

{ **DESARROLLO** DE **APLICACIONES** **WEB** EN **php**



FAVA - Formación en Ambientes Virtuales de Aprendizaje

SENA - Servicio Nacional de Aprendizaje.

Estructura de contenidos

	Pág.
Introducción	4
Mapa de contenidos	5
1. Construcción del modelo o base de datos	6
1.1. Activación de los servicios en XAMPP	7
1.2. Construcción de la base de datos	8
2. Desarrollo de la interfaz gráfica de usuario (GUI) plantilla	12
2.1. Vista del proyecto	12
2.2. Creación de la estructura de carpetas del proyecto	12
2.3. Maquetación	20
2.3.1. Definición del encabezado	27
2.3.2. Contenido	30
2.3.3. Menú	32
3. Desarrollo de la interfaz gráfica de usuario	43
3.1. Inicio y asignar cita	43
3.1.1. Inicio	43
3.2. Asignar cita	47
4. Desarrollo de la interfaz gráfica de usuario: consultar cita y cancelar cita	51
4.1. Consultar cita	52
4.2. Cancelar cita	53
5. Codificación de las clases del modelo	55
5.1. Clase "Paciente"	55
5.2. Clase "Cita"	58
5.3. La clase "Conexion"	59
5.4. La clase "GestorCita"	61
5.5. Codificación del método "consultarCitaPorId()"	62
5.6. Codificación del método "consultarCitasPorDocumento()"	62
5.7. Codificación del método "consultarPaciente()"	63
5.8. Codificación del método "agregarPaciente()"	63
5.9. Codificación del método "consultarMedicos()"	64

6. Codificación de las clases del controlador	64
6.1. Arranque (index.php) y la clase "Controlador"	64
6.2. Codificación del método "AgregarCita()"	66
6.3. Codificación del método "ConsultarCitas()"	72
6.4. Codificación del método "CancelarCitas()"	73
7. Incluyendo Javascript y JQuery parte 1	76
7.1. Consultar paciente.....	76
7.2. Ingresar paciente.....	82
8. Incluyendo Javascript y JQuery parte 2	88
8.1. Llenar el elemento "select" de médicos.....	89
8.2. Llenar el elemento "select" de horas	90
8.3. Codificación del método "consultarHorasDisponibles()"	92
8.4. Llenar el elemento "select" de consultorios.....	95
8.5. Consultar citas.....	96
8.6. Cancelar citas.....	99
9. Despliegue de la aplicación y construcción del instalador	101
9.1 Instalación del ambiente XAMPP en el equipo de cómputo de destino	101
9.2 Creación de la base de datos.....	102
9.3 Copia de los archivos fuentes de la aplicación	103
Glosario	104
Bibliografía.....	106
Control de documento	107

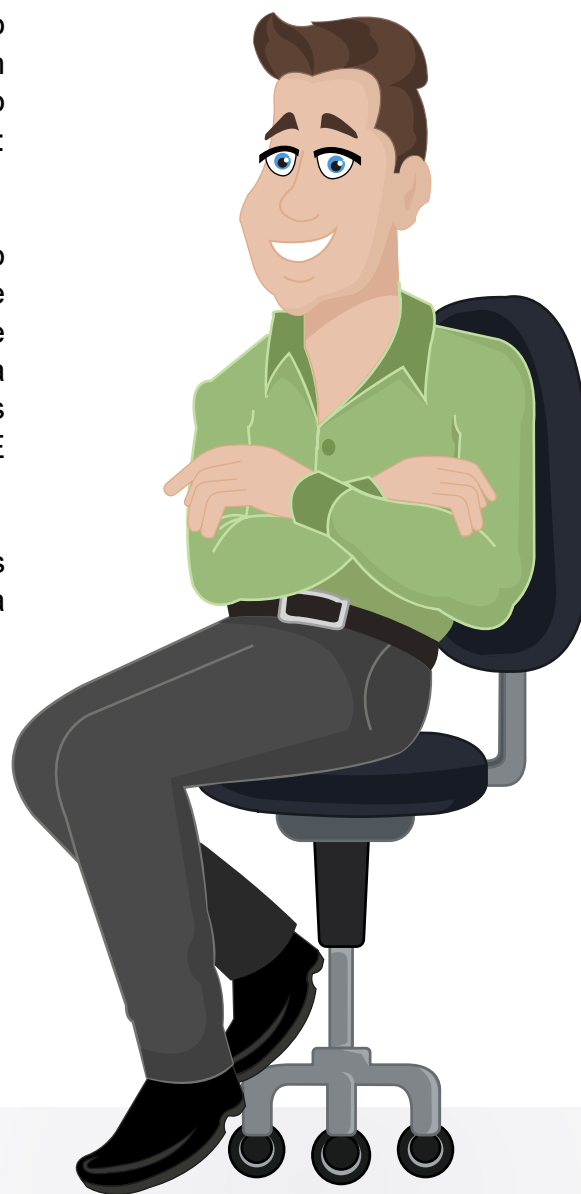
Desarrollo de aplicaciones Web en PHP

Introducción

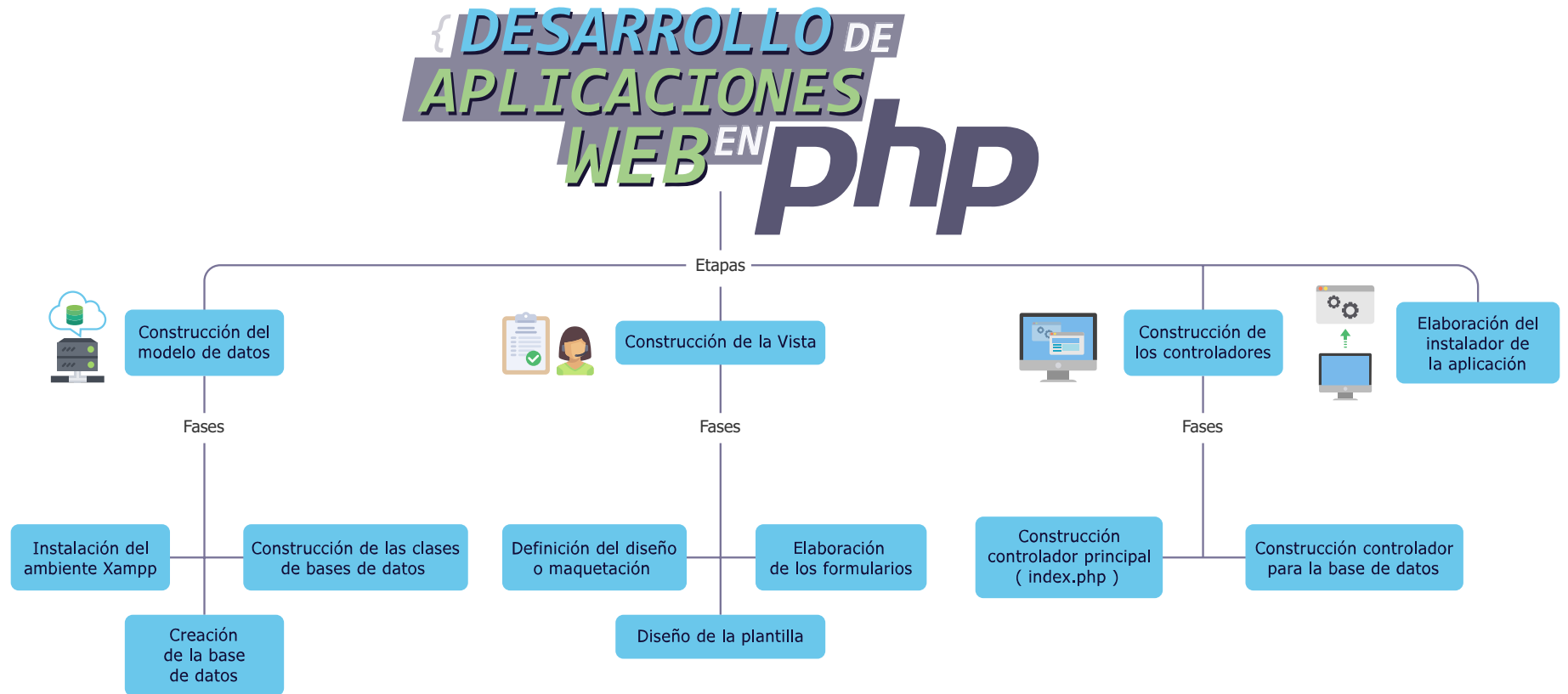
El presente recurso es una guía paso a paso sobre el desarrollo completo de una aplicación en el lenguaje de programación PHP apoyado en las tecnologías web disponibles como son: HTML, CSS, Javascript y JQuery.

El ambiente de desarrollo está compuesto por la herramienta XAMPP para windows que suministra el servidor web Apache, el lenguaje PHP y la base de datos Mysql de manera integrada. Por otra parte se usará Netbeans como entorno de desarrollo integrado o IDE para realizar la codificación.

El caso de uso del Sistema de Gestión de Citas que servirá de fuente para el desarrollo de la aplicación es “Gestionar citas”.



Mapa de contenidos



Desarrollo de contenidos

1. Construcción del modelo o base de datos

Para la construcción del sistema de información se utilizarán las siguientes tecnologías:

- HTML.
- CSS.
- JavaScript.
- Ajax.
- PHP.
- MySQL.
- Apache.
- NetBeans.

PHP es un lenguaje de programación del lado del servidor por tal motivo es necesario utilizar un servidor web sobre el cual se instalará el intérprete de PHP. Adicional a esto el DBMS que se utilizará será MySQL.

Por esta razón se utilizará un programa llamado XAMPP el cual contiene estas tres herramientas configuradas para su utilización. El enlace de descarga del XAMPP es: <http://www.apachefriends.org/es/xampp.html>.

1.1. Activación de los servicios en XAMPP

Luego de instalado el XAMPP es necesario activar los servicios con que cuenta.

Para ello se ejecuta el acceso directo a XAMPP o se ejecuta el programa “xampp-control.exe” el cual se encuentra generalmente en la carpeta d:\xampp.

Luego de ello se muestra la ventana del XAMPP Control Panel Application, y en esta se debe dar clic sobre los botones “Start” tanto de Apache como de MySQL.

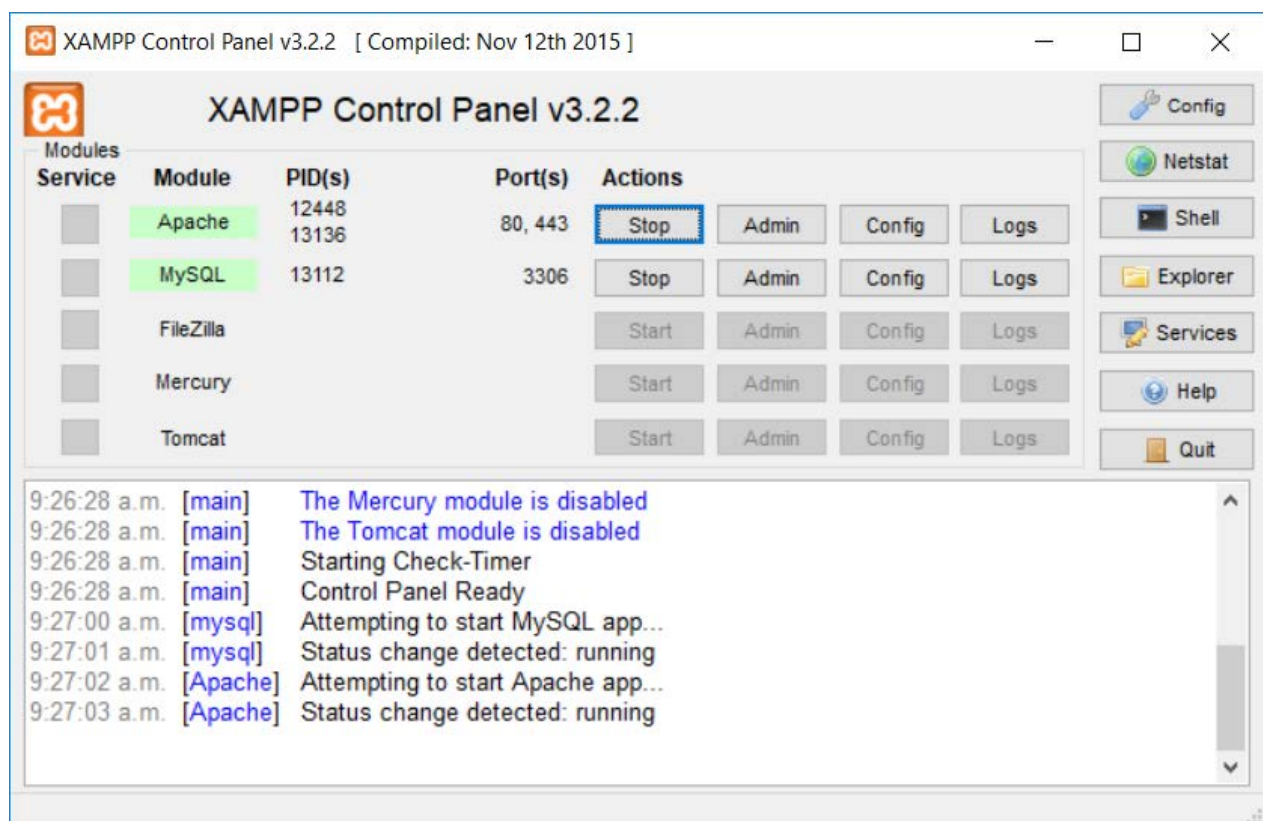


Figura 1.1. Panel de control de Xampp.

Con este proceso ya están activos los servicios y es posible empezar a trabajar.

Para verificar que los servicios se están ejecutando, se abre el navegador, (Internet Explorer, Opera, Google Chrome), y en la barra de direcciones se copia una de estas 2 direcciones:

- <http://localhost>
- <http://127.0.0.1>

Para ambos casos, se mostrará la pantalla de inicio del XAMPP.



Figura 1.2. Página principal de Xampp

1.2. Construcción de la base de datos

El primer paso para la construcción del sistema de información, es la construcción de la base de datos. Para este proyecto se utilizará MySQL como DBMS.

Nota: MySQL y MariaDB son dos motores de bases de datos que comparten una misma base o estructura y por tanto son totalmente compatibles. Ambas usan la licencia GPL que los hace libres tanto en su licenciamiento como en su código fuente. El paquete XAMPP en sus nuevas versiones hace uso de MariaDB en lugar de MySQL.



El modelo de datos a utilizar es el mismo que fue expuesto en el recurso de aprendizaje “Introducción a la base de datos MySQL”. El modelo relacional es el siguiente:

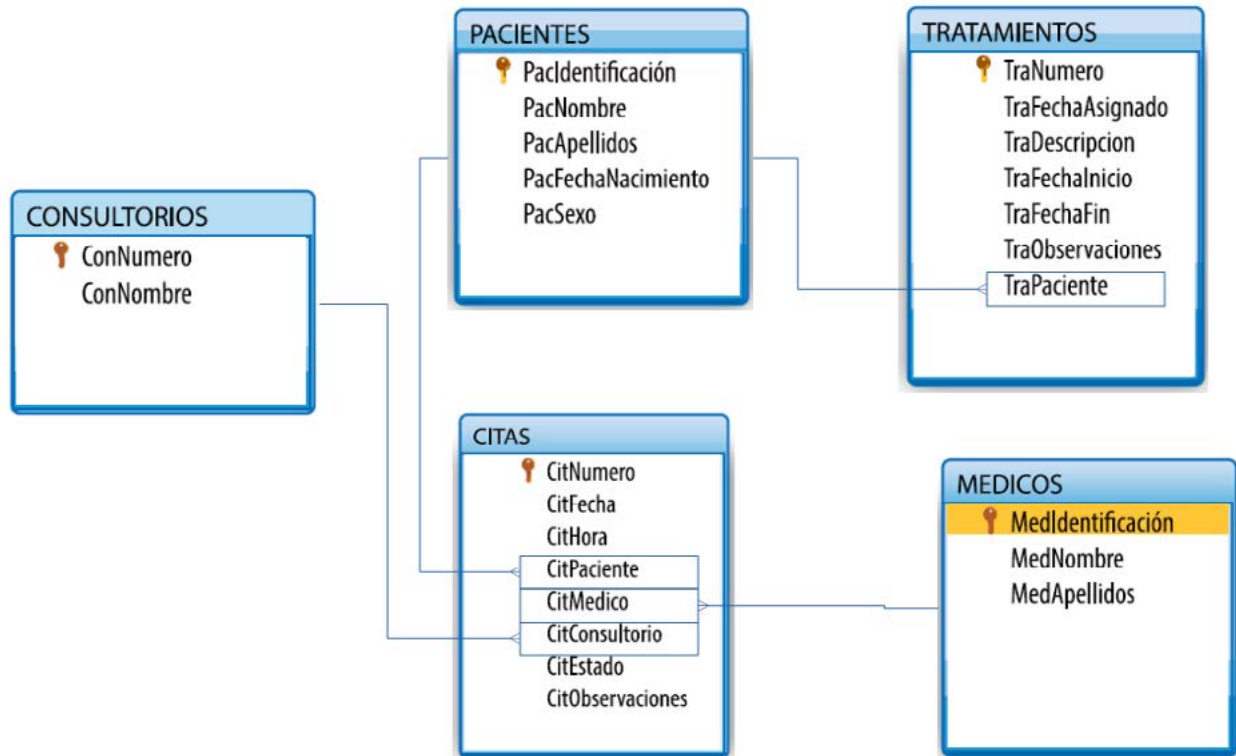


Figura 1.3. Modelo relacional de la base de datos.

El script con la base de datos es la 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;

```

Para ejecutar el script se descarga con el nombre “database_citas.sql” y desde la consola del sistema operativo se ejecuta así:

```
[consola os]
# mysql -u root -p < database_citas.sql ;
[consola]
```

Una vez ejecutado el script se puede revisar en la consola de MySQL que las tablas se hayan creado así:

```
mysql> show tables ;
+-----+
| Tables_in_citas |
+-----+
| Consultorios    |
| Medicos         |
| Pacientes       |
| Tratamientos    |
| citas           |
+-----+
5 rows in set (0,01 sec)

mysql>
```

2. Desarrollo de la interfaz gráfica de usuario (GUI) plantilla

En esta sesión se construirá la interfaz web teniendo en cuenta el patrón Modelo Vista Controlador.

Se debe tener en cuenta que el caso de uso que se desarrollará en el presente recurso es “Gestionar citas”.

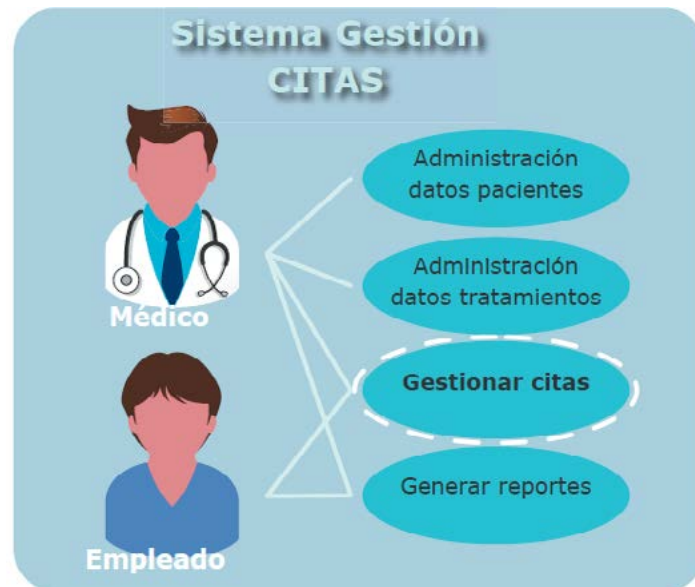


Figura 2.1. Casos de uso de la aplicación.

2.1. Vista del proyecto

Para el desarrollo del sistema de información se utilizará el patrón Modelo Vista Controlador

La particularidad del patrón Modelo Vista Controlador es que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

Para la vista se utilizarán básicamente tres tecnologías que son HTML para el contenido, CSS para los estilos del sitio web y JavaScript para las validaciones.

El IDE o entorno de desarrollo a utilizar es NetBeans.

2.2. Creación de la estructura de carpetas del proyecto

Para organizar los archivos fuentes del proyecto se crea una estructura de carpetas que guarda relación con el modelo MVC.

- Para ello se ejecuta NetBeans.

Se selecciona la opción “File->New Project...”

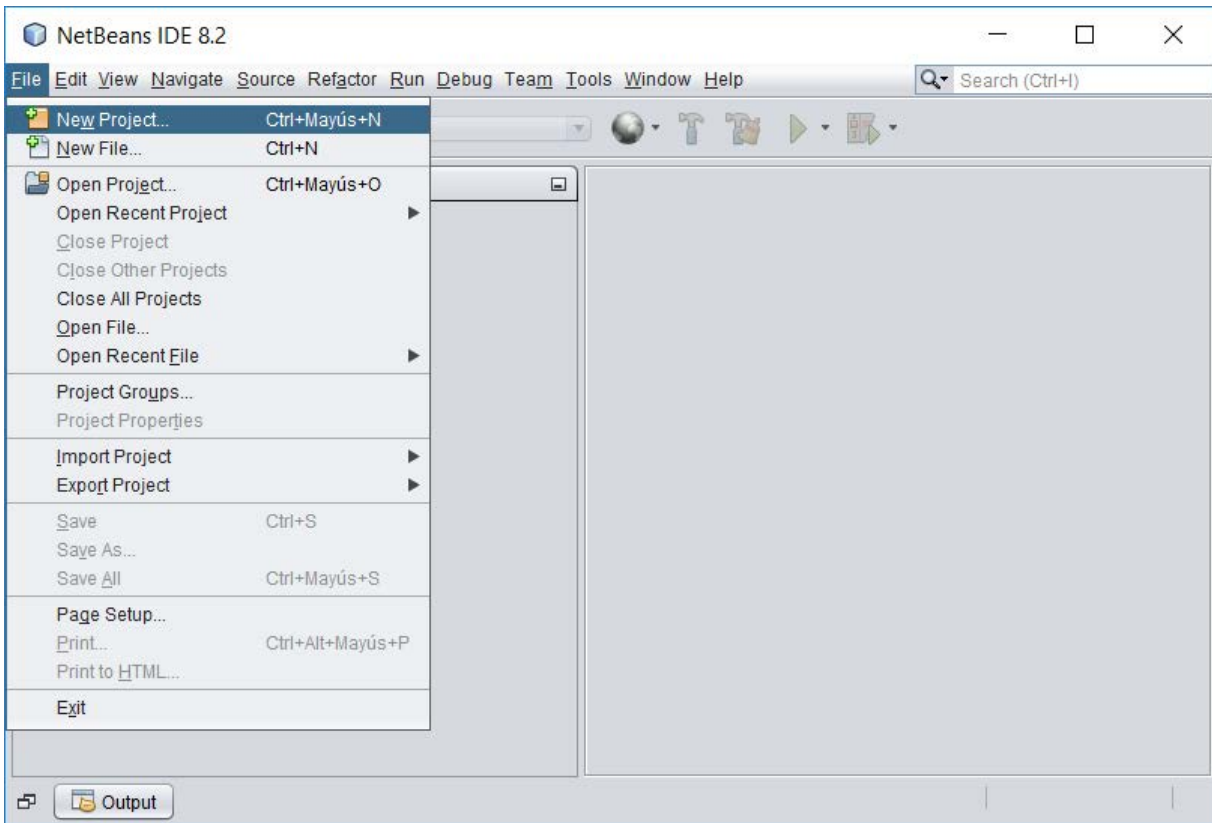


Figura 2.2. Crear proyecto en Netbeans paso 1.

En el campo “Categories” se selecciona “PHP” y en “Projects” se selecciona “PHP Application”.

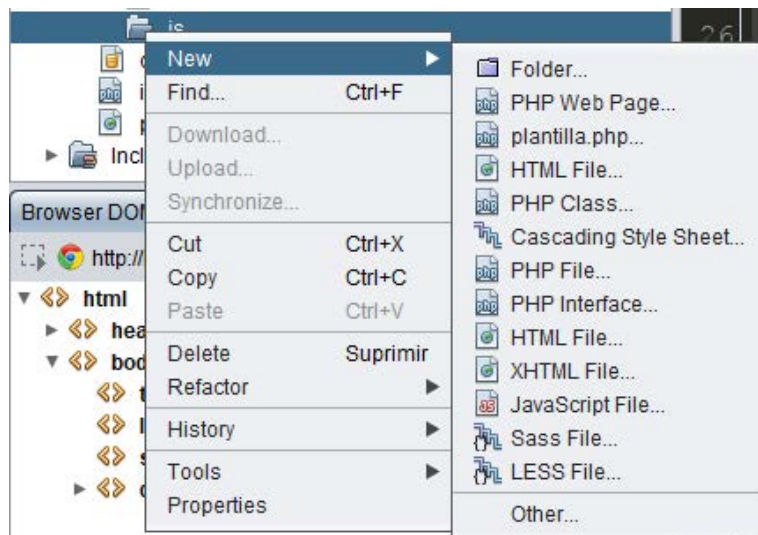


Figura 2.3. Crear proyecto en Netbeans paso 2.

Se oprime “Next” para el siguiente paso donde se asigna el nombre y la ubicación al proyecto.

En el campo “Project Name” se introduce “GestionOdontologica”. En “Sources Folder” deberá aparecer la ruta del directorio raíz de Xampp más el nombre de la carpeta donde se albergará el proyecto.

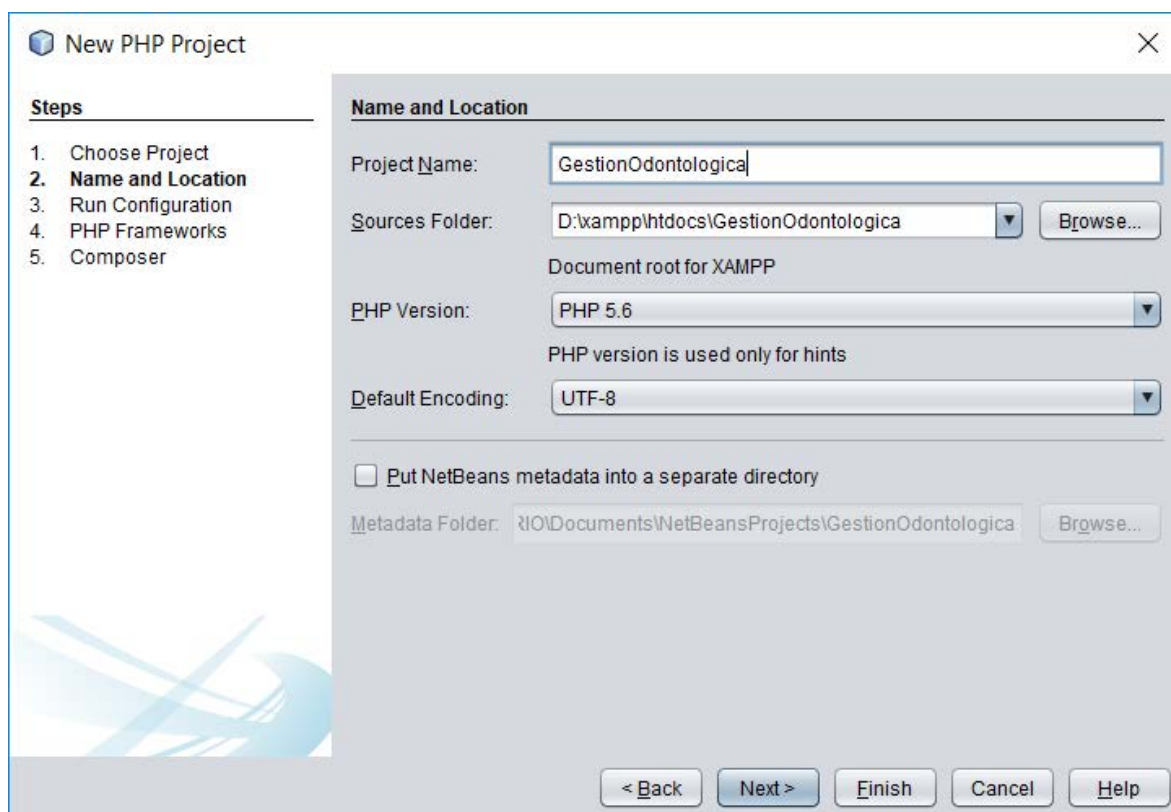


Figura 2.4. Crear proyecto en Netbeans paso 3.

Se oprime “Next” para el siguiente paso. Los valores se dejan como están.

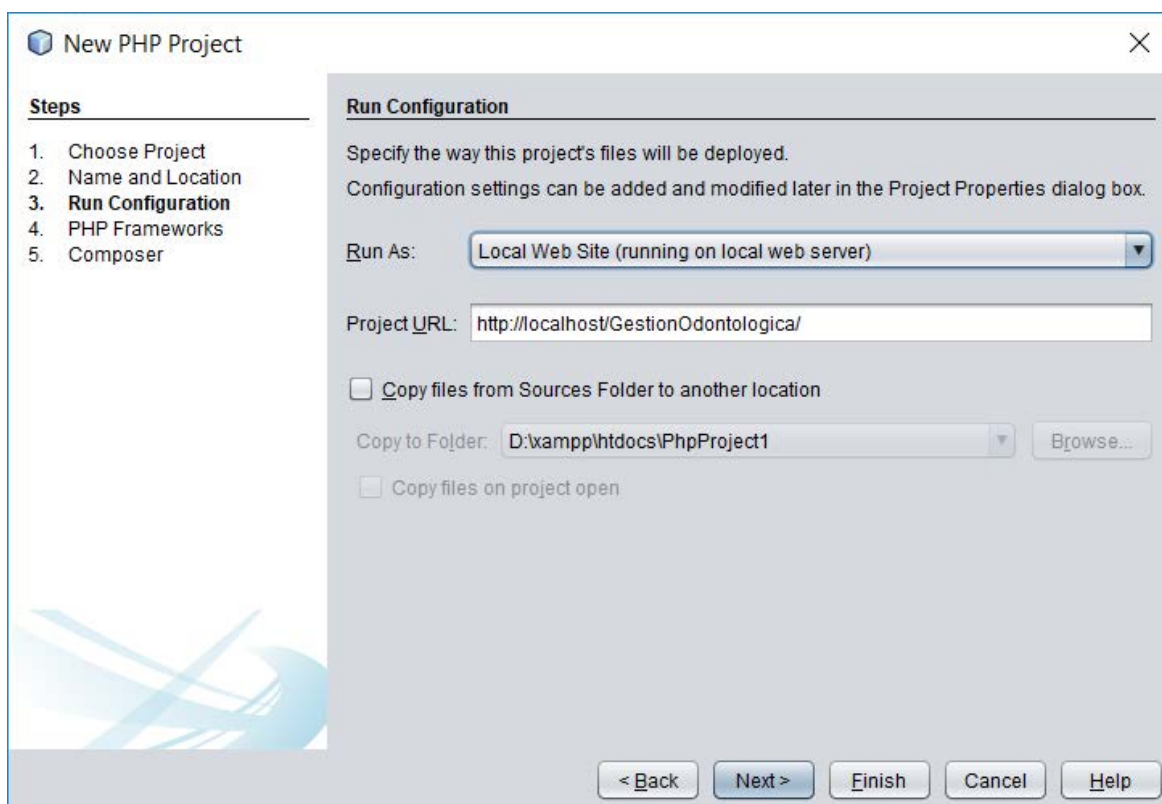
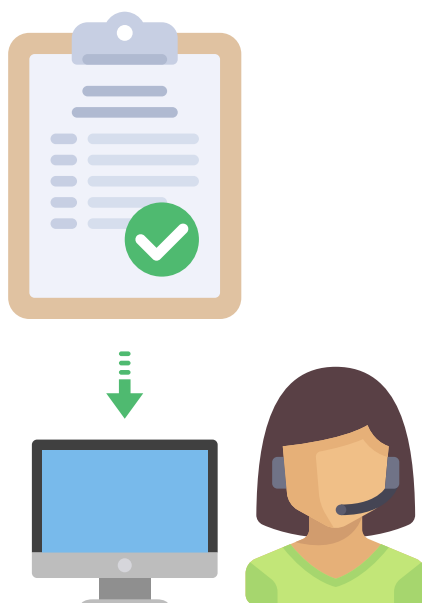


Figura 2.5. Crear proyecto en Netbeans paso 4.



Se oprime “Next” para el siguiente paso. No se selecciona ninguno de los frameworks descritos.

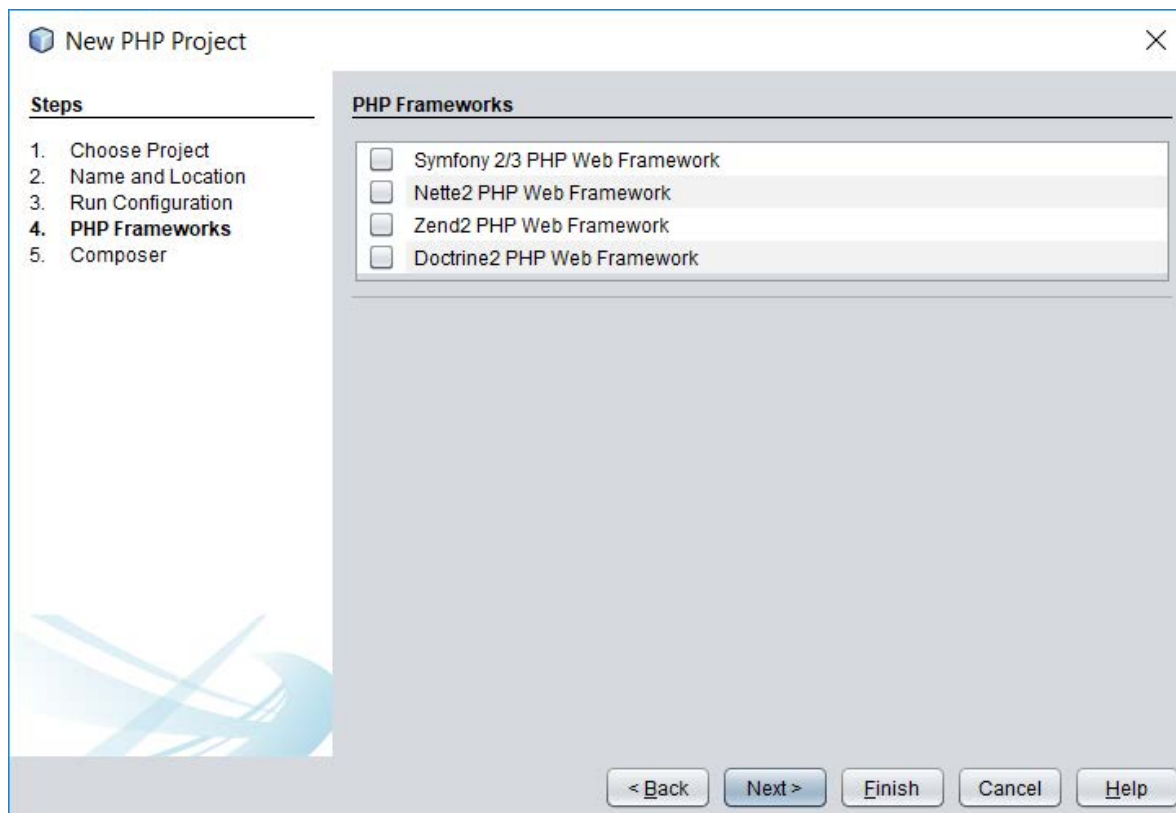


Figura 2.6. Crear proyecto en Netbeans paso 5.

Por último se oprime el botón “Finish” y aparece la estructura del proyecto así:

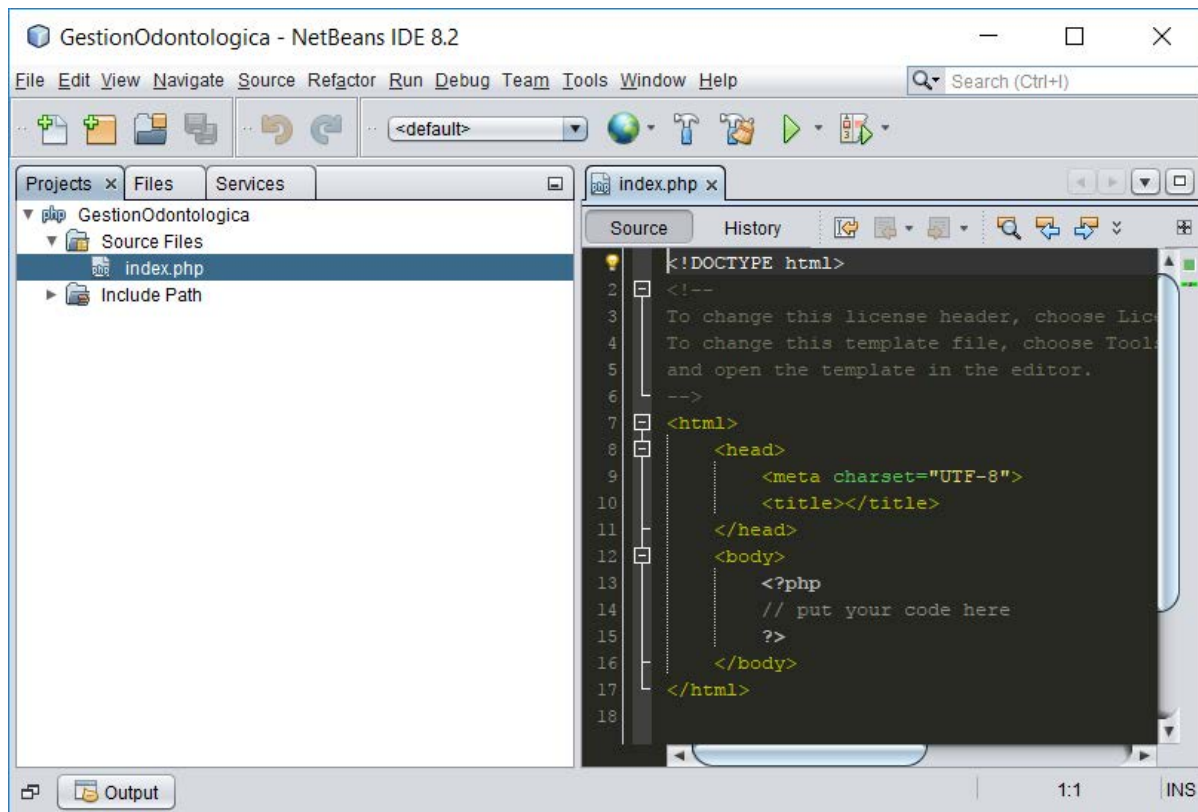


Figura 2.7. Crear proyecto en Netbeans paso 6.

En la sección izquierda está la carpeta “Source Files” y dentro de ella se encuentra “index.php”, que a partir de este momento se llamará “Arranque (index.php)”, el cual es el archivo que por defecto se abre al iniciar el sitio.

Como el sitio va a utilizar el patrón MVC solamente una página podrá ser accedida desde el navegador, dicha página es index.php, la cual será el Arranque (index.php) o controlador principal de nuestro sitio web.

Todas las llamadas son procesadas por el Arranque (index.php) (index.php). Desde éste se llama al controlador para mostrar la página correcta.

En la sección derecha nos muestra el área de edición de código.

Para probar que el entorno esté operativo se ejecuta el siguiente código:

```
<html>
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>
    <?php
      echo "Hola mundo";
    ?>
  </body>
</html>
```

Luego se oprime el botón “Ejecutar” y aparece lo siguiente:

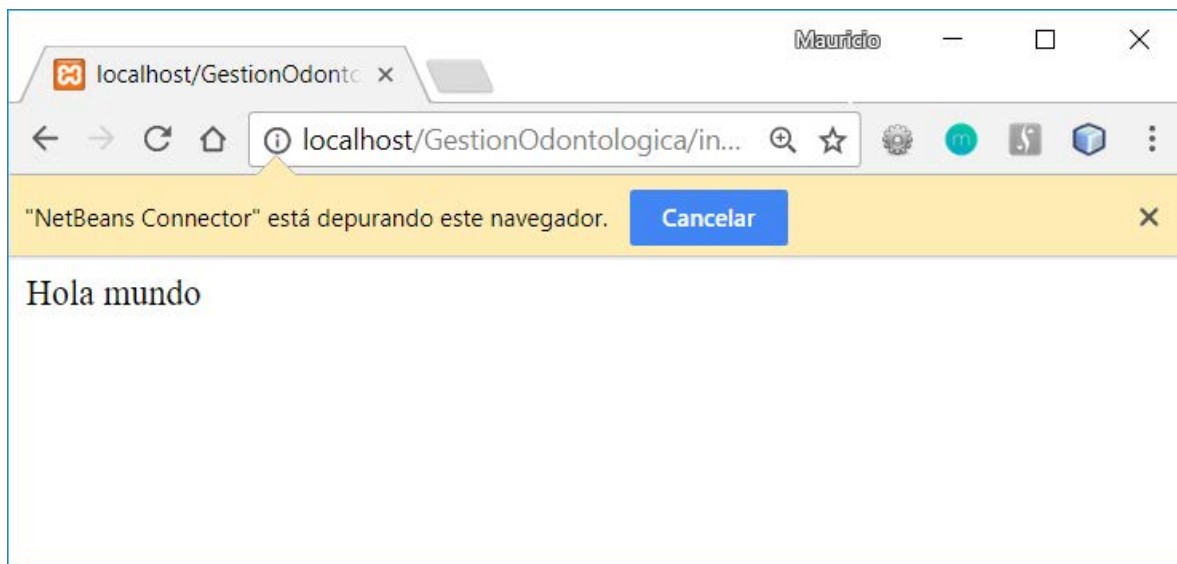


Figura 2.8. Prueba de PHP

La información que aparece antes del texto indica que Netbeans está en modo de depuración. En modo de depuración el navegador se recarga automáticamente los archivos cada vez que se graban los cambios.

A continuación se crean las tres carpetas principales: una llamada Modelo, otra llamada Vista y la última llamada Controlador.

Para ello se da clic derecho sobre la carpeta Source Files->New->Folder...

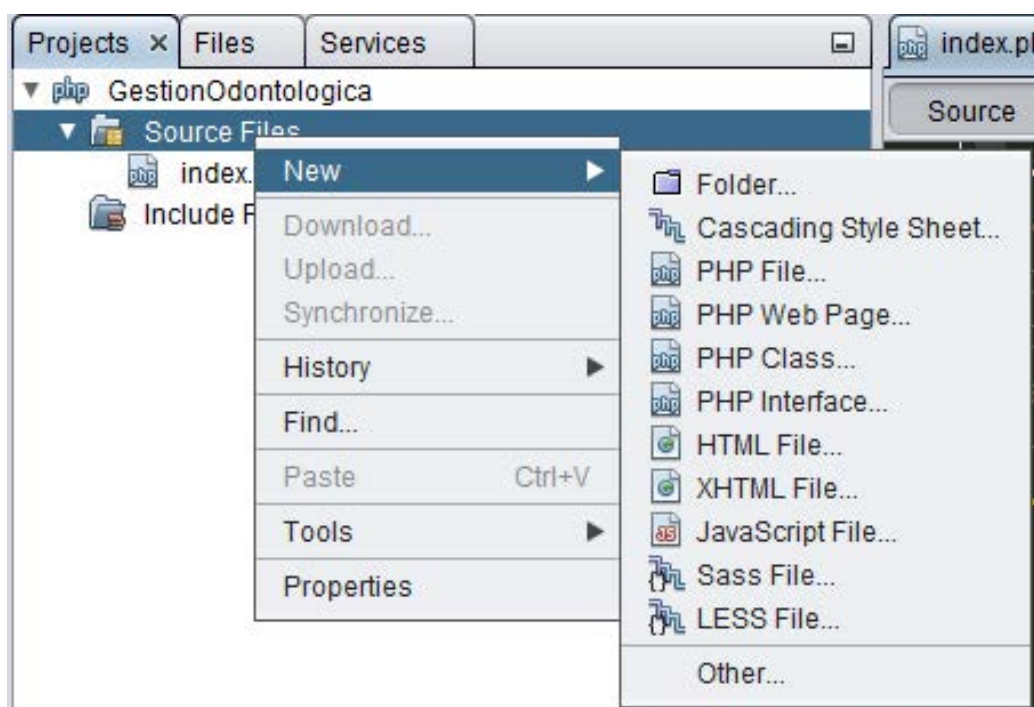


Figura 2.9. Crear carpeta en proyecto Netbeans parte 1.

Se crea la carpeta “Modelo”.

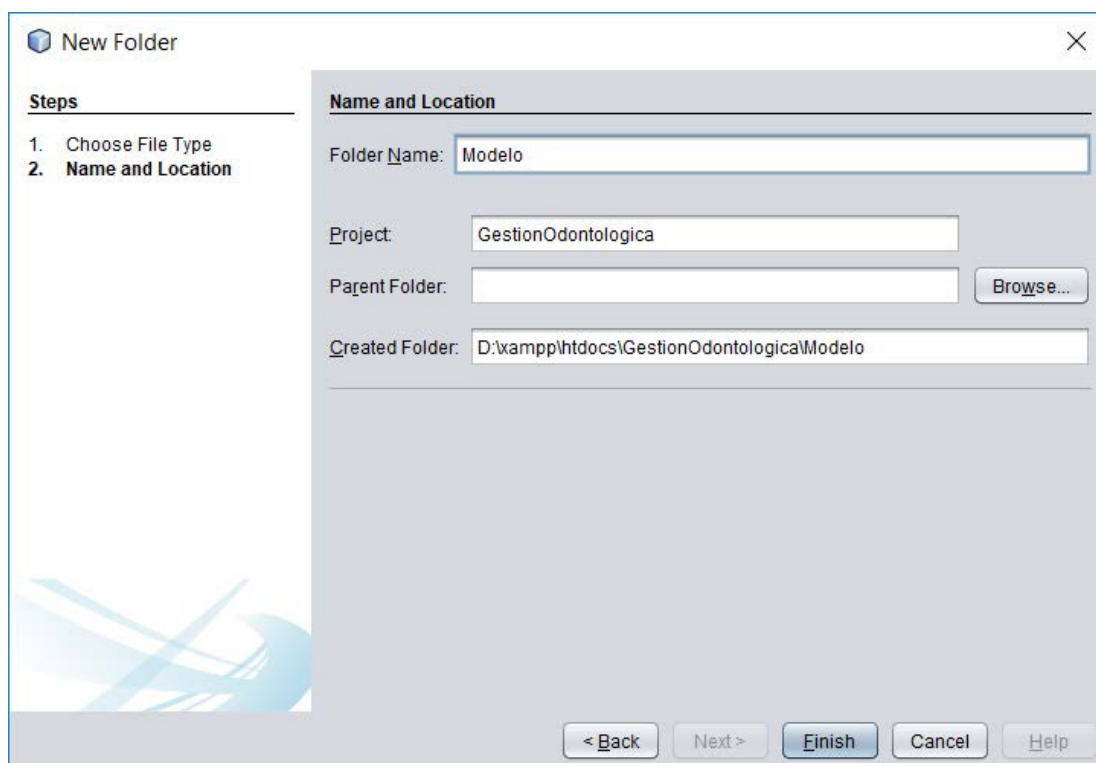


Figura 2.10. Crear carpeta en proyecto Netbeans parte 2.

Se hace lo mismo para las carpetas “Vista” y “Controlador”.
Dentro de la carpeta “Vista” se crean las carpetas “html”, “imagenes”, “css”, y “js”.

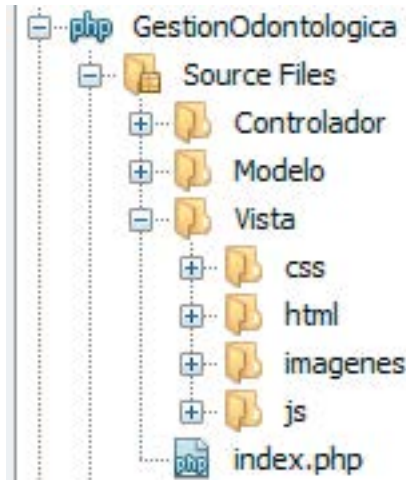


Figura 2.11. Crear carpeta en proyecto Netbeans parte 3.

2.3. Maquetación

La maquetación es la forma como están distribuidos las secciones y elementos en una página web. Se va a utilizar la siguiente estructura:

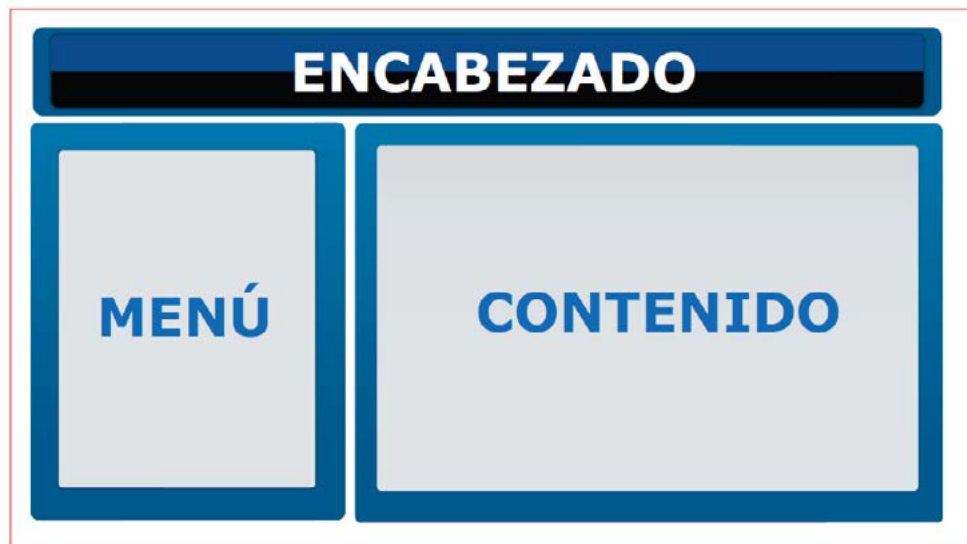


Figura 2.12. Maqueta de la aplicación.

- a. En la estructura se identifican cuatro bloques.
- b. El primero corresponde al contenedor y está delineado en rojo.

- c. El segundo elemento es un encabezado.
- d. El tercer elemento es un menú.
- e. El cuarto elemento es el contenido.

Ahora se crea una plantilla inicial con la estructura definida la cual servirá de base para las páginas del sitio. Para esto se necesita crear dos archivos:

El primer archivo se llama “plantilla.php” y se creará dentro de la carpeta “vista/html”.

Se oprime el botón derecho sobre la carpeta html y en el menú contextual se selecciona la opción “New PHP Web Page...”:

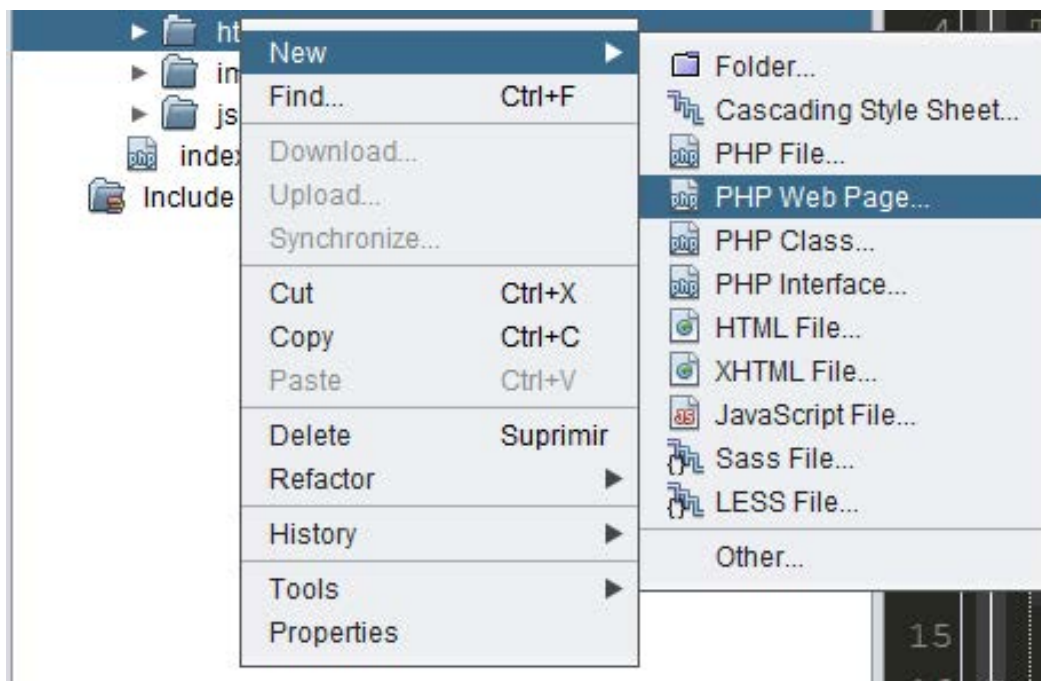


Figura 2.13. Crear página tipo web en PHP en Netbeans.

En el cuadro de diálogo “New PHP Web Page” en el campo “File Name” se escribe “plantilla”, se verifica que el campo Folder tenga la ruta Vista/html.

Se oprime el botón “Finish”.

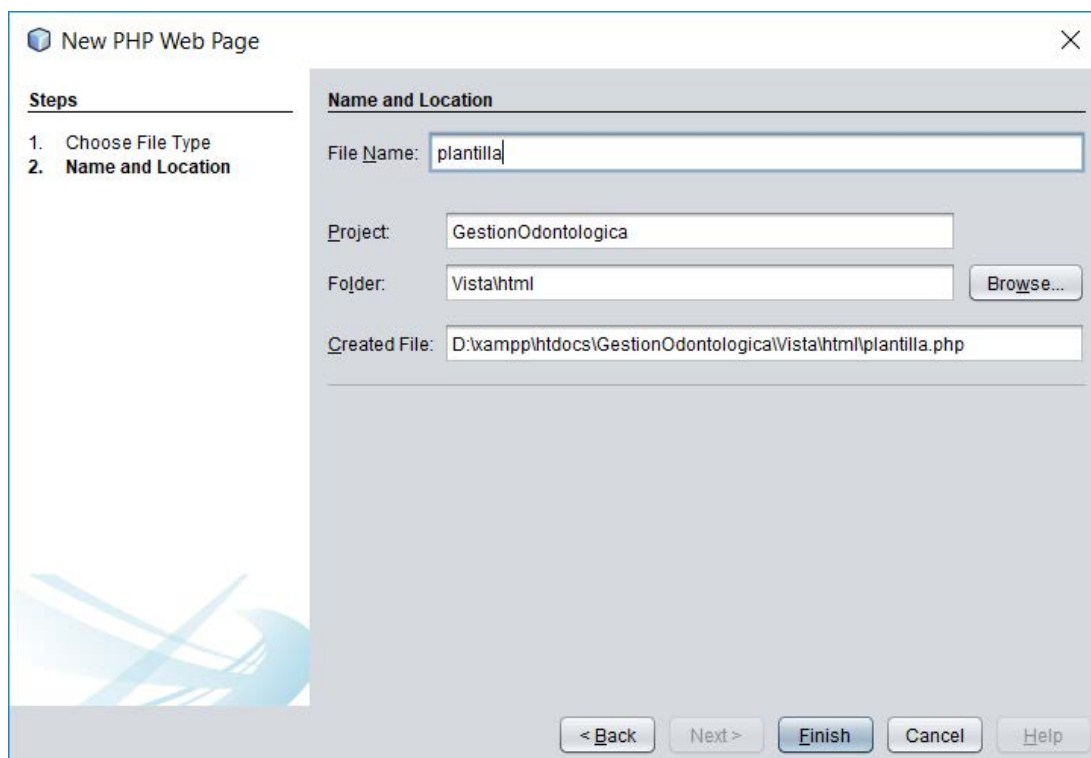


Figura 2.14. Crear página tipo web en PHP en Netbeans parte 2.

El segundo archivo a crear es “estilos.css”. Dicho archivo será creado dentro de la carpeta “Vista/css”.

Se oprime el botón derecho del ratón sobre la carpeta “css” y en el menú contextual se selecciona la opción “New->Cascading Style Sheet...”:

En el cuadro de diálogo “New Cascading Style Sheet”, en el campo “File Name” se escribe “estilos”. Se verifica que el campo Folder tenga la ruta “Vista/css”.

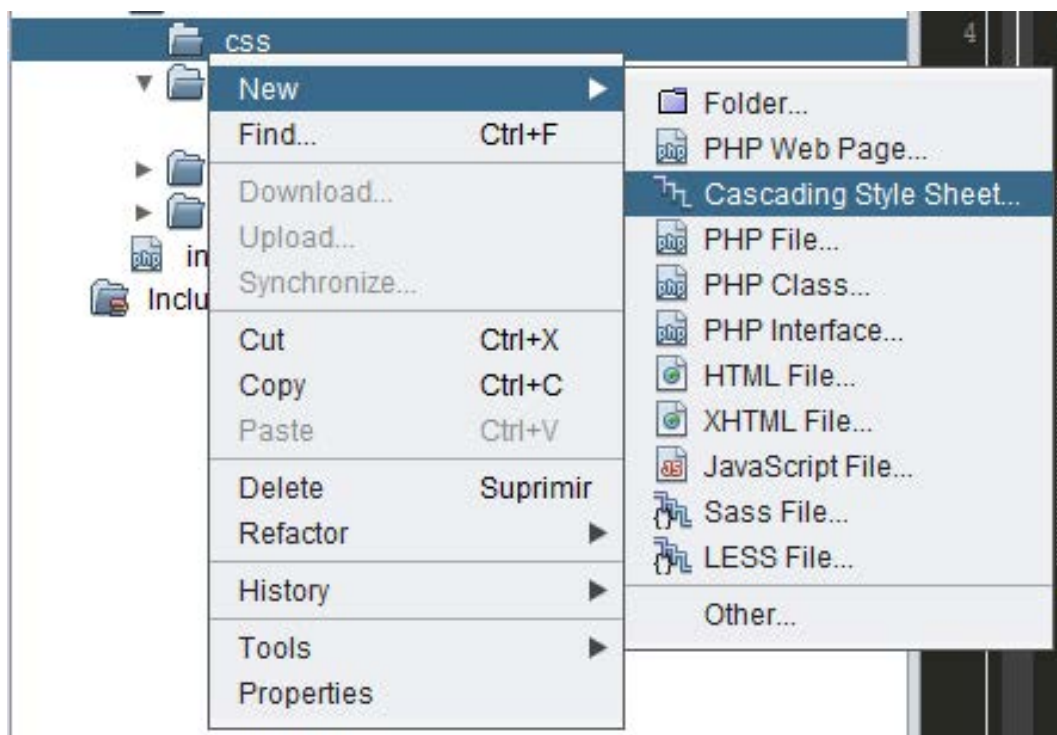


Figura 2.15. Crear página tipo CSS en Netbeans.

Se da clic en "Finish".

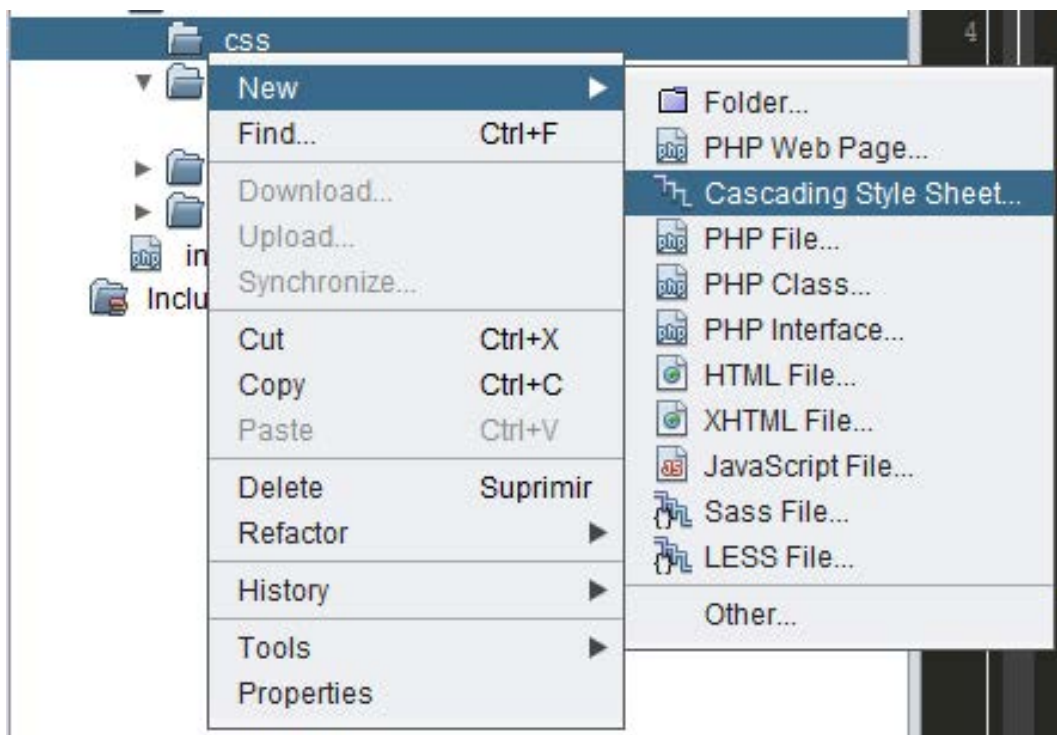


Figura 2.16. Crear página tipo CSS en Netbeans parte 2.

Se modifica el código del Arranque (index.php) para desde allí poder tener acceso al contenido de la plantilla. Como se muestra a continuación:

```
<?php
    require_once 'Vista/html/plantilla.php';
?>
```

Se crea un contenido inicial en la plantilla para ver los resultados de la maquetación. Inicialmente la plantilla tendrá el aspecto mostrado en la imagen.



Figura 2.17. Vista preliminar de la plantilla en el navegador.

El código es el siguiente y debe ser ingresado en el archivo “plantilla.php”.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Sistema de Gestión Odontológica</title>
    <link rel="stylesheet" type="text/css" href="Vista/css/estilos.css">
  </head>
  <body>
    <div id="contenedor">
      <div id="encabezado">
        <h1>Sistema de Gestión Odontológica</h1>
      </div>
      <ul id="menu">
        <li><a href="index.php">inicio</a> </li>
        <li><a href="index.php?accion=asignar">Asignar</a> </li>
        <li><a href="index.php?accion=consultar">Consultar Cita</a> </li>
        <li><a href="index.php?accion=cancelar">Cancelar Cita</a> </li>
      </ul>
      <div id="contenido">
        <h2>Título de página</h2>
        <p>Contenido de la página</p>
      </div>
    </div>
  </body>
</html>
```

Con esta estructura de maquetación, el tamaño del contenedor principal será fijo, por lo tanto es necesario asignarle un color al fondo.

El color gris se aplicará a la etiqueta “body” a través de la hoja de estilos del proyecto “Vista/css/estilos.css”.

Se agrega el siguiente código en el archivo de estilos.

```
body {
  background-color: #dfdfdf;
}
```

Una vez grabada la línea de código el resultado es el siguiente:



Figura 2.18. Vista preliminar de la plantilla en el navegador con fondo.

Ahora se crea un selector de id para el contenedor principal.

El tipo de letra será “Lucida Sans Unicode”, como segunda opción “sans-serif”. Esta letra tendrá un tamaño de 1em. El color de fondo es blanco. Va a tener un borde color gris de un pixel sólido, la alineación será justificada, tendrá un margen superior e inferior de 10px y los laterales serán automáticos y el ancho será de 900px.

En ese orden el estilo se definirá de la siguiente manera y será agregado a la hoja de estilos.

```
#contenedor {
    font-size:1em;
    background-color: #ffffff;
    border:#d3d3d3 1px solid;
    text-align: justify;
    margin: 10px auto;
    width : 900px ;
}
```

- Luego de agregar el estilo el resultado será el siguiente.



Figura 2.19. Vista preliminar aplicando regla para el contenedor

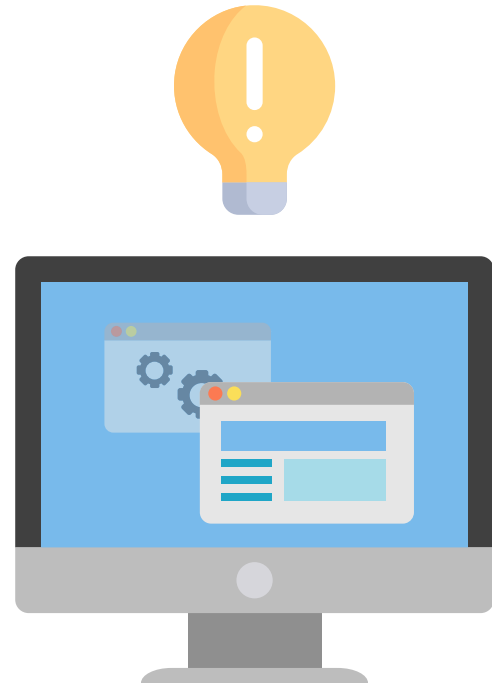
2.3.1. Definición del encabezado

El encabezado tendrá una imagen de fondo como banner y un texto el cual ya está incluido como contenido en la página web.

Para esto se crea un selector de id con el nombre “encabezado”. El selector tendrá como estilos un ancho de 900 pixels, una altura de 150 pixels, y una imagen de fondo.

Para conservar el orden del proyecto es necesario que la imagen que se muestre en el encabezado este almacenada en la carpeta “Vista/imagenes”.

Para crear la imagen del encabezado se puede usar cualquier editor de imágenes. Una vez diseñada se guarda como un archivo “png” o “jpg” y se copia a la carpeta de imágenes.



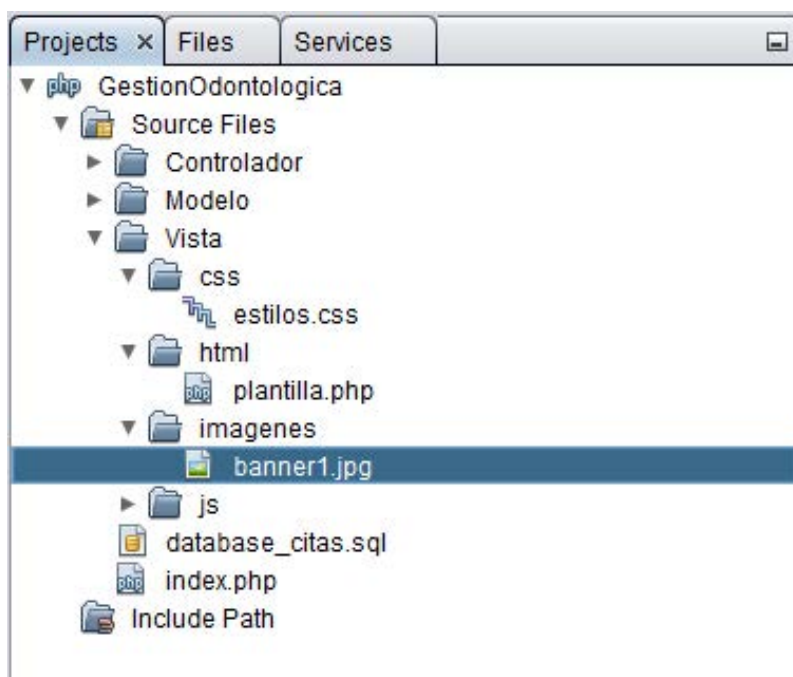


Figura 2.20. Ubicación del banner en la cartera de imágenes.

Teniendo la imagen y las características que posee el selector #encabezado, su código para generar los estilos es el siguiente.

```
#encabezado {  
    width: 900px;  
    height: 150px;  
    background : url('../imagenes/banner1.jpg') no-repeat ;  
    border: #d3d3d3 1px solid ;  
}
```

El resultado se muestra en la imagen:

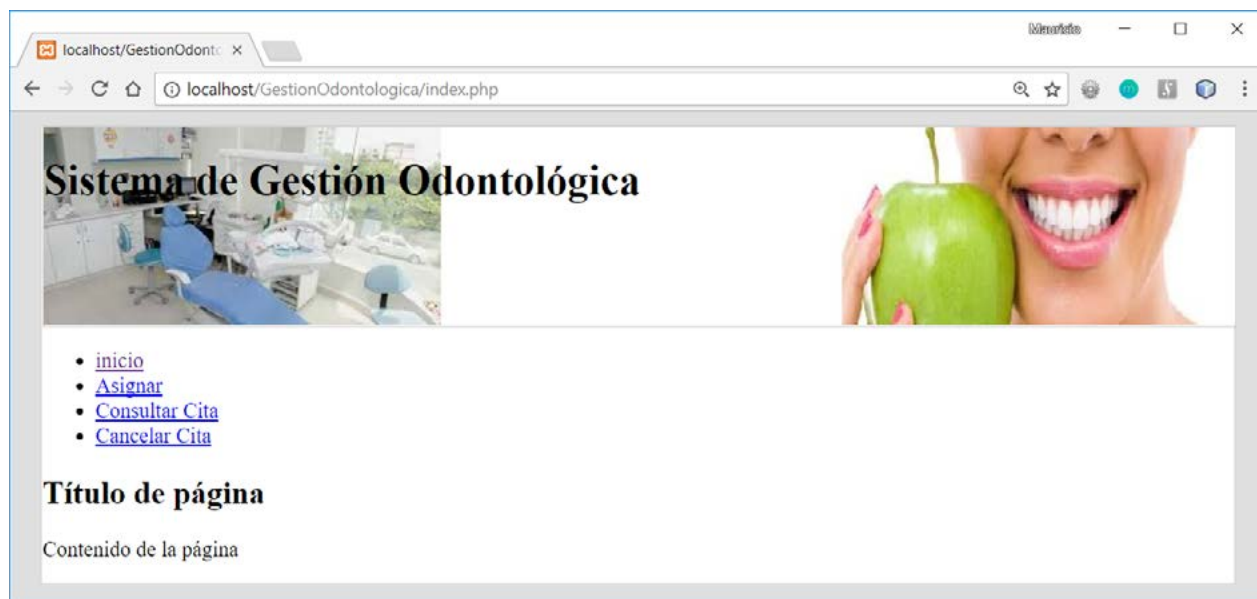


Figura 2.21. Vista preliminar con el banner del encabezado.

El banner fue agregado pero el título de la página no aparece en una posición adecuada, por lo tanto es necesario centrarlo y ubicarlo más abajo. Estas modificaciones se realizan en el selector de elemento “h1” el cual queda con la siguiente estructura.

```
#encabezado h1 {  
    text-align: center ;  
    margin-top: 50px;
```

El encabezado ya tiene la apariencia adecuada, por lo tanto se puede continuar trabajando con otra de las secciones.

Una vez realizados los cambios el navegador se actualizará para mostrar lo siguiente:



Figura 2.22. Vista preliminar del encabezado con estilos aplicados.

2.3.2. Contenido

Se continúa con el contenido de la plantilla. Primero se asigna un borde de un pixel, además se le establecerán márgenes tanto externos como internos (padding).

La regla css para ese estilo es la siguiente:

```
#contenido {
    border: #d3d3d3 1px solid ;
    margin: 20px 20px 10px;
    padding: 10px 15px;
}
```

El resultado con este cambio se muestra a continuación.



Figura 2.23. Vista preliminar de la sección de contenido.

Se continúa con el título de la página el cual se define con el elemento “h2”. Este tendrá color azul, y un borde inferior de color gris. La regla css es la siguiente:

```
#contenido h2 {
    color : #0075ff;
    border-bottom: #CFBFB5 1px solid;
}
```

El resultado es el siguiente.



Figura 2.24. Vista preliminar de la sección de contenido con estilos aplicados.

2.3.3. Menú

Una vez realizada la sección de contenido se sigue con el menú que según el diseño debe estar al lado izquierdo de la página mientras que el contenido a la derecha.

Se define un estilo de tipo id para el menú el cual flotará a la izquierda y tendrá un ancho de 120px. Este será el contenedor que alojará el menú a la izquierda.

La regla css es la siguiente:

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


Una vez se actualice el archivo de estilos aparece lo siguiente:



Figura 2.25. Vista preliminar de la sección del menú.

Se puede observar que el contenido se ha traslapado con el menú. Por lo anterior es necesario desplazar a la derecha el margen del contenido. Esa modificación se realiza en la regla de “div#contenido” así:

```
div#contenido {
    border: #d3d3d3 1px solid ;
    margin: 20px 20px 10px 180px;
    padding: 10px 15px;
}
```

Se ha asignado un desplazamiento al margen izquierdo. Ya no traslapa el menú como se muestra a continuación:



Figura 2.26. Vista preliminar de la sección del menú con la regla CSS aplicada.

Los elementos del menú ya se encuentran ubicados en la posición deseada pero no poseen el aspecto que se espera por tal motivo se utilizarán reglas CSS para modificar su aspecto.

Para observar los cambios efectuados a la lista se asignará un fondo diferente al selector “#menu”. Este color se eliminará al final. La regla se modifica así:

```
#menu {
    float: left;
    width: 120px;
    background-color: #D4FFFF ;
}
```

Se puede observar el cambio en el estilo en el menú de la plantilla a continuación:



Figura 2.27. Vista preliminar de la sección del menú con la regla CSS aplicada.

Ahora se aplica un estilo a cada uno de los elementos de la lista para que tenga el aspecto de botón. Por lo tanto se debe crear un selector descendente para la etiqueta “li” dentro del selector “#menu” así “#menu li”.

El selector descendente aplica el conjunto de estilos a los elementos que se encuentran dentro de otros elementos definidos en la regla.

Definiendo el selector de esta forma el estilo únicamente se aplicará a los elementos “li” que se encuentren dentro de una etiqueta con id “menu”.

Lo que se hace en cada elemento del menú será:

- a. Asignarle un borde gris de un pixel, definir un color de fondo azul.
- b. Establecer un margen inferior que separe los botones.
- c. Disminuir el tamaño de la letra.
- d. Centrar el texto.
- e. Establecer un interlineado para que el texto quede a buena distancia de los bordes superior e inferior del botón.

Teniendo en cuenta esas características la regla CSS queda así:

```
#menu li {
    border: #d3d3d3 3px solid;
    background-color: #0075ff;
    margin-bottom: 5px;
    font-size: 0.9em;
    text-align: center;
    line-height: 30px;
}
```

Ahora se elimina el “background-color” del selector “#menu” para que no se muestre el fondo azul que nos sirvió de referencia.

Con estas modificaciones el resultado es el siguiente.



Figura 2.27. Vista preliminar de la sección del menú con la regla CSS aplicada.

En este momento el menú de la plantilla está tomando el diseño esperado pero es necesario realizar las siguientes modificaciones:

- El texto del enlace no debe estar subrayado.
- El color del texto debe ser negro.
- El enlace esté habilitado sobre cualquier región del botón.

La etiqueta <a> es un elemento HTML en línea que se ajusta a su contenido. Se puede cambiar este comportamiento aplicándole a los enlaces del menú el atributo “display:block”. Esto lo convierte en un elemento de bloque y hace que ocupe todo el contenedor.

Con estos cambios, el selector quedaría con la siguiente estructura.

```
#menu a {
    display: block;
    text-decoration: none;
    color : #000000;
}
```

Una vez guardados los cambios al archivo de estilos aparece lo siguiente:



Figura 2.28. Vista preliminar de la sección del menu con la regla CSS para el ancla aplicada.

La plantilla está lista solo falta agregarle algunos comportamientos al menú. El primer comportamiento es un cambio de fondo al botón cada que el mouse pasa sobre él.

La regla CSS que genera este estilo es:

```
#menu li:hover {
    background-color: #7092be;
}
```

El resultado se muestra en la siguiente imagen teniendo en cuenta que el puntero del ratón está sobre el segundo enlace.



Figura 2.29. Vista preliminar de la sección del menú con la regla CSS para la lista aplicada.

La regla CSS que se acaba de crear: “#menu li: hover”, se aplicará también para identificar la página actual así:

```
#menu li:activa,#menu li:hover {
    background-color: #7092be;
}
```

Si se guarda y actualiza se puede observar que no ha surtido ningún cambio la página web. Esto es debido a que aún no se ha definido la clase en la etiqueta “li” de acuerdo a la página en la cual se está trabajando.

Por lo tanto a cada página web se le debe modificar su etiqueta “li” en el documento plantilla.php. En este caso se asignará la clase al “li” de inicio.



Figura 2.30. Vista de la plantilla realizada.

La plantilla ya ha quedado lista pero es necesario guardarla como plantilla para agregarla cada vez que se necesite. Para lo anterior se oprime el botón izquierdo del ratón sobre el menú “Tools” y se selecciona la opción “Templates”.

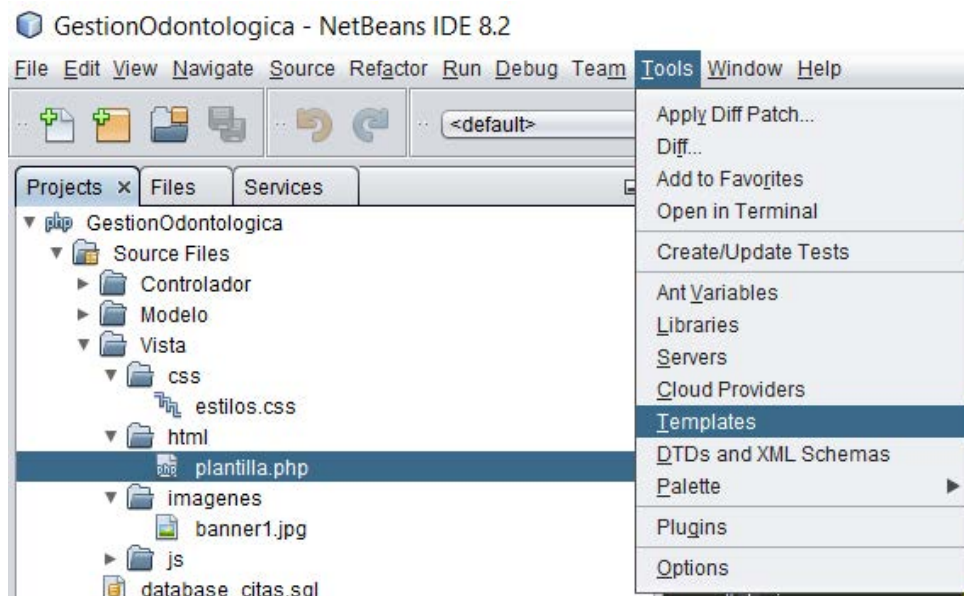


Figura 2.31. Creación de plantilla en Netbeans parte 1.

Inmediatamente se nos muestra el cuadro de diálogo “Template Manager” así:

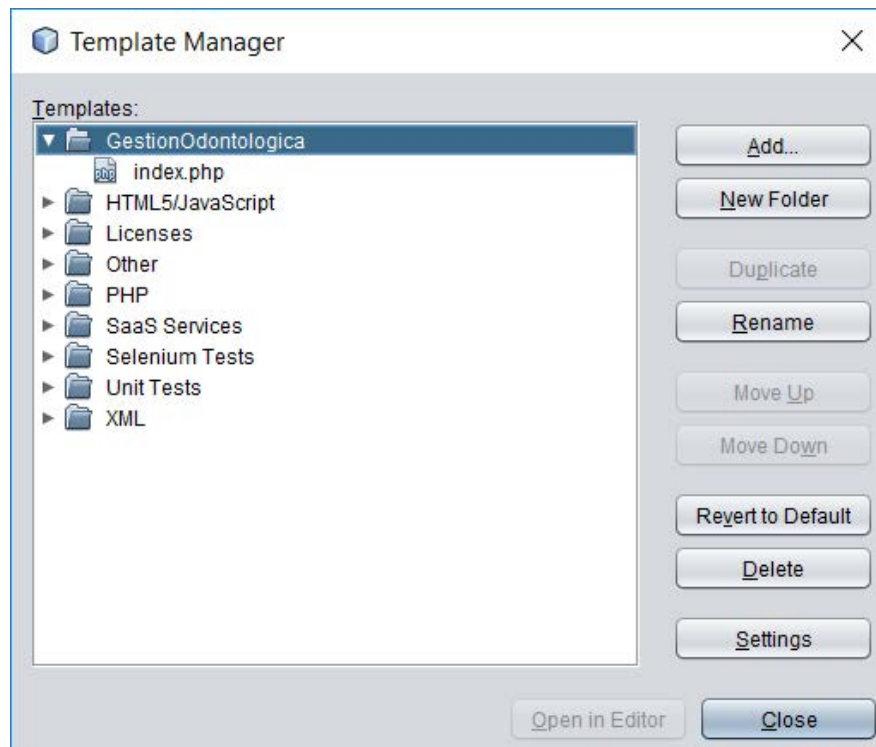


Figura 2.32. Creación de plantilla en Netbeans parte 2.

Para no modificar la estructura de NetBeans, se creará una nueva carpeta para almacenar la plantilla. Para ello se oprime el botón “New Folder” y se cambia el nombre por defecto de la carpeta a “GestionOdontologica” y se oprime el botón “OK”.

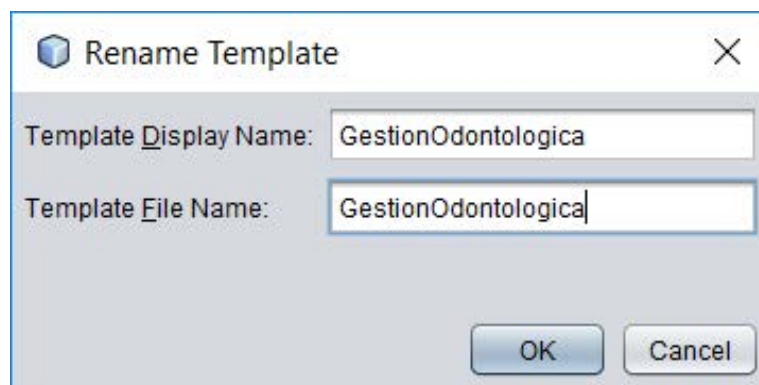


Figura 2.33. Creación de plantilla en Netbeans parte 3.

La carpeta ya ha sido creada. El siguiente paso es agregar la plantilla pero este proceso se realizará por otro medio.

Se oprime el botón “Close” del cuadro de diálogo “Template Manager”.

Se oprime el botón derecho del ratón sobre el archivo “plantilla.php” y en el menú contextual se selecciona la opción “Save As Template”:

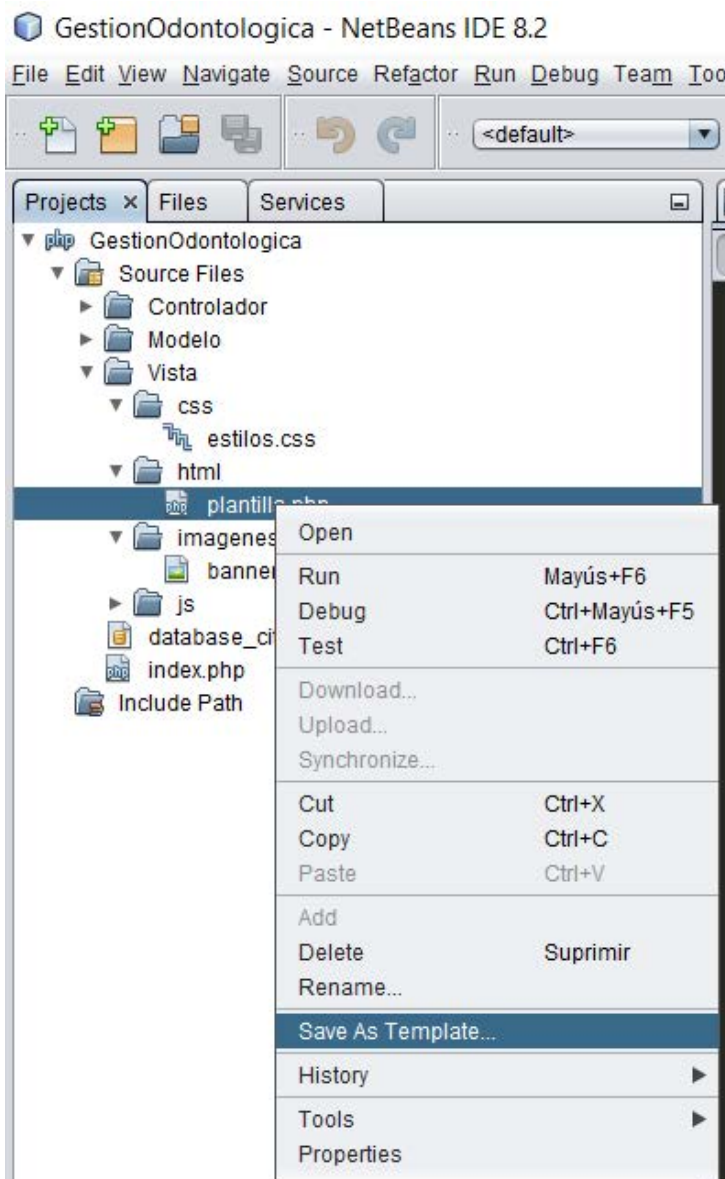


Figura 2.34. Creación de plantilla en Netbeans parte 4.

Aparece el cuadro de diálogo “Save As Template”. En el árbol de directorios se selecciona la carpeta que se acabó de crear y se oprime el botón "OK".

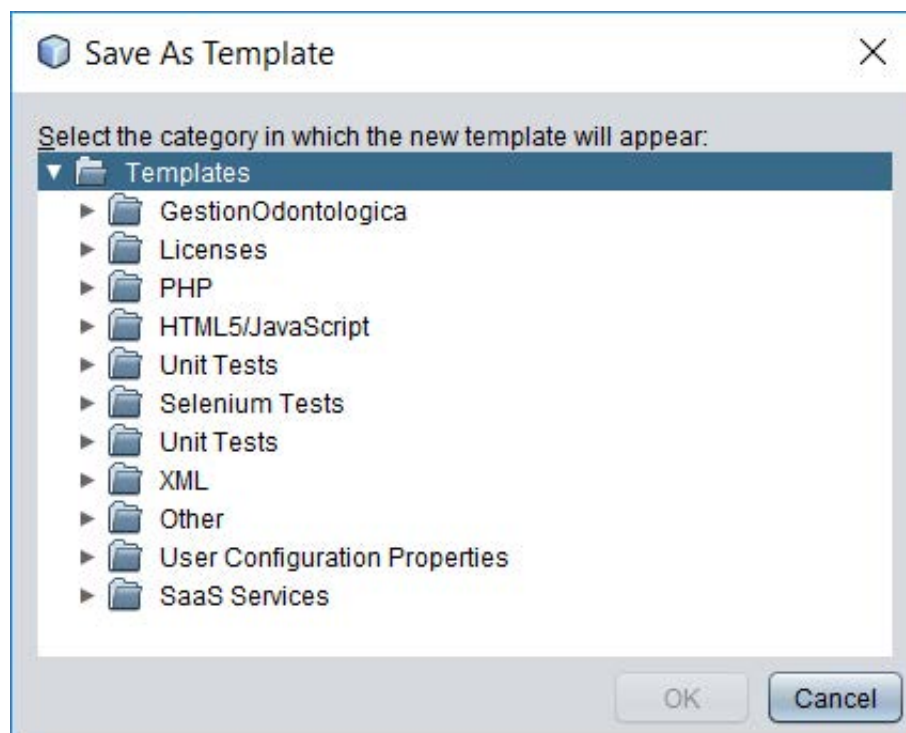


Figura 2.35. Creación de plantilla en Netbeans parte 5.

De esta forma finaliza la creación de la plantilla la cual se requerirá más adelante en el desarrollo del sitio web.

3. Desarrollo de la interfaz gráfica de usuario

3.1. Inicio y asignar cita

En la sesión anterior se creó la plantilla que servirá de base para la vista o interfaz gráfica de usuario del sitio web.

El resultado de dicho proceso se muestra a continuación.



Figura 3.1. Vista de la plantilla de la aplicación.

En esta sesión se construirán las páginas “Inicio” y “Asignar cita” del sistema de información.

3.1.1. Inicio

La página de inicio es solamente informativa y ofrece una descripción general de las acciones que se realizarán con el sistema de información.

Lo primero que se debe hacer es crear una nueva página web a partir de la plantilla. Los pasos son los siguientes:

Dar clic derecho sobre la carpeta “html”. En el menú contextual seleccionar la opción “New-Other...”

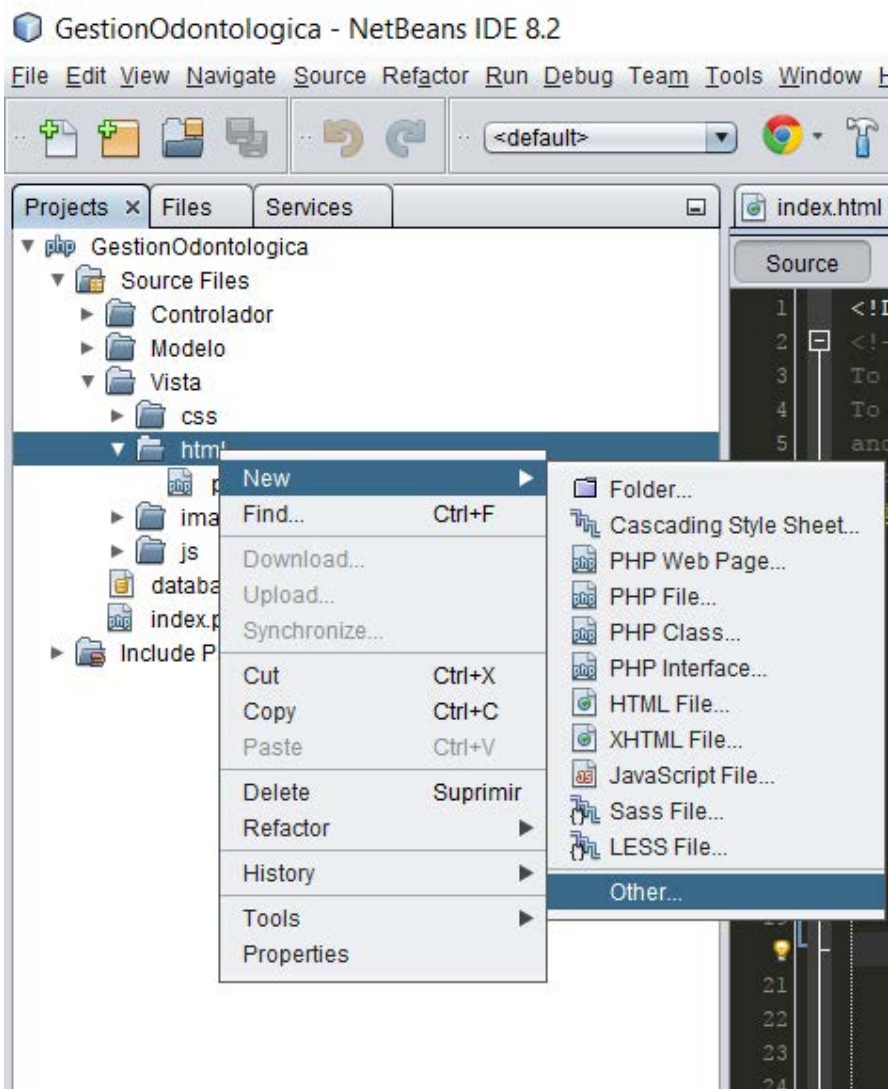


Figura 3.2. Creación de página a partir de plantilla en Netbeans parte 1.

Se abre el cuadro de diálogo “New File”. En la sección “Categories” se selecciona la carpeta “Gestion Odontologica” y en “File Types” se selecciona la opción “plantilla.php”.

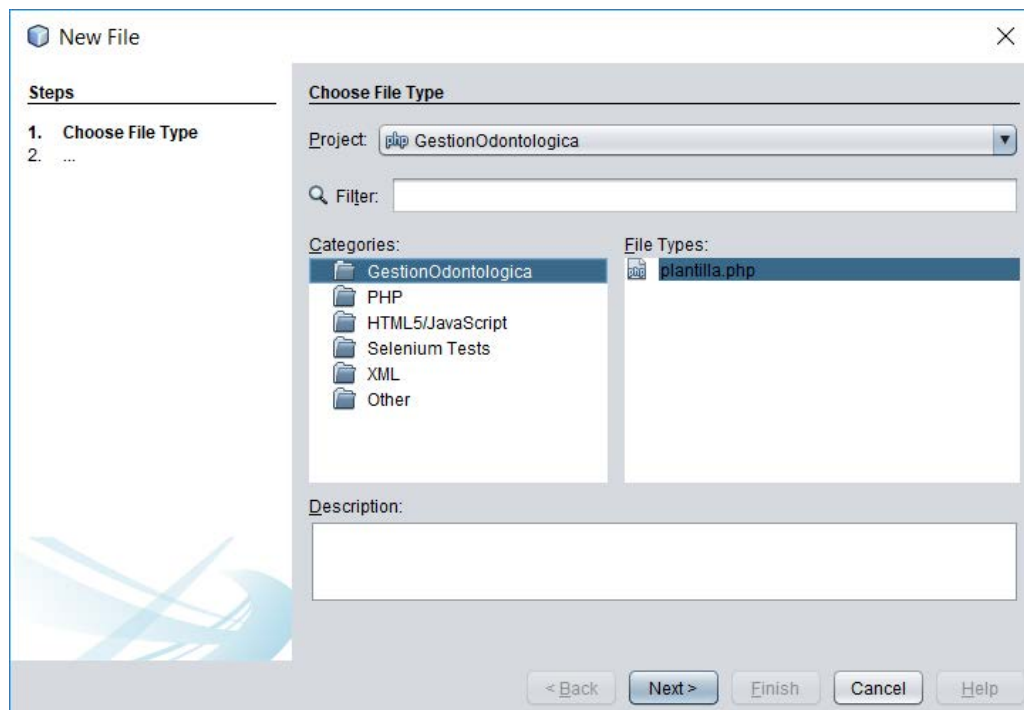


Figura 3.3. Creación de página a partir de plantilla en Netbeans parte 2.

Luego se oprime el botón “Next” y se abre el cuadro de diálogo “New plantilla.php”. En el campo “File Name” se escribe “inicio”.

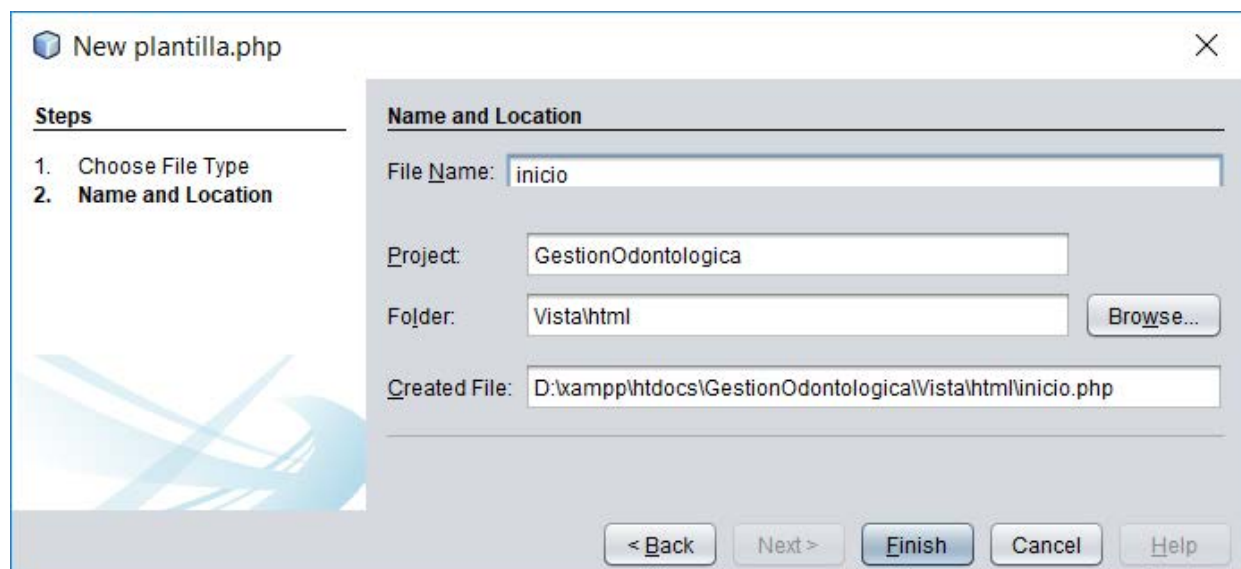


Figura 3.4. Creación de página a partir de plantilla en Netbeans parte 3.

Se revisa que los campos “Project”, “Folder” y “Created File”, tengan la información correcta. Luego se oprime el botón “Finish”.

Dentro del sitio se va a tener acceso a cuatro secciones. Esto se realiza desde el controlador el cual invoca el Arranque (index.php). Para conocer cuál de las páginas se debe mostrar se utilizará la variable “accion” la cual llega a través de la URL y se obtiene a través de la variable súper global “\$_GET”.

Primero se debe modificar el Arranque (index.php) para que de acuerdo a la opción del menú que se seleccione se muestre la página correcta.

Para hacer lo anterior se edita el archivo “index.php” y se aplican los cambios que se muestran a continuación:

```
<body>
    <?php
        if( isset($_GET["accion"])){
            if($_GET["accion"] == "asignar"){
                require_once 'Vista/html/asignar.php';
            }
            if($_GET["accion"] == "consultar"){
                require_once 'Vista/html/consultar.php';
            }
            if($_GET["accion"] == "cancelar"){
                require_once 'Vista/html/cancelar.php';
            }
        } else {
            require_once 'Vista/html/inicio.php';
        }
    ?>
</body>
```

Ahora se trabajará sobre el contenido de la página, es decir, en el archivo “Vista/html/inicio.php”. Se va a modificar el título de la página por “Información General”.

El contenido de esta página tendrá un par de párrafos con la información general del sistema de información.

Por tal motivo el código que se debe reemplazar en la página de inicio se muestra a continuación:

```
<div id="contenido">
    <h2>Información General</h2>
    <p>El Sistema de Gestión Odontológica permite administrar la información de los
    pacientes,
        tratamientos y citas a través de una interfaz web.</p>
    <p>El sistema cuenta con las siguientes secciones:
    <ul>
        <li>Asignar cita</li>
        <li>Consultar cita</li>
        <li>Cancelar cita</li>
    </ul>
    </p>
</div>
```

- El resultado se muestra en la imagen.

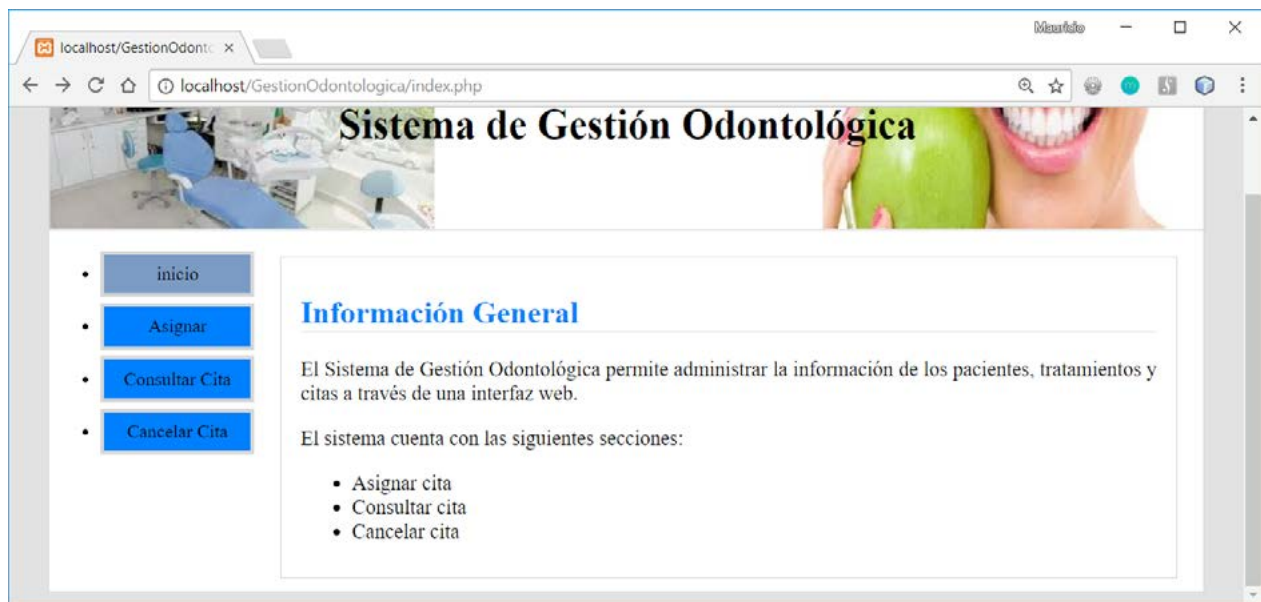


Figura 3.5. Vista preliminar página inicio.php

3.2. Asignar cita

En esta sección se realizará la página para la asignación de citas para lo anterior se genera una nueva página a partir de la plantilla como se hizo con la página de inicio. El nombre del archivo será “asignar.php”.

El título de la página se modifica y se asigna el texto: “Asignar Cita”.

En el menú se elimina el atributo “class=activa” de la opción “Inicio” y se le asigna a la opción “Asignar cita”.


```
<ul id="menu">
  <li><a href="index.php">inicio</a> </li>
  <li class="activa"><a href="index.php?accion=asignar">Asignar</a> </li>
  <li><a href="index.php?accion=consultar">Consultar Cita</a> </li>
  <li><a href="index.php?accion=cancelar">Cancelar Cita</a> </li>
</ul>
<div id="contenido">
  <h2>Asignar cita</h2>
  <p>Contenido de la página</p>
</div>
```

Para asignar una cita es necesario ingresar los datos en un formulario pero primero se revisa que campos posee la tabla “Citas”, y con base en eso se crea el formulario.

No	Campo	Descripción
1	CitNumero	Este campo es auto incrementable, por tal motivo no es necesario solicitarlo.
2	CitFecha	Este campo corresponde a la fecha en que se atenderá la cita, por tal motivo es necesario solicitarla.
3	CitHora	Este campo corresponde a la hora en que se atenderá la cita, por tal motivo es necesario solicitarla.
4	CitPaciente	En este campo se ingresa la cédula, es necesario asegurarnos que el usuario existe, de lo contrario es necesario crearlo.
5	CitMedico	En este campo se ingresa el documento del médico que atenderá la cita, por lo tanto es necesario solicitarlo.
6	CitConsultorio	En este campo se ingresará el consultorio en el cual se atenderá la cita, por lo tanto es necesario solicitarlo.
7	CitEstado	Este campo determina el estado actual de la cita, el cual puede tener tres valores que son: Solicitada, Atendida, Cancelada. Cuando se solicita una cita, automáticamente el estado será “Solicitada”, por tal motivo este campo no se solicita en este formulario.
8	CitObservaciones	Al momento de solicitar una cita, no se da ningún tipo de observación, por lo tanto automáticamente se debe asignar el valor Ninguna, por tal motivo este campo no se solicitaría en este formulario.

Con base en los campos a capturar se diseña un formulario cuyas propiedades sean id=“frmasignar”, action=“index.php?accion=guardarCita” y method=“post”.


```
<form id="frmasignar" action="index.php?accion=guardarCita" method="post">
</form>
```

Pasos del diseño:

El diseño del formulario se apoyará en una tabla por simplicidad y facilitar la explicación de los demás temas. Lo ideal es manejar un framework CSS como Bootstrap (<http://getbootstrap.com/>) que además realice un diseño adaptativo (Responsive Design) pero su explicación está fuera del alcance de este recurso.

1. El primer campo corresponde a los datos del usuario que en primera instancia sería un campo para la cédula y un botón para verificar su existencia en la base de datos. Como el dato del paciente se va a verificar el contenido se cambiará dinámicamente, por lo tanto se incluye una sección “<div>” cuyo id será “paciente”. En este se harán las modificaciones cuando se requiera. Los siguientes elementos corresponden a la cita, y contendrá los elementos mostrados en el siguiente orden:
 - El campo “Medico”. Será un “select”, que cargará los datos de la base de datos, pero por ahora no se realizara dicha carga, sino que se incluirán a mano los valores en el formulario.
 - El campo “Fecha” será de tipo “date” el cual tiene incorporado un control para seleccionar la misma.
 - El campo “Hora”. Esta se creará dentro de un “select”, teniendo en cuenta que el horario de los médicos inicia a las 8 a. m. y finaliza a las 2 p.m. y las citas se atienden cada 20 minutos. Este “select” se cargará automáticamente teniendo en cuenta la disponibilidad de acuerdo al médico y la fecha ya que no pueden existir dos citas a la misma hora con el mismo médico y en la misma fecha. Por ahora no se cargará automáticamente.
 - El campo “Consultorio”. Será un “select” que cargará los datos de la base de datos. Por ahora no se cargarán automáticamente.

De acuerdo al anterior diseño el código de la página “asignar.php” es el siguiente:

```
<form id="frmasignar" action="index.php?accion=guardarCita" method="post">
  <table>
    <tr>
      <td>Documento del paciente</td>
      <td><input type="text" name="asignarDocumento" id="asignarDocumento"></td>
    </tr>
    <tr>
      <td colspan="2"><input type="button" value="Consultar" name="asignarConsultar" id="asignarConsultar"></td>
```

```

        </tr>
        <tr><td colspan="2"><div id="paciente"></div></td>
    </tr>
    <tr>
        <td>Médico</td>
        <td>
            <select id="medico" name="medico">
                <option value="-1" selected="selected">---Seleccione el
Médico</option>
                <option value="12345">12345-Pepito Pérez</option>
                <option value="67890">67890-Pepita Mendieta</option>
            </select>
        </td>
    </tr>
    <tr>
        <td>Fecha</td>
        <td>
            <input type="date" id="fecha" name="fecha">
        </td>
    </tr>
    <tr>
        <td>Hora</td>
        <td>
            <select id="hora" name="hora">
                <option value="-1" selected="selected">---Seleccione la
hora ---</option>
                <option>08:00:00</option>
                <option>08:20:00</option>
                <option>08:40:00</option>
                <option>09:00:00</option>
            </select>
        </td>
    </tr>
    <tr>
        <td>Consultorio</td>
        <td>
            <select id="consultorio" name="consultorio">
                <option value="-1" selected="selected">---Seleccione el
Consultorio---</option>
                <option value="1">1 Consultas 1</option>
                <option value="2">2 Tratamientos 1</option>
            </select>
        </td>
    </tr>
    <tr>
        <td colspan="2">
            <input type="submit" name="asignarEnviar" value="Enviar"

```

```
id="asignarEnviar">
    </td>
</tr>
</table>
</form>
```

El resultado es el siguiente.



Figura 3.6. Vista preliminar módulo de asignación de citas.

De esta forma se han creado las dos primeras páginas del sitio web.

4. Desarrollo de la interfaz gráfica de usuario: consultar cita y cancelar cita

En la sesión anterior se crearon las páginas “Inicio” y “Asignar cita”.

En esta sesión se construirán las páginas “Consultar cita” y “Cancelar cita” del sistema de información.



4.1. Consultar cita

Para crear la página “Consultar Cita” se utiliza la plantilla “plantilla.php” del mismo modo que se hizo en las sesiones anteriores. El nombre del archivo será “consultar.php”. El título de la página se modifica y se le asigna el valor “Consultar Cita”.

En el menú se elimina el atributo “class=activa” de la opción “inicio” y se le asigna a la opción “Consultar cita”.

```
<ul id="menu">
  <li><a href="index.php">inicio</a> </li>
  <li><a href="index.php?accion=asignar">Asignar</a> </li>
  <li class="activa"><a href="index.php?accion=consultar">Consultar Cita</a> </li>
  <li><a href="index.php?accion=cancelar">Cancelar Cita</a> </li>
</ul>
```

Para consultar una cita es necesario ingresar el documento del paciente. Por lo tanto en este momento ese es el único campo que se requiere dentro del formulario. Con base en el resultado obtenido se mostrará el contenido dinámico que informará acerca de las citas del usuario.

El código de la página “consultar.php” es el siguiente.

```
<div id="contenido">
  <h2>Consultar Cita</h2>
  <form action="index.php?accion=consultarCita" method="post" id="frmconsultar">
    <table>
      <tr>
        <td>Documento del Paciente</td>
        <td><input type="text" name="consultarDocumento" id="consultarDocumento"></td>
      </tr>
      <tr>
        <td colspan="2"><input type="submit" name="consultarConsultar" value="Consultar" id="consultarConsultar"></td>
      </tr>
      <tr>
        <td colspan="2"><div id="paciente2"></div></td>
      </tr>
    </table>
  </form>
</div>
```

El resultado se muestra a continuación.



Figura 4.1. Vista preliminar módulo de consulta de citas.

4.2. Cancelar cita

Este ejercicio continúa con la página para cancelar una cita.

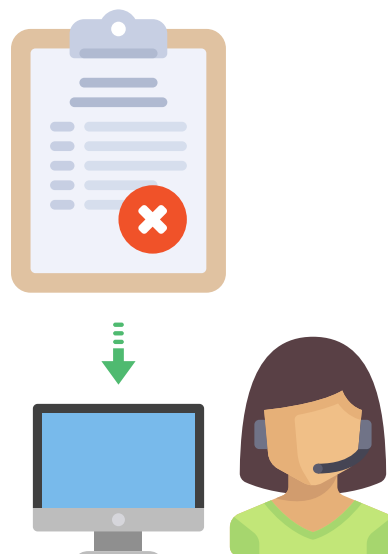
Se genera una nueva página a partir de la plantilla, como se hizo en la sesión anterior y el nombre será “cancelar.php”.

El título de la página se asigna el texto “Cancelar Cita”.

En el menú se elimina el atributo “class=activa” de la opción “inicio” y se le asigna a la opción “Cancelar cita”.

Para consultar una cita es necesario ingresar el documento del paciente. Por lo tanto en este momento ese es el único campo que se requiere dentro del formulario. Con base en el resultado obtenido se mostrará el contenido dinámico que informará acerca de las citas del usuario.

El código de la página “cancelar.php” es el siguiente.



```
<div id="contenido">
    <h2>Cancelar Cita</h2>
    <form action="index.php?accion=cancelarCita" method="post"
id="frmcancelar">
        <table>
            <tr>
                <td>Documento del Paciente </td>
                <td><input type="text" name="cancelarDocumento"
id="cancelarDocumento"></td>
            </tr>
            <tr>
                <td colspan="2"><input type="submit"
name="cancelarConsultar" value="Consultar" id="cancelarConsultar"></td>
            </tr>
            <tr>
                <td colspan="2"><div id="paciente3"></div></td>
            </tr>
        </table>
    </form>
</div>
```

El resultado se muestra a continuación.



Figura 4.2. Vista preliminar módulo de cancelación de citas.

De esta forma se finaliza con la creación de la interfaz gráfica de usuario.

5. Codificación de las clases del modelo

En esta sesión se construirán todas las clases que corresponden al modelo como se muestra en el diagrama de clases.

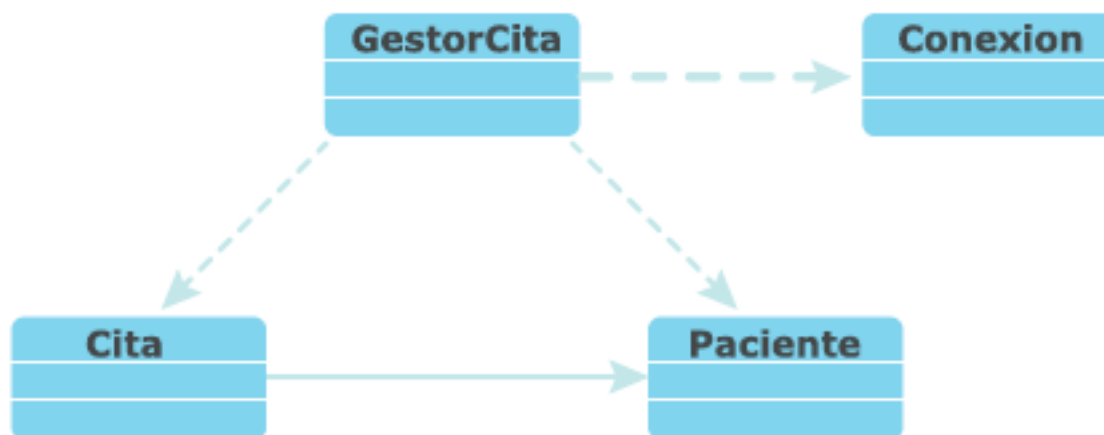


Figura 5.1. Diagrama de clases del caso de uso

5.1. Clase "Paciente"

Se creará la clase "Paciente" de acuerdo al siguiente diagrama de clases.

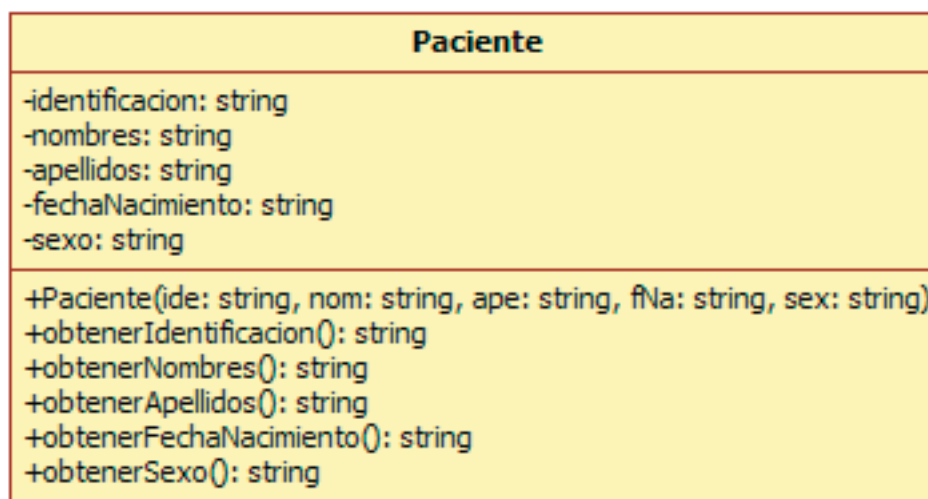


Figura 5.2. Estructura de la clase Paciente

Para ello se da clic derecho sobre la carpeta "Modelo". Luego el menú contextual se selecciona la opción "PHP Class..." así:

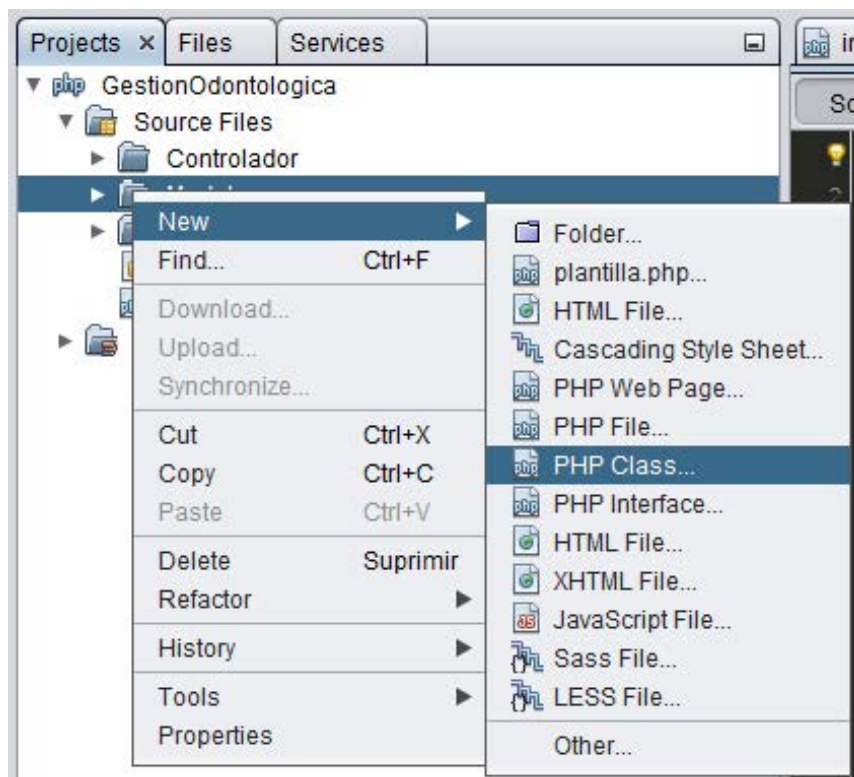


Figura 5.3. Creación de una clase en PHP usando Netbeans parte 1.

Se muestra el cuadro de diálogo: “New PHP Class”. En el campo “File Name” se ingresa “Paciente”. Se verifica que el proyecto sea “GestiónOdontológica” y la carpeta sea “Modelo”. Para finalizar se oprime el botón “Finish”.

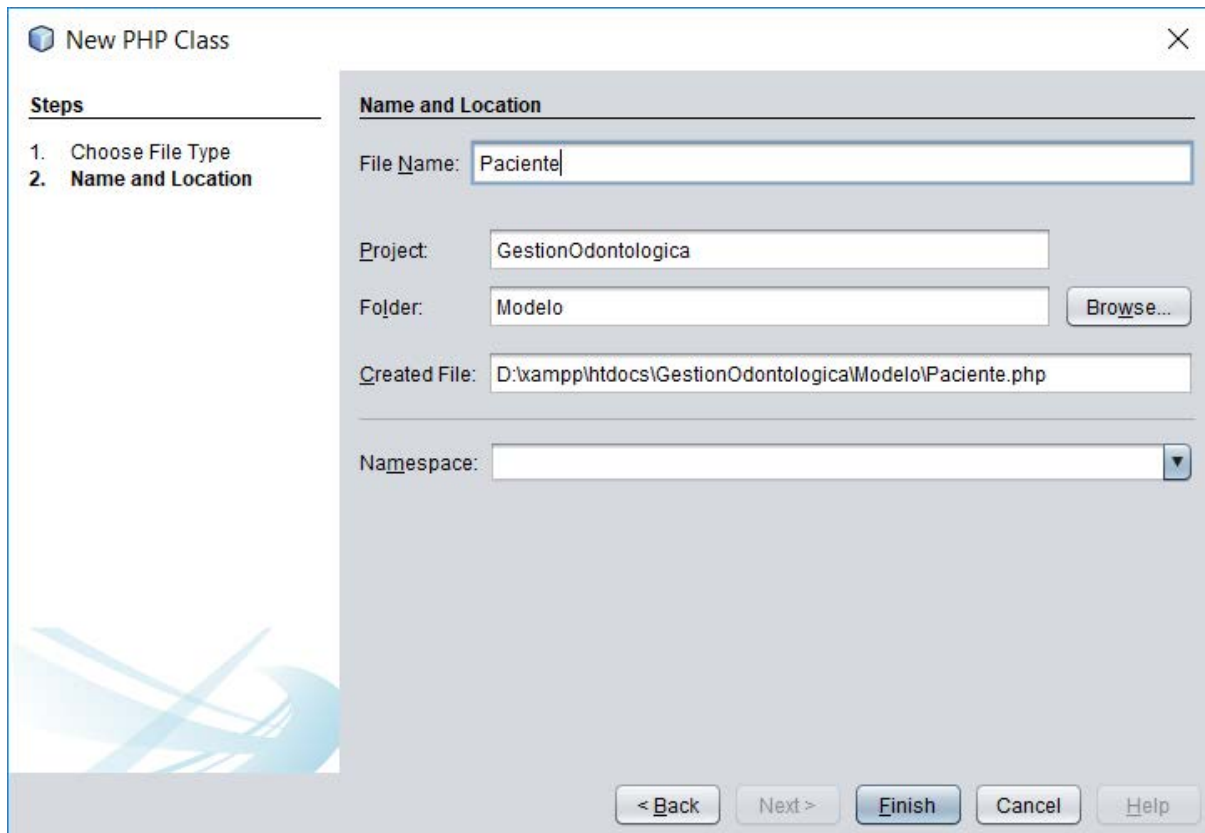


Figura 5.4. Creación de una clase en PHP usando Netbeans parte 2.

El código de la clase se muestra a continuación:

```
<?php
class Paciente {
    private $identificacion;
    private $nombres;
    private $apellidos;
    private $fechaNacimiento;
    private $sexo;

    public function __construct($ide,$nom,$ape,$fNa,$sex) {
        $this->identificacion=$ide;
        $this->nombres=$nom;
        $this->apellidos=$ape;
        $this->fechaNacimiento=$fNa;
        $this->sexo=$sex;
    }
    public function obtenerIdentificacion(){
        return $this->identificacion;
    }
    public function obtenerNombres(){
```

```

        return $this->nombres;
    }
    public function obtenerApellidos(){
        return $this->apellidos;
    }
    public function obtenerFechaNacimiento(){
        return $this->fechaNacimiento;
    }
    public function obtenerSexo(){
        return $this->sexo;
    }
}

```

5.2. Clase "Cita"

Se creará la clase “Cita” de acuerdo al siguiente diagrama de clases:

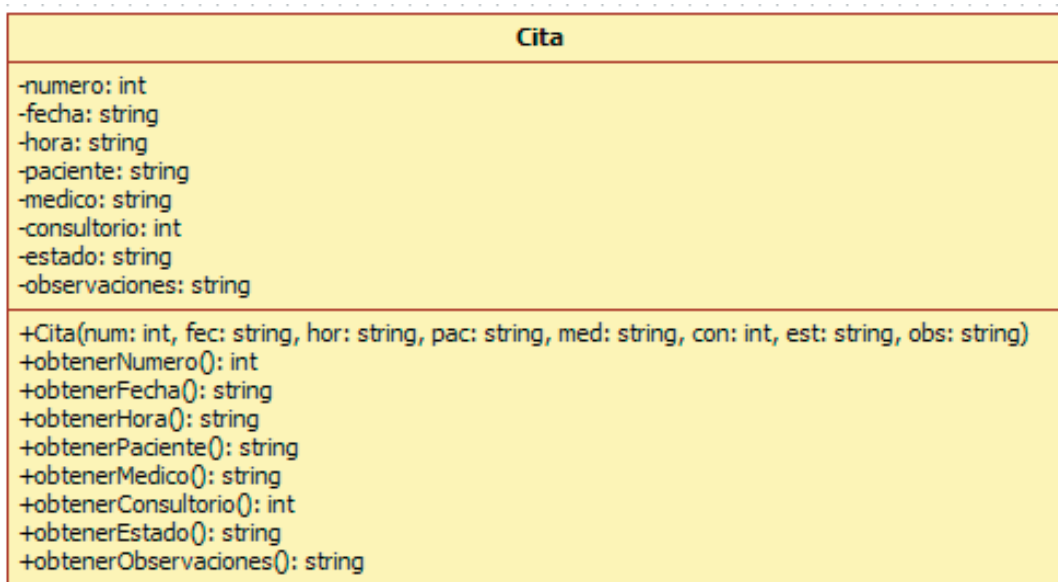


Figura 5.5. Estructura de la clase Cita.

Se siguen las mismas instrucciones que en el paso anterior para crear el archivo, solo que en este caso el archivo se llamará “Cita.php”.

El código de “Cita.php” se muestra a continuación:

```

<?php
class Cita {
    private $numero;
    private $fecha;
    private $hora;

```

```

private $paciente;
private $medico;
private $consultorio;
private $estado;
private $observaciones;

public function __construct($num,$fec,$hor,$pac,$med,$con,$est,$obs) {
    $this->numero=$num;
    $this->fecha=$fec;
    $this->hora=$hor;
    $this->paciente=$pac;
    $this->medico=$med;
    $this->consultorio=$con;
    $this->estado=$est;
    $this->observaciones=$obs;
}
public function obtenerNumero(){
    return $this->numero;
}
public function obtenerFecha(){
    return $this->fecha;
}
public function obtenerHora(){
    return $this->hora;
}
public function obtenerPaciente(){
    return $this->paciente;
}
public function obtenerMedico(){
    return $this->medico;
}
public function obtenerConsultorio(){
    return $this->consultorio;
}
public function obtenerEstado(){
    return $this->estado;
}
public function obtenerObservaciones(){
    return $this->observaciones;
}
}

```

5.3. La clase "Conexion"

Se creará la clase "Conexion" de acuerdo al siguiente diagrama de clases:

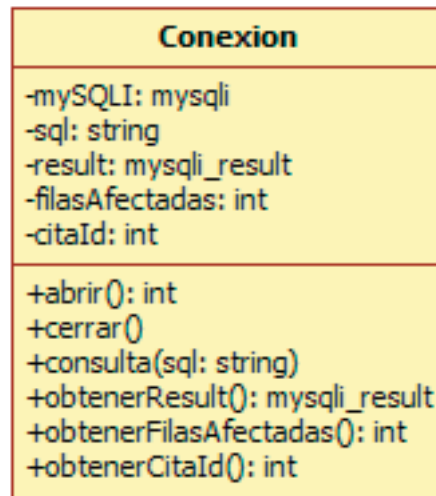


Figura 5.6. Estructura de la clase Conexion.

Se siguen las mismas instrucciones que en el paso anterior para crear el archivo, solamente que en este caso se llamará “Conexion.php”. El código de “Conexion.php” se muestra a continuación:

```
<?php
class Conexion {
    private $mysqli;
    private $sql;
    private $result;
    private $filasAfectadas;
    private $citaId;

    public function abrir(){
        $this->mysqli=new mysqli("localhost","root","","citas");
        if(mysqli_connect_error()){
            return 0;
        } else {
            return 1;
        }
    }
    public function cerrar(){
        $this->mysqli->close();
    }
    public function consulta($sql){
        $this->sql = $sql;
        $this->result = $this->mysqli->query($this->sql);
        $this->filasAfectadas = $this->mysqli->affected_rows;
        $this->citaId = $this->mysqli->insert_id;
    }
    public function obtenerResult(){
```

```

        return $this->result;
    }
    public function obtenerFilasAfectadas(){
        return $this->filasAfectadas;
    }
    public function obtenerCitaId(){
        return $this->citaId;
    }
}

```

5.4. La clase "GestorCita"

La clase "GestorCita" es la encargada de gestionar todos los procesos relacionados con los accesos a datos que realiza el sistema de información. Por tal motivo debe tener referencia a las tres clases creadas anteriormente para poderlas manipular.

- El diagrama de la clase "GestorCita" se muestra a continuación:

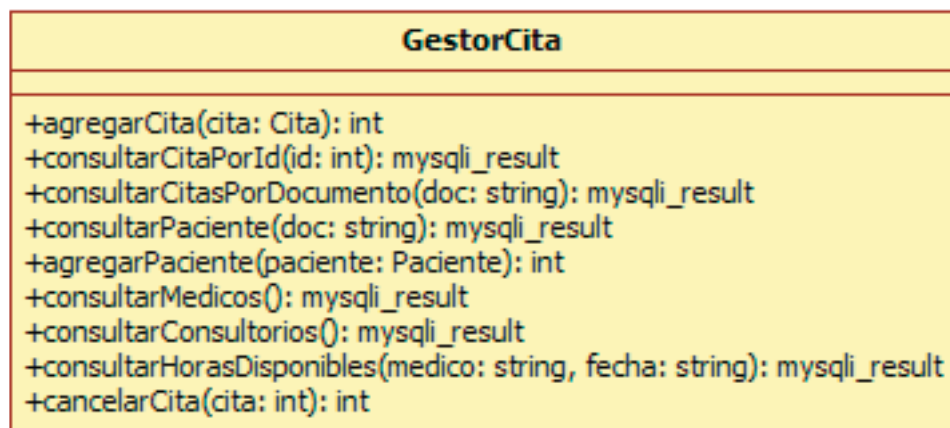


Figura 5.7. Estructura de la clase GestorCita.

Se siguen las mismas instrucciones que en el paso anterior para crear el archivo, solamente que en este caso se llamará "GestorCita.php". A continuación se muestra el código para la creación de la clase "GestorCita" y del método "agregarCita":

```

<?php
class GestorCita {
    public function agregarCita(Cita $cita){
        $conexion = new Conexion();
        $conexion->abrir();
        $fecha = $cita->obtenerFecha();
        $hora = $cita->obtenerHora();
        $paciente = $cita->obtenerPaciente();
        $medico = $cita->obtenerMedico();
        $consultorio = $cita->obtenerConsultorio();
    }
}

```

```

        $estado          = $cita->obtenerEstado();
        $observaciones    = $cita->obtenerObservaciones();
        $sql              = "INSERT INTO Citas VALUES ( null, '$fecha', '$hora',
'$paciente', '$medico', '$consultorio', '$estado', '$observaciones')";
        $conexion->consulta($sql);
        $citaId = $conexion->obtenerCitaId();
        $conexion->cerrar();
        return $citaId ;
    }
}

```

5.5. Codificación del método "consultarCitaPorId()"

Este método permite obtener todos los datos de una cita específica para mostrarlos luego como reporte.

Como el objetivo es mostrar al paciente todos los datos de la cita, incluyendo datos del consultorio, paciente y médico, la consulta que se realizará en este caso involucra varias tablas.

El código de este método se muestra a continuación.

```

public function consultarCitaPorId($id){
    $conexion = new Conexion();
    $conexion->abrir();
    $sql      = "SELECT pacientes.* , pedicos.*, consultorios.*, citas.*"
        . "FROM Pacientes as pacientes, Medicos as medicos, Consultorios
as consultorios ,citas "
        . "WHERE citas.CitPaciente = pacientes.PacIdentificacion "
        . " AND citas.CitMedico    = medicos.MedIdentificacion "
        . " AND citas.CitNumero    = $id";
    $conexion->consulta($sql);
    $result = $conexion->obtenerResult();
    $conexion->cerrar();
    return $result ;
}

```

5.6. Codificación del método "consultarCitasPorDocumento()"

Este método permite obtener todos los datos de las citas correspondientes a un paciente específico de acuerdo a su número de documento. El código de este método se muestra a continuación.

```
public function consultarCitasPorDocumento($doc){
    $conexion = new Conexion();
    $conexion->abrir();
    $sql      = "SELECT * FROM citas "
                . "WHERE CitPaciente    = '$doc' "
                . "AND CitEstado        = 'Solicitada' ";
    $conexion->consulta($sql);
    $result = $conexion->obtenerResult();
    $conexion->cerrar();
    return $result ;
}
```

5.7. Codificación del método "consultarPaciente()"

Este método permite obtener todos los datos de un paciente específico de acuerdo a su número de documento. El código de este método se muestra a continuación.

```
public function consultarPaciente($doc){
    $conexion = new Conexion();
    $conexion->abrir();
    $sql      = "SELECT * FROM Pacientes WHERE PacIdentificacion = '$doc' ";
    $conexion->consulta($sql);
    $result = $conexion->obtenerResult();
    $conexion->cerrar();
    return $result ;
}
```

5.8. Codificación del método "agregarPaciente()"

Este método permite ingresar los datos de un nuevo paciente a la base de datos. El código de este método se muestra a continuación.

```
public function agregarPaciente(Paciente $paciente){
    $conexion = new Conexion();
    $conexion->abrir();
    $identificacion = $paciente->obtenerIdentificacion();
    $nombres        = $paciente->obtenerNombres();
    $apellidos       = $paciente->obtenerApellidos();
    $fechaNacimiento = $paciente->obtenerFechaNacimiento();
    $sexo            = $paciente->obtenerSexo();
    $sql             = "INSERT INTO Pacientes VALUES (
'$identificacion','$nombres','$apellidos',"
                . "'$fechaNacimiento','$sexo')";
    $conexion->consulta($sql);
    $filasAfectadas = $conexion->obtenerFilasAfectadas();
}
```



```
$conexion->cerrar();  
return $filasAfectadas;  
}
```

5.9. Codificación del método "consultarMedicos()"

Este método permite consultar todos los médicos que están registrados en la base de datos. El código de este método se muestra a continuación.

```
public function consultarMedicos(){  
    $conexion = new Conexion();  
    $conexion->abrir();  
    $sql      = "SELECT * FROM Medicos ";  
    $conexion->consulta($sql);  
    $result = $conexion->obtenerResult();  
    $conexion->cerrar();  
    return $result ;  
}
```

Dentro de la clase "GestorCita", quedan pendientes algunos métodos pero para un mejor entendimiento se codificarán a medida que se vayan necesitando.

6. Codificación de las clases del controlador

En la sesión anterior se crearon todas las **clases del modelo** descritas en el diagrama de clases. En esta sesión se codificará el controlador para darle funcionalidad al sistema de información.

En este sistema de información el controlador estará definido por dos programas. El primero de ellos es el Arranque, correspondiente al archivo "index.php" la cual define la navegabilidad dentro del sistema de información. Este a su vez crea un objeto de la clase "Controlador" la cual estará definida en el archivo "controlador.php" ubicada dentro de la carpeta "Controlador" y ejecuta el método correspondiente.

6.1. Arranque (index.php) y la clase "Controlador"

Se va a crear la clase "Controlador" de acuerdo al siguiente diagrama de clases.

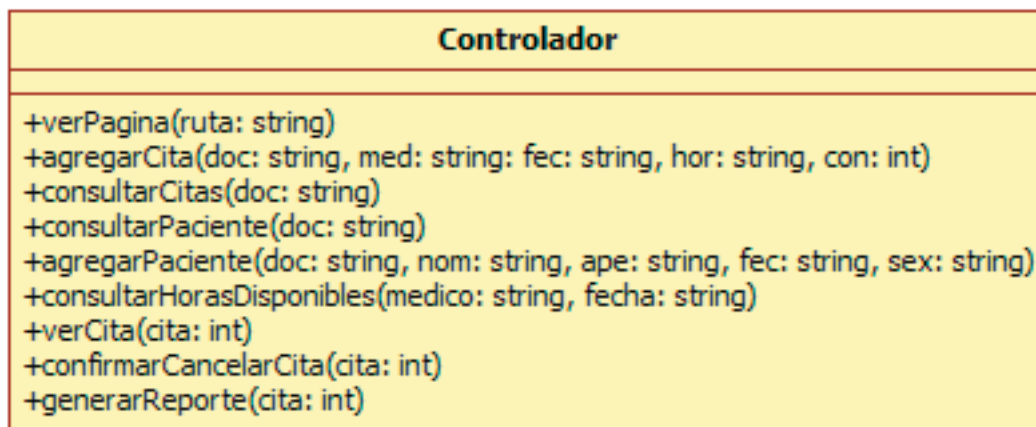


Figura 6.1. Estructura de la clase Controlador

Para ello se da clic derecho sobre la carpeta Controlador. En el menú contextual se selecciona la opción “PHP Class...” de la misma forma que se hizo en el ejercicio anterior. En el campo “File name” se debe ingresar: “Controlador”.

El primer método que se creará es “verPagina(ruta: string)”. Su código se muestra a continuación.

```

<?php
class Controlador {
    public function verPagina($ruta){
        require_once $ruta;
    }
}

```

Este método sirve para mostrar las páginas correspondientes a las vistas. Como la clase Controlador se instancia a través del Arranque (index.php) es necesario hacer las modificaciones a dicho archivo.

Se va a cambiar todo el contenido de “index.php” y se modificará de la siguiente forma:

1. Se debe hacer referencia a todas las clases creadas tanto en el Modelo como en el Controlador de la siguiente forma.

```

require_once 'Controlador/Controlador.php';
require_once 'Modelo/GestorCita.php';
require_once 'Modelo/Cita.php';
require_once 'Modelo/Paciente.php';
require_once 'Modelo/Conexion.php';

```

2. Se crea una una instancia de la clase Controlador.

```
$controlador = new Controlador();
```

3. El código selector queda así:

```
if( isset($_GET["accion"])){
    if($_GET["accion"] == "asignar"){
        $controlador->verPagina('Vista/html/asignar.php');
    }
    if($_GET["accion"] == "consultar"){
        $controlador->verPagina('Vista/html/consultar.php');
    }
    if($_GET["accion"] == "cancelar"){
        $controlador->verPagina('Vista/html/cancelar.php');
    }
} else {
    $controlador->verPagina('Vista/html/inicio.php');
}
```

Se puede probar el sistema de información para verificar que se tiene navegabilidad como al principio pero esta vez a través de la clase “Controlador”.

6.2. Codificación del método "AgregarCita()"

Ahora se creará el método “AgregarCita()” dentro de la clase “Controlador”. Este método recibe el documento del paciente, el documento del médico, la fecha y hora de la cita y el consultorio.

Con estos datos se construye un objeto de la clase “Cita”. Hay que tener en cuenta que el número de la cita lo genera Mysql. Por lo tanto allí se debe enviar un valor nulo o “null”. Para el estado de la cita se envía el texto: “Solicitada”. En “observaciones” se envía el texto “Ninguna”.

El código del método es el siguiente.

```
public function agregarCita($doc,$med,$fec,$hor,$con){
    $cita = new Cita(null, $fec, $hor, $doc, $med, $con, "Solicitada",
"Ninguna");
    $gestorCita = new GestorCita();
    $gestorCita->agregarCita($cita);
}
```

Con el método “agregarCita()” en el controlador es necesario modificar el Arranque (index.php) teniendo en cuenta que la acción del formulario que envía los datos de la cita posee la siguiente referencia.

Por tanto se incluirá en el Arranque (index.php) el siguiente código.

```
elseif($_GET["accion"] == "guardarCita"){
    $controlador->agregarCita($_POST["asignarDocumento"],
        $_POST["medico"], $_POST["fecha"], $_POST["hora"],
        $_POST["consultorio"]);
}
```

Para verificar que los datos de la cita se insertan en la base de datos se llenará el formulario con datos.

El documento del paciente debe existir en la tabla “Pacientes” de lo contrario se genera un error.

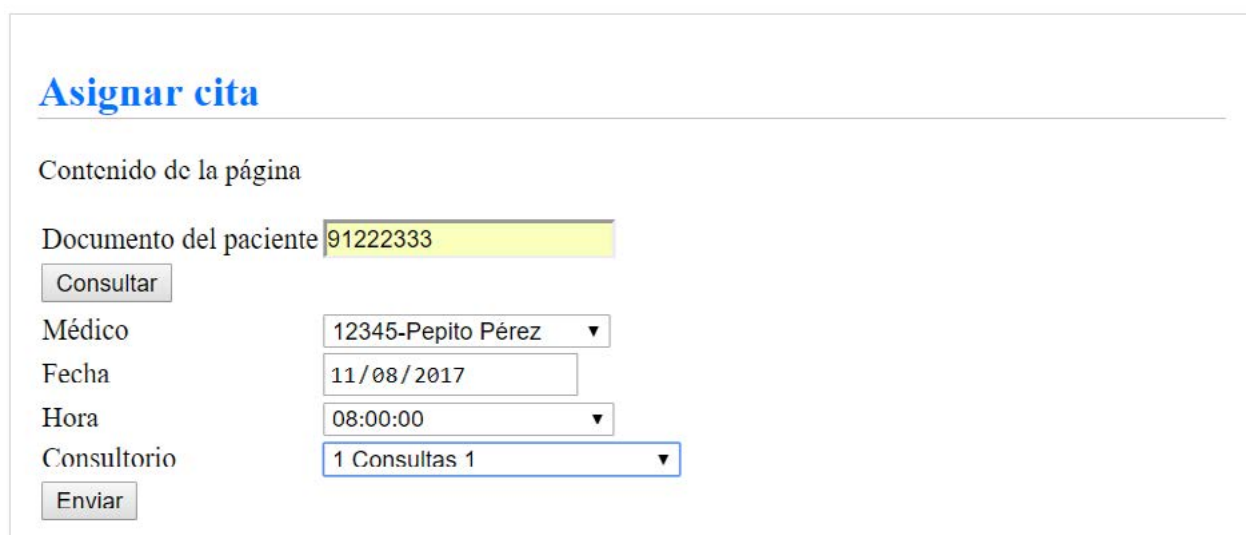


Figura 6.2. Formulario de captura para la asignación de citas.

Al enviar los datos, y si todo ha salido correctamente, nos aparecerá una página en blanco, esta página no da ninguna información y en el siguiente punto se hará la modificación necesaria para que se muestren los datos de la cita. Pero se puede verificar en la consola de Mysql así:

```
MariaDB [citas]> select * from citas ;
```

CitNumero	CitFecha	CitHora	CitPaciente	CitMedico	CitConsultorio
1	2017-08-11	08:00:00	91222333	12345	1
	Ninguna				

```
+-----+-----+-----+-----+-----+
--+-----+-----+
1 row in set (0.00 sec)

MariaDB [citas]>
```

El sistema de información debe generar un reporte en caso de que la cita se haya creado con éxito. Este reporte se debe mostrar en una nueva página. Más adelante se creará la opción de imprimir para entregarle el soporte físico al paciente.

Su estructura podría ser similar a la siguiente.

Datos del Paciente	
Documento	102030
Nombre	Jose Perez
Datos del Medico	
Documento	12345
Nombre	Camilo Robledo
Datos de la Cita	
Numero	1
Fecha	2012-09-09
Hora	08:20:00
Numero Consultorio	1
Nombre Consultorio	Consultas1
Estado	Solicitada
Observaciones	Ninguna

Figura 6.3. Diseño del reporte de la cita.

Lo primero que se debe hacer es crear una nueva página web con la estructura de la plantilla. Para ello se siguen los pasos para crear dentro de la carpeta “html” un nuevo archivo basado en la plantilla.

Este nuevo archivo se llamará “confirmarCita.php”. El código de este nuevo archivo es el siguiente:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Sistema de Gestión Odontológica</title>
    <link rel="stylesheet" type="text/css" href="Vista/css/estilos.css">
```

```

</head>
<body>
    <div id="contenedor">
        <div id="encabezado">
            <h1>Sistema de Gestión Odontológica</h1>
        </div>
        <ul id="menu">
            <li class="activa"><a href="index.php">inicio</a> </li>
            <li><a href="index.php?accion=asignar">Asignar</a> </li>
            <li><a href="index.php?accion=consultar">Consultar Cita</a> </li>
            <li><a href="index.php?accion=cancelar">Cancelar Cita</a> </li>
        </ul>
        <div id="contenido">
            <?php $fila = $result->fetch_object();?>
            <h2>Información Cita</h2>
            <table>
                <tr>
                    <th colspan="2">Datos del Paciente</th>
                </tr>
                <tr>
                    <td>Documento</td>
                    <td><?php echo $fila->PacIdentificacion;?></td>
                </tr>
                <tr>
                    <td>Nombre</td>
                    <td><?php echo $fila->PacNombres . " " . $fila->PacApellidos;?></td>
                </tr>
                <tr>
                    <th colspan="2">Datos del Médico</th>
                </tr>
                <tr>
                    <td>Documento</td>
                    <td><?php echo $fila->MedIdentificacion;?></td>
                </tr>
                <tr>
                    <td>Nombre</td>
                    <td><?php echo $fila->MedNombres . " " . $fila->MedApellidos;?></td>
                </tr>
                <tr>
                    <th colspan="2">Datos de la Cita</th>
                </tr>
                <tr>
                    <td>Número</td>
                    <td><?php echo $fila->CitNumero;?></td>
            </table>
        </div>
    </div>
</body>
</html>

```

```

        </tr>
        <tr>
            <td>Fecha</td>
            <td><?php echo $fila->CitFecha;?></td>
        </tr>
        <tr>
            <td>Hora</td>
            <td><?php echo $fila->CitHora;?></td>
        </tr>
        <tr>
            <td>Número de Consultorio</td>
            <td><?php echo $fila->ConNombre;?></td>
        </tr>
        <tr>
            <td>Estado</td>
            <td><?php echo $fila->CitEstado;?></td>
        </tr>
        <tr>
            <td>Observaciones</td>
            <td><?php echo $fila->CitObservaciones;?></td>
        </tr>
    </table>
</div>
</div>
</body>
</html>

```

Se puede observar que en la línea resaltada aparece una variable llamada “\$result” la cual no se tiene como referencia pero tendrá alcance desde el controlador.

Por esto es necesario modificar el método “agregarCita()” de la clase “Controlador” para que muestre la plantilla que se creó anteriormente mostrando los datos de la cita ingresada.

El código del método “agregarCita()” se muestra a continuación:

```

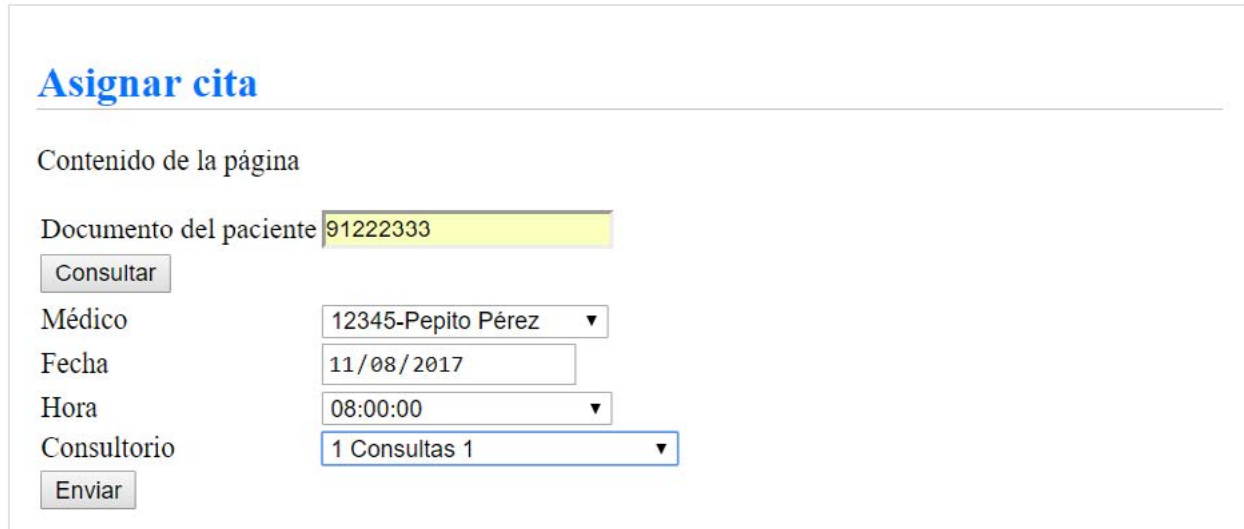
<?php
class Controlador {
    public function verPagina($ruta){
        require_once $ruta;
    }
    public function agregarCita($doc,$med,$fec,$hor,$con){
        $cita = new Cita(null, $fec, $hor, $doc, $med, $con, "Solicitada",
"Ninguna");
        $gestorCita = new GestorCita();
    }
}

```



```
$id      = $gestorCita->agregarCita($cita);
$result  = $gestorCita->consultarCitaPorId($id);
require_once 'Vista/html/confirmarCita.php';
}
}
```

Para verificar el código anterior se hace una prueba así:



Asignar cita

Contenido de la página

Documento del paciente 91222333

Médico 12345-Pepito Pérez ▼

Fecha 11/08/2017

Hora 08:00:00 ▼

Consultorio 1 Consultas 1 ▼

Figura 6.4. Datos de prueba para la asignación de una cita.

El resultado es el siguiente:

Información Cita

Datos del Paciente

Documento 91222333
Nombre Carlos Jesus Rodriguez Cala

Datos del Médico

Documento 12345
Nombre Pepito Perez

Datos de la Cita

Número 8
Fecha 2017-08-11
Hora 08:00:00
Número de Consultorio Consultas
Estado
Observaciones Ninguna

Figura 6.5. Resultado de prueba para la asignación de una cita.

6.3. Codificación del método "ConsultarCitas()"

Cuando el usuario hace la consulta de la cita se debe mostrar la información de las citas asignadas al paciente formateadas en una tabla. Por tal motivo antes de proceder a crear el método dentro del controlador se creará el documento PHP que mostrará el resultado.

En este caso no se crea el archivo a partir de la plantilla sino que se crea dentro de la carpeta "html" un nuevo archivo de tipo "PHP Web Page".

El nombre del archivo será "consultarCitas.php" y su código se muestra a continuación:

La razón principal por la que esta página no fue creada como plantilla obedece a que más adelante este contenido se incluirá dinámicamente dentro de la misma página "consultar.php"

Ahora se creará el método "consultarCitas()" dentro de la clase "Controlador". Este método recibe el documento del paciente, con el cual se llamará al método "consultarCitasPorDocumento()" perteneciente a la clase "GestorCitas", y se mostrará la página "consultarCitas.php."

El código del método es el siguiente:

```
public function consultarCitas($doc){  
    $gestorCita    = new GestorCita();  
    $result        = $gestorCita->consultarCitasPorDocumento($doc);  
    require_once 'Vista/html/consultarCitas.php';  
}
```

Para que funcione es necesario realizar el cambio en el archivo "index.php" de la siguiente forma:

```
if( isset($_GET["accion"])){  
    if($_GET["accion"] == "asignar"){  
        $controlador->verPagina('Vista/html/asignar.php');  
    }  
    elseif($_GET["accion"] == "consultar"){  
        $controlador->verPagina('Vista/html/consultar.php');  
    }  
    elseif($_GET["accion"] == "cancelar"){  
        $controlador->verPagina('Vista/html/cancelar.php');  
    }  
    elseif($_GET["accion"] == "guardarCita"){  
        $controlador->agregarCita(  

```


```

        $_POST["asignarDocumento"],
        $_POST["medico"],
        $_POST["fecha"],
        $_POST["hora"],
        $_POST["consultorio"]);
    }
    elseif($_GET["accion"] == "consultarCita"){
        $controlador->consultarCitas($_POST["consultarDocumento"]);
    }

} else {
    $controlador->verPagina('Vista/html/inicio.php');
}

```


Al hacer la prueba tanto con un paciente que posea citas aparece lo siguiente:



Número	Fecha	Hora
10	2017-08-13 08:40:00	Ver
11	2017-08-24 08:20:00	Ver

Figura 6.6. Resultado de la opción de consulta de citas cuando existen.

Uno que no tenga citas aparece lo siguiente:



El paciente no tiene citas asignadas

Figura 6.7. Resultado de la opción de consulta de citas cuando no existen.

En este momento no se tiene formato. Este contenido se insertará dinámicamente en la misma página “consultar” más adelante en este recurso.

6.4. Codificación del método "CancelarCitas()"

Igual que en la consulta de citas se debe mostrar la información de las citas canceladas al paciente formateadas en una tabla. Por tal motivo se creará un nuevo documento “php” que mostrará el resultado.

Nuevamente se crea el archivo dentro de la carpeta “html” de tipo “PHP Web Page”. El nombre del archivo será “cancelarCitas.php” y su código se muestra a continuación:

```
<!DOCTYPE html>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<html>
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>
    <?php
      if($result->num_rows > 0){
    ?>
    <table>
      <tr>
        <th>Número</th><th>Fecha</th><th>Hora</th>
      </tr>
      <?php
        while($fila=$result->fetch_object()){
          ?>
          <tr>
            <td><?php echo $fila->CitNumero;?></td>
            <td><?php echo $fila->CitFecha;?></td>
            <td><?php echo $fila->CitHora;?></td>
            <td><a href="#" onclick="confirmarCancelar(<?php echo $fila->CitNumero; ?>)">Cancrelar</a></td>
          </tr>
          <?php
            }
          ?>
        </table>
        <?php
          }
          else {
            ?>
            <p>El paciente no tiene citas asignadas</p>
            <?php
              }
            ?>
          </body>
        </html>
```

Como se puede observar esta página por ahora tiene prácticamente el mismo código que “consultarCitas.php”, solo tiene de más una columna con la palabra “Cancelar”, se puede pensar en reutilizar el código anterior, eso sería posible, pero es necesario diferenciar, posiblemente a través de una variable.

Ahora se creará el método “cancelarCitas()” dentro de la clase “Controlador”.

Este método recibe el documento del paciente, con el cual se llamará al método “consultarCitasPorDocumento()” perteneciente a la clase “GestorCitas”, y se mostrará la página “cancelarCitas.php”.

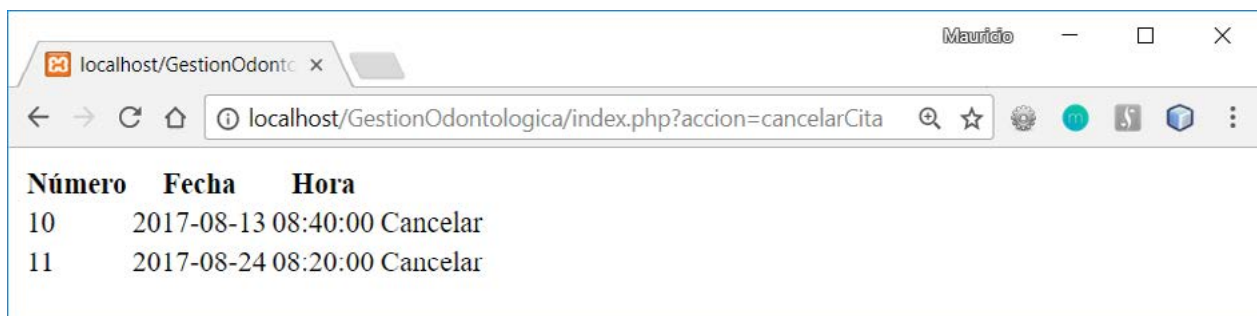
El código de dicho método es el siguiente.

```
public function cancelarCitas($doc){
    $gestorCita    = new GestorCita();
    $result        = $gestorCita->consultarCitasPorDocumento($doc);
    require_once 'Vista/html/cancelarCitas.php';
}
```

Para que esto funcione, es necesario realizar el cambio en el archivo “index.php” de la siguiente forma.

```
elseif($_GET["accion"] == "cancelarCita"){
    $controlador->cancelarCitas($_POST["cancelarDocumento"]);
}
```

Al hacer la prueba tanto con un paciente que posea citas se obtiene lo siguiente:



Número	Fecha	Hora
10	2017-08-13 08:40:00	Cancelar
11	2017-08-24 08:20:00	Cancelar

Figura 6.8. Resultado de la opción de cancelación de citas cuando existen.

Si no tiene citas aparece lo siguiente:

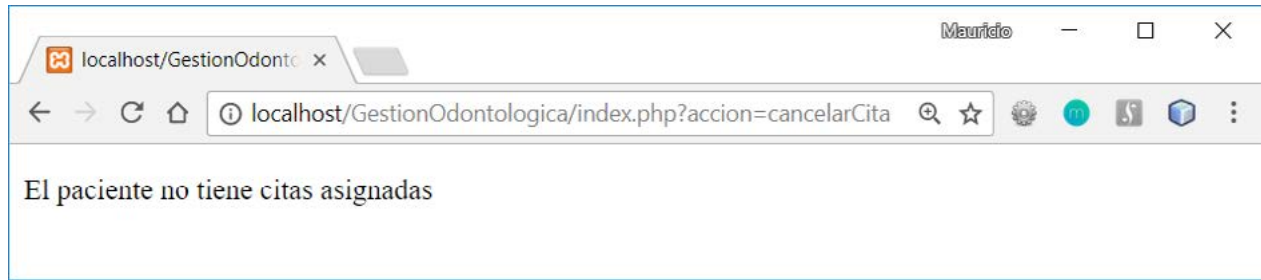


Figura 6.9. Resultado de la opción de cancelación de citas cuando no existen.

7. Incluyendo Javascript y JQuery parte 1

En la sesión anterior se trabajó con los archivos del controlador, tanto el Arranque (index.php) como la clase “Controlador”. Se ingresó y verificó el acceso a datos a los diferentes procesos del sistema de información como asignación, búsqueda y cancelación de citas.

En esta sesión se empezará a realizarle algunos ajustes al sistema de información apoyándonos en el lenguaje Javascript y la librería JQuery.

7.1. Consultar paciente

Cuando una cita se asigna esta irá asociada a un paciente específico pero dicho paciente debe estar registrado en la base de datos del sistema de información. De lo contrario, si dicho paciente no existe, será necesario crearlo en el sistema.

Lo primero que se hace es consultar si el paciente existe en la base de datos, en caso de no existir, se dará la opción al usuario de ingresar los datos de dicho paciente.

La estructura que hasta el momento se tiene es la siguiente:

Asignar cita

Contenido de la página

Documento del paciente

Médico

Fecha

Hora

Consultorio

Figura 7.1 Formulario de asignación de citas.

- Y el código de esta parte se muestra a continuación:

```
<tr>
    <td>Documento del paciente</td>
    <td><input type="text" name="asignarDocumento" id="asignarDocumento"></td>
</tr>
<tr>
    <td colspan="2"><input type="button" value="Consultar" name="asignarConsultar"
id="asignarConsultar"></td>
</tr>
<tr><td colspan="2"><div id="paciente"></div></td>
</tr>
```

Lo que se pretende es que el usuario ingrese el documento del paciente y de clic sobre el botón “Consultar”. Inmediatamente se realiza la consulta a la base de datos validando la información del paciente. Con base en esta validación se mostrará en esta misma página específicamente en el “<div> “con el id “paciente” la información del paciente y en caso que no exista mostrará el mensaje con la opción de crearlo.

Como se va a utilizar la librería JQuery se debe crear dentro de la carpeta “Vista” del proyecto una nueva carpeta llamada “jquery”. Una vez realizada esta actividad dentro de la carpeta “Vista” se encontraran las carpetas css, html, imagenes, jquery y js.

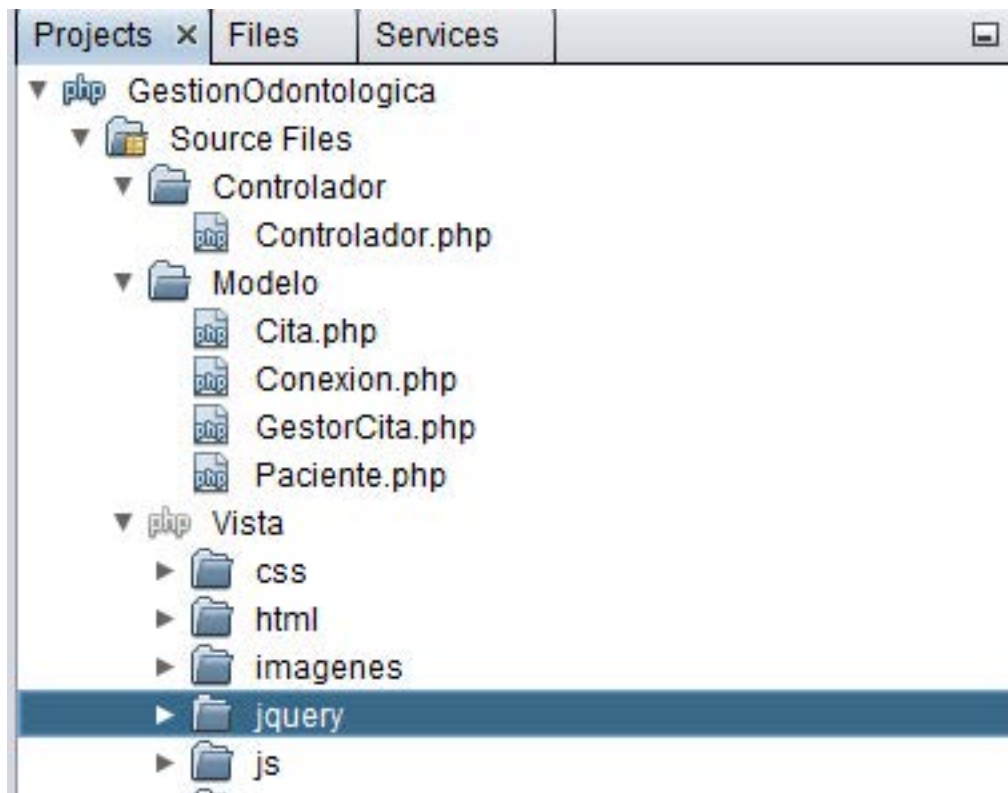


Figura 7.2 Ubicación de la carpeta jquery dentro de Netbeans.

Para descargar una librería llamada jquery se hace desde la página <https://jquery.com/download/>. Se puede descargar la versión comprimida o sin comprimir. Para este caso se eligió la versión comprimida la cual se caracteriza porque tiene la palabra “min” dentro del nombre (jquery-X.X.X-min.js). La librería se debe guardar dentro de la carpeta “jquery” que se creó en el punto anterior.

Luego de descargado el archivo es necesario incluirlo dentro de las páginas que lo utilizarán dentro del sitio web. Por tal motivo se incluirá dentro de los archivos “asignar.php”, “consultar.php”, “cancelar.php” de la siguiente forma.

```
<head>
    <title>Sistema de Gestión Odontológica</title>
    <link rel="stylesheet" type="text/css" href="Vista/css/estilos.css">
    <script type="text/javascript" src="Vista/html/jquery/jquery-3.2.1-min.js"></script>
</head>
```

Se continúa con el archivo “asignar.php”. Se ubica el botón del formulario identificado con el atributo “name” e “id” igual a “asignarConsultar”.

Al botón se le agrega un manejador del evento “onclick” de la siguiente forma:

```
<tr>
    <td colspan="2"><input type="button" value="Consultar" name="asignarConsultar"
id="asignarConsultar" onclick="consultarPaciente()"></td>
</tr>
```

Este manejador de eventos indica que cada vez que se oprima el botón “Consultar” el flujo del programa irá hasta la función “consultarPaciente()” en JavaScript.

Para codificar la función lo primero que se debe hacer es crear el archivo que contendrá el código JavaScript. Por lo tanto hay que ubicar el directorio “js” y se da clic derecho. En el menú contextual se selecciona la opción “New->JavaScript File...”:

- En caso de que esta opción no esté disponible en el menú contextual, se selecciona la opción Other, para buscar la plantilla.

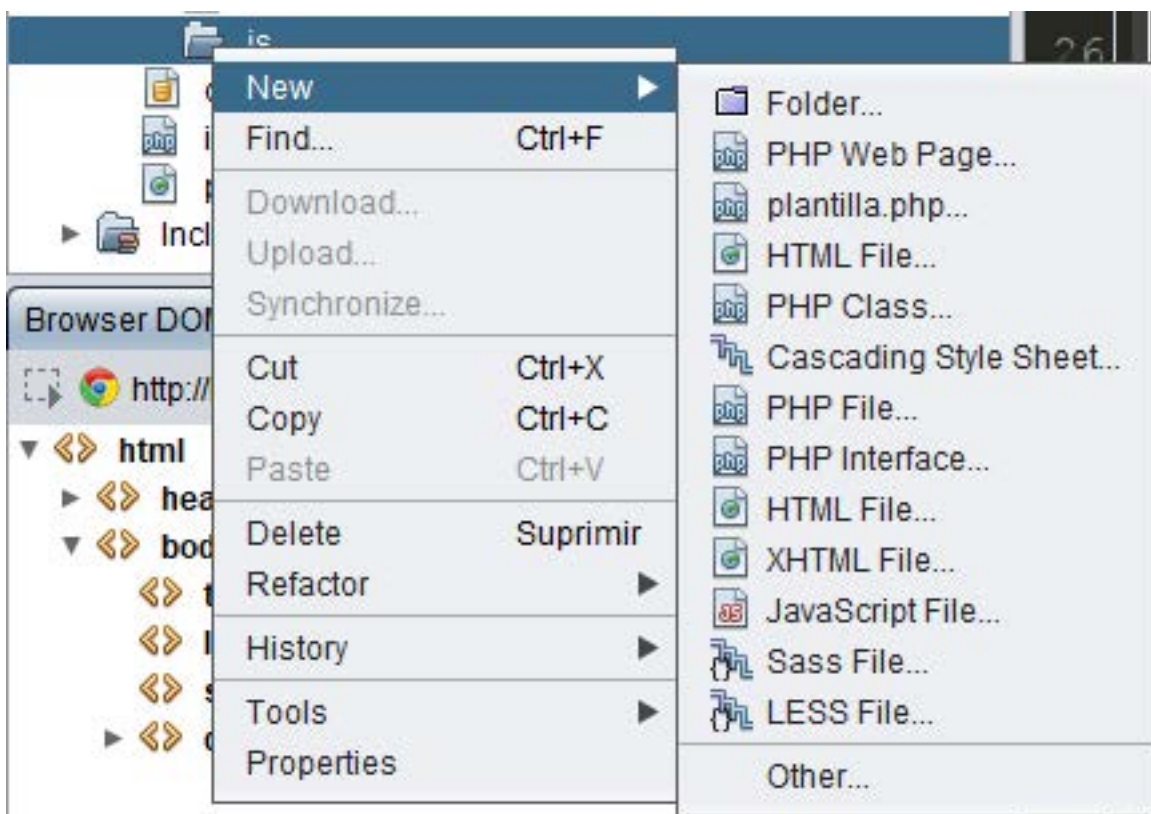


Figura 7.3 Creación de un archivo Javascript en Netbeans.

El nombre del archivo será script.js el cual va a controlar todas las funciones Javascript que se implementa en el sitio web. El archivo se debe incluir también en los archivos “asignar.php”, “consultar.php”, “cancelar.php” igual que se hizo con JQuery.

Ahora se codificará la función “consultarPaciente()” dentro del archivo “script.js” para que al momento de consultar un paciente, si este existe, se muestre la información del mismo en la misma página. De lo contrario se mostrará un mensaje indicando que no existe y un botón que permita crearlo.

Para realizar estas acciones, se utilizará una función de jquery llamada “load” la cual realiza la petición y reemplaza el contenido de un elemento HTML específico de la página.

El código de esta función se muestra a continuación.

```
function consultarPaciente(){
    var url = "index.php?accion=consultarPaciente&documento=" +
    $("#asignarDocumento").val();
    $("#paciente").load(url,function(){
    });
}
```

A continuación se crea un nuevo documento PHP que mostrará el resultado de la consulta o en su defecto un mensaje indicando que el paciente no existe.

Para esto se crea un archivo dentro de la carpeta “html” de tipo “PHP Web Page”. El nombre del archivo será “consultarPaciente.php” y su código se muestra a continuación:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>
    <?php
      if($result->num_rows > 0){
    ?>
    <table>
      <tr>
        <th>Documento</th><th>Nombre Completo</th><th>Sexo</th>
      </tr>
      <?php
        while($fila=$result->fetch_object()){
          ?>
          <tr>
            <td><?php echo $fila->PacIdentificacion;?></td>
            <td><?php echo $fila->PacNombres . " ". $fila->PacApellidos;?></td>
            <td><?php echo $fila->PacSexo;?></td>
            <td>Ver</td>
          </tr>
        <?php
          }
        ?>
      </table>
      <?php
      }
      else {
    ?>
    <p>El paciente no existe en la base de datos.</p>
    <input type="button" name="ingPaciente" id="ingPaciente" value="Ingresar
Paciente" onclick="ingPaciente()">
    <?php
    }
    ?>
  </body>
</html>
```

Ahora se crea el método “consultarPaciente()” dentro de la clase “Controlador”. Este método recibe el documento del paciente del cual se llamará al método “consultarPaciente()” dentro de la clase “GestorCitas” y se mostrará la página “consultarPaciente.php”.

El código del método es el siguiente:

```
public function consultarPaciente($doc){
    $gestorCita    = new GestorCita();
    $result        = $gestorCita->consultarPaciente($doc);
    require_once 'Vista/html/consultarPaciente.php';
}
```

Para que esto funcione, es necesario realizar el cambio en el archivo index. php de la siguiente forma:

```
elseif($_GET["accion"] == "ConsultarPaciente"){
    $controlador->consultarPaciente($_GET["documento"]);
}
```

Al hacer la prueba tanto con un paciente que exista, como con uno que no exista se obtienen los siguientes resultados.



Asignar cita

Contenido de la página

Documento del paciente

Documento	Nombre Completo	Sexo
91222333	Carlos Jesus Rodriguez Cala M	Ver

Médico

Fecha

Hora

Consultorio

Figura 7.4 Resultado de la consulta de pacientes cuando están creados.

Asignar cita

Contenido de la página

Documento del paciente

El paciente no existe en la base de datos.

Médico

Fecha

Hora

Consultorio

Figura 7.5 Resultado de la consulta de pacientes cuando no existe.

7.2. Ingresar paciente

Como se puede observar cuando el paciente no existe se muestra el mensaje y adicional a ello un nuevo botón cuyo texto es “Ingresar Paciente”.

Cuando el usuario de clic sobre este botón se debe mostrar un cuadro de diálogo con el formulario de ingreso de los datos del paciente. Para la creación de este formulario se usará una herramienta de JQuery llamada “dialog”.

Se puede descargar la librería JQueryUI desde la siguiente dirección: <http://jqueryui.com>:

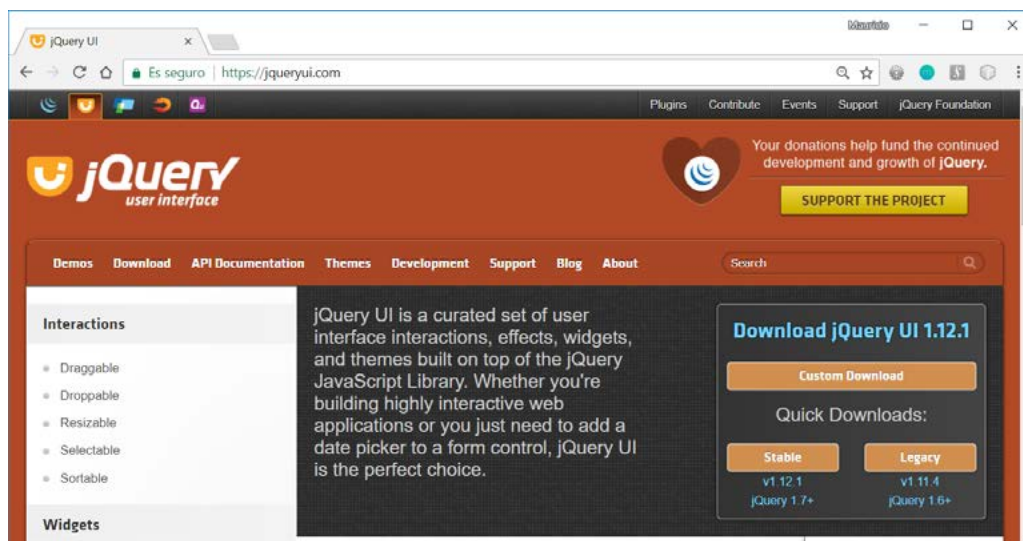


Figura 7.6 Sitio oficial de JQuery UI

El archivo se llamará “dialog.zip” y se almacenará en la carpeta jquery del proyecto. En Xampp se encuentra en la ruta “C:\xampp\htdocs\GestionOdontologica\Vista\jquery”.

Se descomprime la carpeta y luego la estructura dentro del proyecto debe ser la misma que se muestra en la imagen:

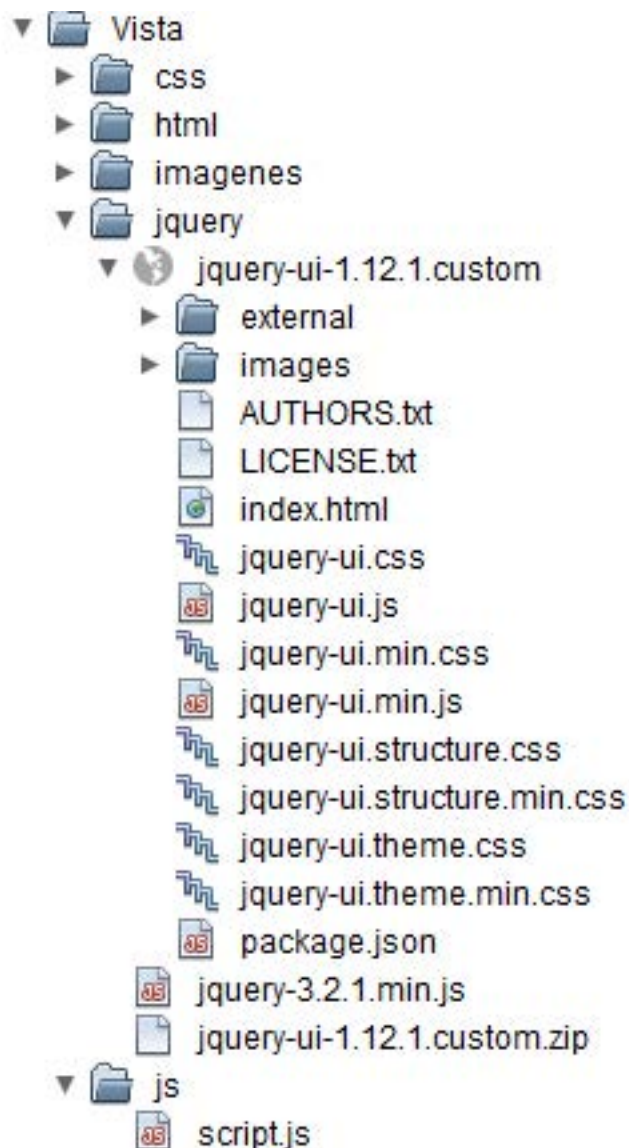


Figura 7.7 Ubicación de la carpeta JQueryUI en el proyecto Netbeans.

Lo primero que se debe hacer es incluir los archivos necesarios dentro del archivo “asignar.php” de la siguiente forma.

```
<head>  
<title>Sistema de Gestión Odontológica</title>
```

```
<link rel="stylesheet" type="text/css" href="Vista/css/estilos.css">
<script src="Vista/jquery/jquery-3.2.1.min.js" type="text/javascript"></script>
<script src="Vista/jquery/jquery-ui-1.12.1.custom/jquery-ui.js" type="text/javascript"></script>
<link href="Vista/jquery/jquery-ui-1.12.1.custom/jquery-ui.min.css" rel="stylesheet" type="text/css"/>
<script src="Vista/js/script.js" type="text/javascript"></script>
<script>
</script>
</head>
```

Dentro de este mismo archivo se creará al final del documento un nuevo formulario dentro de un “<div>”. Dicho formulario solamente se mostrará como cuadro de diálogo cuando el usuario da clic sobre el botón “Ingresar Paciente”.

El código de este formulario se muestra a continuación:

```
<div id="frmPaciente" title="Agregar Nuevo Paciente">
  <form id="agregarPaciente">
    <table>
      <tr>
        <td>Documento</td>
        <td><input type="text" name="PacDocumento" id="PacDocumento"></td>
      </tr>
      <tr>
        <td>Nombres</td>
        <td><input type="text" name="PacNombres" id="PacNombres"></td>
      </tr>
      <tr>
        <td>Apellidos</td>
        <td><input type="text" name="PacApellidos" id="PacApellidos"></td>
      </tr>
      <tr>
        <td>Fecha de Nacimiento</td>
        <td><input type="text" name="PacNacimiento" id="PacNacimiento"></td>
      </tr>
      <tr>
        <td>Sexo</td>
        <td><select id="pacSexo" name="PacSexo">
          <option value="-1">
```

```
selected="selected">--Seleccione el sexo ---</option>
                                <option value="M">Masculino</option>
                                <option value="F">Femenino</option>
                                </select>
                                </td>
                                </tr>
                                </table>
                                </form>
                                </div>
```

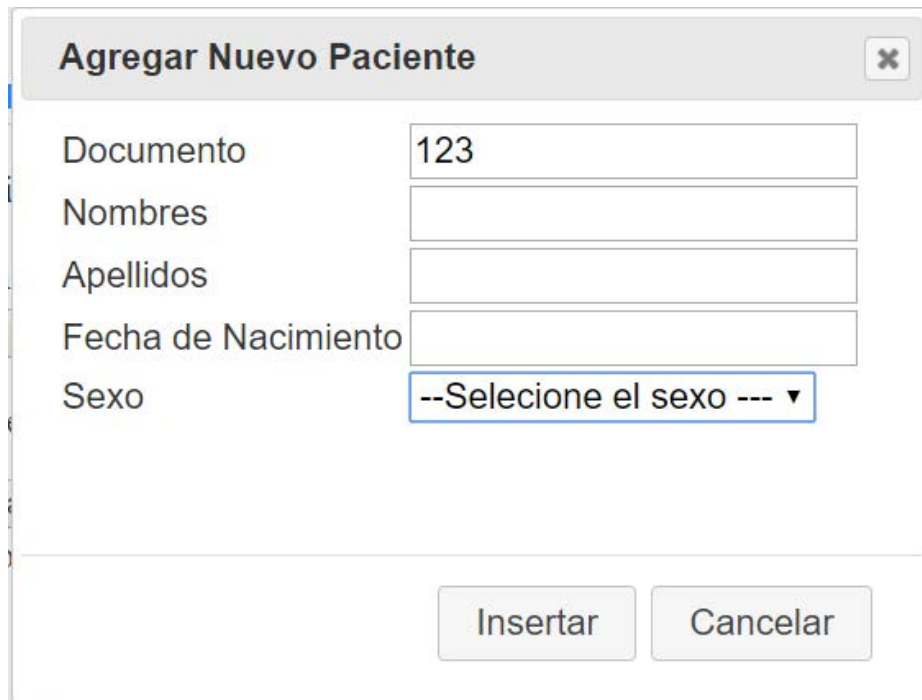
Para que funcione este contenido dentro de un elemento JQueryUI del tipo “dialog”, es necesario agregar unas líneas de código JavaScript del archivo “script.js”. El programa completo queda como se muestra a continuación.

```
$(document).ready(function(){
    $("#frmPaciente").dialog({
        autoOpen: false,
        height: 310,
        width: 400,
        modal: true,
        buttons: {
            "Insertar":insertarPaciente,
            "Cancelar":cancelar
        }
    });
});

function consultarPaciente(){
    var url = "index.php?accion=consultarPaciente&documento=" +
$("#asignarDocumento").val();
    $("#paciente").load(url,function(){
    });
}

function mostrarFormulario(){
    documento = "" + $("#asignarDocumento").val();
    $("#PacDocumento").attr("value",documento);
    $("#frmPaciente").dialog('open');
}
```

Si todo está correcto al momento de dar clic sobre el botón “Ingresar Paciente” se mostrará un nuevo cuadro de diálogo con el formulario y el documento del paciente que aún no está registrado.



El formulario 'Agregar Nuevo Paciente' contiene los siguientes campos:

- Documento: 123
- Nombres: [Campo vacío]
- Apellidos: [Campo vacío]
- Fecha de Nacimiento: [Campo vacío]
- Sexo: --Seleccione el sexo --

Los botones 'Insertar' y 'Cancelar' están ubicados en la parte inferior derecha del formulario.

Figura 7.8 Formulario para la creación de pacientes usando JQueryUI.

Se va a codificar en JavaScript las funciones “insertarPaciente()” y “cancelar()”, las cuales están asociadas a los botones del formulario.

El código se muestra a continuación.

```
function insertarPaciente(){
    queryString = $("#agregarPaciente").serialize();
    url = "index.php?accion=ingresarPaciente&" + queryString ;
    $("#paciente").load(url);
    $("#frmPaciente").dialog('close');
}

function cancelar(){
    $(this).dialog('close');
}
```

Ahora se crea el método “agregarPaciente()” dentro de la clase “Controlador”. El código del método es el siguiente:

```
public function agregarPaciente($doc,$nom,$ape,$fec,$sex){
    $paciente      = new Paciente($doc, $nom, $ape, $fec, $sex);
    $gestorCita    = new GestorCita();
    $registros     = $gestorCita->agregarPaciente($paciente);
    if($registros > 0){
```

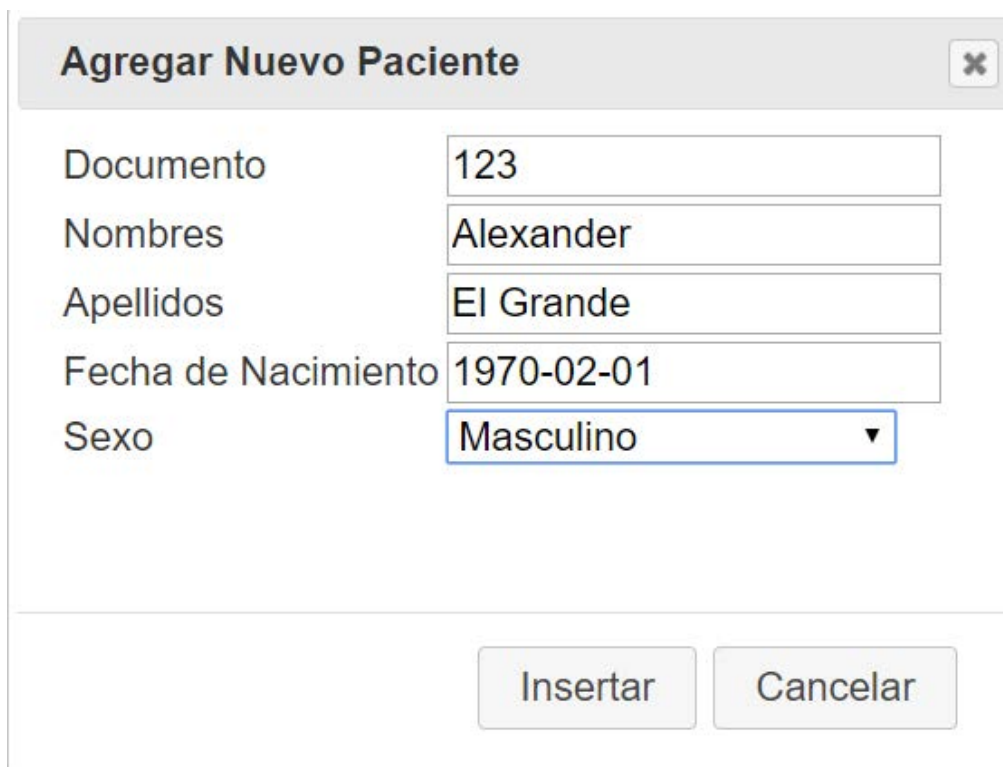


```
        echo "Se insertó el paciente con exito";  
    } else {  
        echo "Error al grabar el paciente";  
    }  
}
```

Para que esto funcione es necesario realizar el cambio en el archivo "index.php" de la siguiente forma:

```
elseif($_GET["accion"] == "ingresarPaciente"){  
    $controlador->agregarPaciente(  
        $_GET["PacDocumento"],  
        $_GET["PacNombres"],  
        $_GET["PacApellidos"],  
        $_GET["PacNacimiento"],  
        $_GET["PacSexo"]  
    );  
}
```

Con estos cambios se insertará un paciente con los datos que se muestran a continuación:



El formulario "Agregar Nuevo Paciente" contiene los siguientes campos:

Documento	123
Nombres	Alexander
Apellidos	El Grande
Fecha de Nacimiento	1970-02-01
Sexo	Masculino ▼

En la parte inferior del formulario hay dos botones: "Insertar" y "Cancelar".

Figura 7.9 Prueba de creación de paciente

Si todo ha salido bien se mostrará un mensaje confirmándolo:



Asignar cita

Contenido de la página

Documento del paciente

Se insertó el paciente con éxito

Médico

Fecha

Hora

Consultorio

Figura 7.10 Resultado de la prueba de creación de paciente

Se puede verificar consultando el paciente.



Asignar cita

Contenido de la página

Documento del paciente

Documento	Nombre Completo	Sexo
123	Alexander El Grande M	Ver

Médico

Fecha

Hora

Consultorio

Figura 7.11 Verificación del resultado de la prueba de creación de paciente

8. Incluyendo Javascript y JQuery parte 2

En la sesión anterior con las librerías Javascript JQuery y JQueryUI se realizó la consulta para verificar la existencia del paciente y en caso de que no existiera se utilizó una herramienta llamada “dialog” para agregar un nuevo usuario.

En esta sesión se continuará con el formulario de asignación de citas, cargando los datos de los médicos, y las citas disponibles de acuerdo a la fecha seleccionada.

8.1. Llenar el elemento "select" de médicos

Al momento de asignar una cita se debe seleccionar el médico que la atenderá. Cuando se construyó la vista del sistema de información se trabajó de manera estática ingresando en el documento HTML los médicos que en el momento se encontraban en la base de datos. Pero ocurre que si ingresan nuevos médicos o se retiran este listado se debe modificar. Por este motivo es necesario que esos datos se carguen de la base de datos.

Actualmente se tiene el siguiente código que controla estas actividades.

```
<tr>
    <td>Médico</td>
    <td>
        <select id="medico" name="medico">
            <option value="-1" selected="selected">---Seleccione el
Médico</option>
            <option value="12345">12345-Pepito Pérez</option>
            <option value="67890">67890-Pepita Mendieta</option>
        </select>
    </td>
</tr>
```

Se va a realizar la siguiente modificación a ese segmento de código como se muestra en la imagen:

```
<tr>
    <td>Médico</td>
    <td>
        <select id="medico" name="medico" onchange="cargarHoras()">
            <option value="-1" selected="selected">---Seleccione el
Médico</option>
            <?php
                while( $fila = $result->fetch_object())
                {
                    ?>
                    <option value = <?php echo $fila->MedIdentificacion; ?> >
                    <?php echo $fila->MedIdentificacion . " " . $fila-
>MedNombres . " ". $fila->MedApellidos; ?>
                }
            </?php>
        </select>
    </td>
</tr>
```

```

                </option>
            <?php } ?>
        </select>
    </td>
</tr>

```

Para que todo esto funcione como se desea es necesario hacer unas modificaciones tanto al “index.php” como al controlador. Se inicia con el archivo “index.php”.

Actualmente la acción de cargar la página “asignar.php” está controlada de la siguiente manera.

```

if($_GET["accion"] == "asignar"){
    $controlador->verPagina('Vista/html/asignar.php');
}

```

Se va a modificar para que quede de la siguiente forma.

```

if($_GET["accion"] == "asignar"){
    $controlador->cargarAsignar();
}

```

A continuación se crea el método “cargarAsignar()” en la clase “Controlador”. Este tendrá el siguiente código:

```

public function cargarAsignar(){
    $gestorCita    = new GestorCita();
    $result        = $gestorCita->consultarMedicos();
    require_once 'Vista/html/asignar.php';
}

```

Con estos cambios al momento de ingresar a la página para asignar una cita, dinámicamente se cargarán los médicos existentes en la base de datos.

8.2. Llenar el elemento “select” de horas

Para llenar el select con los horarios disponibles de un médico cierto día, es necesario que se tenga seleccionado tanto un médico como una fecha. Por tal motivo es necesario que al momento de cambiar el médico o la fecha de la cita se carguen las horas dentro las opciones del “select”.

Se va a asignar un manejador al evento “change” tanto del “select” del médico como del “input” de la fecha. Dichos cambios se realizarán en el archivo “asignar.php” como se muestra a continuación:

```
<select id="medico" name="medico" onchange="cargarHoras()">
```

```
<input type="date" id="fecha" name="fecha" onchange="cargarHoras()">
```

Se debe crear ahora la función “cargarHoras()” dentro del archivo “script.js”. Dicha función posee el siguiente código:

```
function cargarHoras(){
    if( ($("#medico").val() == -1 ) || ($("#fecha").val() == "")){
        $("#hora").html("<option value='-1' selected='selected'>--Seleccione la
hora </option>")
    } else {
        queryString = "medico="+$("#medico").val()+"&fecha="+$("#fecha").val();
        url          = "index.php?accion=consultarHora&" + queryString ;
        $("#hora").load(url);
    }
}
```

Se va a asignar un manejador al evento “mousedown” del “select” hora para validar que estén seleccionados el médico y la fecha. Para que en caso de no estar seleccionado muestre una alerta. El código anterior era el siguiente:

```
<tr>
    <td>Hora</td>
    <td>
        <select id="hora" name="hora">
            <option value="-1" selected="selected">---Seleccione
la hora ---</option>
            <option>08:00:00</option>
            <option>08:20:00</option>
            <option>08:40:00</option>
            <option>09:00:00</option>
        </select>
    </td>
</tr>
```

Se va a modificar para asignarle el manejador de eventos. Además se eliminarán todas las opciones ya que estas se cargarán dinámicamente. Entonces el segmento de código anterior se reemplaza por:

```
<tr>
    <td>Hora</td>
    <td>
        <select id="hora" name="hora" onmousedown="seleccionarHora()">
            <option value="-1" selected="selected">---Seleccione
la hora ---</option>
```

```

                </select>
            </td>
        </tr>

```

Se debe codificar en JavaScript la función “seleccionarHora()” y agregar a “script.js”. El código es el siguiente:

```

function seleccionarHora(){
    if($("#medico").val() == -1 ){
        alert("Debe seleccionar un médico");
    } else if ($("#fecha").val() == ""){
        alert("Debe seleccionar una fecha");
    }
}

```

8.3. Codificación del método "consultarHorasDisponibles()"

En la sesión 5 se trabajó con las clases del modelo, y entre ellas la clase “GestorCita”. Dentro de esta clase quedaron pendientes algunos métodos los cuales no se codificaron en dicha sesión y se irán codificando a continuación. El método consultarHorasDisponibles() permite consultar el listado de horas disponibles para atender una cita un médico en una fecha específica.

Antes de proceder a codificar este código, es necesario crear una nueva tabla en la base de datos la cual almacenará las horas en las cuales se atenderán citas en el consultorio odontológico. Lo anterior se puede hacer con el siguiente script de base de datos:

```

create table horas ( hora time );
insert into horas values ('08:00:00'),('08:20:00'),('08:40:00'),('09:00:00'),('09:20:00'),('09:40:00') ;
insert into horas values ('10:00:00'),('10:20:00'),('10:40:00'),('11:00:00'),('11:20:00'),('11:40:00') ;

```

Para verificar se puede hacer una consulta en la consola de Mysql:

```

MariaDB [citas]> select * from horas ;
+-----+
| hora |
+-----+
| 08:00:00 |
| 08:20:00 |
| 08:40:00 |
| 09:00:00 |
| 09:20:00 |

```

```
| 09:40:00 |
| 10:00:00 |
| 10:20:00 |
| 10:40:00 |
| 11:00:00 |
| 11:20:00 |
| 11:40:00 |
```

```
+-----+
```

```
12 rows in set (0.00 sec)
```

```
MariaDB [citas]>
```

Ahora se procede a la creación del método “consultarHorasDisponibles()” dentro de la clase “GestorCitas” el cual tiene el siguiente código:

```
public function consultarHorasDisponibles($medico,$fecha){
    $conexion = new Conexion();
    $conexion->abrir();
    $sql      = "SELECT hora FROM horas WHERE hora NOT IN "
                . "( SELECT CitHora FROM citas WHERE CitMedico = '$medico' AND
CitFecha = '$fecha' "
                . " AND CitEstado = 'Solicitada' ) ";
    $conexion->consulta($sql);
    $result = $conexion->obtenerResult();
    $conexion->cerrar();
    return $result ;
}
```

En este momento el código de la función en JavaScript y el método “consultarHorasDisponibles()” dentro de la clase “GestorCita”. Falta crear el método dentro del controlador y acceder a él a través del “index.php”. Además se necesita como en las anteriores ocasiones un código en php que cargue las horas disponibles para incluirlas en el “select”.

Dentro de la carpeta “Vista” dentro de la carpeta “html” se debe crear un nuevo “PHP Web File”. A este archivo se asigna el nombre “consultarHoras.php” y tendrá el siguiente código.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>
    <option value="-1" selected="selected">---Seleccione la hora ---</
```

```

option>
    <?php
        while($fila = $result->fetch_object()){
        ?>
        <option><?php echo $fila->hora; ?></option>
        <?php
            }
        ?>
    </body>
</html>

```

A continuación se crea el método “consultarHorasDisponibles()”, el cual es miembro de la clase “Controlador”. Este posee el código mostrado en la imagen.

```

    public function consultarHorasDisponibles($medico,$fecha){
        $gestorCita      = new GestorCita();
        $result           = $gestorCita->consultarHorasDisponibles($medico,
$fecha);
        require_once 'Vista/html/consultarHoras.php';
    }

```

Pero para que este código funcione es necesario hacer la modificación en el “index.php” como se muestra a continuación:

```

        elseif($_GET["accion"] == "consultarHora"){
            $controlador->consultarHorasDisponibles($_GET["medico"], $_
GET["fecha"]);
        }

```

Si se verifica se evidencia que al dar clic sobre el “select” de la hora, si no hay un médico seleccionado, se mostrará un alert, igual si no hay una fecha seleccionada, también se mostrará un alert, y si ambos campos poseen datos, se mostrará un listado con las horas disponibles para citas que posee el médico esa fecha.

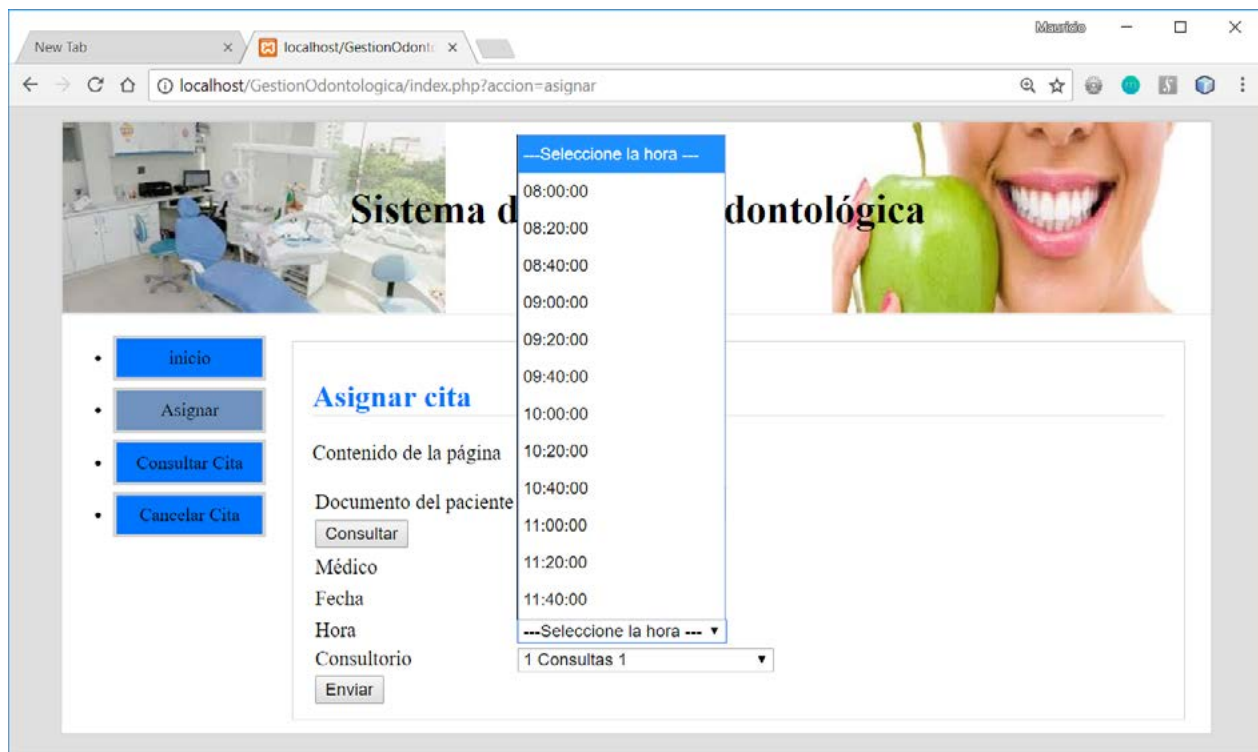


Figura 8.1 Vista del Select para las horas de las citas

8.4. Llenar el elemento "select" de consultorios

Este proceso tiene los mismos pasos que se realizaron al llenar el select de médicos. Actualmente se tiene el siguiente código que controla esta actividad.

```
<tr>
    <td>Consultorio</td>
    <td>
        <select id="consultorio" name="consultorio">
            <option value="-1" selected="selected">---Seleccione el Consultorio---</option>
            <option value="1">1 Consultas 1</option>
            <option value="2">2 Tratamientos 1</option>
        </select>
    </td>
</tr>
```

Se debe realizar la siguiente modificación a ese segmento de código como se muestra a continuación:

```
<tr>
    <td>Consultorio</td>
```

```

        <td>
            <select          id="consultorio"          name="consultorio"
onchange="cargarHoras()">
                <option value="-1" selected="selected">---Seleccione el
Consultorio</option>
                <?php
                    while( $fila = $result2->fetch_object())
                    {
                        ?>
                        <option value = <?php echo $fila->ConNumero; ?> >
                        <?php echo $fila->ConNumero . " - " . $fila->ConNombre ;
                    ?>
                        </option>
                    <?php } ?>
                </select>
            </td>
        </tr>

```

Para que todo esto funcione es necesario modificar el método “cargarAsignar()” del controlador. Este tendrá las modificaciones señaladas en la imagen.

```

public function cargarAsignar(){
    $gestorCita    = new GestorCita();
    $result        = $gestorCita->consultarMedicos();
    $result2       = $gestorCita->consultarConsultorios();
    require_once 'Vista/html/asignar.php';
}

```

También se debe crear el método “consultarConsultorios()” en la clase “GestorCita” así:

```

public function consultarConsultorios(){
    $conexion = new Conexion();
    $conexion->abrir();
    $sql      = "SELECT * FROM consultorios ";
    $conexion->consulta($sql);
    $result = $conexion->obtenerResult();
    $conexion->cerrar();
    return $result ;
}

```

8.5. Consultar citas

Ya se han terminado prácticamente todos los procesos correspondientes a la asignación de una cita ahora se trabajará sobre la opción “Consultar cita”.

En la sesión 6 se trabajó este punto el cual muestra un formulario para ingresar el

documento del paciente y en otra página mostraba las citas que tenía asignadas. Esto se modificará de tal forma que al usuario dar clic sobre el botón “consultar” el resultado se cargue dinámicamente en la misma página.

El primer cambio a realizar es en el archivo “consultar.php”. Allí se tiene un botón el cual posee el siguiente código:

```
<tr>
    <td colspan="2"><input type="submit" name="consultarConsultar"
value="Consultar" id="consultarConsultar"></td>
</tr>
```

Este código se reemplazará por el siguiente:

```
<tr>
    <td colspan="2"><input type="submit" name="consultarConsultar"
value="Consultar" id="consultarConsultar" onclick="consultarCita()"></td>
</tr>
```

A continuación se requiere crear la función “consultarCita()” dentro del archivo “script.php”. Esta función contendrá el código que se muestra a continuación:

```
function consultarCita(){
    url = "index.php?accion=consultarCita&consultarDocumento=" +
$("#consultarDocumento").val() ;
    $("#paciente2").load(url);
}
```

El siguiente cambio se debe realizar en el archivo “index.php” ya que anteriormente el documento del paciente se consultaba a través de la matriz “\$_POST” y con los cambios anteriores se consultará en la matriz \$_GET.

```
elseif($_GET["accion"] == "consultarCita"){
    $controlador->consultarCitas($_GET["consultarDocumento"]);
}
```

Con estos cambios la carga se realizaría dinámicamente mostrando el siguiente resultado:



Consultar Cita

Documento del Paciente

Número	Fecha	Hora
10	2017-08-13	08:40:00 Ver
11	2017-08-24	08:20:00 Ver

Figura 8.2 Vista de la relación de citas de un paciente.

Se puede observar que a la derecha de la tabla de “citas” aparece la palabra “Ver”. Se va a generar un enlace de tal forma que al dar clic sobre este enlace se muestre toda la información correspondiente a la cita seleccionada.

Para lo anterior se realizará la siguiente modificación al archivo “consultarCitas.php” justo en la celda donde se encuentra la palabra “Ver”.

```
<td><a href="index.php?accion=verCita&numero=<?php echo $fila->CitNumero;
?>">Ver</a></td>
```

A continuación se inserta el código en el archivo “index.php” que se encargará de llamar al método adecuado del controlador cuando la acción sea “verCita”. El código a insertar en el “index.php” se muestra a continuación:

```
elseif($_GET["accion"] == "verCita"){
    $controlador->verCita($_GET["numero"]);
}
```

Ahora se crea el método “verCita()” dentro de “Controlador.php”. Este tendrá el siguiente código.

```
public function verCita($cita){
    $gestorCita = new GestorCita();
    $result = $gestorCita->consultarCitaPorId($cita);
    require_once 'Vista/html/confirmarCita.php';
}
```

Si todo ha quedado bien al momento de dar clic sobre el enlace “Ver” en alguna de las citas aparecerá la información de la cita de la siguiente forma.

Información Cita	
Datos del Paciente	
Documento	91222333
Nombre	Carlos Jesus Rodriguez Cala
Datos del Médico	
Documento	12345
Nombre	Pepito Perez
Datos de la Cita	
Número	10
Fecha	2017-08-13
Hora	08:40:00
Número de Consultorio	Consultas
Estado	Solicitada
Observaciones	Ninguna

Figura 8.3 Vista del reporte de la cita.

8.6. Cancelar citas

En este proceso se deben realizar unos cambios muy similares a los realizados en el punto cinco de esta sesión. El primer cambio se realizará en el archivo “cancelar.php”. En este archivo se tiene un botón el cual posee el siguiente código:

```
<tr>
    <td colspan="2"><input type="submit" name="cancelarConsultar"
value="Consultar" id="cancelarConsultar"></td>
</tr>
```

Este código se reemplazará por el siguiente:

```
<tr>
    <td colspan="2"><input type="button" value="Consultar"
onclick="cancelarConsultar()"></td>
</tr>
```

A continuación se requiere crear la función “cancelarCita()” dentro del archivo “script.js”. Esta función contendrá lo siguiente:

```
function cancelarCita(){
    url = "index.php?accion=cancelarCita&cancelarDocumento=" +
$("#cancelarDocumento").val() ;
    $("#paciente3").load(url);
}
```

El siguiente cambio se debe realizar en el archivo “index.php” ya que anteriormente el documento del paciente llegaba a través de la matriz \$_POST, y con los cambios anteriores llegará en la matriz \$_GET.

```
elseif($_GET["accion"] == "cancelarCita"){
    $controlador->cancelarCitas($_GET["cancelarDocumento"]);
}
```

Con estos cambios la carga se realizaría dinámicamente mostrando lo siguiente:

Cancelar Cita

Documento del Paciente

Número	Fecha	Hora	
10	2017-08-13	08:40:00	Cancelar
11	2017-08-24	08:20:00	Cancelar

Figura 8.4 Relación de citas que se pueden cancelar.

Como se puede observar a la derecha de la tabla aparece la palabra “Cancelar”. Este texto actuará como enlace, y permitirá cancelar una cita en especial. Se va a realizar la siguiente modificación al archivo “cancelarCitas.php” en la celda donde se encuentra la palabra “cancelar”.

```
<td><a href="#" onclick="confirmarCancelar(<?php echo $fila->CitNumero;
?>)">Cancelar</a></td>
```

Ahora es necesario crear la función “confirmarCancelar()” dentro del archivo “script.js con el siguiente código:

```
function confirmarCancelar(numero){
    if(confirm("Esta seguro de cancelar la cita " + numero)){
        $.get("index.php",{accion:'confirmarCancelar',numero:numero},function(mensaje)
        {
            alert(mensaje);
        });
    }
    $("#cancelarConsultar").trigger("click");
}
```

A continuación se inserta el código en el archivo “index.php” que se encargará de llamar al método adecuado del controlador cuando la acción sea “confirmarCancelar”. El código a insertar en el “index.php” es el siguiente:

```
elseif($_GET["accion"] == "confirmarCancelar"){
    $controlador->confirmarCancelarCita($_GET["numero"]);
}
```

Ahora se crea el método “confirmarCancelarCita()” dentro de “Controlador.php”. Este tendrá el siguiente código.

```
public function confirmarCancelarCita($cita){
    $gestorCita = new GestorCita();
    $registros = $gestorCita->cancelarCita($cita);
    if($registros > 0){
        echo "La cita se ha cancelado con éxito";
    } else {
        echo "Hubo un error al cancelar la cita";
    }
}
```

Para finalizar es necesario crear el método “cancelarCita()” dentro de la clase “GestorCita”. método tendrá el siguiente código.

```
public function cancelarCita($cita){
    $conexion = new Conexion();
    $conexion->abrir();
    $sql      = "UPDATE citas SET CitEstado = 'Cancelada' "
        . " WHERE CitNumero = $cita ";
    $conexion->consulta($sql);
    $filasAfectadas = $conexion->obtenerFilasAfectadas();
    $conexion->cerrar();
    return $filasAfectadas;
}
```

Si todo ha quedado bien, al momento de dar clic sobre el enlace “Cancelar” en alguna de las citas aparecerá un mensaje de confirmación así:

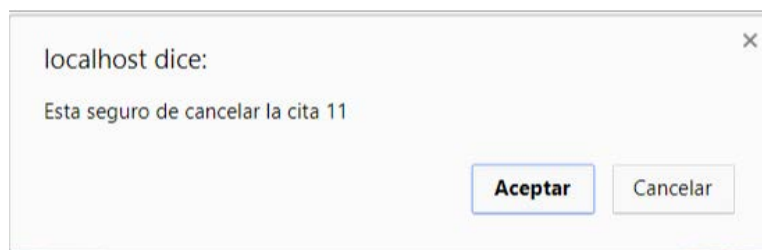


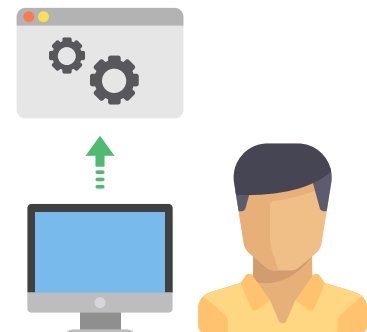
Figura 8.5 Diálogo de confirmación de la cancelación de la cita.

Una vez se da clic sobre el botón “aceptar” el sistema realiza la cancelación de la cita.

9. Despliegue de la aplicación y construcción del instalador

Una vez concluido el proceso de desarrollo y culminadas las pruebas se procede al despliegue de la aplicación.

Las aplicaciones web desarrolladas con PHP a diferencia de las aplicaciones nativas para Windows, Mac o Linux no generan código objeto que se pueda distribuir en un único archivo. El despliegue generalmente consiste en copiar la estructura del proyecto en Netbeans al computador destino.



Se pueden identificar los siguientes pasos:

9.1 Instalación del ambiente XAMPP en el equipo de cómputo de destino

La máquina destino debe tener instalado Mysql, Apache2 y PHP en el sistema operativo que use. Antes de proceder con los siguientes pasos es importante revisar que el lenguaje PHP esté operativo. Esto se puede hacer con la siguiente línea de código que se copia en un archivo PHP llamado “prueba.php”.

```
<?php echo phpinfo(); ?>
```

Por último se copia este archivo a la carpeta “htdocs” y se visualiza en el navegador. Si PHP está instalado debe aparecer lo siguiente:

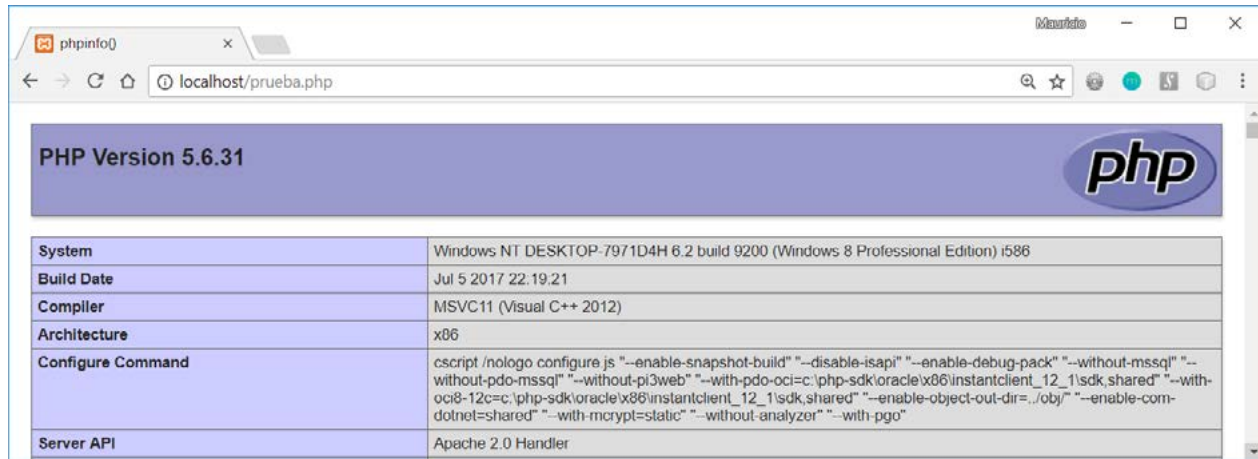


Figura 9.1 Información suministrada por la función phpinfo().

La página muestra la versión de PHP, el sistema operativo donde está operando.

9.2 Creación de la base de datos

En Mysql se puede usar la herramienta “mysqldump” para generar una copia de la base de datos ya sea con datos de prueba o sólo la estructura. A continuación se muestra el uso de esta herramienta para hacer un backup o copia de respaldo completo de la base de datos Citas:

```
Setting environment for using XAMPP for Windows.
```

```
USUARIO@DESKTOP-7971D4H d:\xampp
```

```
# mysqldump citas -B -u root -p > citas_db.sql
```

```
Enter password: *****
```

```
USUARIO@DESKTOP-7971D4H d:\xampp
```

```
# notepad++ citas_db.sql
```

Para recuperar la base de datos en el equipo destino se ejecuta el archivo de la copia que está en lenguaje SQL como se muestra a continuación:

```
USUARIO@DESKTOP-7971D4H d:\xampp
```

```
# mysql -u root -p < citas_db.sql
```

```
Enter password: *****
```

```
USUARIO@DESKTOP-7971D4H d:\xampp
```

```
#
```


9.3 Copia de los archivos fuentes de la aplicación

Los archivos fuente de la aplicación se pueden distribuir en un formato comprimido como zip o rar. Para generar un archivo .zip se puede usar el menú contextual del explorador de archivos así:

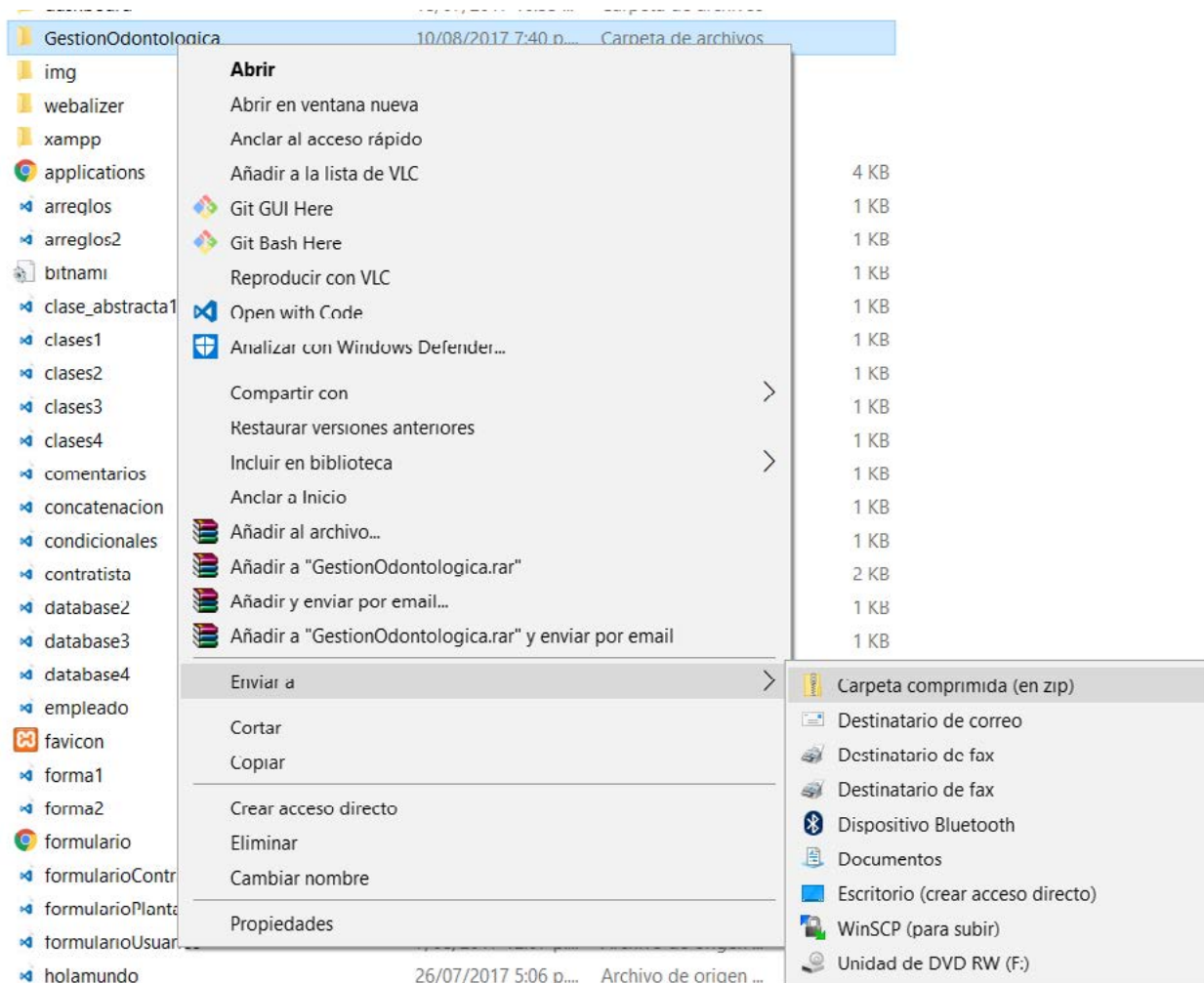


Figura 9.2 Generación de un archivo comprimido de la carpeta GestionOdontologica.

Glosario

AJAX: acrónimo de Asynchronous Javascript and XML. Tecnología que permite a las páginas web actualizar los datos sin tener que recargar la página.

Atributo: característica o propiedad de un elemento HTML que puede ser definida por el programador.

Browser: navegador o visualizador de páginas web.

Bootstrap: framework CSS que simplifica y estandariza el diseño web.

CSS: acrónimo de Cascade-Style Sheet. Hojas de estilos en cascada. Tecnología que permite personalizar la presentación de las páginas web.

Diseño adaptativo: estrategia para abordar el diseño web para distintos tamaños de pantalla al mismo tiempo.

DOM: acrónimo de Document Object Model. Modelo de objeto para documentos que permite ver un documento como un conjunto de nodos en forma de árbol.

Etiqueta: elementos que en su conjunto conforman una página web.

Framework: conjunto de programas y librerías que apoyan varios aspectos de la programación.

Jquery: librería Javascript que simplifica tareas comunes de programación web.

Herencia: en CSS es la cualidad que tienen los nodos hijos de tener los mismos atributos que sus padres.

HTML: acrónimo de Hypertext Markup Language. Lenguaje de marcación de hipertexto.

Javascript: lenguaje de programación empotrado dentro de los navegadores web.

Librería: conjunto de programas afines que apoyan un aspecto de la programación.

Layout: maqueta o prototipo de un diseño web.

MVC: acrónimo de Model-View-Controller. Patrón de diseño compuesto por tres capas que son el modelo, la vista y el controlador.

Selector: parte de una regla CSS que identifica los elementos HTML a los cuales aplicará.

WWW: acrónimo de World Wide Web o Red Mundial que se comparte a través de la Internet.

XML: acrónimo de eXtensible Markup Language. Lenguaje de marcación extensible.

Bibliografía

Niederst Robbins J. (2012). *Learning Web Design*. Beijing: Cambridge, O'Reilly.

W3C The World Wide Web Consortium (2016). *HTML 5.1 W3C Recommendation*. Recuperado de: <https://www.w3.org/TR/html>

W3C The World Wide Web Consortium (2017). *CSS Snapshot 2017*. Recuperado de: <https://www.w3.org/TR/CSS>

Control de documento

CONSTRUCCIÓN OBJETO DE APRENDIZAJE



DESARROLLO DE APLICACIONES WEB EN PHP

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: Leydy Carolina Muñoz (V1)
Nelson Mauricio Silva M. (V2)

Diseño multimedia: Luis Gabriel Urueta Alvarez

Programador: Francisco José Lizcano Reyes

Producción de audio: Víctor 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.



**creative
commons**

PHP®, Copyright © PHP Group.
Software libre publicado bajo la licencia PHP.



Registered trademark