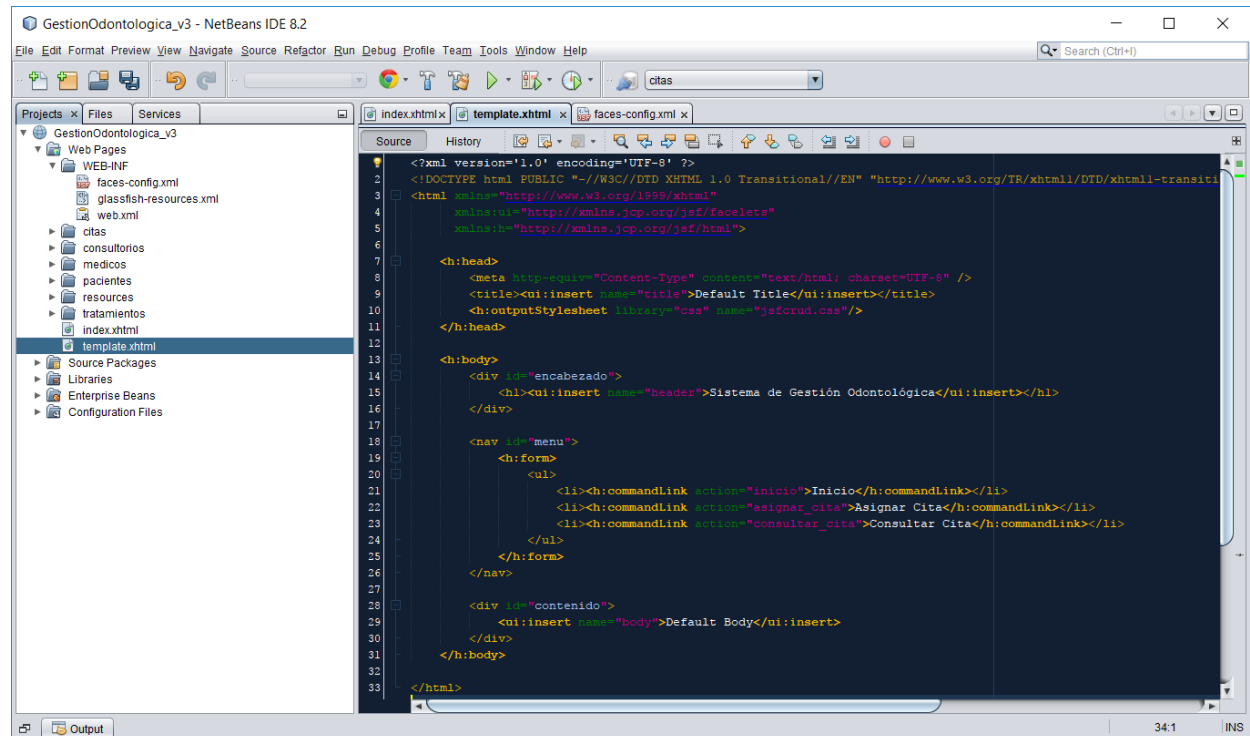


Figura 6.4 Plantilla con las opciones del menú.

En este punto ya tenemos lista la plantilla que aplicará a todos los archivos de presentación del proyecto. Al ejecutar el proyecto aparece lo siguiente en pantalla:

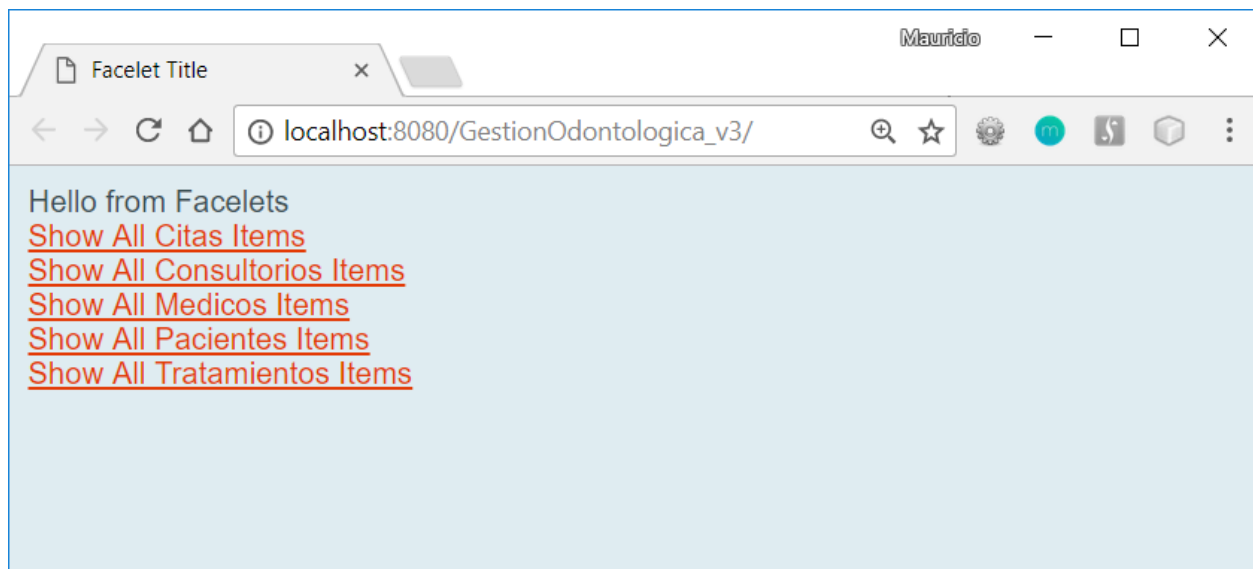


Figura 6.5. Prueba de la aplicación.

Se puede observar que el index.xhtml que generó automáticamente el sistema no está actualizado con la plantilla que se acabó de diseñar. Para aplicar la plantilla al index.xhtml se introducen los siguientes cambios:

```
<ui:composition template="/template.xhtml">

    <ui:define name="title">
        <h:outputText value="INICIO"/></h:outputText>
    </ui:define>

    <ui:define name="body">
    </ui:define>

</ui:composition>
```

Se puede observar la utilización de la etiqueta de JSF llamada “ui:composition”. Esta última hace uso de la plantilla “/template.xhtml”.

Las etiquetas “<ui:define>” sirven para definir los parámetros establecidos en la plantilla. En este caso los parámetros “title” y “body” se definieron usando esta etiqueta. En caso que no se diligencien los parámetros la plantilla incluirá el código por defecto.

El archivo index.html con los cambios queda así:

```
<?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://xmlns.jcp.org/jsf/html"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
  <ui:composition template="/template.xhtml">

    <ui:define name="title">
      <h:outputText value="INICIO"/></h:outputText>
    </ui:define>

    <ui:define name="body">
    </ui:define>

  </ui:composition>
</html>
```

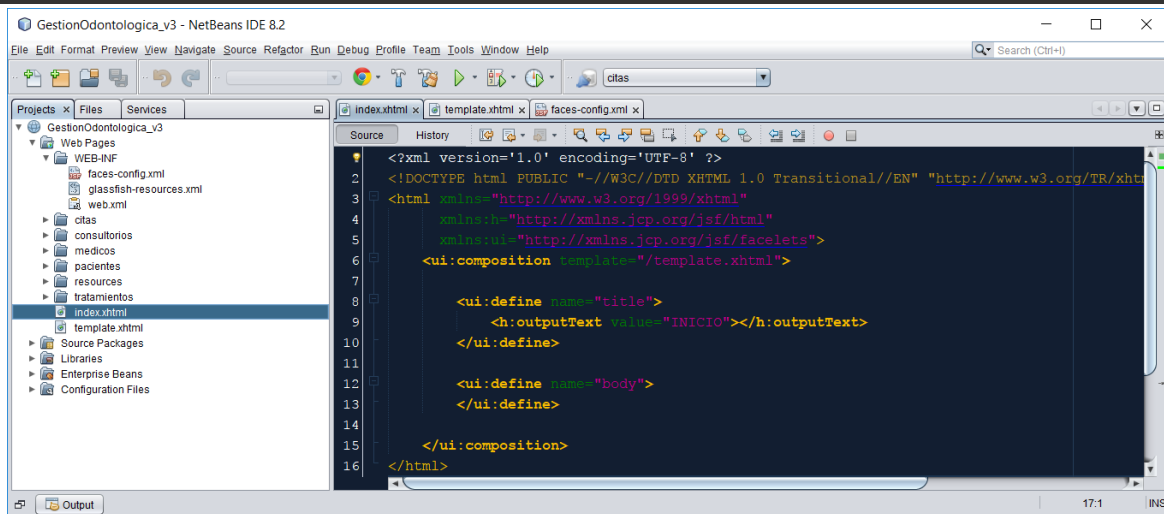


Figura 6.6. Archivo index.xhtml usando la plantilla de la aplicación.

En este punto se puede nuevamente probar la aplicación ejecutando “run” en la barra de herramientas de Netbeans. Aparece lo siguiente en pantalla:



Figura 6.7. Prueba de index.xhtml que usa la plantilla de la aplicación.

6.2 Aplicación de estilos CSS.

Los archivos de presentación de JSF permiten incluir etiquetas tanto de su mismo dominio como aquellas provistas por html. Además se pueden incluir hojas de estilo CSS. En esta sección se hará uso de estos estilos para mejorar la presentación de la plantilla.

Cuando Netbeans creó las clases de entidad y las clases controladoras también genera un archivo de estilos por defecto llamado “jsfcrud.css” ubicado en el directorio “resources/css”. La ubicación en el proyecto se muestra a continuación:

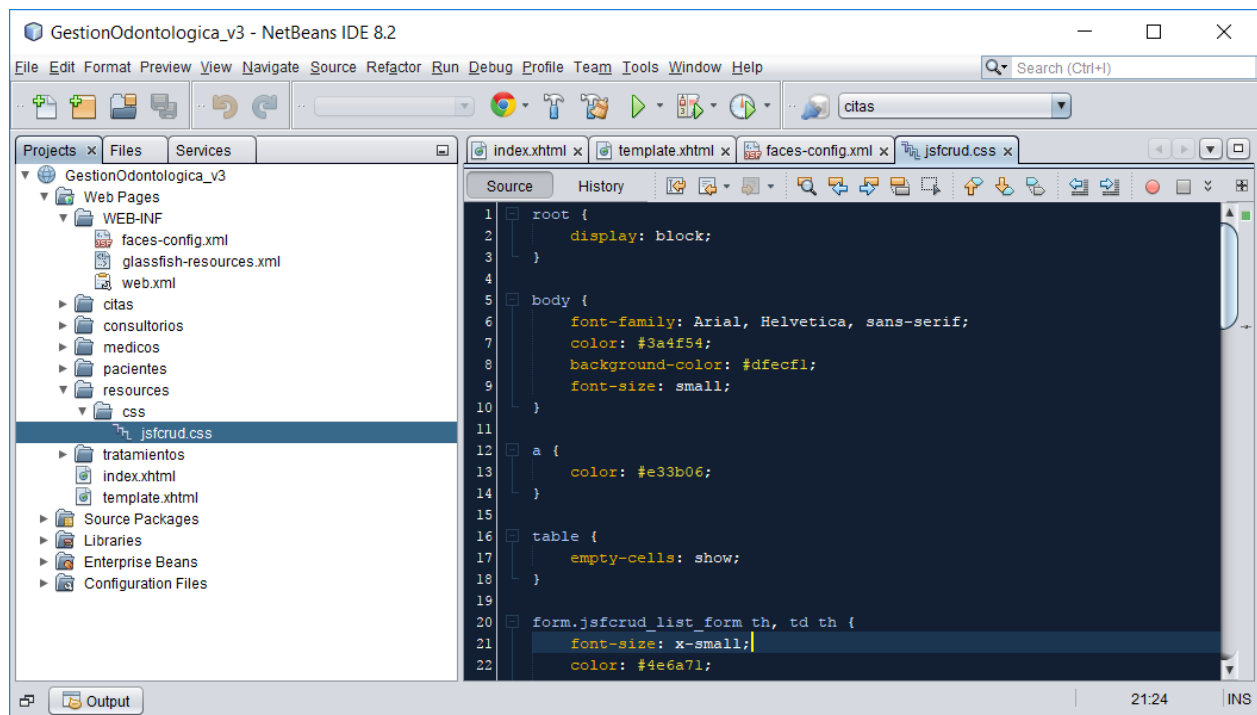


Figura 6.8. Hoja de estilos del generador de código.

Esta hoja de estilos es usada también por la plantilla general. A continuación se introducirán los siguientes cambios para mejorar la apariencia de las páginas:

6.2.1. Se incluirá un banner en la sección “encabezado”.

La imagen del banner se diseña por separado y se debe ubicar en el directorio “resources/ images”. Para crear esta carpeta se da click derecho sobre el directorio “resources” y posteriormente “New” → “Folder”.

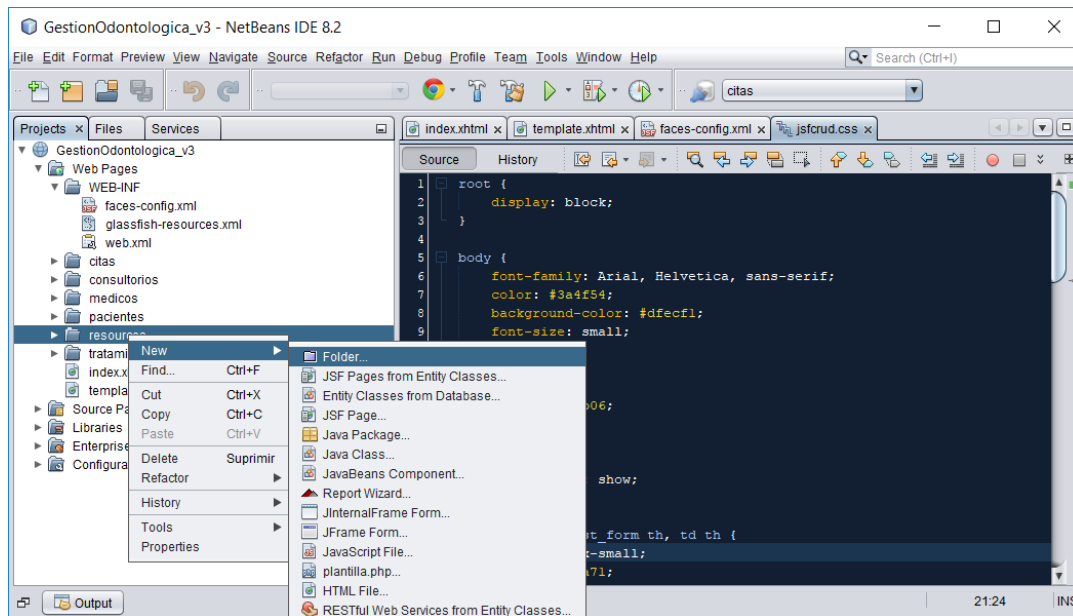


Figura 6.9. Creación de la carpeta de imágenes

Una vez se selecciona esta opción aparece lo siguiente:

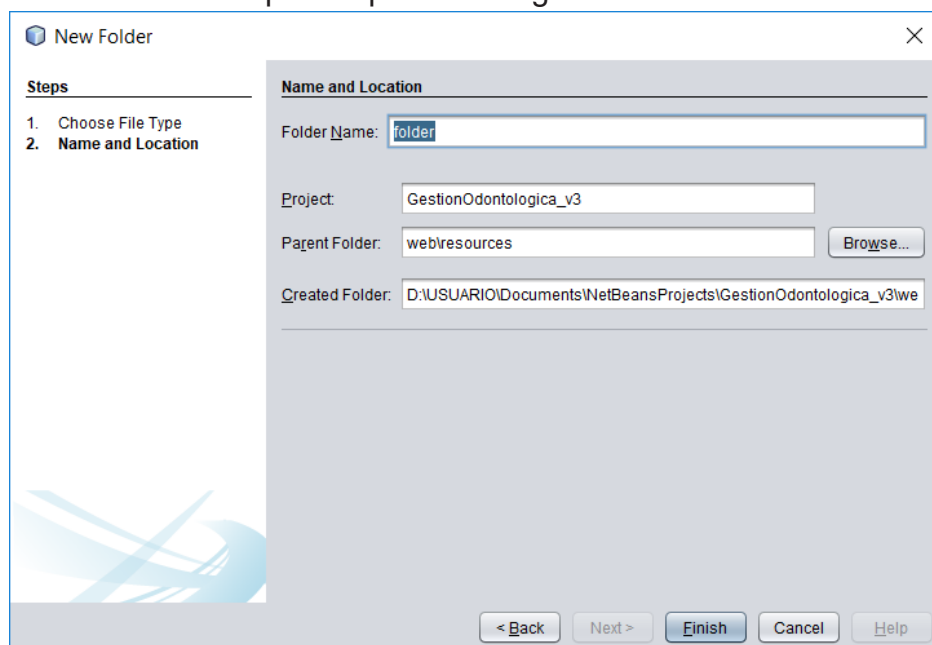


Figura 6.10. Creación de la carpeta de imágenes

En la opción “Folder Name” se introduce “images”. Por último se oprime el botón “Finish”.

Por último se copia el archivo a la carpeta “images” usando el explorador de archivo del sistema operativo.

La regla CSS para aplicar el banner en el encabezado es la siguiente:

```
#encabezado {
    width: 900px;
    height: 150px;
    background: url("http://localhost:8080/GestionOdontologica_v2/faces/resources/images/banner1.jpg") no-repeat ;
    border: #d3d3d3 1px solid ;
}
#encabezado h1 {
    text-align: center ;
    margin-top: 50px;
}
div#contenido {
    border: #d3d3d3 1px solid ;
    margin: 20px 20px 10px 180px;
    padding: 10px 15px;
}
```

Se puede observar que se usó una ruta absoluta para la url del banner. Lo anterior porque la plantilla es usada en distintos niveles de directorios.

6.2.2. Aplicación de estilos para la sección de contenido.

Para la sección de contenido de la plantilla se usarán las siguientes reglas:

```
#contenido {
    border: #d3d3d3 1px solid ;
    margin: 20px 20px 10px;
    padding: 10px 15px;
    width: 690px;
}
#contenido h2 {
    color: #0075ff;
    border-bottom: #CFBFB5 1px solid;
}
```

6.2.3. Aplicación de estilos para la sección del menú.

Para la sección del menú se usará el siguiente estilo:

```
#menu {
    float: left;
    width: 120px;
}
```

Se puede observar que se usó el atributo “float: left” para que el menú se ubique a la izquierda de la página.

Una vez aplicados los estilos y se ejecuta la aplicación aparece lo siguiente:



Figura 6.11. Aspecto de la aplicación con los estilos aplicados.

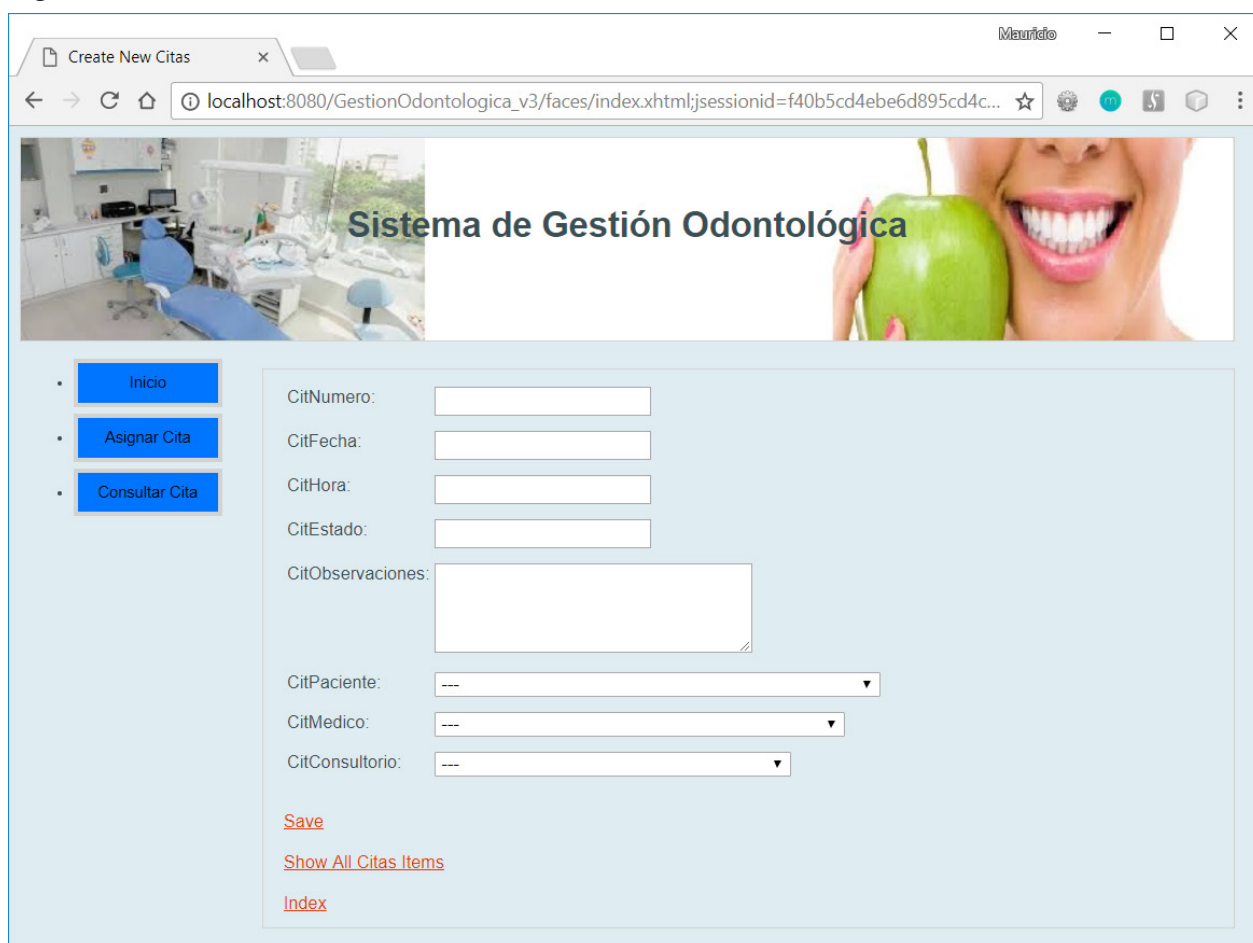
7. Desarrollo de los casos de uso.

Una vez generadas la capa de persistencia, la capa de controladores y la plantilla o template se procede a desarrollar los casos de uso.

Debido a que Netbeans, basado en el framework JSF, generó el código para las capas mencionadas en el párrafo anterior lo que se va a realizar en adelante es el ajuste o adaptación de ese código a los requerimientos del caso de uso.

7.1. Caso de uso “Asignar Cita”.

En este punto al dar click sobre la opción del menú “Asignar Cita” el sistema muestra lo siguiente:



The screenshot displays a web browser window with the title 'Create New Citas'. The address bar shows the URL: `localhost:8080/GestionOdontologica_v3/faces/index.xhtml;jsessionid=f40b5cd4ebe6d895cd4c...`. The page features a header with a dental clinic image and the text 'Sistema de Gestión Odontológica'. On the left, there is a navigation menu with three buttons: 'Inicio', 'Asignar Cita', and 'Consultar Cita'. The main content area contains a form for creating a new appointment. The form includes the following fields: 'CitNumero:' (text input), 'CitFecha:' (text input), 'CitHora:' (text input), 'CitEstado:' (text input), 'CitObservaciones:' (text area), 'CitPaciente:' (dropdown menu), 'CitMedico:' (dropdown menu), and 'CitConsultorio:' (dropdown menu). Below the form, there are three links: 'Save', 'Show All Citas Items', and 'Index'.

Figura 7.1. Estado inicial del caso de uso Asignar Cita.

Se puede observar lo siguiente:

No	Descripción	Ajuste a realizar
1	Los nombres de los campos son traídos de la base de datos y se deben ajustar por unos que sean más descriptivos para las personas.	Ajustar el archivo Source Packages/resources/Bundle
2	El campo CitNumero se asigna automáticamente por la base de datos por tanto no se requiere que esté en el formulario de creación.	Retirar el campo del archivo /citas/Create.xhtml
3	Las opciones de navegación “Show all Citas Items” e “Index” no se requieren porque ya están contempladas en el menú lateral izquierdo.	Retirar estas opciones del archivo /citas/Create.xhtml

Al revisar los campos de selección para los pacientes, médicos y consultorios aparece lo siguiente:

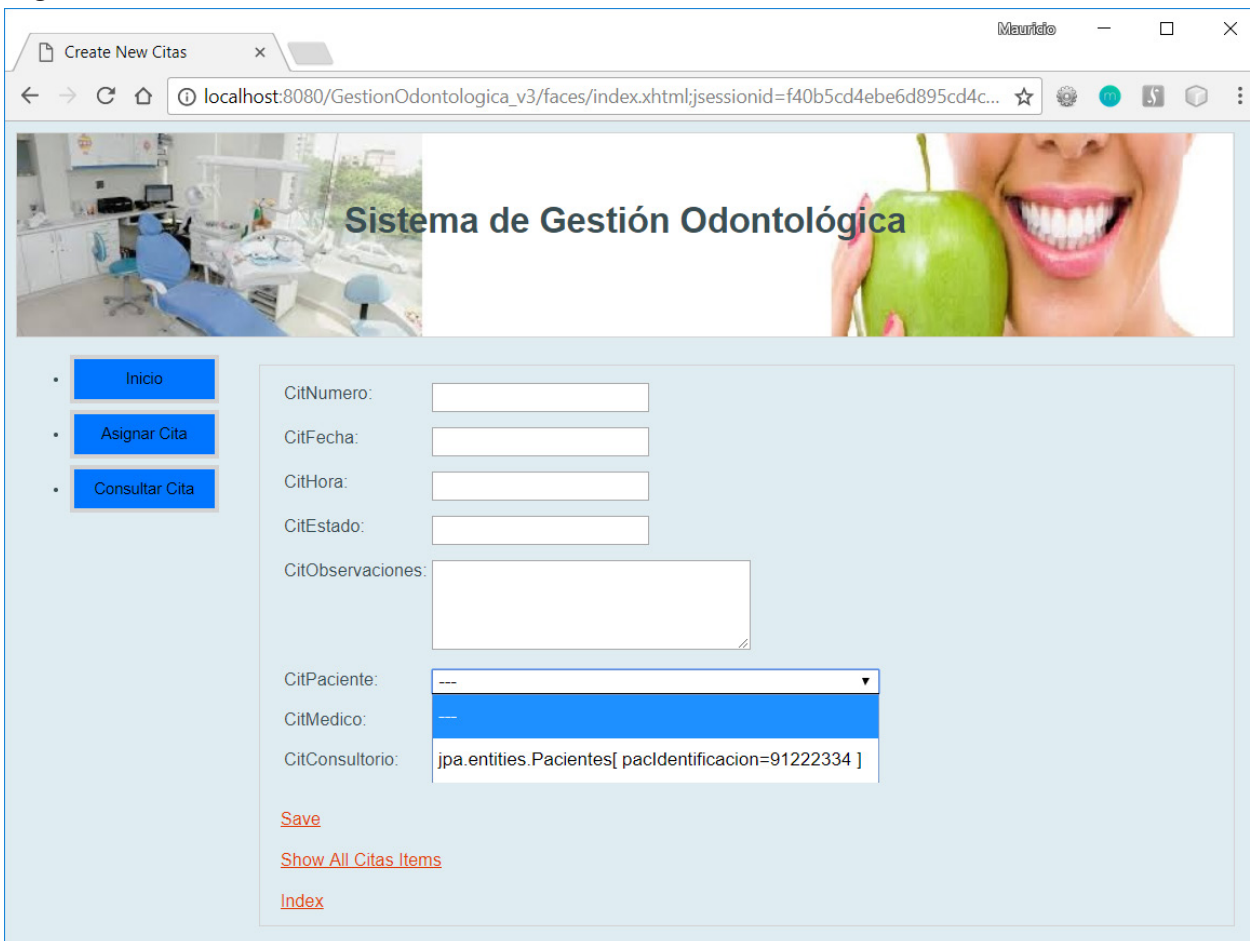


Figura 7.2. Estado inicial del caso de uso Asignar Cita.

Se puede observar que el nombre del paciente contiene una estructura técnica y sólo muestra la cédula de paciente. Por tanto se deben realizar los siguientes ajustes:

No	Descripción	Ajuste a realizar
4	El campo de selección del paciente no muestra los pacientes de forma adecuada.	Ajustar el método toString() de la clase /jpa.entities/Pacientes.java
5	El campo de selección del médico no muestra los médicos de forma adecuada.	Ajustar el método toString() de la clase /jpa.entities/Medicos.java
6	El campo de selección de los consultorios no muestra los consultorios de forma adecuada.	Ajustar el método toString() de la clase /jpa.entities/Consultorios.java

A continuación se desarrollan los ajustes.

7.1.1. Ajustar el nombre de los campos.

Por defecto el generador de código construye un archivo de localización o internacionalización donde se centralizan todos los mensajes del sistema. Este archivo se llama “Bundle” y se ubica en la sección de “Source Packages” del proyecto como se muestra a continuación:

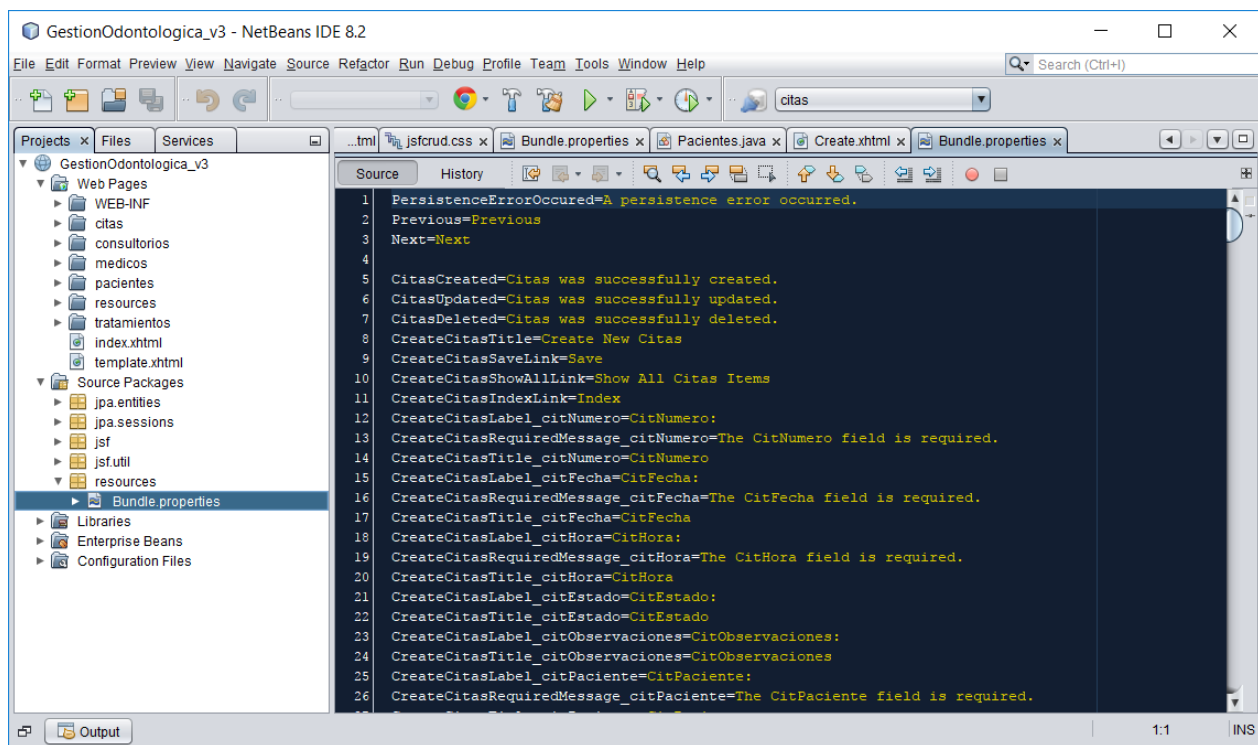


Figura 7.3. Ubicación del archivo de localización o internacionalización.

Primero se deben identificar los textos a cambiar. Para esto se abre al archivo /citas/ Create.xhtml 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">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">

    <ui:composition template="/template.xhtml">
        <ui:define name="title">
            <h:outputText value="#{bundle.CreateCitasTitle}" /></h:outputText>
        </ui:define>
        <ui:define name="body">
            <h:panelGroup id="messagePanel" layout="block">
                <h:messages errorStyle="color: red" infoStyle="color: green" layout="table" />
            </h:panelGroup>
            <h:form>
                <h:panelGrid columns="2">
                    <h:outputLabel value="#{bundle.CreateCitasLabel_citNumero}" for="citNumero" />
                    <h:inputText id="citNumero" value="#{citasController.selected.citNumero}"
                        title="#{bundle.CreateCitasTitle_citNumero}" required="true" requiredMessage="#{bundle.CreateCitasRequiredMessage_citNumero}" />
                    <h:outputLabel value="#{bundle.CreateCitasLabel_citFecha}" for="citFecha" />
                    <h:inputText id="citFecha" value="#{citasController.selected.citFecha}"
                        title="#{bundle.CreateCitasTitle_citFecha}" required="true" requiredMessage="#{bundle.CreateCitasRequiredMessage_citFecha}" />
                    <f:convertDateTime pattern="MM/dd/yyyy" />
                    </h:inputText>
                    <h:outputLabel value="#{bundle.CreateCitasLabel_citHora}" for="citHora" />
                    <h:inputText id="citHora" value="#{citasController.selected.citHora}"
                        title="#{bundle.CreateCitasTitle_citHora}" required="true" requiredMessage="#{bundle.CreateCitasRequiredMessage_citHora}" />
                    <h:outputLabel value="#{bundle.CreateCitasLabel_citEstado}" for="citEstado" />
                    <h:inputText id="citEstado" value="#{citasController.selected.citEstado}"
                        title="#{bundle.CreateCitasTitle_citEstado}" />
                    <h:outputLabel value="#{bundle.CreateCitasLabel_citObservaciones}" for="citObservaciones" />
                    <h:inputTextarea rows="4" cols="30" id="citObservaciones" value="#{citasController.selected.citObservaciones}"
                        title="#{bundle.CreateCitasTitle_citObservaciones}" />
                    <h:outputLabel value="#{bundle.CreateCitasLabel_citPaciente}" for="citPaciente" />
                    <h:selectOneMenu id="citPaciente" value="#{citasController.selected.citPaciente}"
                        title="#{bundle.CreateCitasTitle_citPaciente}" required="true" requiredMessage="#{bundle.CreateCitasRequiredMessage_citPaciente}" />
                    <f:selectItems value="#{pacientesController.itemsAvailableSelectOne}" />
                    </h:selectOneMenu>
                    <h:outputLabel value="#{bundle.CreateCitasLabel_citMedico}" for="citMedico" />
                    <h:selectOneMenu id="citMedico" value="#{citasController.selected.citMedico}"
                        title="#{bundle.CreateCitasTitle_citMedico}" required="true" requiredMessage="#{bundle.CreateCitasRequiredMessage_citMedico}" />
                    <f:selectItems value="#{medicosController.itemsAvailableSelectOne}" />
                    </h:selectOneMenu>
                    <h:outputLabel value="#{bundle.CreateCitasLabel_citConsultorio}" for="citConsultorio" />
                    <h:selectOneMenu id="citConsultorio" value="#{citasController.selected.citConsultorio}"
                        title="#{bundle.CreateCitasTitle_citConsultorio}" required="true" requiredMessage="#{bundle.CreateCitasRequiredMessage_citConsultorio}" />
                    <f:selectItems value="#{consultoriosController.itemsAvailableSelectOne}" />
                    </h:selectOneMenu>
                </h:panelGrid>
                <br />
                <h:commandLink action="#{citasController.create}" value="#{bundle.CreateCitasSaveLink}" />
                <br />
                <h:commandLink action="#{citasController.prepareList}" value="#{bundle.CreateCitasShowAllLink}" immediate="true" />
                <br />
                <h:link outcome="/index" value="#{bundle.CreateCitasIndexLink}" />
            </h:form>
        </ui:define>
    </ui:composition>
</html>
```

Los textos a cambiar son:

Texto	Nuevo texto
bundle.CreateCitasLabel_citNumero	Número
bundle.CreateCitasLabel_citFecha	Fecha
bundle.CreateCitasLabel_citHora	Hora
bundle.CreateCitasLabel_citEstado	Estado
bundle.CreateCitasLabel_citObservaciones	Observaciones
bundle.CreateCitasLabel_citPaciente	Paciente
bundle.CreateCitasLabel_citMedico	Médico
bundle.CreateCitasLabel_citConsultorio	Consultorio

Una vez realizados los cambios al ejecutar la aplicación aparece lo siguiente en pantalla:

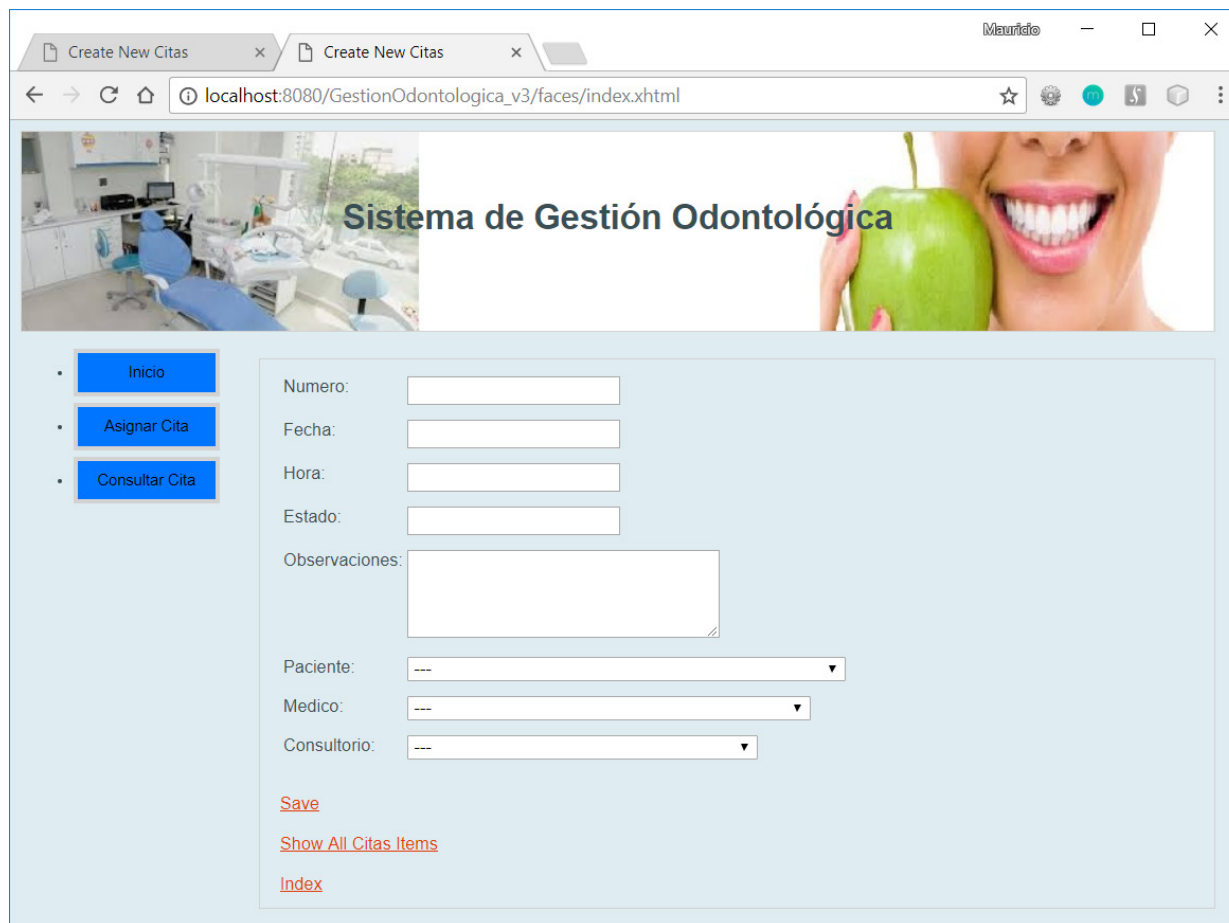


Figura 7.4. Ubicación del archivo de localización o internacionalización.

7.1.2. Retirar el campo Número de cita.

Para realizar este ajuste se edita el archivo /citas/Create.xhtml y se usa el atributo: `rendered="false"` de la etiqueta JSF `<h:outputLabel>` y `<h:inputText>`. El atributo "valor" del campo se deja en cero. Lo anterior porque los valores que son generados automáticamente por la base de datos Mysql, es decir aquellos definidos con el atributo "auto_increment", se requiere que sean enviados con el valor cero.

Una vez realizados los cambios el código queda así:

```
<h:outputLabel value="0" for="citNumero" rendered="false"/>
<h:inputText id="citNumero" value="#{citasController.selected.citNumero}" title="#{bundle.CreateCitasTitle_citNumero}"
required="true" requiredMessage="#{bundle.CreateCitasRequiredMessage_citNumero}" rendered="false"/>
```

Al ejecutar la aplicación aparece lo siguiente:



Figura 7.5. Asignar cita con el campo número retirado.

7.1.3. Eliminar las opciones de navegación del formulario.

Para realizar este ajuste se edita el archivo / citas/Create.xhtml y se eliminan las líneas que se muestran resaltadas 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">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">

    <ui:composition template="/template.xhtml">
        <ui:define name="title">
            <h:outputText value="#{bundle.CreateCitasTitle}" /></h:outputText>
        </ui:define>
        <ui:define name="body">
            <h:panelGroup id="messagePanel" layout="block">
                <h:messages errorStyle="color: red" infoStyle="color: green" layout="table" />
            </h:panelGroup>
            <h:form>
                <h:panelGrid columns="2">
                    <h:outputLabel value="0" for="citNumero" rendered="false" />
                    <h:inputText id="citNumero" value="#{citasController.selected.citNumero}"
                        title="#{bundle.CreateCitasTitle_citNumero}" required="true" requiredMessage="#{bundle.CreateCitasRequiredMessage_citNumero}" rendered="false" />
                    <h:outputLabel value="#{bundle.CreateCitasLabel_citFecha}" for="citFecha" />
                    <h:inputText id="citFecha" value="#{citasController.selected.citFecha}"
                        title="#{bundle.CreateCitasTitle_citFecha}" required="true" requiredMessage="#{bundle.CreateCitasRequiredMessage_citFecha}"
                        <f:convertDateTime pattern="MM/dd/yyyy" />
                    </h:inputText>
                    <h:outputLabel value="#{bundle.CreateCitasLabel_citHora}" for="citHora" />
                    <h:inputText id="citHora" value="#{citasController.selected.citHora}"
                        title="#{bundle.CreateCitasTitle_citHora}" required="true" requiredMessage="#{bundle.CreateCitasRequiredMessage_citHora}" />
                    <h:outputLabel value="#{bundle.CreateCitasLabel_citEstado}" for="citEstado" />
                    <h:inputText id="citEstado" value="#{citasController.selected.citEstado}"
                        title="#{bundle.CreateCitasTitle_citEstado}" />
                    <h:outputLabel value="#{bundle.CreateCitasLabel_citObservaciones}" for="citObservaciones" />
                    <h:inputTextarea rows="4" cols="30" id="citObservaciones" value="#{citasController.selected.citObservaciones}"
                        title="#{bundle.CreateCitasTitle_citObservaciones}" />
                    <h:outputLabel value="#{bundle.CreateCitasLabel_citPaciente}" for="citPaciente" />
                    <h:selectOneMenu id="citPaciente" value="#{citasController.selected.citPaciente}"
                        title="#{bundle.CreateCitasTitle_citPaciente}" required="true" requiredMessage="#{bundle.CreateCitasRequiredMessage_citPaciente}"
                        <f:selectItems value="#{pacientesController.itemsAvailableSelectOne}" />
                    </h:selectOneMenu>
                    <h:outputLabel value="#{bundle.CreateCitasLabel_citMedico}" for="citMedico" />
                    <h:selectOneMenu id="citMedico" value="#{citasController.selected.citMedico}"
                        title="#{bundle.CreateCitasTitle_citMedico}" required="true" requiredMessage="#{bundle.CreateCitasRequiredMessage_citMedico}"
                        <f:selectItems value="#{medicosController.itemsAvailableSelectOne}" />
                    </h:selectOneMenu>
                    <h:outputLabel value="#{bundle.CreateCitasLabel_citConsultorio}" for="citConsultorio" />
                    <h:selectOneMenu id="citConsultorio" value="#{citasController.selected.citConsultorio}"
                        title="#{bundle.CreateCitasTitle_citConsultorio}" required="true" requiredMessage="#{bundle.CreateCitasRequiredMessage_citConsultorio}"
                        <f:selectItems value="#{consultoriosController.itemsAvailableSelectOne}" />
                    </h:selectOneMenu>
                </h:panelGrid>
                <br />
                <h:commandLink action="#{citasController.create}" value="#{bundle.CreateCitasSaveLink}" />
                <br />
                <h:commandLink action="#{citasController.prepareList}" value="#{bundle.CreateCitasShowAllLink}" immediate="true" />
                <br />
                <h:link outcome="/index" value="#{bundle.CreateCitasIndexLink}" />
            </h:form>
        </ui:define>
    </ui:composition>
</html>
```

También se debe cambiar el texto asociado a "bundle.CreateCitasSaveLink" para que muestre "Grabar Cita" en vez de "Save". Esto se realiza en el archivo de Bundle como se explicó anteriormente.

Una vez realizados los cambios al ejecutar la aplicación aparece lo siguiente:

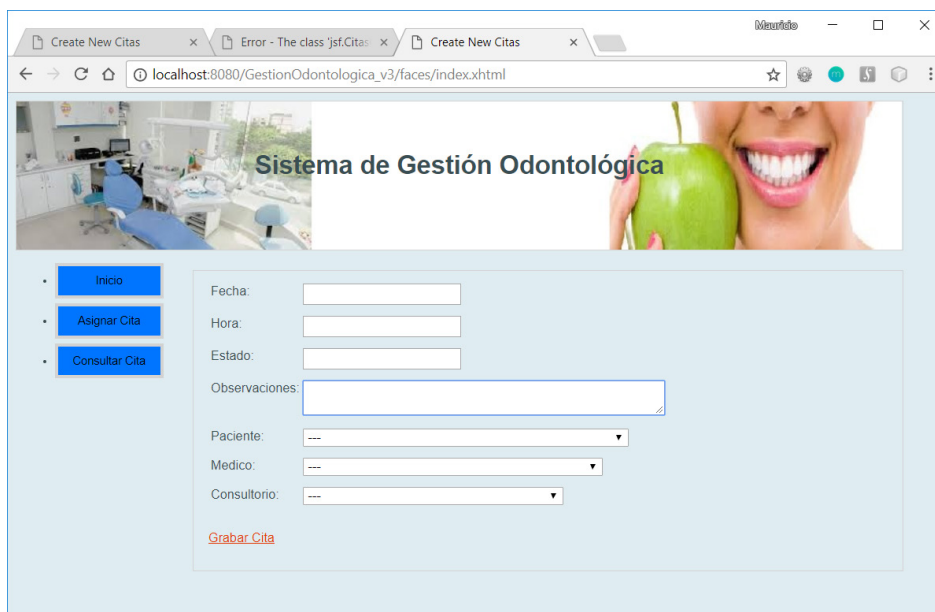


Figura 7.6. Asignación de citas con los cambios aplicados.

7.1.4. Ajustar la clase Paciente.

Para que el nombre del paciente que se muestra en el campo de selección múltiple aparezca de una manera más adecuada se procede a modificar el método toString() de la clase `jpa.entities/Pacientes.java`.

La línea de código generada es:

```
@Override
public String toString() {
    return "jpa.entities.Pacientes[ pacIdentificacion=" + pacIdentificacion + " ]";
}

Se debe cambiar a :

@Override
public String toString() {
    return pacIdentificacion + " " + pacNombres + " " + pacApellidos;
}
```

7.1.5. Ajustar la clase Medico.

Para que el nombre del médico que se muestra en el campo de selección múltiple aparezca de una manera más adecuada se procede a modificar el método toString() de la clase `jpa.entities/Medicos.java`.

La línea de código generada es:

```
@Override
public String toString() {
    return "jpa.entities.Medicos[ medIdentificacion=" + medIdentificacion + " ]";
}
Se debe cambiar a :
@Override
public String toString() {
    return medIdentificacion + " " + medNombres + " " + medApellidos ;
}
```

7.1.6. Ajustar la clase Consultorios.

Para que el nombre del consultorio que se muestra en el campo de selección múltiple aparezca de una manera más adecuada se procede a modificar el método toString() de la clase jpa.entities/Consultorios.java.

La línea de código generada es:

```
@Override
public String toString() {
    return "jpa.entities.Consultorios[ conNumero=" + conNumero + " ]";
}
Se debe cambiar a :
@Override
public String toString() {
    return conNumero + " " + conNombre;
}
```

Una vez realizados los ajustes al ejecutar la aplicación aparece lo siguiente en pantalla:



Figura 7.7. Asignación de citas con los cambios aplicados.

7.2. Caso de uso “Consultar Citas”.

El caso de uso Consultar Citas corresponde a la acción “List” implementada en el archivo /citas/List.xhtml.

El estado actual del diseño es el siguiente:



CitNumero	CitFecha	CitHora	CitEstado	CitObservaciones	CitPaciente	CitMedico	CitConsultorio
24	12/31/1989	800	Asignada	sdasdf	91222334	12345	2

Figura 7.8. Formulario para Consulta de citas

Se puede observar lo siguiente:

No	Descripción	Ajuste a realizar
1	Cambiar los nombres de las columnas por unos más adecuados	Identificar y actualizar los bundles o texto en los archivos correspondientes.
2	En la última columna sólo dejar la opción de eliminar cita	Ajustar el archivo /citas/List.xhtml
3	Retirar los enlaces “Create new Citas” e “Index”	Ajustar el archivo /citas/List.xhtml

7.2.1 Cambiar los nombres de las columnas.

Para realizar este cambio primero se debe identificar los nombres de los bundles en el archivo /citas/List.xhtml. A continuación se resaltan los textos a cambiar:

```
<?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:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">

  <ui:composition template="/template.xhtml">
    <ui:define name="title">
```

```

<h:outputText value="#{bundle.ListCitasTitle}"></h:outputText>
</ui:define>
<ui:define name="body">
<h:form styleClass="jsfcrud_list_form">
<h:panelGroup id="messagePanel" layout="block">
<h:messages errorStyle="color: red" infoStyle="color: green" layout="table"/>
</h:panelGroup>
<h:outputText escape="false" value="#{bundle.ListCitasEmpty}"
rendered="#{citasController.items.rowCount == 0}"/>
<h:panelGroup rendered="#{citasController.items.rowCount > 0}">
<h:outputText value="#{citasController.pagination.pageFirstItem + 1}..
#{citasController.pagination.pageLastItem + 1}/#{citasController.pagination.itemsCount}"/>&nbsp;
<h:commandLink action="#{citasController.previous}" value="#{bundle.Previous}" #{citasController.pagination.pageSize}"
rendered="#{citasController.pagination.hasPreviousPage}"/>&nbsp;
<h:commandLink action="#{citasController.next}" value="#{bundle.Next}" #{citasController.pagination.pageSize}"
rendered="#{citasController.pagination.hasNextPage}"/>&nbsp;
<h:dataTable value="#{citasController.items}" var="item" border="0" cellpadding="2"
cellspacing="0" rowClasses="jsfcrud_odd_row, jsfcrud_even_row" rules="all" style="border:solid 1px">
<h:column>
<f:facet name="header">
<h:outputText value="#{bundle.ListCitasTitle_citNumero}"/>
</f:facet>
<h:outputText value="#{item.citNumero}"/>
</h:column>
<h:column>
<f:facet name="header">
<h:outputText value="#{bundle.ListCitasTitle_citFecha}"/>
</f:facet>
<h:outputText value="#{item.citFecha}">
<f:convertDateTime pattern="MM/dd/yyyy" />
</h:outputText>
</h:column>
<h:column>
<f:facet name="header">
<h:outputText value="#{bundle.ListCitasTitle_citHora}"/>
</f:facet>
<h:outputText value="#{item.citHora}"/>
</h:column>
<h:column>
<f:facet name="header">
<h:outputText value="#{bundle.ListCitasTitle_citEstado}"/>
</f:facet>
<h:outputText value="#{item.citEstado}"/>
</h:column>
<h:column>
<f:facet name="header">
<h:outputText value="#{bundle.ListCitasTitle_citObservaciones}"/>
</f:facet>
<h:outputText value="#{item.citObservaciones}"/>
</h:column>
<h:column>
<f:facet name="header">
<h:outputText value="#{bundle.ListCitasTitle_citPaciente}"/>
</f:facet>
<h:outputText value="#{item.citPaciente}"/>
</h:column>
<h:column>
<f:facet name="header">
<h:outputText value="#{bundle.ListCitasTitle_citMedico}"/>
</f:facet>
<h:outputText value="#{item.citMedico}"/>
</h:column>
<h:column>
<f:facet name="header">
<h:outputText value="#{bundle.ListCitasTitle_citConsultorio}"/>
</f:facet>
<h:outputText value="#{item.citConsultorio}"/>
</h:column>
<h:column>
<f:facet name="header">
<h:outputText value="#&nbsp;"/>
</f:facet>
<h:commandLink action="#{citasController.prepareView}" value="#{bundle.ListCitasViewLink}"/>
<h:outputText value=" "/>
<h:commandLink action="#{citasController.prepareEdit}" value="#{bundle.ListCitasEditLink}"/>
<h:outputText value=" "/>
<h:commandLink action="#{citasController.destroy}" value="#{bundle.ListCitasDestroyLink}"/>
</h:column>
</h:dataTable>
</h:panelGroup>
<br />
<h:commandLink action="#{citasController.prepareCreate}" value="#{bundle.ListCitasCreateLink}"/>
<br />

```

```
<br />
<h:link outcome="/index" value="#{bundle.ListCitasIndexLink}"/>
</h:form>
</ui:define>
</ui:composition>

</html>
```

Los textos a cambiar son:

Bundle	Nuevo texto
bundle.ListCitasTitle_citNumero	Nro
bundle.ListCitasTitle_citFecha	Fecha
bundle.ListCitasTitle_citHora	Hora
bundle.ListCitasTitle_citEstado	Estado
bundle.ListCitasTitle_citObservaciones	Observaciones
bundle.ListCitasTitle_citPaciente	Paciente
bundle.ListCitasTitle_citMedico	Medico
bundle.ListCitasTitle_citConsultorio	Consul

7.2.2 Dejar solamente la opción de eliminar cita.

Para realizar este ajuste se edita el archivo /citas/List.xhtml y se ubican y se eliminan las siguientes líneas:

```
<h:commandLink action="#{citasController.prepareView}" value="#{bundle.ListCitasViewLink}"/>
<h:outputText value=" " />
<h:commandLink action="#{citasController.prepareEdit}" value="#{bundle.ListCitasEditLink}"/>
<h:outputText value=" " />
```

Luego se ubica la línea:

```
<h:commandLink action="#{citasController.destroy}" value="#{bundle.ListCitasDestroyLink}"/>
```

Se puede apreciar que bundle.ListCitasCreateLink contiene el nombre que aparece en la tabla como "Destroy". Se busca en el archivo de bundles y se cambia por "Eliminar".

7.2.3 Eliminar los enlaces del formulario.

Para eliminar los enlaces a otras páginas del formulario se buscan y se eliminan las siguientes líneas en el archivo “/citas/List.xhtml”:

```
<h:commandLink action="#{citasController.prepareCreate}" value="#{bundle.ListCitasCreateLink}"/>
<br />
<br />
<h:link outcome="/index" value="#{bundle.ListCitasIndexLink}"/>
```

Una vez aplicados los cambios al ejecutar la aplicación se muestra lo siguiente:



Figura 7.9. Formulario para Consulta de citas ajustado

Este ajuste también desarrolla el caso de uso: Eliminar cita. Para lo anterior se debe dar click sobre el enlace “Eliminar”. Una vez eliminado el registro se muestra lo siguiente en pantalla:



Figura 7.10. Caso de uso Eliminar Cita.

8. Construcción del instalador de la aplicación.

Para realizar la generación del instalador de la aplicación o “deployment” las aplicaciones JSF se apoyan en los archivo tipo WAR. Los archivo WAR son un tipo de archivo comprimido usando el formato de compresión zip.

Cuando se hacían pruebas de la aplicación a lo largo de este recurso no se generaba instalador sino que Netbeans copiaba internamente la estructura de directorios para que el servidor GlassFish los utilizara.

Sin embargo este esquema es útil solo en la fase de desarrollo. Una vez la aplicación esté lista se debe trasladar hasta el servidor o servidores de producción.

Este recurso se apoyará en Netbeans para la creación del archivo WAR.

8.1 Creación del instalador usando Netbeans.

Para realizar esta operación se da click derecho sobre el nodo principal y se elige la opción “Clean and Build” del menú contextual así:

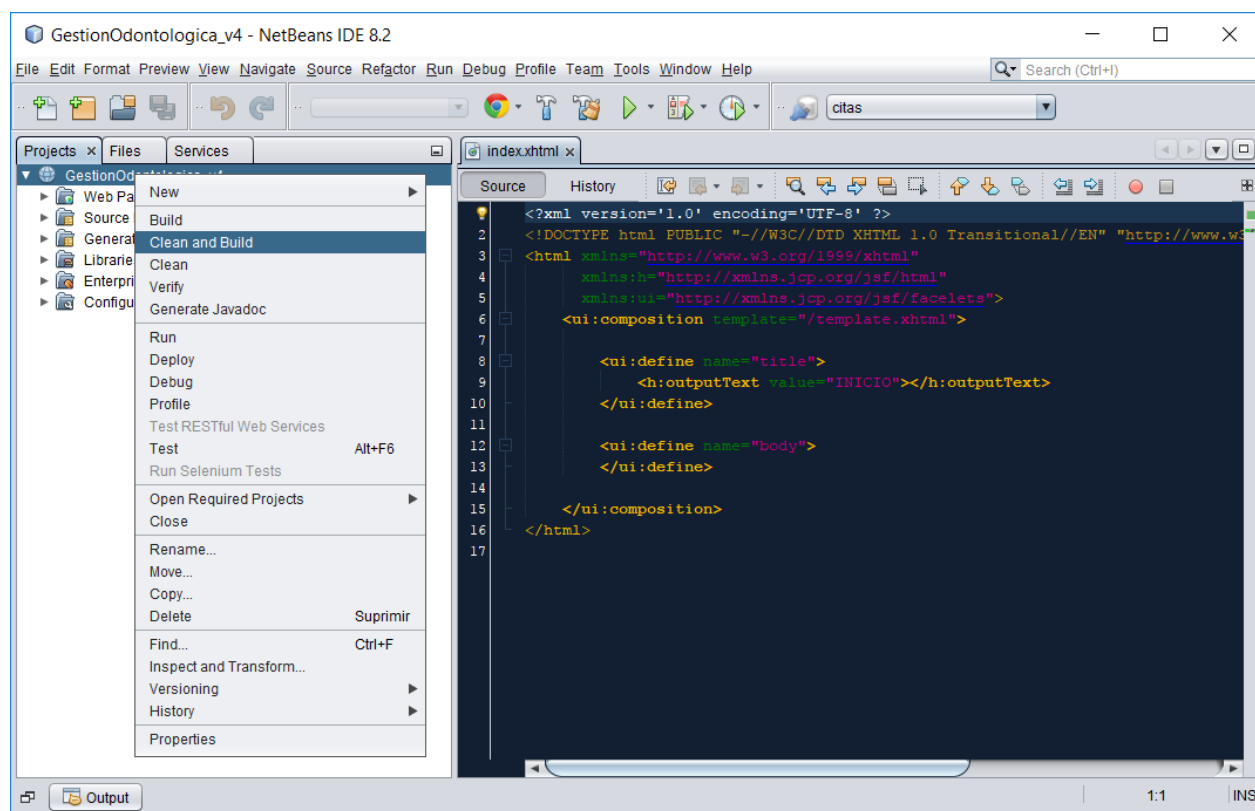


Figura 8.1. Creación del instalador de la aplicación.

Una vez seleccionada la opción el sistema muestra lo siguiente:

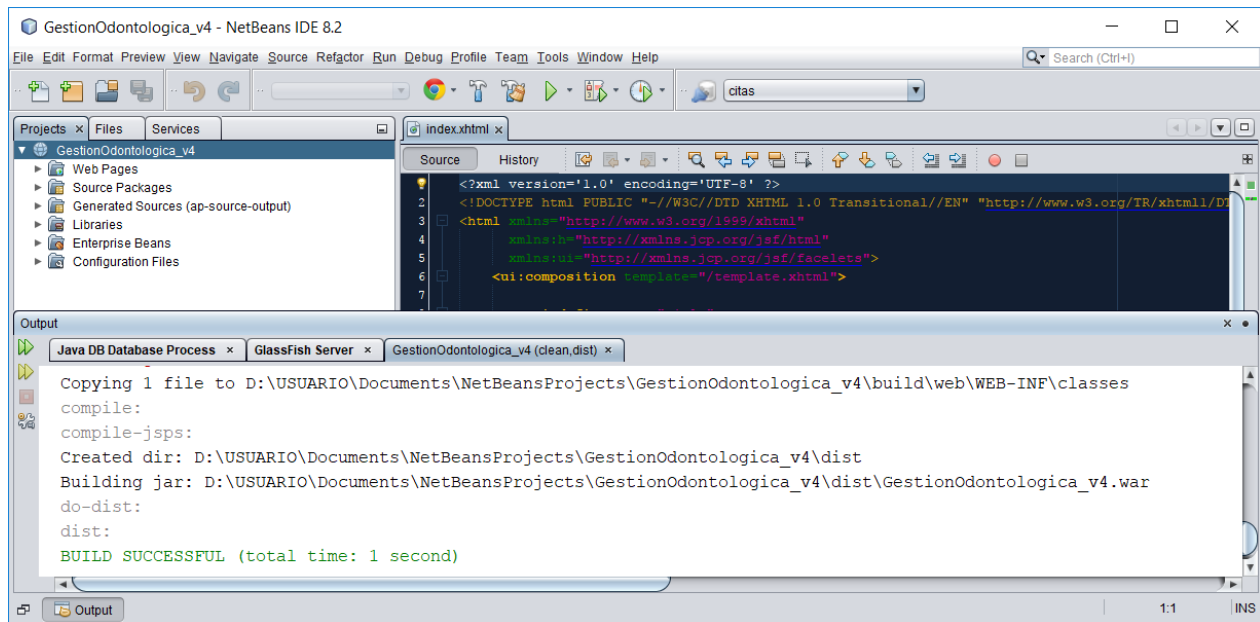


Figura 8.2. Creación del instalador de la aplicación.

La pantalla de mensajes (output) muestra que se creó el archivo “GestionOdontologica_v4.war” en el directorio: “D:\USUARIO\Documents\NetBeansProjects\GestionOdontologica_v4\dist”.

Este archivo WAR contiene todos los elementos necesarios para la ejecución de la aplicación en otro servidor. Para visualizar su contenido se puede usar una herramienta de compresión y descompresión de archivos como winzip o winrar. Para este ejemplo se muestra lo siguiente:

Nombre	Tamaño	Compri...	Tipo	Modificado
..			Carpeta de arc...	
citas			Carpeta de arc...	23/10/2017...
consultorios			Carpeta de arc...	23/10/2017...
medicos			Carpeta de arc...	23/10/2017...
META-INF			Carpeta de arc...	23/10/2017...
pacientes			Carpeta de arc...	23/10/2017...
resources			Carpeta de arc...	23/10/2017...
tratamientos			Carpeta de arc...	23/10/2017...
WEB-INF			Carpeta de arc...	23/10/2017...
index.xhtml	548	548	Chrome HTML ...	23/10/2017...
template.xhtml	1.244	1.244	Chrome HTML ...	23/10/2017...

Figura 8.3. Creación del instalador de la aplicación.

Para instalar este archivo WAR en otro servidor se copia al mismo y se siguen las instrucciones específicas de GlassFish para la instalación de aplicaciones nuevas de acuerdo a la plataforma donde se despliegue.

Glosario

Bean: modelo para la construcción de componentes de código en la plataforma Java EE que utiliza técnicas de encapsulamiento.

Componente: en desarrollo de software un componente es una porción de código reutilizable que encapsula una lógica y provee una interfaz única.

Facelet: programa para la generación de páginas web dinámicas que usa la extensión xhtml y la definición de etiquetas de la tecnología JSF.

Faces Servlet: Servlet que atiende las peticiones HTTP de los facelets.

JSF: JavaServer Faces. tecnología de la plataforma Java Platform EE que soporta la creación de páginas web dinámicas.

JPA: acrónimo de Java Persistence API. Es la capa de persistencia que contiene las clases que interactúan directamente con el modelo o base de datos.

JSR: Java Specification Request. Solicitud de cambios a las especificaciones Java.

MVC: Modelo Vista Controlador. Patrón de diseño comúnmente usado en el desarrollo web.

Servlet: programa o servicio que se ejecuta en un servidor de aplicaciones y atiende peticiones de los clientes.

Bibliografía

Goncalves, A. (2013). *Beginning Java EE 7*. New York: Apress.

Geary, D. & Horstmann, C. (2010). *Core JavaServer Faces*. Madrid: Prentice-Hall.

CONTROL DEL DOCUMENTO

CONSTRUCCIÓN OBJETO DE APRENDIZAJE



DESARROLLO DE APLICACIONES WEB CON JAVA

Centro Industrial de Mantenimiento Integral - CIMI
Regional Santander

Líder línea de producción: Santiago Lozada Garcés

Asesores pedagógicos: Rosa Elvia Quintero Guasca
Claudia Milena Hernández Naranjo

Líder expertos temáticos: Rita Rubiela Rincón Badillo

Experto temático: Nelson Mauricio Silva Maldonado

Diseño multimedia: Jesús Antonio Vecino Valero

Programador: Francisco José Lizcano Reyes

Producción de audio: Victor Hugo Tabares Carreño

Este material puede ser distribuido, copiado y exhibido por terceros si se muestra en los créditos. No se puede obtener ningún beneficio comercial y las obras derivadas tienen que estar bajo los mismos términos de la licencia que el trabajo original.

NETBEANS IDE 8.2. Copyright © 1997, 2017, Oracle and/or its affiliates. All rights reserved.

creative
commons



BY



NC



SA



REGISTERED TRADEMARK