

**Tema**

FORMULARIO WEB CON MVC UTILIZANDO APACHE NETBEANS

**Tutor**

Ing. Eduardo Mauricio Campaña Ortega

MIS.MDU.CCNA.CCIA.

PhD. (c) Ingeniería de Software

PhD. (c) Seguridad Información

**Fecha**

14/11/2022

FORMULARIO WEB SIN MVC EN APACHE NETBEANS

[MARCO TEÓRICO 3](#_Toc119317923)

[JAVA WEB 3](#_Toc119317924)

[JAVA SERVER PAGES (JSP) 4](#_Toc119317925)

[SERVLETS 4](#_Toc119317926)

[JAVABEANS 4](#_Toc119317927)

[ARQUITECTURA DE SOFTWARE 5](#_Toc119317928)

[PATRÓN MODELO VISTA CONTROLADOR 5](#_Toc119317929)

[MODELO 5](#_Toc119317930)

[VISTA 5](#_Toc119317931)

[CONTROLADOR 5](#_Toc119317932)

[CREACIÓN DEL PROYECTO 6](#_Toc119317933)

[OBJETIVO 6](#_Toc119317934)

[CREAR UN SITIO WEB LLAMADO FORMULARIOWEBSINMVC 6](#_Toc119317935)

[\_APELLIDO\_ NOMBRE 6](#_Toc119317936)

[DESARROLLO DEL PATRÓN MVC 10](#_Toc119317937)

[CREACIÓN DE LOS PAQUETES Y REESTRUCTURACIÓN DEL PROYECTO 10](#_Toc119317938)

[CODIFICACIÓN DE LOS ARCHIVOS .JSP 11](#_Toc119317939)

[CODIFICACIÓN DEL ARCHIVO INDEX.JSP 11](#_Toc119317940)

[CODIFICACIÓN DEL ARCHIVO DATOS ENCUESTA 15](#_Toc119317941)

[CODIFICACIÓN DEL ARCHIVO SERVLET CONTROLLER 19](#_Toc119317942)

[CODIFICACIÓN DEL ARCHIVO SALIDA.JSP 23](#_Toc119317943)

[EJECUCIÓN DEL PROYECTO 27](#_Toc119317944)

[CONCLUSIONES 28](#_Toc119317945)

[RECOMENDACIONES 28](#_Toc119317946)

[BIBLIOGRAFÍA 29](#_Toc119317947)

**INDICE DE IMÁGENES**

[Figura 1. Logotipo identificativo de Java Enterprise Edition 3](#_Toc119317814)

[Figura 2. Logotipo identificativo de Java Server Pages 4](#_Toc119317815)

[Figura 3. Patrón Modelo Vista Controlador (MVC) 6](#_Toc119317816)

[Figura 4. Creación de carpetas para él proyecto. 7](#_Toc119317817)

[Figura 5. Logotipo identificativo de Apache NetBeans 15. 7](#_Toc119317818)

[Figura 6. Creación de un nuevo proyecto. 7](#_Toc119317819)

[Figura 7. Selección del tipo de aplicación a desarrollar 8](#_Toc119317820)

[Figura 8. Selección del nombre y directorio donde guardar el proyecto 8](#_Toc119317821)

[Figura 9. Selección del servidor y versión de Java EE. 9](#_Toc119317822)

[Figura 10. Selección del Frameworks y finalización de la creación del proyecto. 9](#_Toc119317823)

[Figura 11. Estructura del proyecto. 9](#_Toc119317824)

[Figura 12. Eliminación del archivo index.html. 10](#_Toc119317825)

[Figura 13. Creación del paquete controlador. 10](#_Toc119317826)

[Figura 14. Creación del paquete modelo. 10](#_Toc119317827)

[Figura 15. Selección de un archivo del tipo JSP 11](#_Toc119317828)

[Figura 16. Asignación del nombre al archivo a crear. 11](#_Toc119317829)

[Figura 17. Codificación del archivo index.jsp (Formulario) 12](#_Toc119317830)

[Figura 18. Creación del archivo DatosEncuesta.java 15](#_Toc119317831)

[Figura 19. Asignación de nombre al archivo a crear. 15](#_Toc119317832)

[Figura 20. Codificación del archivo DatosEncuesta (Persistencia de datos en el paquete modelo) 16](#_Toc119317833)

[Figura 21. Creación del archivo ServletController.java 19](#_Toc119317834)

[Figura 22. Asignación de nombre al archivo a crear. 19](#_Toc119317835)

[Figura 23. Codificación del archivo ServletController.java 20](#_Toc119317836)

[Figura 24. Selección de un archivo del tipo JSP 23](#_Toc119317837)

[Figura 25. Asignación del nombre al archivo a crear. 23](#_Toc119317838)

[Figura 26. Codificación del archivo salida.jsp 24](#_Toc119317839)

[Figura 27. Ejecución del proyecto 27](#_Toc119317840)

[Figura 28. Formulario del proyecto. 27](#_Toc119317841)

[Figura 29. Ingreso de Datos en el formulario index.jsp 28](#_Toc119317842)

[Figura 30. Despliegue del formulario salida.jsp 28](#_Toc119317843)

**ÍNDICE DE TABLAS**

[Tabla 1. Codificación del archivo index.jsp 15](#_Toc119318144)

[Tabla 2. Codificación del archivo DatosEncuesta.java 19](#_Toc119318145)

[Tabla 3. Codificación del archivo ServeltController.java 23](#_Toc119318148)

[Tabla 4. Codificación del archivo salida.jsp 27](#_Toc119318150)

# MARCO TEÓRICO

## JAVA WEB



Figura 1. Logotipo identificativo de Java Enterprise Edition

Java Enterprise Edition es una plataforma que se encarga de definir las APIs y los requerimientos necesarios para poder ejecutar aplicaciones Java servidoras las cuales pueden ser: Cliente-Servidor, Multiusuarios, Transaccionalidad, Escalabilidad entre otros, donde tales características no son importantes o indispensables en aplicaciones de escritorio.

La plataforma Java Enterprise Edition se apoya en Java SE (Java Standard Edition) que a diferencia de Java EE se encarga de definir las APIs y los requerimientos necesarios para poder ejecutar aplicaciones de escritorio ya sea en ordenadores personales o portátiles como se requiera.

Java EE tiene un modelo la cual define una arquitectura para poder implementar servicios como aplicaciones multicapas donde se asegura la escalabilidad, accesibilidad, y la facilidad de gestión necesaria para el ámbito empresarial.

Su modelo divide la implementación en dos partes:

1. La lógica de presentación y de negocio a implementar por el desarrollador.
2. Los servicios estándar que ofrece la plataforma Java EE.

Además, Java EE utiliza un modelo de programación distribuido en distintas capas, donde la lógica de la aplicación se divide en distintos componentes los cuales dependen de su funcionalidad donde son desplegados en las distintas capas dependiendo a que capa pertenecen.

Componentes de Java Enterprise Edition

* Componentes clientes: son aplicaciones Java SE (AWT/Swing, Applets) o de un navegador web (Firefox, Chrome, IExplore) y se despliegan en la capa del cliente.
* Componentes web: son Java Servlets, JavaServer Pages (JSP) o JavaServer Faces (JSF) y se despliegan en la capa web.
* Componentes de negocio: Enterprise JavaBeans (EJB) y se despliegan en la capa de negocio.

## JAVA SERVER PAGES (JSP)

Una página JSP (JavaServer Page) es una página HTML a la que se le incrusta código de lenguaje de programación JAVA. El código JAVA se incrusta entre los siguientes indicadores <% y %>.

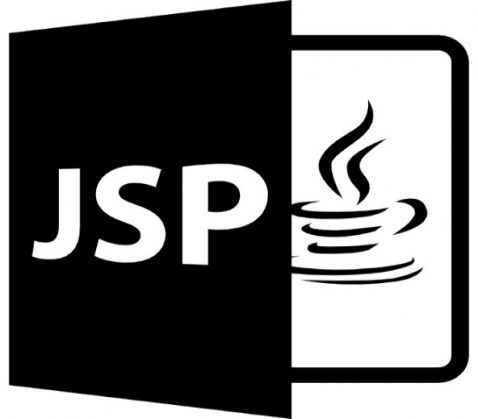


Figura 2. Logotipo identificativo de Java Server Pages

## SERVLETS

Un servlet es una clase Java la cual hereda de la clase (HttpServlet) corre en el servidor. Su palabra se deriva de applet, donde anteriormente un applet era un pequeño programa escrito en lenguaje de programación JAVA y el cuál corría en el navegador del cliente.

La función de un servlet es dar servicio a las peticiones de un navegador web donde recibe una petición, la procesa, y devuelve una respuesta al navegador, al ser escrito en lenguaje JAVA es portable como toda aplicación JAVA lo cual quiere decir que puede funcionar sin necesidad de hacer cambios en diferentes servidores.

Además, en un servlet no solo se puede tener código JAVA también se puede incrustar código HTML.

## JAVABEANS

Un JavaBean también conocido como Bean es un componente el cual se encuentra desarrollado con patrones de diseño bien definidos lo que hace que sea reutilizable y manipulado de manera visual por medio de una herramienta de programación en el lenguaje Java.

Características:

* Introspección: permite analizar a la herramienta de programación o IDE como trabaja en Bean.
* Customización: es la capacidad que tiene el programador de poder manipular y alterar la apariencia y la conducta del Bean.
* Events: se encarga de informar al IDE de aquellos sucesos que se pueden generar en respuesta en base a las acciones del usuario o del sistema. Además, de los sucesos que se pueden mejorar.
* Properties: permite cambiar los valores de las propiedades del Bean con el objetivo de personalizarlo.
* Persistence: tiene la capacidad de guardar el estado de los Beans los cuales han sido personalizados por el programador.

Reglas de los JavaBeans:

* Un Bean debe de tener un constructor por defecto (Constructor sin argumentos).
* Un Bean siempre debe de tener persistencia lo cual quiere decir que se debe de implementar una interfaz Serializable.

## ARQUITECTURA DE SOFTWARE

La arquitectura de software de un sistema es el conjunto de estructuras necesarias para razonar sobre el sistema. Comprende elementos de software, relaciones entre ellos, y propiedades de ambos. (Bass, Clements y Kazman, 2012).

La arquitectura de software es un término el cual surge en los años 60 donde se hace referencia a una planificación basada en modelos, patrones y abstracciones teóricas de tal forma que nos proporciona una visión clara del cómo va a encajar cada una de las piezas de un producto o servicio software.

## PATRÓN MODELO VISTA CONTROLADOR

Es un patrón de arquitectura para el desarrollo de aplicaciones software donde su objetivo es el separar la lógica del negocio de la interfaz de usuario lo cual lo hace escalable facilitando la evolución por separado de sus partes, incrementa la reutilización y flexibilidad.

Tuvo su origen en 1979 y es también conocido como patrón MVC, divide una aplicación software en tres grandes partes bien diferenciadas las cuales son Modelo, Vista y Controlador.

### **MODELO**

Es una capa donde se localiza la funcionalidad central y los datos, se comunica con el controlador y la base de datos.

Es la capa donde se trabaja con los datos, por tanto, contendrá mecanismos para acceder a la información y también para actualizar su estado. Los datos los tendremos habitualmente en una base de datos, por lo que en los modelos tendremos todas las funciones que accederán a las tablas y harán los correspondientes selects, updates, inserts, etc.

No obstante, cabe mencionar que cuando se trabaja con MCV lo habitual también es utilizar otras librerías como PDO o algún ORM como Doctrine, que nos permiten trabajar con abstracción de bases de datos y persistencia en objetos. Por ello, en vez de usar directamente sentencias SQL, que suelen depender del motor de base de datos con el que se esté trabajando, se utiliza un dialecto de acceso a datos basado en clases y objetos.

### **VISTA**

Es la capa donde se muestra la información al usuario donde es posible definir una o más vista de usuario en un software, se comunica con el controlador y el usuario.

Las vistas, como su nombre nos hacen entender, contienen el código de nuestra aplicación que va a producir la visualización de las interfaces de usuario, o sea, el código que nos permitirá renderizar los estados de nuestra aplicación en HTML. En las vistas nada más tenemos los códigos HTML y PHP que nos permite mostrar la salida.

En la vista generalmente trabajamos con los datos, sin embargo, no se realiza un acceso directo a éstos. Las vistas requerirán los datos a los modelos y ellas se generarán la salida, tal como nuestra aplicación requiera.

**CONTROLADOR**  
Es la capa donde se manejan las entradas del usuario se separa la representación interna de la información y la forma en la que se le muestra al usuario, se comunica con el modelo.

Contiene el código necesario para responder a las acciones que se solicitan en la aplicación, como visualizar un elemento, realizar una compra, una búsqueda de información, etc.

En realidad, es una capa que sirve de enlace entre las vistas y los modelos, respondiendo a los mecanismos que puedan requerirse para implementar las necesidades de nuestra aplicación. Sin embargo, su responsabilidad no es manipular directamente datos, ni mostrar ningún tipo de salida, sino servir de enlace entre los modelos y las vistas para implementar las diversas necesidades del desarrollo.

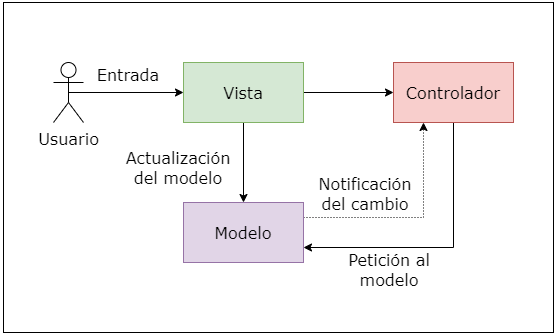


Figura 3. Patrón Modelo Vista Controlador (MVC)

## MVC EN APLICACIONES WEB

El patrón de arquitectura MVC puede ser aplicado en el desarrollo de cualquier software que lo requiera en este caso mencionaremos el desarrollo de aplicaciones web, donde:

* Vista: Páginas HTML
* Controlador: Código que obtiene datos dinámicamente y genera el contenido HTML.
* Modelo: Información almacenada en una base de datos.

# CREACIÓN DEL PROYECTO

## OBJETIVO

Realizar un formulario WEB utilizando la arquitectura MVC (Modelo Vista Controlador) donde se apliquen los conceptos revisados previamente, utilizando como entorno de desarrollo Apache Netbeans 15.

## CREAR UN SITIO WEB LLAMADO FORMULARIOWEBSINMVC

## \_APELLIDO\_ NOMBRE

Para poder comenzar con el desarrollo del proyecto se crean las carpetas donde se almacenará el mismo el cual tiene la siguiente estructura en el **Disco Local C** se crea una carpeta llamada **FORMULARIOS APELLIDO** donde dentro de la carpeta se crea una nueva carpeta con el siguiente nombre **03\_WEB\_CON\_MVC\_JAVA\_APELLIDO**, por último, se crea dos carpetas donde la primera se llamada **APLICATIVO** y **DOCUMENTACIÓN** respectivamente como se muestra en la siguiente imagen.

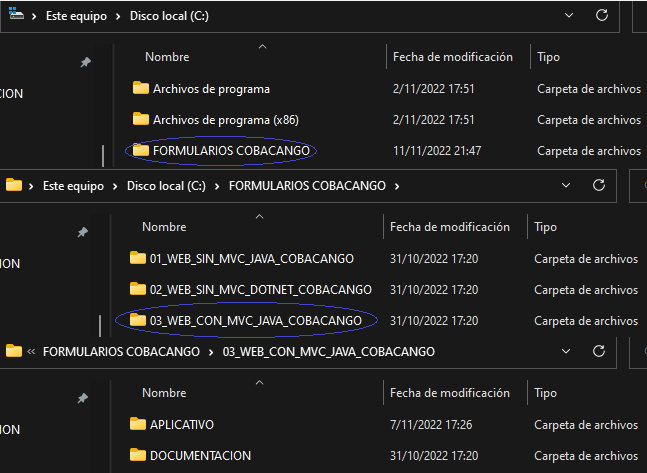


Figura 4. Creación de carpetas para él proyecto.

Una vez creados los directorios se procede a abrir el entorno de desarrollo el cual es Apache Netbeans 15.



Figura 5. Logotipo identificativo de Apache NetBeans 15.

Una vez abierto el entorno de desarrollo (Apache Netbeans 15) se deberá hacer clic en el apartado File donde seguido de esto se selecciona New Project como se muestra en la siguiente figura.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 6. Creación de un nuevo proyecto.

En la siguiente pantalla se selecciona Java Web seguido de eso Web Application como se muestra en la siguiente figura.

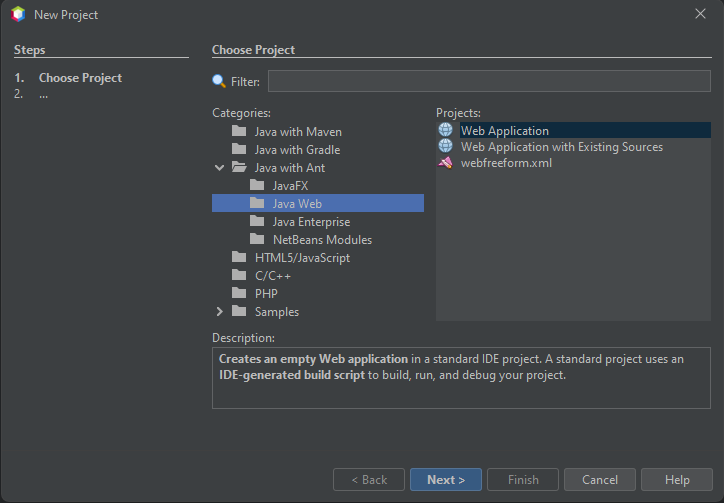


Figura 7. Selección del tipo de aplicación a desarrollar

Se procede a dar el siguiente nombre al proyecto WebConMVCJavaApellidoNombre y se elige la ubicación que se creó anteriormente.

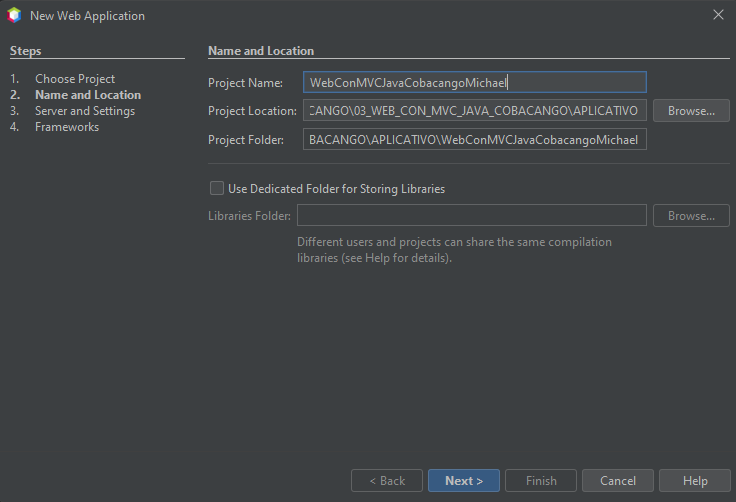


Figura 8. Selección del nombre y directorio donde guardar el proyecto

En la siguiente pantalla se selecciona el Servidor en este caso es Payara al igual que Java EE Version que es la última de Jakarta EE 8 Web como se muestra en la siguiente figura.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 9. Selección del servidor y versión de Java EE.

A continuación, no se deberá seleccionar ningún Frameworks puesto que para la práctica no son requeridos. Finalmente se da clic en Finish.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 10. Selección del Frameworks y finalización de la creación del proyecto.

Una vez finalizada la creación del proyecto obtenemos la siguiente estructura

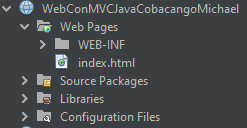


Figura 11. Estructura del proyecto.

## DESARROLLO DEL PATRÓN MVC

### **CREACIÓN DE LOS PAQUETES Y REESTRUCTURACIÓN DEL PROYECTO**

Para poder comenzar con el desarrollo y la codificación de los archivos se procede a eliminar el archivo index.html el cuál se crea de manera automática al momento de crear el proyecto, de igual forma se reestructura el proyecto creando los paquetes para el patrón de arquitectura MVC.

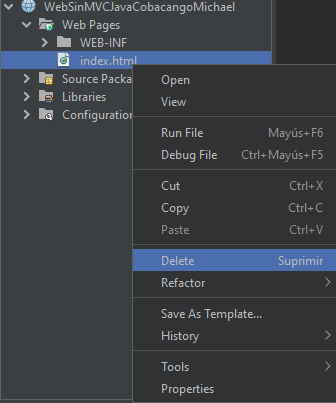


Figura 12. Eliminación del archivo index.html.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Figura 13. Creación del paquete controlador.

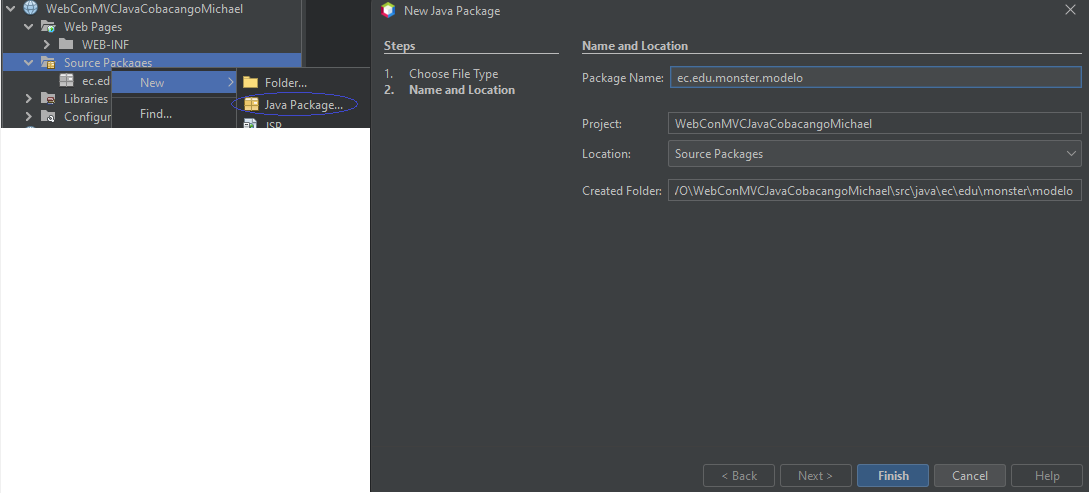


Figura 14. Creación del paquete modelo.

## CODIFICACIÓN DE LOS ARCHIVOS .JSP

### **CODIFICACIÓN DEL ARCHIVO INDEX.JSP**

Seguido de eso se deberá dar clic derecho sobre el apartado Web Pages seleccionar New y elegir JSP como se muestra en la siguiente figura.

Captura de pantalla de computadora

Descripción generada automáticamente

Figura 15. Selección de un archivo del tipo JSP

Se indica el nombre del archivo el cual es index como se muestra en la siguiente figura.

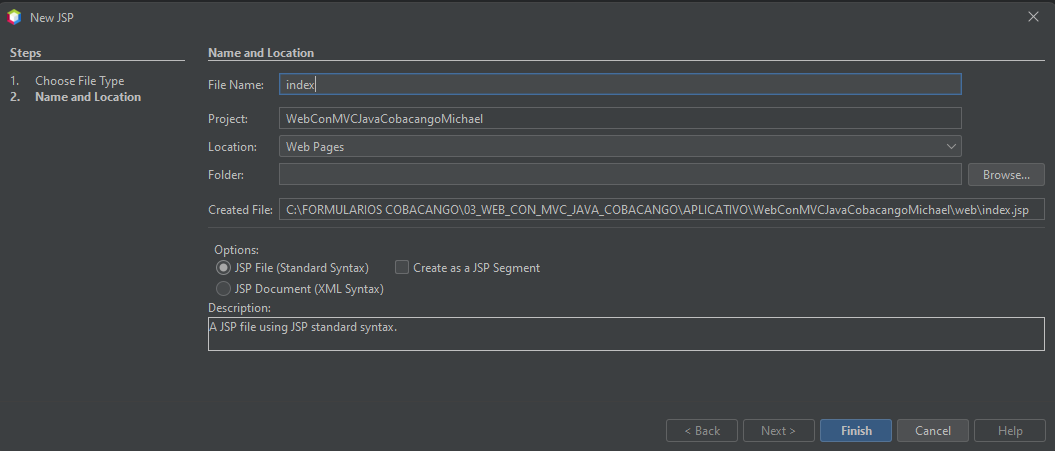


Figura 16. Asignación del nombre al archivo a crear.

Una vez creado el archivo se procede a codificar el formulario que se mostrará al momento de ejecutar el proyecto.

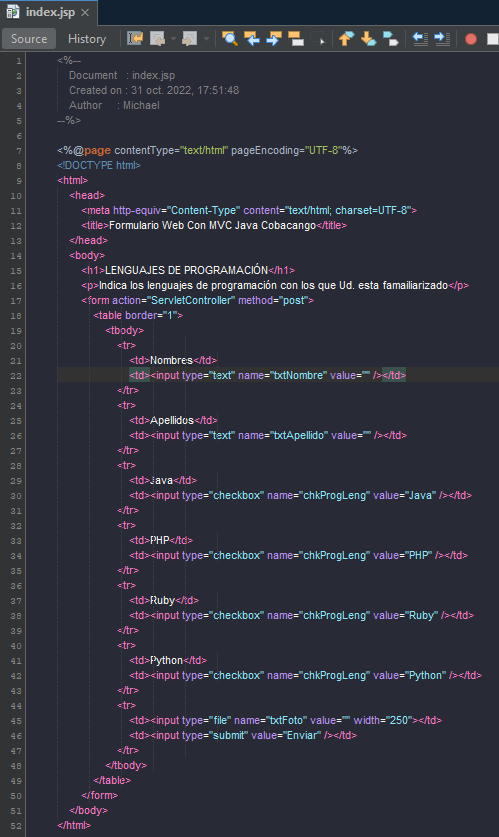


Figura 17. Codificación del archivo index.jsp (Formulario)

Tabla 1. Codificación del archivo index.jsp

<%-- Document : index.jsp Created on : 31 oct. 2022, 17:51:48 Author : Michael

--%> <%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

  <head>

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

    <title>Formulario Web Con MVC Java Cobacango</title>

  </head>

  <body>

    <h1>LENGUAJES DE PROGRAMACIÓN</h1>

    <p>

      Indica los lenguajes de programación con los que Ud. esta famailiarizado

    </p>

    <form action="salida.jsp">

      <table border="1">

        <tbody>

          <tr>

            <td>Nombres</td>

            <td><input type="text" name="txtNombre" value="" /></td>

          </tr>

          <tr>

            <td>Apellidos</td>

            <td><input type="text" name="txtApellido" value="" /></td>

          </tr>

          <tr>

            <td>Java</td>

            <td><input type="checkbox" name="chkProgLeng" value="Java" /></td>

          </tr>

          <tr>

<tr>

            <td>PHP</td>

            <td><input type="checkbox" name="chkProgLeng" value="PHP" /></td>

          </tr>

          <tr>

            <td>Ruby</td>

            <td><input type="checkbox" name="chkProgLeng" value="Ruby" /></td>

          </tr>

          <tr>

            <td>Python</td>

            <td><input type="checkbox" name="chkProgLeng" value="Python" /></td>

          </tr>

          <tr>

            <td>

              <input type="file" name="imgProgramador" value="" width="250" />

            </td>

            <td><input type="submit" value="Enviar" /></td>

          </tr>

        </tbody>

      </table>

    </form>

  </body>

</html>

### **CODIFICACIÓN DEL ARCHIVO DATOS ENCUESTA**

Para poder codificar el archivo DatosEncuesta.java se deberá crear una clase JAVA en el paquete modelo creado anteriormente.

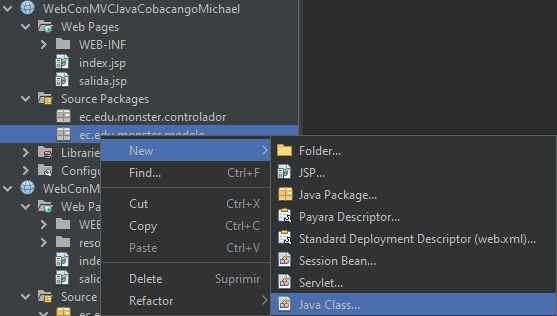


Figura 18. Creación del archivo DatosEncuesta.java

Seguido de eso se le asigna un nombre al archivo y configuración como ubicación.

Interfaz de usuario gráfica

Descripción generada automáticamente

Figura 19. Asignación de nombre al archivo a crear.

Una vez creado el archivo se procede a la codificación como se muestra en la siguiente figura.

Texto

Descripción generada automáticamente

Figura 20. Codificación del archivo DatosEncuesta (Persistencia de datos en el paquete modelo)

Para poder comprender de mejor manera el código anterior se define brevemente en términos generales lo siguiente:

**Private:** cuando se define un atributo o un método con el acceso de “Private” quiere decir que se tiene acceso únicamente desde la clase donde fue definido.

**String:** es un tipo de dato donde se almacena un arreglo de caracteres.

**DatosEncuesta:** es el método constructor de la clase, nos sirve para instanciar objetos, el constructor lleva el mismo nombre de la clase, y puede tener sobrecargas.

**Getters and Setters:** son métodos que se le asignan a los atributos de la clase, los métodos getters nos sirven para obtener aquel atributo y el método set nos permite modificar el valor de los atributos.

Tabla 2. Codificación del archivo DatosEncuesta.java

/\*

 \* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

 \* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

 \*/

package ec.edu.monster.modelo;

/\*\*

 \*

 \* @author Michael

 \*/

public class DatosEncuesta {

    private String txtNombre;

    private String txtApellido;

    private String txtFoto;

    private String[] chkProgLeng;

    public DatosEncuesta() {

    }

    public String getTxtNombre() {

        return txtNombre;

    }

    public void setTxtNombre(String txtNombre) {

        this.txtNombre = txtNombre;

    }

    public String getTxtApellido() {

        return txtApellido;

    }

   public void setTxtApellido(String txtApellido) {

        this.txtApellido = txtApellido;

    }

    public String getTxtFoto() {

        return txtFoto;

    }

    public void setTxtFoto(String txtFoto) {

        this.txtFoto = txtFoto;

    }

    public String[] getChkProgLeng() {

        return chkProgLeng;

    }

    public void setChkProgLeng(String[] chkProgLeng) {

        this.chkProgLeng = chkProgLeng;

    }

}

### **CODIFICACIÓN DEL ARCHIVO SERVLET CONTROLLER**

Para poder codificar el archivo ServletController.java se deberá crear una clase SERVLET en el paquete controlador creado anteriormente.

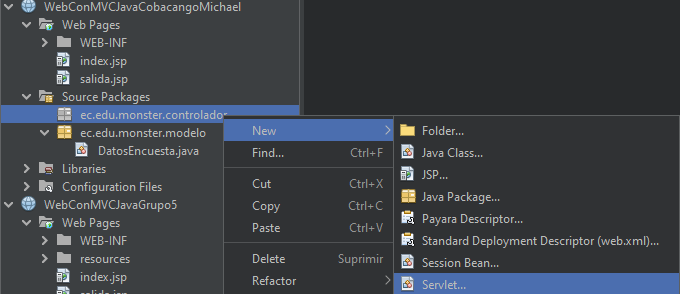


Figura 21. Creación del archivo ServletController.java

Seguido de eso se le asigna un nombre al archivo y configuración como ubicación.

Texto

Descripción generada automáticamente

Figura 22. Asignación de nombre al archivo a crear.

Una vez creado el archivo se procede a la codificación como se muestra en la siguiente figura.

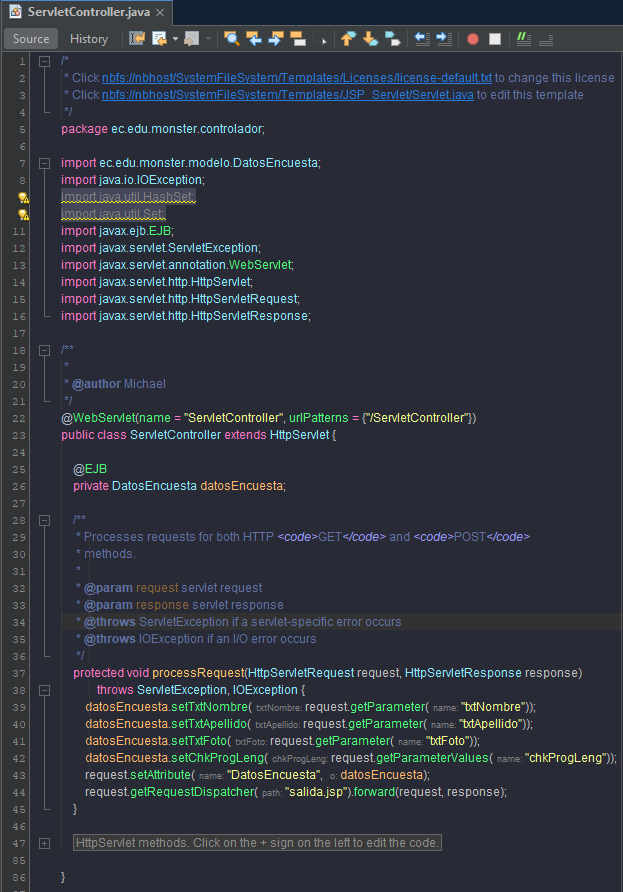


Figura 23. Codificación del archivo ServletController.java

Al igual que el archivo anterior se van a definir algunos de los términos más importantes que se manejan en ServletController.java para poder comprender de manera más clara el código:

**Get:** es un método público el cual se encarga de mostrar un valor a una propiedad o atributo de un objeto.

**Post:** es un método público la cual se encarga de enviar la petición utilizando los objetos URL y HttpURL Connection.

**Request:** es una petición HTTP la cual se conoce como una línea de petición, contiene tres campos: un método HTTP, un identificador universal de recursos (URI), y una versión del protocolo HTTP.

**Response:** se da cuando el servidor recibe y procesa la solicitud, devuelve una respuesta HTTP hacia el cliente.

Tabla 3. Codificación del archivo ServeltController.java

/\*

 \* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

 \* Click nbfs://nbhost/SystemFileSystem/Templates/JSP\_Servlet/Servlet.java to edit this template

 \*/

package ec.edu.monster.controlador;

import ec.edu.monster.modelo.DatosEncuesta;

import java.io.IOException;

import java.util.HashSet;

import java.util.Set;

import javax.ejb.EJB;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

 \*

 \* @author Michael

 \*/

@WebServlet(name = "ServletController", urlPatterns = {"/ServletController"})

public class ServletController extends HttpServlet {

    @EJB

    private DatosEncuesta datosEncuesta;

     \* @param request servlet request

     \* @param response servlet response

 \* @throws ServletException if a servlet-specific error occurs

     \* @throws IOException      if an I/O error occurs

     \*/

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)

            throws ServletException, IOException {

        datosEncuesta.setTxtNombre(request.getParameter("txtNombre"));

        datosEncuesta.setTxtApellido(request.getParameter("txtApellido"));

        datosEncuesta.setTxtFoto(request.getParameter("txtFoto"));

        datosEncuesta.setChkProgLeng(request.getParameterValues("chkProgLeng"));

        request.setAttribute("DatosEncuesta", datosEncuesta);

        request.getRequestDispatcher("salida.jsp").forward(request, response);

    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the

    // + sign on the left to edit the code.">

    /\*\*

     \* Handles the HTTP <code>GET</code> method.

     \*

     \* @param request  servlet request

     \* @param response servlet response

     \* @throws ServletException if a servlet-specific error occurs

     \* @throws IOException      if an I/O error occurs

     \*/

    @Override

    protected void doGet(HttpServletRequest request, HttpServletResponse response)

            throws ServletException, IOException {

    }

 /\*\*

     \* Handles the HTTP <code>POST</code> method.

     \*

     \* @param request  servlet request

     \* @param response servlet response

     \* @throws ServletException if a servlet-specific error occurs

     \* @throws IOException      if an I/O error occurs

     \*/

    @Override

    protected void doPost(HttpServletRequest request, HttpServletResponse response)

            throws ServletException, IOException {

        processRequest(request, response);

    }

    /\*\*

     \* Returns a short description of the servlet.

     \*

     \* @return a String containing servlet description

     \*/

    @Override

    public String getServletInfo() {

        return "Short description";

    }// </editor-fold>

}

### **CODIFICACIÓN DEL ARCHIVO SALIDA.JSP**

Se deberá dar clic derecho sobre el apartado Web Pages seleccionar New y elegir JSP como se muestra en la siguiente figura.

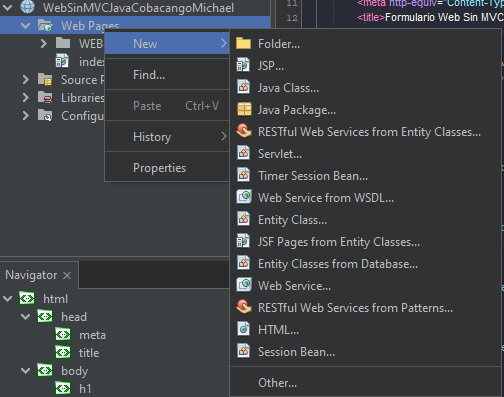


Figura 24. Selección de un archivo del tipo JSP

Se indica el nombre del archivo el cual es salida como se muestra en la siguiente figura.

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

Figura 25. Asignación del nombre al archivo a crear.

Una vez creado el archivo se procede a codificar el archivo de salida el cuál recibirá los datos del formulario para poder mostrarlos al cliente.

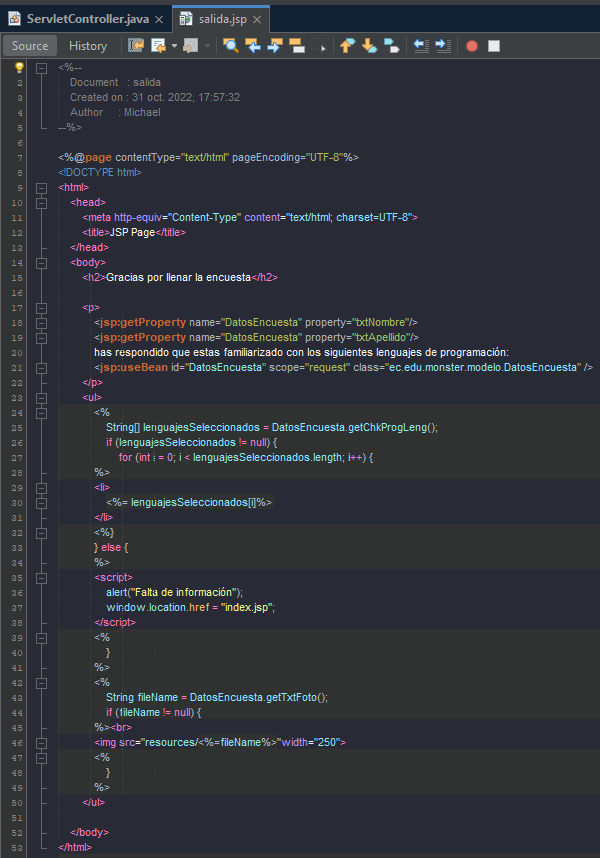


Figura 26. Codificación del archivo salida.jsp

Tabla 4. Codificación del archivo salida.jsp

<%--

    Document   : salida

    Created on : 31 oct. 2022, 17:57:32

    Author     : Michael

--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

    <head>

        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

        <title>JSP Page</title>

    </head>

    <body>

        <h2>Gracias por llenar la encuesta</h2>

        <p>

            <jsp:getProperty name="DatosEncuesta" property="txtNombre"/>

            <jsp:getProperty name="DatosEncuesta" property="txtApellido"/>

            has respondido que estas familiarizado con los siguientes lenguajes de programación:

            <jsp:useBean id="DatosEncuesta" scope="request" class="ec.edu.monster.modelo.DatosEncuesta" />

        </p>

        <ul>

            <%

                String[] lenguajesSeleccionados = DatosEncuesta.getChkProgLeng();

                if (lenguajesSeleccionados != null) {

                    for (int i = 0; i < lenguajesSeleccionados.length; i++) {

            %>

 <li>

                <%= lenguajesSeleccionados[i]%>

            </li>

            <%}

            } else {

            %>

            <script>

                alert("Falta de información");

                window.location.href = "index.jsp";

            </script>

            <%

                }

            %>

            <%

                String fileName = DatosEncuesta.getTxtFoto();

                if (fileName != null) {

            %><br>

            <img src="resources/<%=fileName%>"width="250">

            <%

                }

            %>

        </ul>

    </body>

</html>

## EJECUCIÓN DEL PROYECTO

Para poder ejecutar el proyecto se deberá seleccionar la opción de Run Project como se muestra en la siguiente figura.

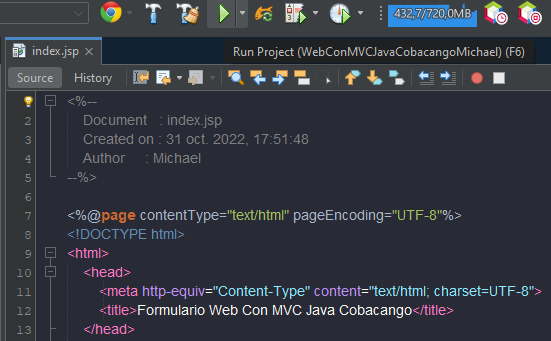


Figura 27. Ejecución del proyecto

Al ejecutar el proyecto se mostrará la siguiente aplicación donde se tiene el formulario que fue codificado en el archivo index.jsp

Imagen que contiene Tabla

Descripción generada automáticamente

Figura 28. Formulario del proyecto.

A continuación, se procede a llenar los campos solicitados y se hace clic en enviar.

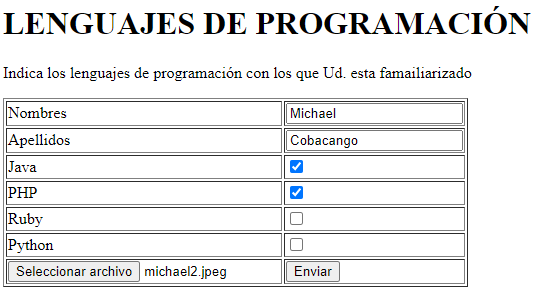


Figura 29. Ingreso de Datos en el formulario index.jsp

Una vez enviado los datos se mostrará la siguiente pantalla con la información ingresada.

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

Figura 30. Despliegue del formulario salida.jsp

## CONCLUSIONES

* Las aplicaciones Java EE se encuentran distribuidas en distintas capas que trabajan en conjunto para resolver una necesidad en común, esto garantiza una mejor organización funcional. En el patrón MVC la capa vista se encarga de mostrar la información del sistema al usuario, la capa controladora permite gestionar la lógica del negocio, y la capa del modelo contiene la información del negocio.
* Se logró desarrollar un proyecto web utilizando el entorno de desarrollo Apache NetBeans 15, donde se implementó un formulario y la salida de información ingresada aplicando los conceptos previos a la práctica, el patrón MVC con JSP, Servlets, y EJB.
* El IDE Apache NetBeans facilita ciertos procesos repetitivos al desarrollar proyectos con JAVA EE, dispone de una gran cantidad de ventanas en donde se llenan los campos ya definidos para definir la configuración o la implementación de ciertas partes del proyecto, al final generándose el código necesario para su ejecución.

## RECOMENDACIONES

* Es recomendable realizar una introducción o revisión previa de los conceptos de JAVA EE como son JSP, Servlets, EJB, y tener un conocimiento básico de tecnologías como HTML, CSS y JavaScript para realizar este tipo de prácticas.
* La plataforma de desarrollo JAVA EE lanza de forma constante actualizaciones de sus herramientas, debido a esto el entorno de desarrollo puede verse afectado, pues ciertos procesos y pasos para realizar ciertas configuraciones pueden cambiar.

## BIBLIOGRAFÍA

* Java EE Specifications <http://www.oracle.com/technetwork/java/javaee/tech/index.html>
* Building Web Apps in Java: Beginning & Intermediate Servlet & JSP Tutorials. <http://courses.coreservlets.com/Course-Materials/csajsp2.html>
* Bryan Basham, Kathy Sierra & Bert Bates: “Head First Servlets and JSP: Passing the Sun Certified Web Component Developer Exam” O’Reilly, 2008. ISBN 0596516681.
* Richard Monson-Haefel: Enterprise JavaBeans, 3rd Edition. Ed. O’Reilly, September 2001.
* Len Bass, Paul Clements and Rick Kazman. Software Architecture in Practice. Edited by SEI Series In Software Engineering. second ed: Addison Wesley, 2003.
* Paul Clements, Rick Kazman and Mark Klein. Evaluating Software Architectures, Methods and case studies. Edited by SEI Series In Software Engineering. Addison Wesley, 2002.
* Pavón, Juan. (2009) Estructura de las Aplicaciones Orientadas a Objetos. El Patrón Modelo-Vista-Controlador (MVC). Universidad Complutense Madrid
* Sommerville, Ian (2005). Ingeniera del Software. Pearson Educación.