# Instalación de componentes

## Instalación de componentes.

Para generar de manera practica una solución web con java utilizaremos el ambiente de desarrollo NetBeans en su versión 12.0 ya que es una de las ultimas actualizaciones más estables a la fecha; esta versión nos permite incorporar y desplegar soluciones web básicas y con componentes complejos.

Este ambiente de desarrollo nos permite desplegar nuestras soluciones en algunos servidores conocidos tal es el caso de Apache Tomcat en su versión 10.0.7, que es uno de los servidores de despliegue más nombrados entre desarrolladores.

Del mismo modo utilizaremos a MySQL como sistema gestor de bases de datos ya con el almacenaremos de forma sistemática la información utilizada por el sistema que se estará definiendo en ejercicios posteriores y junto con este utilizaremos un conector JDBC que nos permitirá la fluidez de la información con código Java que estaremos proporcionando.

Otros elementos importantes son los frameworks como Struts que nos permitirá controlar la información desde las vistas del sistema hasta él envió de la información a la base de datos.

Estos son algunos componentes básicos que un Sistema Web utiliza para ser desarrollado de manera correcta.

## Instalación del ambiente de desarrollo NetBeans 12.0 apache

Para iniciar con la instalación de NetBeans 12.0 requerimos ir a la siguiente dirección: <https://netbeans.apache.org/download/index.html> lo cual nos brindara las opciones generales de descarga, para el cual utilizaremos la última versión estable del IDE denominado **Apache NetBeans 12 LTS (NB 12.0).**

Para continuar con la instalación utilizaremos el instalador que nos arroja la dirección web antes mencionada.

Al abrir el archivo mostrará la imagen 1.0.1 que nos especificará que es lo que se instalará en nuestro equipo de cómputo, para continuar dará clic en el botón Next/Siguiente.

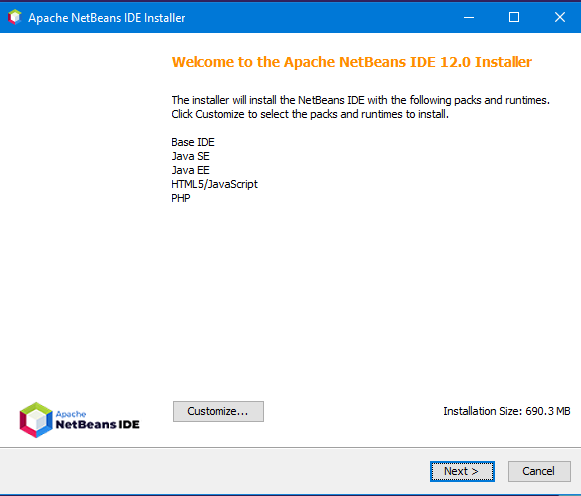


Imagen 1.0.1 Instalación de NetBeans 12.0

Para continuar con la instalación se mostrará la imagen 1.0.2 donde se debe considerar los términos y condiciones aceptando y dando clic en Next/Siguiente.

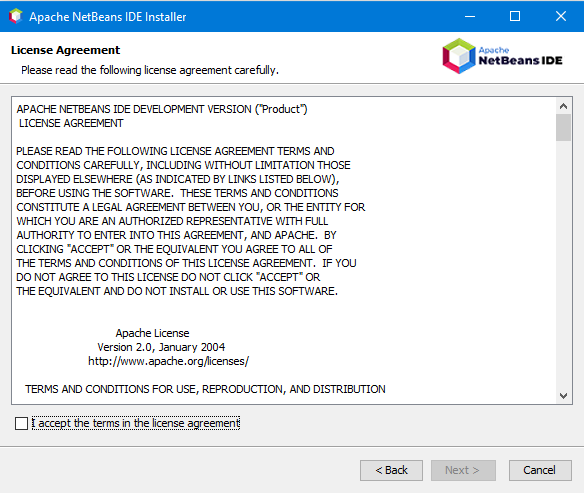


Imagen 1.0.2 Aceptación de términos y condiciones de Instalación de NetBeans 12.0.

Para finalizar con la instalación básica del ambiente de desarrollo de NetBeans 12.0 se debe hacer clic en Install/Instalar, en este punto se instalarán los componentes básicos como se muestra en la imagen 1.0.3.

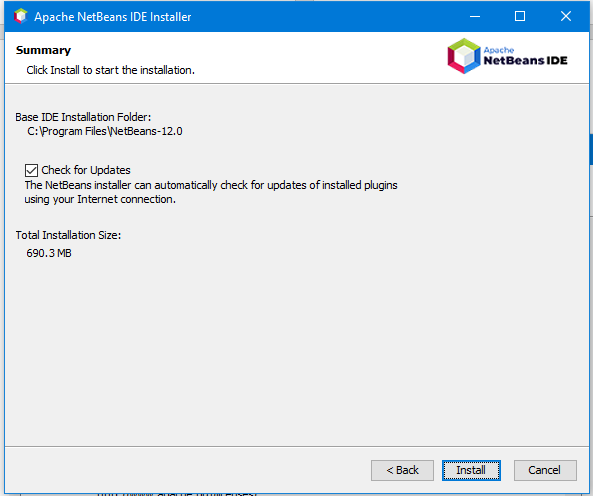


Imagen 1.0.3 Instalación de NetBeans 12.0.

Al terminar la instalación si no hay ningún problema mostrará la imagen 1.0.4, esto nos confirmará que ya podemos iniciar nuestro IDE NetBeans 12.0 de manera correcta.

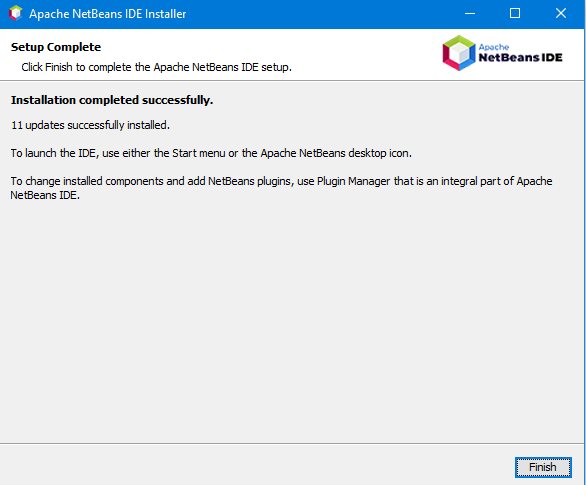


Imagen 1.0.4 Finalización de la Instalación de NetBeans 12.0.

## Instalación del MySQL server

Para realizar una correcta instalación de MySQL quien será nuestro motor de base de datos, primeramente, se requiere ir a la siguiente dirección web: <https://dev.mysql.com/downloads/mysql/> y seleccionar nuestra plataforma en base a nuestro sistema operativo, por ejemplo, utilizaremos la versión del sistema operativo Windows para ello tendremos que seleccionar Microsoft Windows como se muestra en la imagen 1.1.0, posteriormente requerimos dar clic en *Go to Download Page*, utilizaremos la versión que nos muestra la imagen 1.1.0.1 donde nos descargará un archivo ejecutable; para iniciar con la instalación de los componentes básicos que estaremos utilizando.

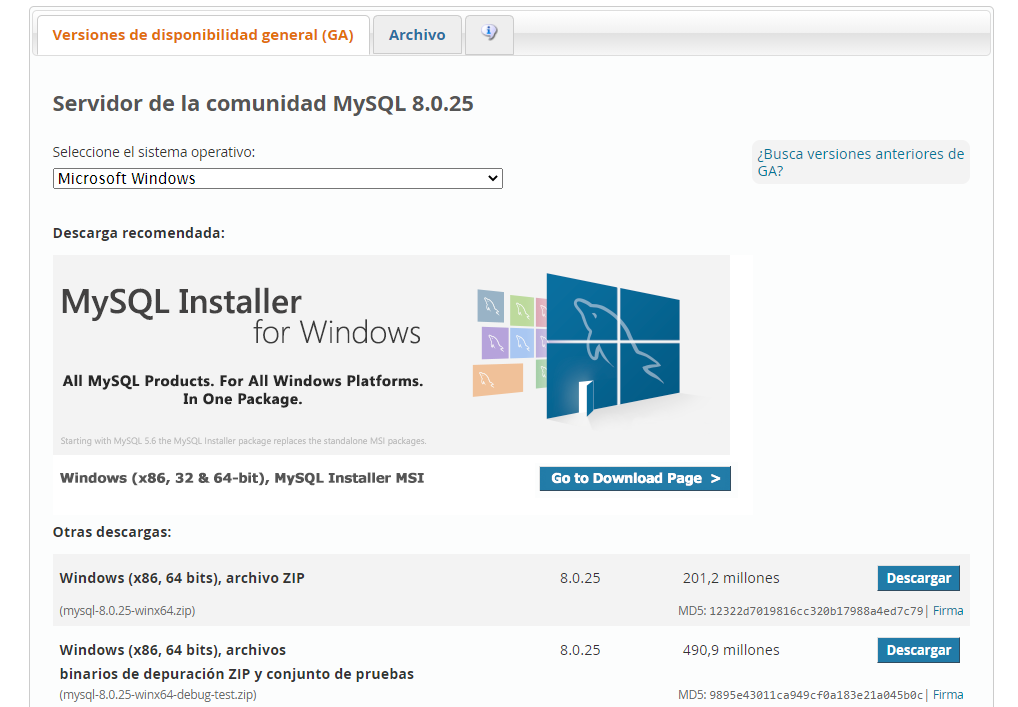


Imagen 1.1.0 Pagina de descarga de componentes MySQL.



Imagen 1.1.0.1 Versión de descarga de MySQL.

Cuando iniciemos el instalador generado por la descripción anterior iniciara el programa como la imagen 1.1.1 en donde nos permitirá agregar a nuestro equipo de computo el servidor de base de datos, el espacio de trabajo grafico denominado en el instalador como MySQL Workbench el cual será de gran apoyo para generar gráficamente nuestro modelo de base de datos, asi como llevar de manera mas sencilla la administración de las múltiples bases de datos que sean requeridas en nuestro sistema que será propuesto en bloques posteriores a este.

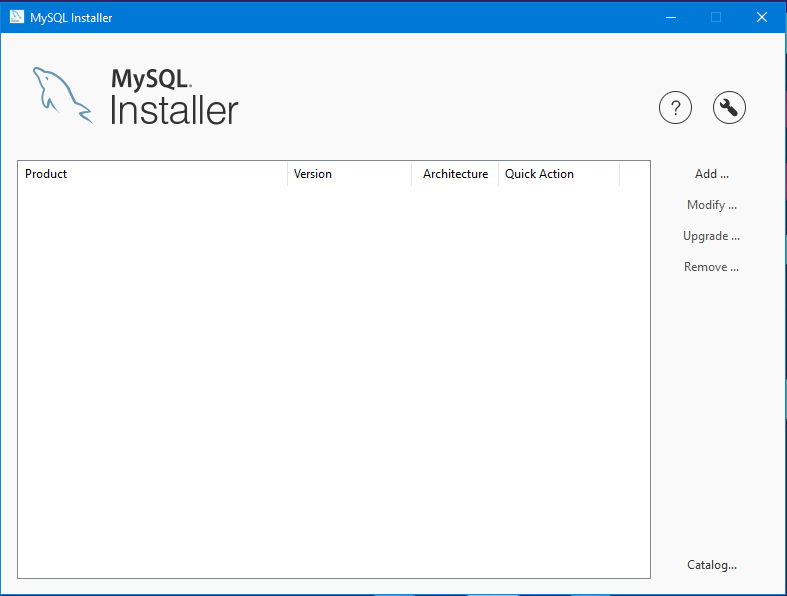


Imagen 1.1.1 Instalación de componentes MySQL.

Para agregar componentes en la imagen 1.1.1 tendremos que dar clic en Add/Agregar para que nos solicite el tipo de instalación, para una instalación básica de componentes seleccionaremos la opción **Developer Default** y dar Clic en Next/Siguiente para continuar con la instalación.

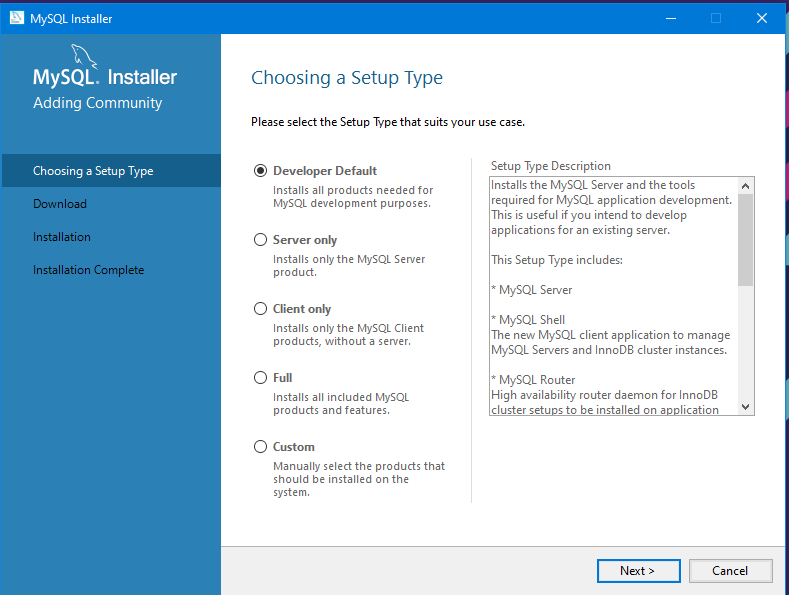


Imagen 1.1.2 Selección del tipo de instalación de MySQL.

En este apartado seleccionaremos la ruta de instalación de nuestro equipo del servidor MySQL y MySQL Workbench, para continuar daremos clic en Next/Siguiente como se muestra en la imagen 1.1.3.

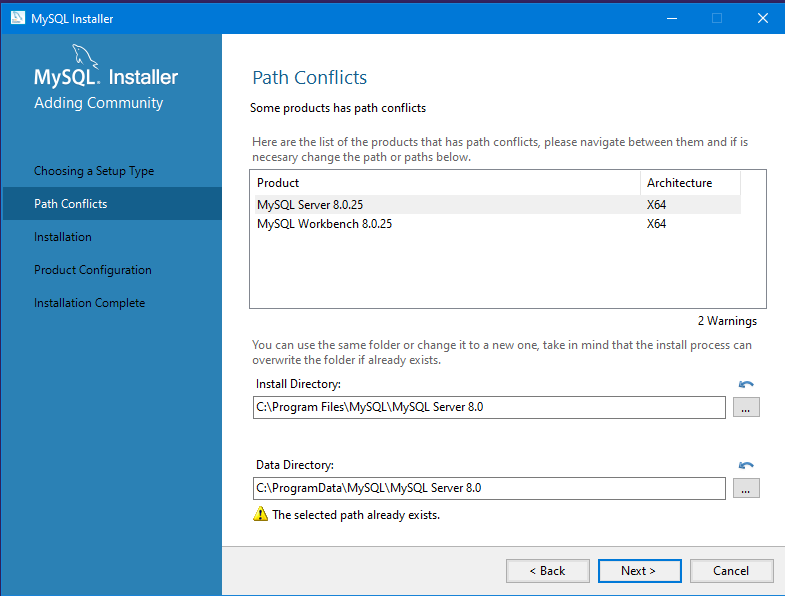


Imagen 1.1.3 Selección de ruta de instalación.

En esta parte daremos clic en Execute/Ejecutar para instalar los componentes, además en esta parte nos brindara conectores para el desarrollo como .Net u ODBC para la unión entre sistemas y bases de datos como se muestra en la imagen 1.1.4.

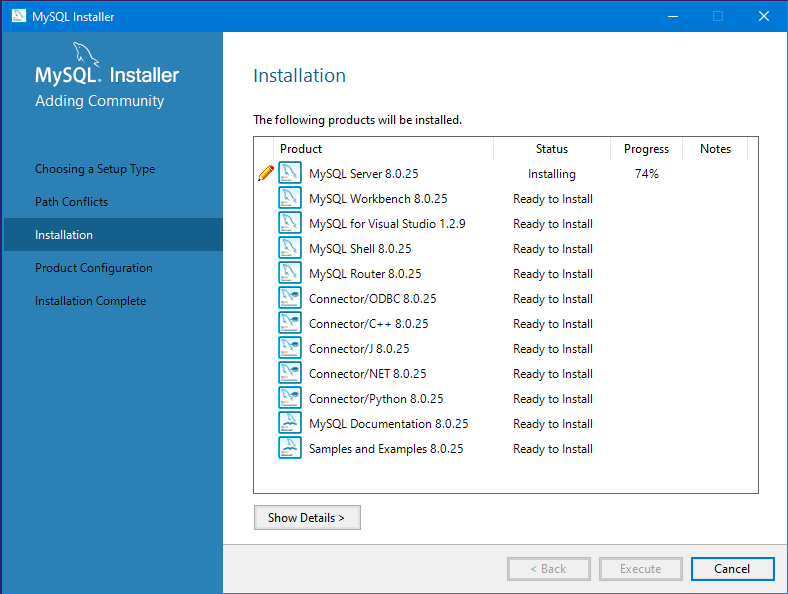


Imagen 1.1.4 Instalación final de componentes.

En la imagen 1.1.5 nos muestra que la instalación se ha completado con normalidad para iniciar la configuración de nuestro servidor.

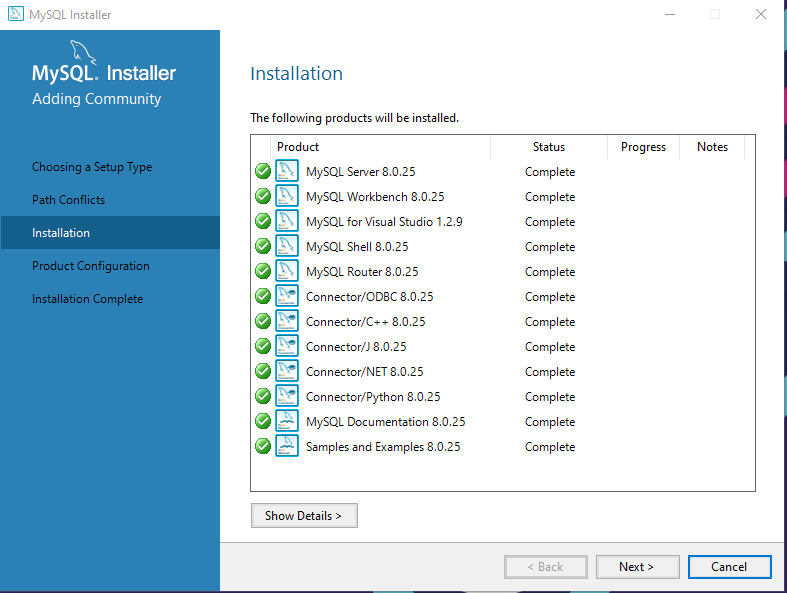


Imagen 1.1.5 Instalación de componentes finalizado.

En este apardado podemos configurar el puerto por la cual el servidor nos conectara con las distintas bases de datos, tambien nos permitira nombrar esa conexión, la recomendación si no hay inconvenientes es dejar los valores por default brindados por el instalador como se muesta en la imagen 1.1.6.

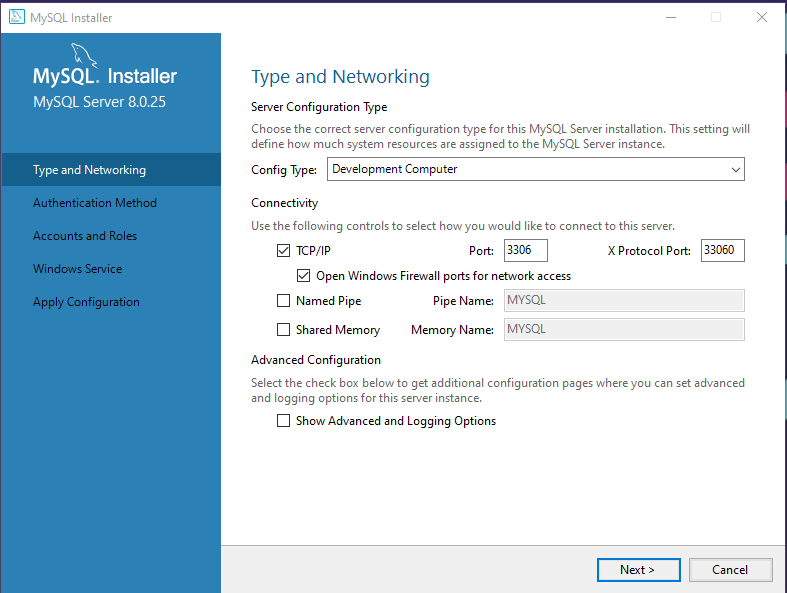


Imagen 1.1.6 Configuración de puertos de red.

En el siguiente aparado podemos configurar los métodos de autenticación, por default el instalador nos permite tener una encriptación fuerte, la cual es recomendada para la seguridad de las bases de datos como se muestra en la imagen 1.1.7, al terminar daremos clic en **Next/Siguiente** para ingresar la contraseña del usuario ROOT que configura el instalador, para ello en la imagen 1.1.8 ingresaremos la contraseña ya que no viene una predeterminada y al dar clic en el botón de **Check** nos verificara si nuestra contraseña cumple con los requisitos mínimos.

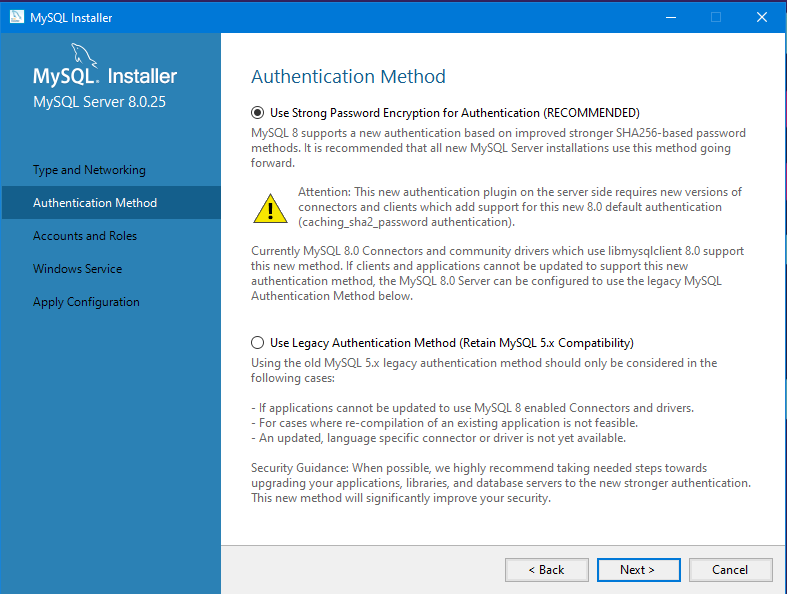


Imagen 1.1.7 Selección de tipo de autenticación.

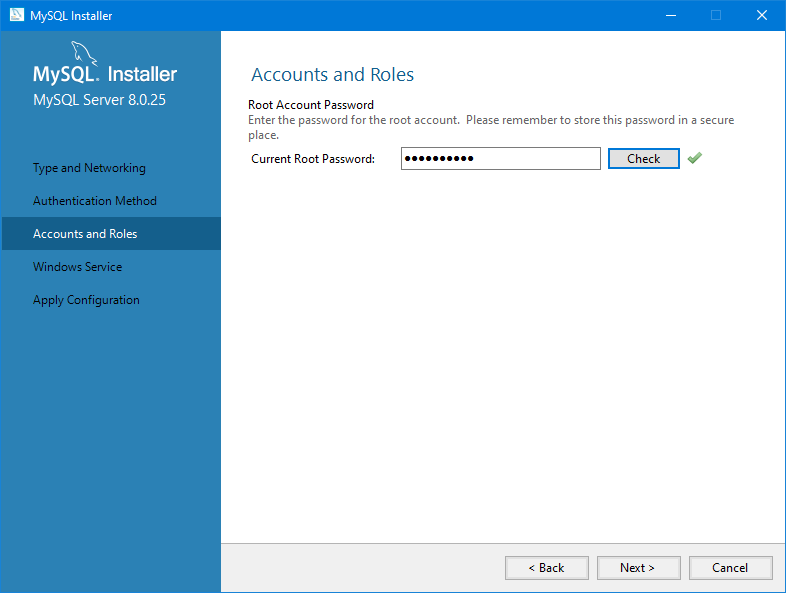


Imagen 1.1.8 Ingreso y verificación de contraseña del usuario ROOT.

En la siguiente imagen 1.1.9 nos permitirá especificar el nombre de nuestro servicio y levantar el servidor, para continuar con la verificación y escritura de la configuración que hemos estado realizando en anteriores explicaciones, para terminar en la imagen 1.1.10

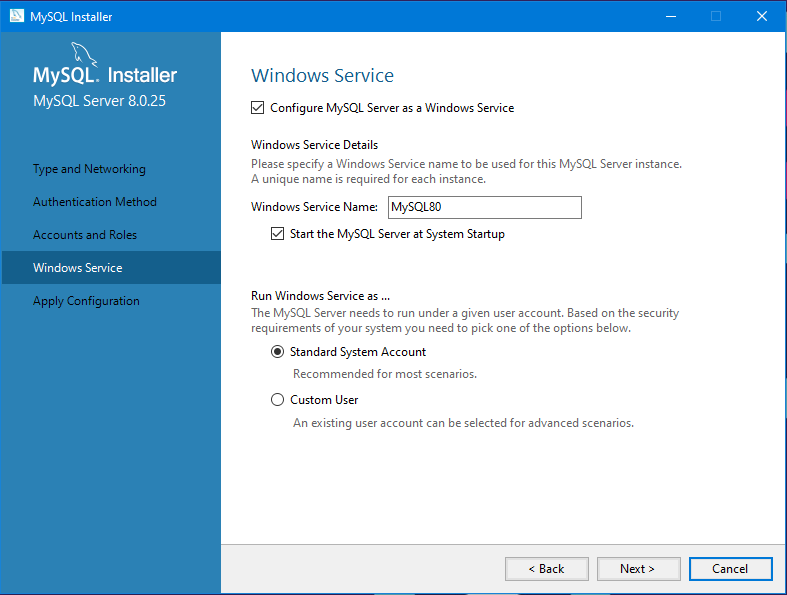


Imagen 1.1.9 Nombramiento del servicio.

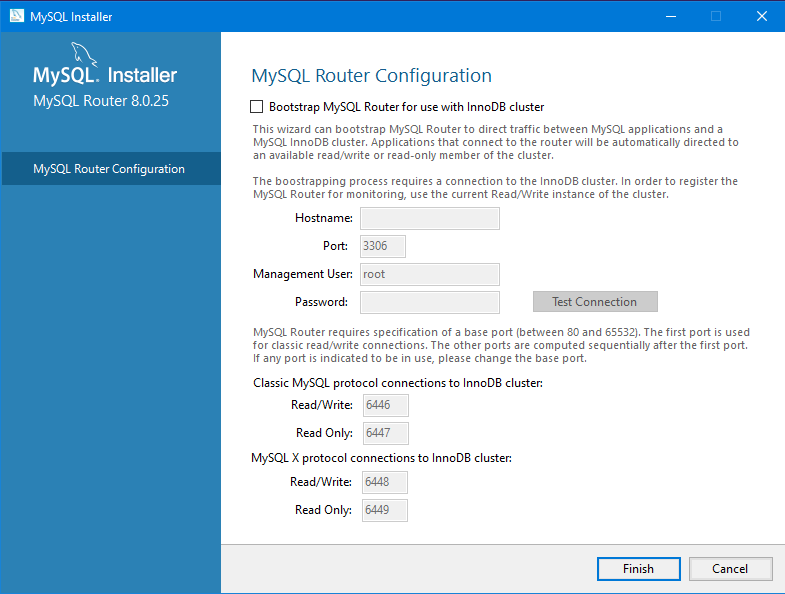


Imagen 1.1.10 Resumen de la configuración

Para finalizar con la instalación levantaremos el servicio ingresando las credenciales que anteriormente configuramos y dando clic en Check conectaremos el servidor, tal como se muestra en la imagen 1.1.11; para finalizar damos clic en botón de Finalizado para terminar y cerrar el instalador en las imágenes 1.1.12 y la imagen 1.1.13.

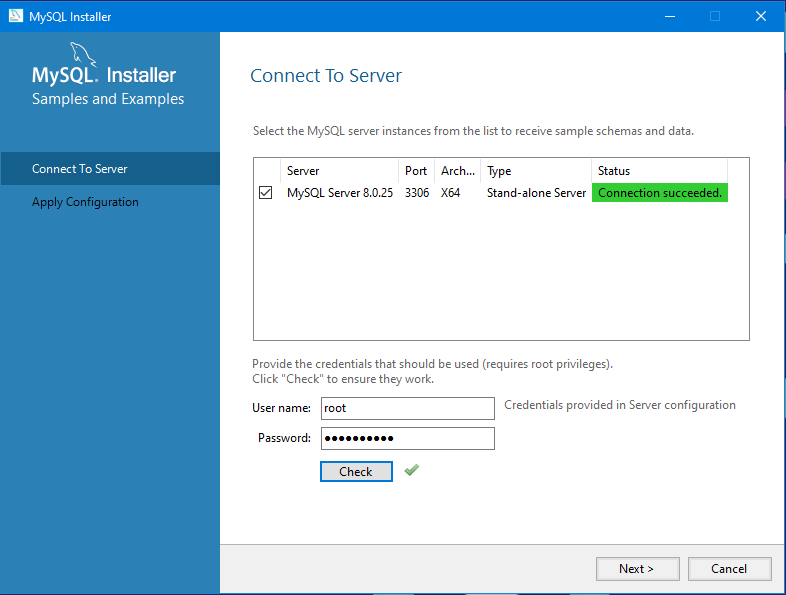


Imagen 1.1.11 Levantamiento del servidor.

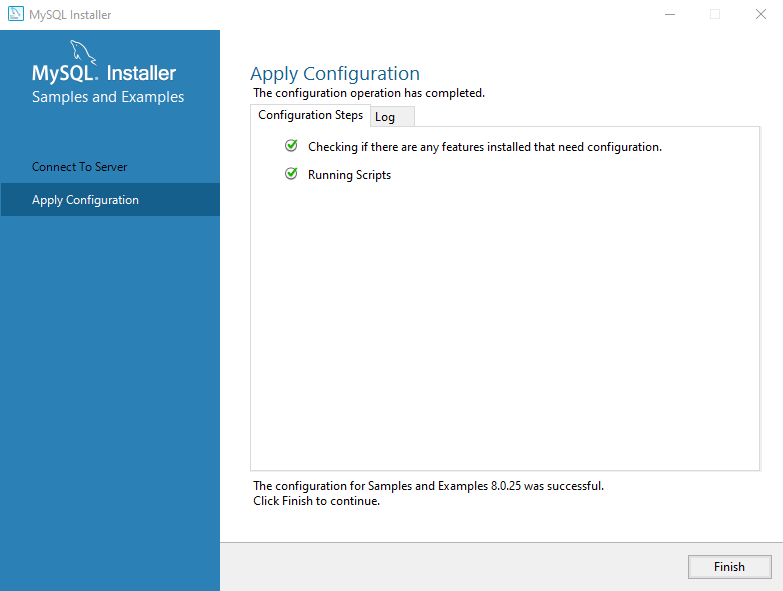


Imagen 1.1.12 Levantamiento del servidor.

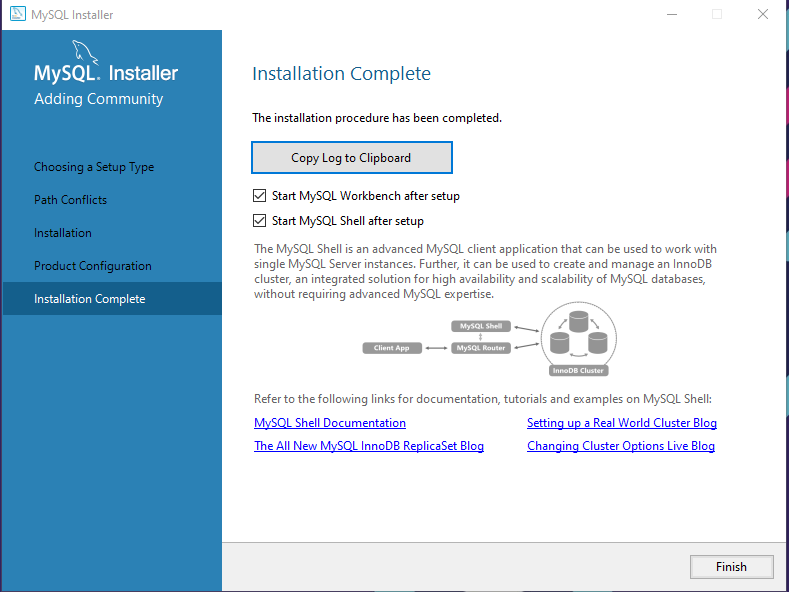


Imagen 1.1.13 Levantamiento del servidor.

## Instalación del servidor de páginas Apache Tomcat 10.

Para desplegar proyectos web regularmente se utilizan servidores instalados o dedicados a administrar los diversos proyectos web que nos permiten ver los sitios en formato HTML dentro de cualquier navegador web como Google Chrome, Internet Explorer, Mozilla Firefox, entre otros más.

Dentro del desarrollo que estaremos dedicando en bloques posteriores requeriremos desplegar nuestro proyecto web en algún navegador y asi verificar el correcto funcionamiento de este, utilizaremos Apache Tomcat en su versión mas estable 10.0, para ello nos a la siguiente dirección web: <https://tomcat.apache.org/download-10.cgi> como nos muestra la imagen 1.2.0.

Descargaremos la opción ZIP y lo colocaremos en nuestro equipo de computo preferentemente en la raíz de nuestro equipo como se muestra en la imagen 1.2.1 para localizarlo fácilmente a la hora de incorporarlo con nuestro IDE o entorno de desarrollo.

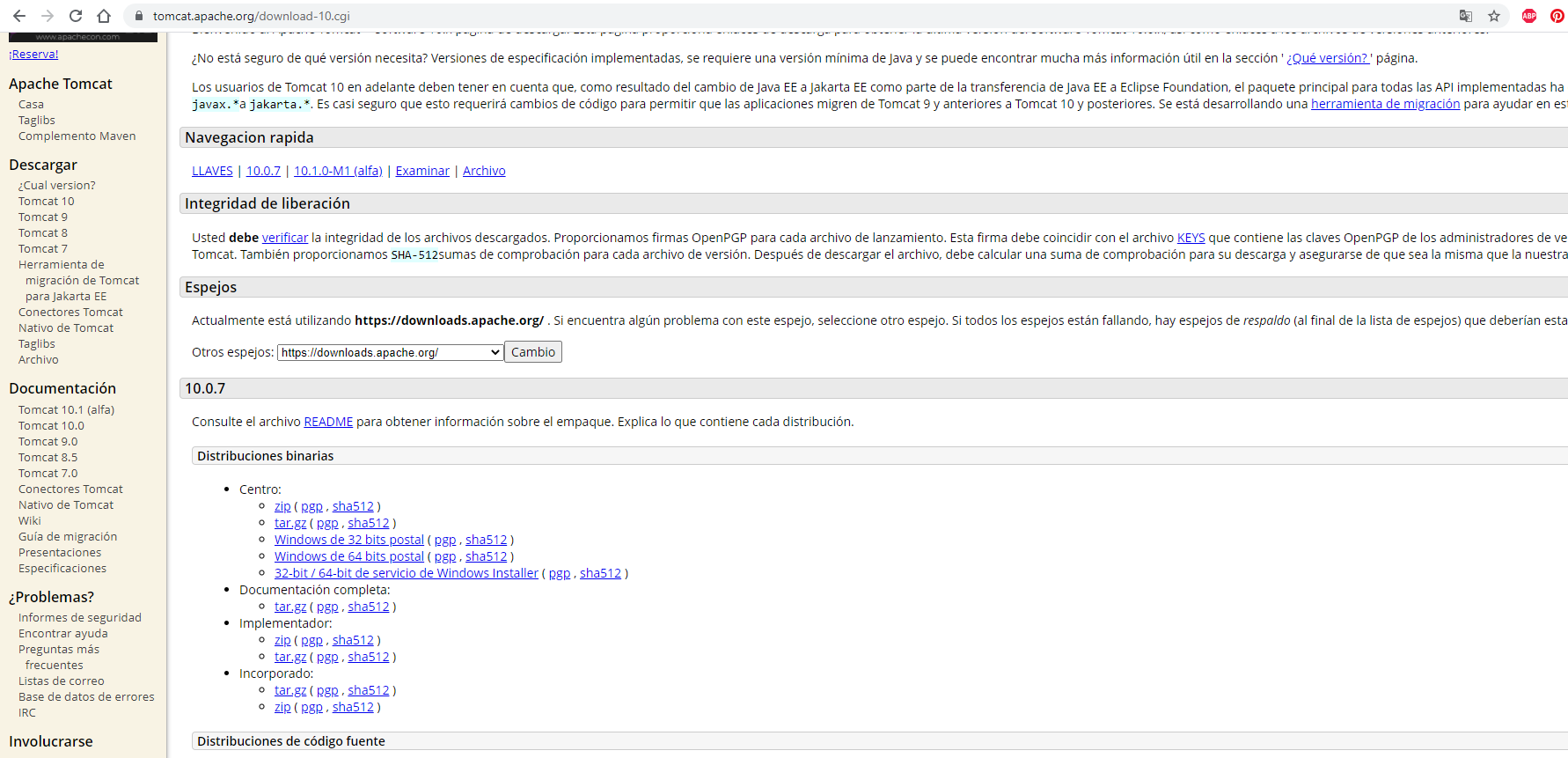


Imagen 1.2.0 Descarga de los archivos ZIP.

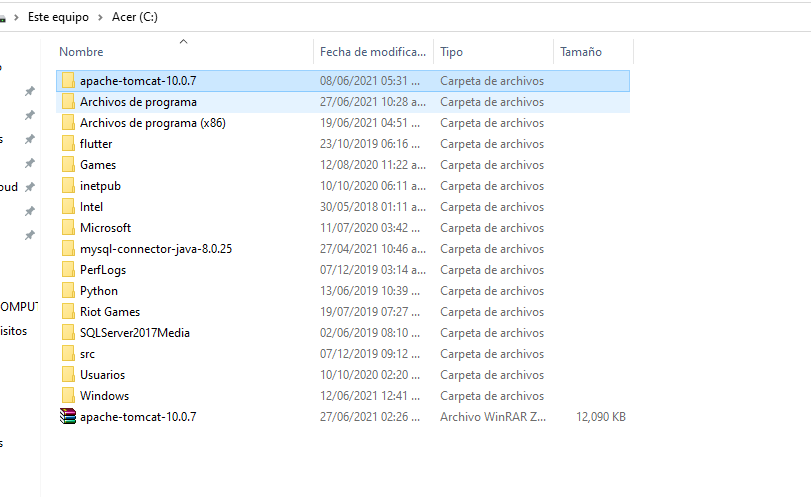


Imagen 1.2.1 Descarga y descompresión de archivos ZIP de Apache Tomcat.

## 2.0 Diseño de la base de datos relacional básico de un sistema web de ventas.

Cuando se desarrolla un sistema web de tipo ventas es de manera prioritario tener un reposito de datos para el cual mantener y genera un flujo de información que va desde generar una venta, hasta consultar productos, lo cual hace interactiva y dinámico cualquier sistema web de este tipo, en este caso consideraremos 3 entidades principales para generar una venta dentro del sistema web las cuales son: Usuarios, Producto y Ventas. Estas entidades nos permitirán expandir las necesidades de los usuarios desde lo más básico hasta algo más complejo como la administración de direcciones o ventas; categorizando cada una de ellas para que la información fluya de manera más comprensiva para el desarrollador que está llevando a cabo la parte dinámica a la hora de generar flujos completos de información bajo un patrón de diseño como el MVC que en bloques posteriores mencionaremos de manera básica la utilización, desarrollo y despliegue de cada elemento que conforma este patrón.

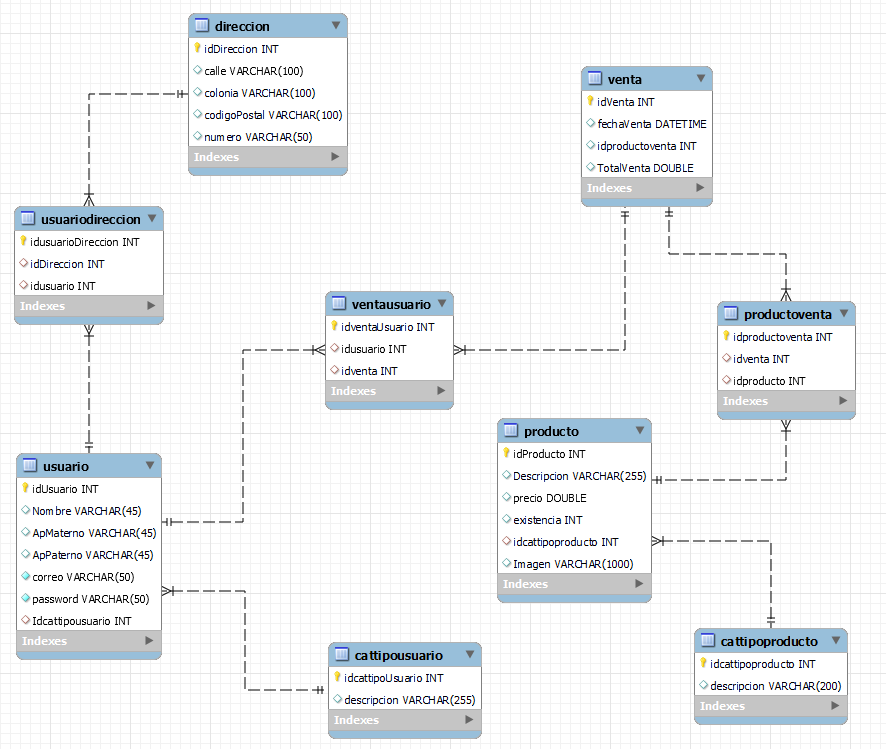


Imagen 1.2.2 Modelo Relacional

Podemos generar el modelo con la siguiente información de un Diccionario de datos desglosado en cada tabla que utilizaremos en nuestro modelo Relacional que observamos en la imagen 1.2.2.

En el sistema web se utilizarán catálogos como el catálogo de productos (Tabla 1.0.1) que nos permitiría categorizar distintos tipos de productos por su descripción, además del catálogo de usuarios (Tabla 1.0.2) que nos permitirá gestionar distintos tipos de usuarios y roles en el sistema web como gestionar productos en nuestro sistema de ventas.

Tabla 1.0.1 Cattipoproducto.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nombre** | **Tipo de dato** | **PK** | **FK** | **Descripción** |
| idcattipoproducto | int | X |  | identificador |
| descripcion | Varchar (255) |  |  | Descripcion corta |

Tabla 1.0.2 Cattipousuario.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Columna** | **Tipo de dato** | **PK** | **FK** | **Descripción** |
| idcattipoUsuario | int | X |  | identificador |
| descripcion | Varchar (255) |  |  | Descripcion corta |

Para casi cualquier sistema web del tipo de ventas se requiere gestionar el acceso al sistema para lo cual tenemos nuestra entidad Usuario que modelado en tabla obtendremos las credenciales del usuario que utilizará el sistema web, como se muestra en la tabla 1.0.3.

Tabla 1.0.3 Usuario

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Columna** | **Tipo de dato** | **PK** | **FK** | **Descripción** |
| id Usuario | int | X |  | identificador |
| Nombre | Varchar (50) |  |  | Nombre del usuario. |
| Materno | Varchar (50) |  |  | Apellido materno del usuario. |
| Paterno | Varchar (50) |  |  | Apellido Paterno del usuario. |
| correo | Varchar (50) |  |  | Correo del usuario. |
| password | Varchar (50) |  |  | Password del del usuario. |
| IdcattipoUsuario | int |  | X | Identificador tabla cattipousuario |

Dentro de nuestra definición en el modelo relacional utilizaremos distintas direcciones asi que para ello requerimos la table de Dirección establecida en la tabla 1.0.4 y por definición una tabla de UsuarioDireccion en la tabla 1.0.5.

Tabla 1.0.4 Direccion

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Columna** | **Tipo de dato** | **PK** | **FK** | **Descripción** |
| idDireccion | int | X |  | identificador |
| calle | Varchar (100) |  |  | Nombre de la calle |
| colonia | Varchar (100) |  |  | Nombre de la colonia |
| codigoPostal | Varchar (100) |  |  | Identificador postal |
| numero | Varchar (100) |  |  | Numero de domicilio |

Tabla 1.0.5 UsuarioDireccion.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Columna** | **Tipo de dato** | **PK** | **FK** | **Descripción** |
| idusuarioDireccion | int | X |  | identificador |
| idDireccion | int |  | X | Identificador tabla dirección |
| idusuario | int |  | X | Identificador tabla usuario |

Para que un usuario pueda generar múltiples ventas se utilizaron las tablas ventausuario y venta para gestionar lo que cada usuario pueda generar a partir de la selección de productos; la tabla ventausuario establecida en la tabla 1.0.6 y la tabla venta generada en la tabla 1.0.7.

Nos permitirá establecer un flujo de venta entre usuario y las ventas que este pueda generar.

Tabla 1.0.6 VentaUsuario.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Columna** | **Tipo de dato** | **PK** | **FK** | **Descripción** |
| idventausuario | int | X |  | identificador |
| idusuario | int |  | X | Identificador tabla usuario |
| idventa | int |  | X | Identificador tabla venta |

Tabla 1.0.7 Venta.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Columna** | **Tipo de dato** | **PK** | **FK** | **Descripción** |
| idVenta | int | X |  | identificador |
| fechaVenta | datetime |  |  | Fecha de creación de la venta |
| idproductoventa | int |  | X | Identificador tabla productoventa |
| TotalVenta | double |  |  | Precio acumulado de los productos |

Las ventas en los sistemas web vienen acompañados de la obtención de productos que son entidades con variaciones por definición ya que esta entidad cuenta con actualizaciones seguidas muy comunes en el desarrollo web a continuación mostramos las tablas de productos, al mismo tiempo como se muestra en la tabla 1.0.8. También por definición en las ventas presentemos múltiples productos en las ventas, para ello utilizamos la relación en una tabla denominada productoventa establecida en la tabla 1.0.9.

Tabla 1.0.8 Producto.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Columna** | **Tipo de dato** | **PK** | **FK** | **Descripción** |
| idProducto | int | X |  | identificador |
| Descripcion | Varchar (100) |  |  | Descripcion corta |
| precio | double |  |  | Precio del Producto |
| existencia | int |  |  | Numero de Productos |
| idcattipoproducto | int |  | X | Identificador tabla cattipoproducto |
| Imagen | Varchar (1000) |  |  | Imagen del producto |

Tabla 1.0.9 ProductoVenta.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Columna** | **Tipo de dato** | **PK** | **FK** | **Descripción** |
| idproductoventa | int | X |  | identificador |
| idventa | int |  | X | Identificador tabla venta |
| idproducto | int |  | X | Identificador tabla producto |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Columna** | **Tipo de dato** | **PK** | **FK** | **Descripcion** |
| idwishlist | int | X |  | identificador |
| Descripcion | Varchar (255) |  |  | Descripción de la lista |
| idProducto | int |  | X | Identificador tabla producto |
| idusuario | int |  | X | Identificador tabla producto |

## 3.0 Integración del servidor Tomcat 10 con NetBeans 12.0

Para instalar ciertos componentes en un IDE completo como NetBeans es necesario seguir ciertas características, ya que muchas veces en tipo de desarrollo la integración con servidores locales para desplegar un proyecto web, es necesario seguir ciertos pasos con características que son propias del IDE NetBeans, en este apartado integraremos estos dos componentes para hacer un primer despliegue de un proyecto llamado *TestWeb* creado desde NetBeans y se mostrara en nuestro navegador local sobre un servidor web Tomcat 10.

Para ello requerimos abrir nuestro IDE NetBeans, para levantar un servidor Tomcat requerimos ir al apartado Tools/Herramientas dando clic en Servers/Servidores como se muestra en nuestra Imagen 1.2.3.

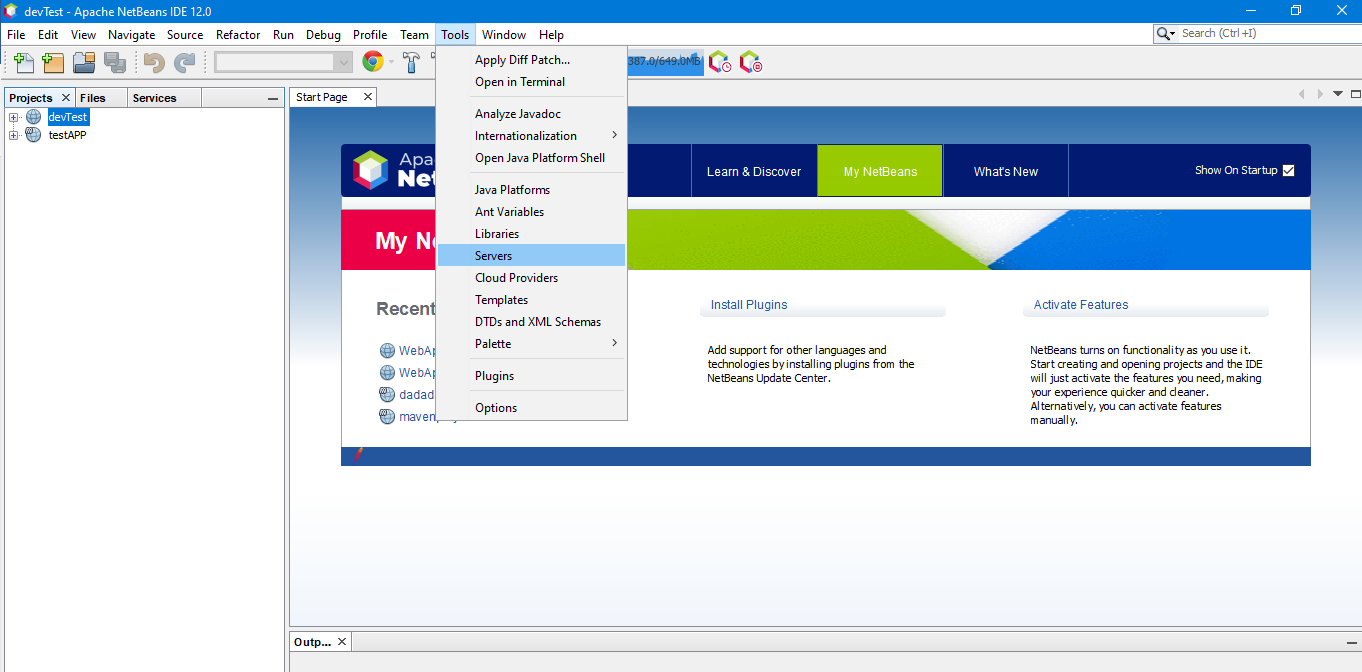


Imagen 1.2.3 Levantamiento de Servidor web Tomcat.

A continuación, nos muestra la imagen 1.2.4 donde se encuentran la configuración necesaria para la integración y levantamiento del servidor. Para agregar un servidor damos clic en Add Server/Agregar servidor.

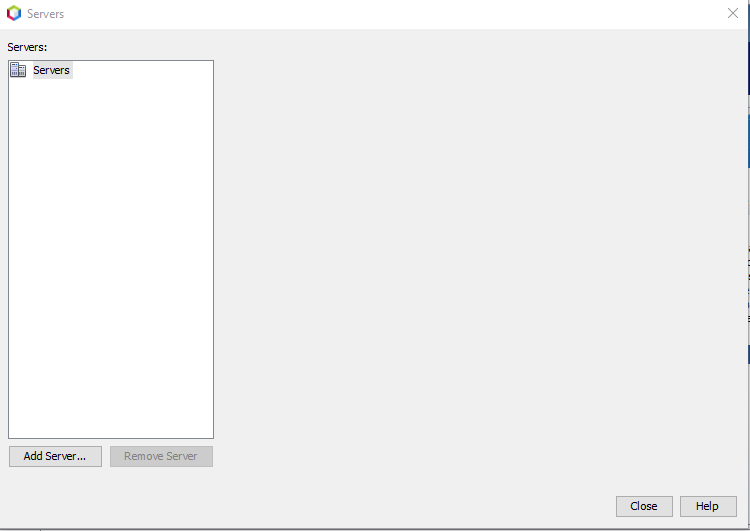


Imagen 1.2.4 Agregar servidor.

Este paso nos mostrará los servidores web que el IDE NetBeans soporta para esta versión; para continuar con la integración seleccionamos el servidor *Apache Tomcat or TomEE* y damos clic en siguiente tal como se muestra en la imagen 1.2.5.

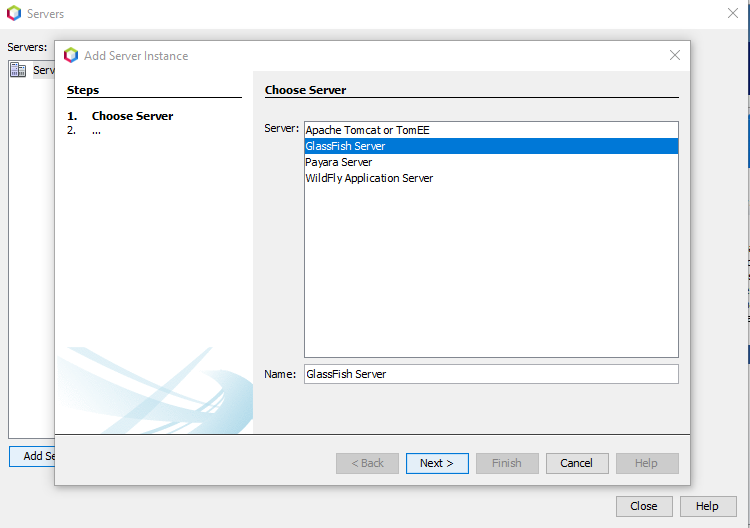


Imagen 1.2.5 Selección de servidor nuevo.

Como anteriormente mostramos en el apartado 1.2 utilizaremos el archivo ZIP que descargamos anteriormente y extraemos en nuestro equipo de cómputo, para lo cual en la opción *Server Location* damos clic en el botón *Browse* y seleccionamos la ruta de nuestro archivo extraído como se muestra en la imagen 1.2.6.

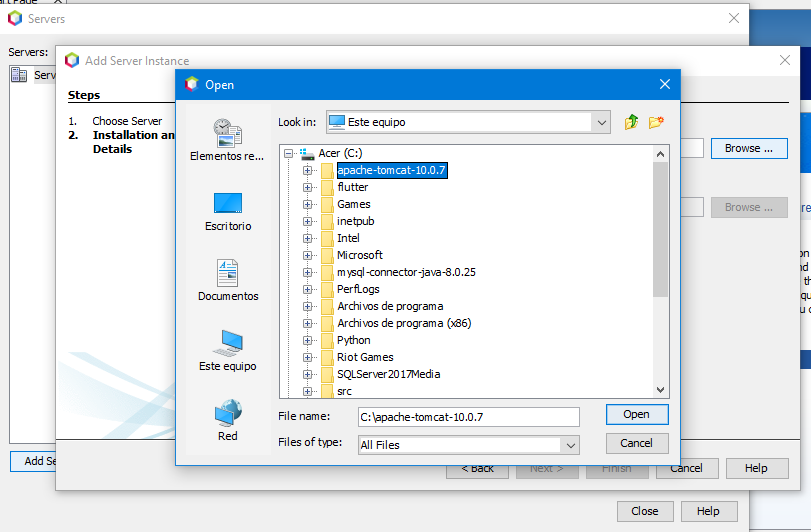


Imagen 1.2.6 Selección de archivos propios del servidor.

A continuación, ingresamos las credenciales iniciales para nuestro servidor web, tenga presente que estas se utilizaran al levantar el servicio en apartados posteriores; al ingresar las credenciales damos clic al botón *Finish* como se muestra en la imagen 1.2.7*.*

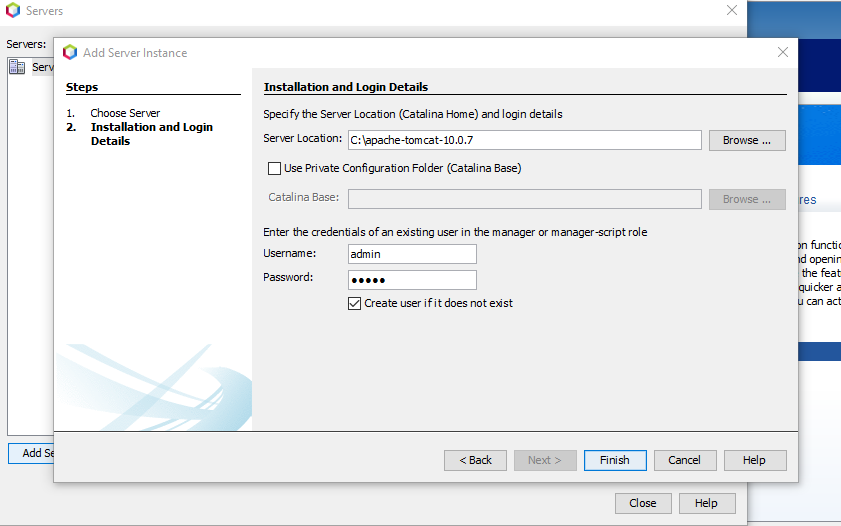


Imagen 1.2.7 Configuración de credenciales del servidor web.

Automáticamente colocará la configuración básica de nuestro servidor web Tomcat 10, en donde se muestra el servidor de puerto predeterminado que es configurable si es que otro servicio no esta utilizándolo, la recomendación para estos casos es dejar el puerto predeterminado, también mostrará las credenciales antes ingresadas de nuestro servidor web, para terminar, damos clic en *Close* para terminar la instalación del componente, como se muestra en la imagen 1.2.8.

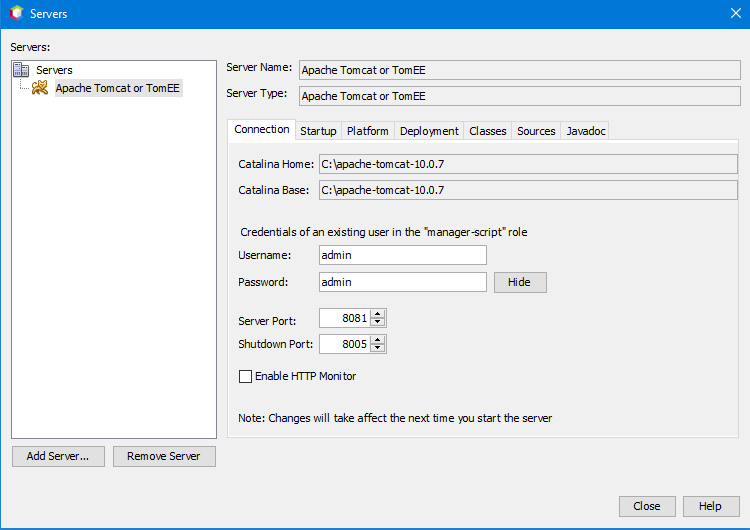


Imagen 1.2.8 Finalización de instalación del servidor web con NetBeans 12.0

## 3.1 Creación de un proyecto web TestWeb y levantamiento del servidor Tomcat 10.

Para hacer uso de nuestro servidor web antes instalado en el apartado 3.0, debemos regresar a nuestro IDE NetBeans donde crearemos un proyecto nuevo tal como se muestra en la imagen 1.2.9 donde el tipo será un *Web Application*, en este apartado nombraremos el proyecto como *TestWeb* y daremos clic en el botón de *Next*.

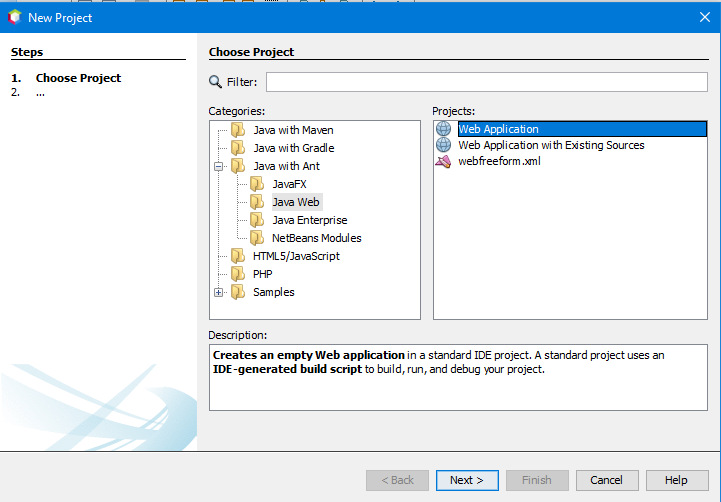


Imagen 1.2.9 Creación del proyecto web TestWeb.

Para continuar nombraremos el proyecto y elegiremos el lugar donde estará alojado nuestro proyecto, terminado esto daremos clic en el botón Nextcomo se muestra en la imagen 1.3.0.

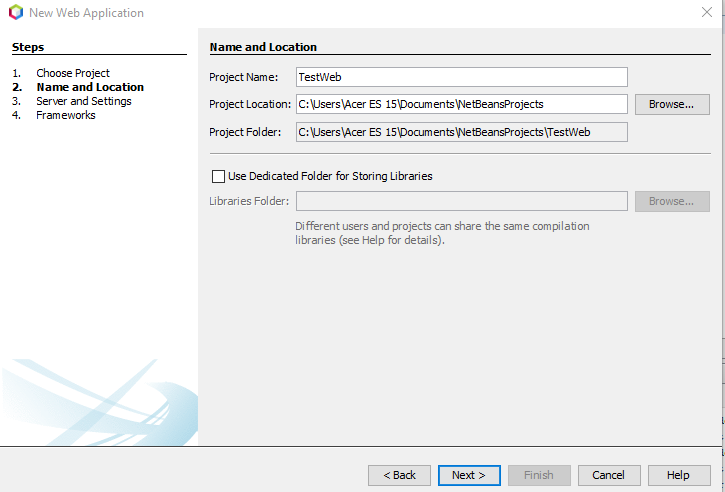


Imagen 1.3.0 Nombramiento del proyecto web.

En el siguiente apartado nos mostrará el servidor mediante el cual se estará desplegando dentro de nuestro navegador web, en este apartado se pueden tener más de un servidor, pero solo se puede seleccionar uno principalmente; en este caso elegimos el servidor web *Apache Tomcat or TomEE* tal como se muestra en la imagen 1.3.1, al finalizar damos clic en el botón *Next*.

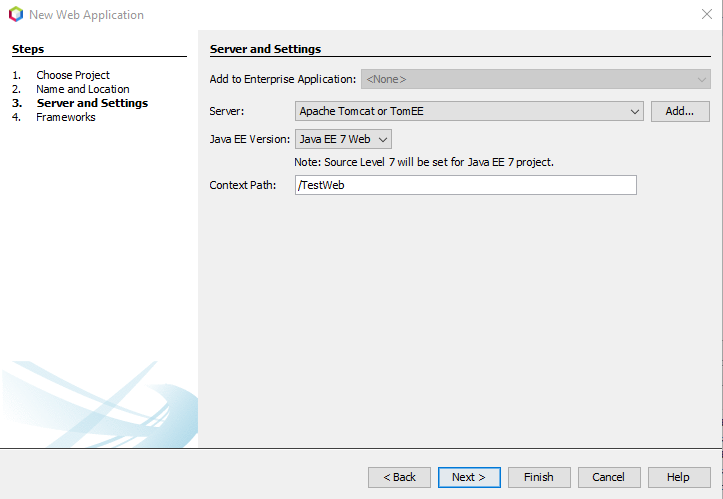


Imagen 1.3.1 Elección de servidores.

Para concluir con la generación del proyecto damos clic en el botón *Finish* como se muestra en la imagen 1.3.2; en este apartado esperaremos un momento para que el IDE genere el proyecto con nuestra configuración antes generada, al terminar nos mostrará nuestro proyecto como la imagen 1.3.3 con las librerías básicas.

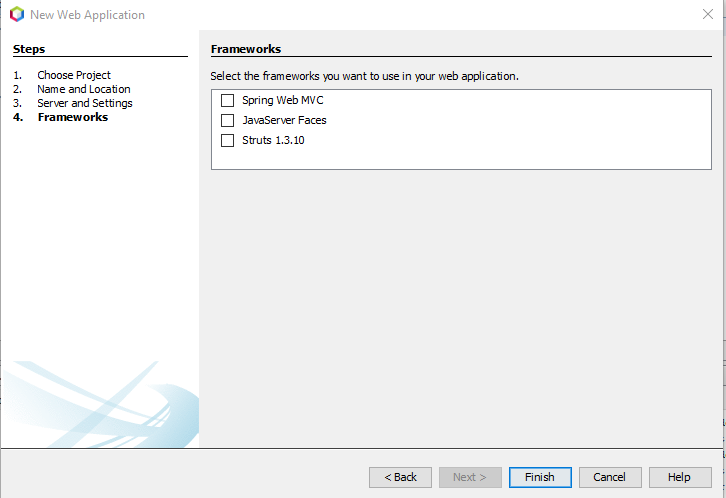


Imagen 1.3.2 Finalización de instalación del proyecto.

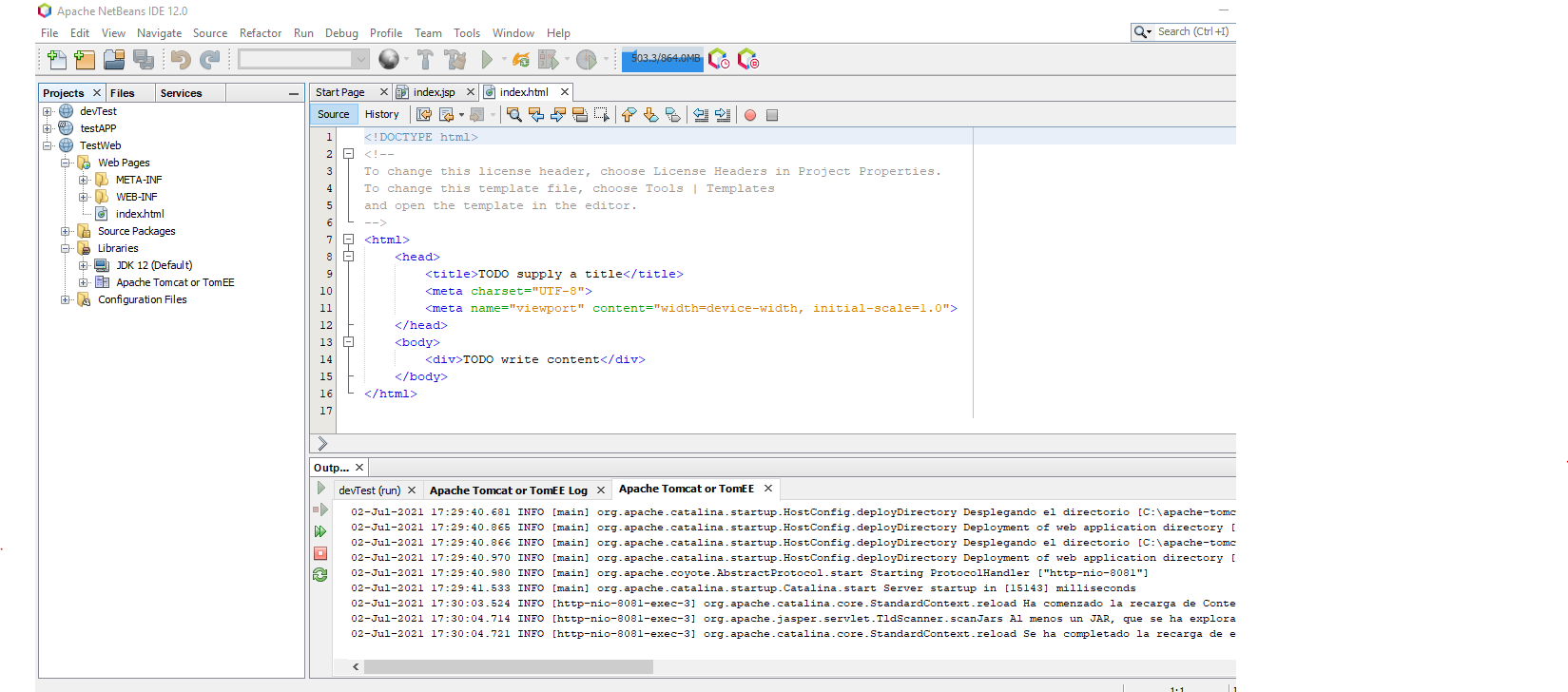


Imagen 1.3.3 Generación un proyecto web.

Si todo ha sido satisfactorio abrimos el archivo index.html que el proyecto contiene por inicio y corremos el proyecto para desplegar el proyecto con el servidor Tomcat 10, como se muestra la imagen 1.3.4, esto nos indicará que el proyecto se generó de manera correcta y que el servidor Tomcat funciona de manera correcta en la integración con NetBeans 12.0 tal como se muestra en la imagen.

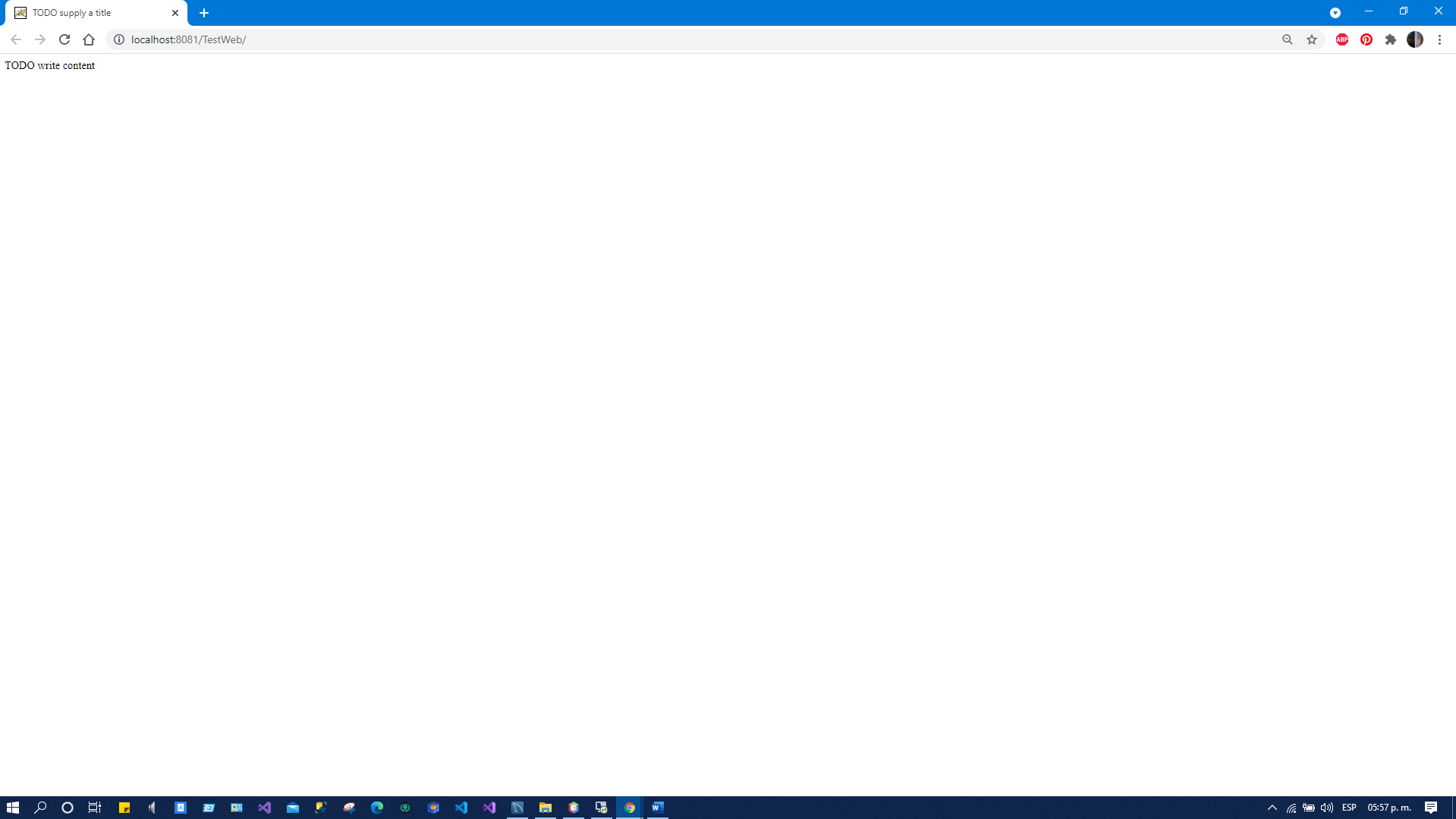
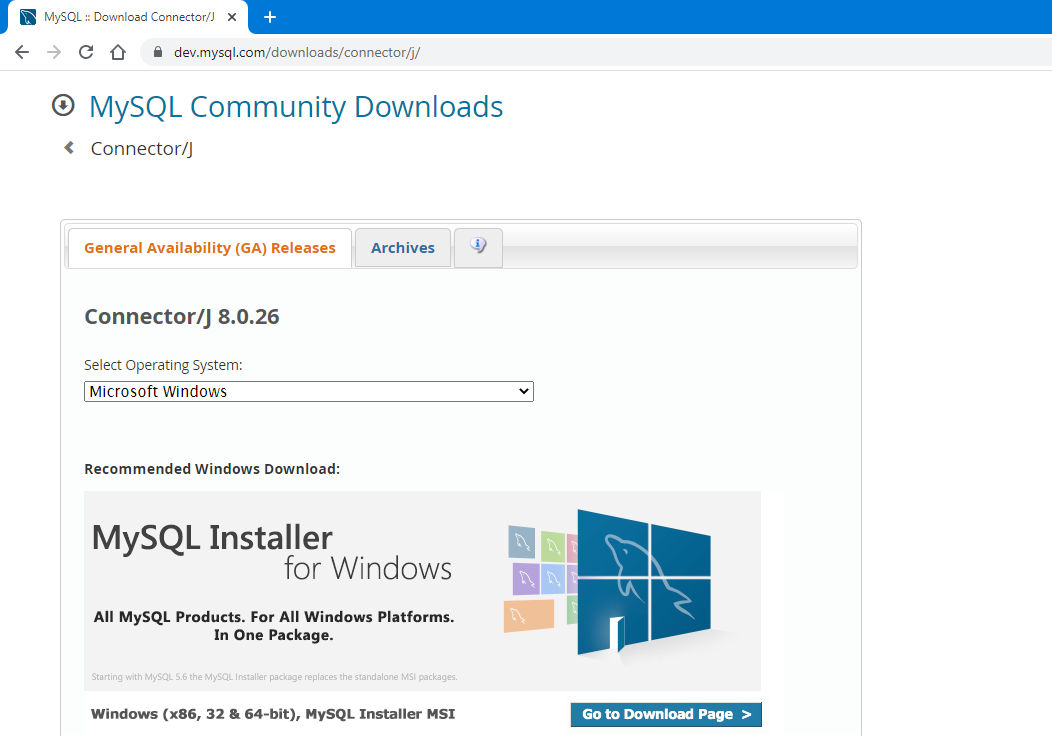


Imagen 1.3.4 Despliegue y levantamiento del servidor

## Instalación del componente JDBC en el proyecto de desarrollo en el IDE NetBeans 12.0.

Como mencionamos en apartados anteriores requerimos utilizar un conector JDBC para realizar consultas directo a la base de datos MySQL con el fin de interactuar con la información desde la vista del cliente o también llamado usuario.

Para descargar el componente necesario es requerido ir a la siguiente direccion web: <https://dev.mysql.com/downloads/connector/j/> dentro de esta nos desplegará una página como se muestra en la imagen 1.4.0:



1.4.0 Sitio de descarga del conector JDBC.

Para obtener directamente el conector elegiremos la opción plataforma independiente en el apartado de elección del sistema operativo y daremos clic en la opción de descarga; esta opción nos descargará un archivo zip nombrado *mysql-connector-java-8.0.26.zip* que contiene el archivo JAR como se muestra en la imagen 1.4.1 para incorporar al proyecto anteriormente generado.

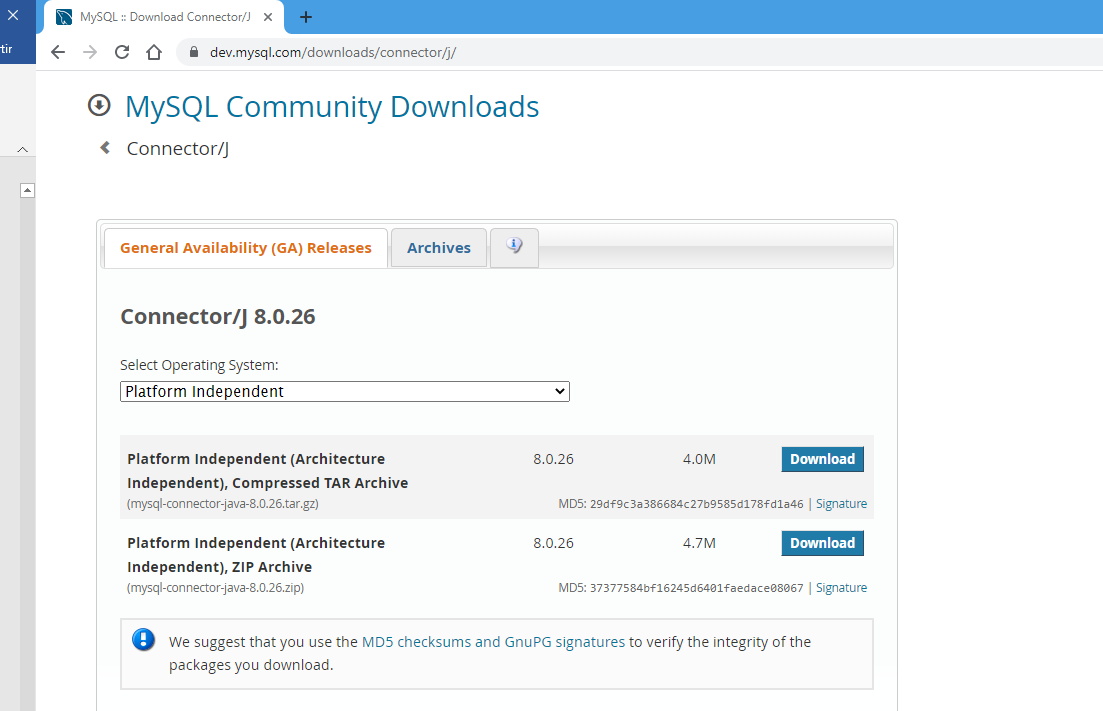


Imagen 1.4.1 descarga del archivo.

Para llevar el control de nuestros archivos es recomendable moverlos de ruta en nuestro equipo de cómputo para hacer uso de ellos de manera más sencilla.

Para hacer uso del conector JDBC es necesario ir al proyecto *TestWeb* de nuestro IDE NetBeans 12, en apartado de librerías damos clic derecho, seleccionamos *Add JAR/Folder* donde nos permitirá seleccionar el archivo JAR que previamente descargamos y también descomprimimos en nuestra ruta establecida, como se muestra en la imagen 1.4.2.

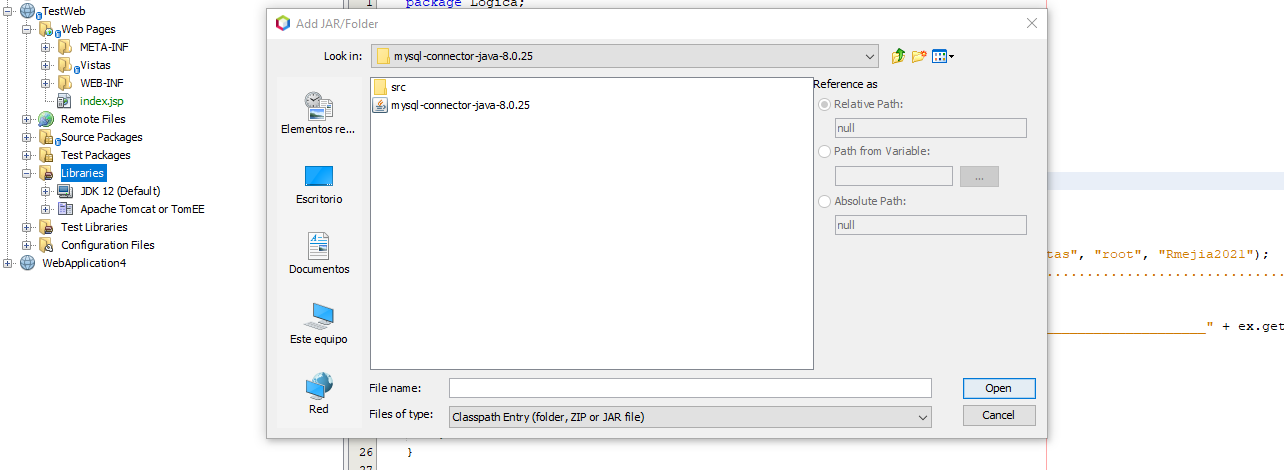


Imagen 1.4.2 Selección de archivo JAR.

# 4.0 Estructuración del patrón de diseño MVC en NetBeans 12.0.

Dentro del desarrollo de sistemas es necesario estructurar los proyectos de acuerdo a ciertas características ya que con ello nos permite seguir un patrón de diseño de manera más fácil y efectiva. Dentro de nuestra solución para estructurar el proyecto de acuerdo con el patrón de diseño MVC es necesario realizar las siguientes acciones:

1.-En el proyecto web *TestWeb* agregaremos los paquetes necesarios en el apartado de *Source Package* (Paquete de recursos), para ello para generar cada paquete dando clic derecho *New* (Nuevo) > *Java Package,* donde siguiendo la estructura del patrón de diseño MVC generaremos 3 paquetes los cuales nombraremos nuestros paquetes de la siguiente manera: Controlador(es) llamada también capa Controladora, Modelo(s) nombrada también capa de Modelo y Lógica (El paquete de lógica se realiza de manera opcional para dividir la capa Lógica del sistema de la parte Controladora ya que muchos métodos y clases regularmente se generan en el paquete Controlador).Al generar los paquetes necesarios para seguir el diseño de MVC estructuraremos nuestro proyecto de la siguiente manera como se muestra en la imagen 1.4.3.

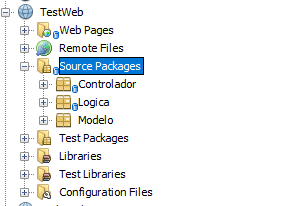


Imagen 1.4.3 Estructuración de paquetes.

2.-En el apartado *Web Pages* agregaremos la parte o también llamada capa de presentación de MVC, para ello dando clic derecho New (Nuevo) > Folder donde nombraremos nuestra estructura de vistas nombrándola de la misma manera tal como se muestra en la imagen 1.4.4.

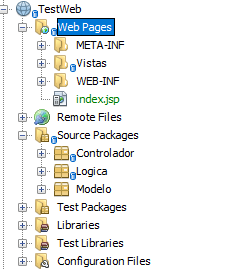


Imagen 1.4.4 Estructuración de la capa de presentación.

# 5.0 Prueba de integración con la base de datos MySQL.

En el proyecto podemos generar pruebas de integración con el conector JDBC, esto nos permitirá saber si tenemos acceso a nuestra base de datos siguiendo el MVC, para ello se apoyará de una clase esencial que a partir de aquí nombraremos como *clase* *Conexión.* En el proyecto *TestWeb* generaremos una clase Java dentro del paquete Lógica con el nombre de Conexión, para ello dando clic derecho en el paquete lógica *New (Nuevo) > Java Class* la cual nos generará un archivo .java como se muestra en la imagen 1.4.5.

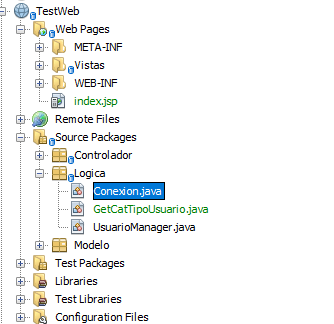


Imagen 1.4.5 Generación de clases de acceso a base de datos.

Para hacer uso del archivo JAR que nos permite la interacción con la base de datos MySQL que generamos anteriormente llamada *DBVENTAS* dentro de la clase *Conexión* utilizaremos el siguiente código que nos permitirá ingresar la cadena de conexión con la(s) base de datos, como se muestra a continuación.

package Logica**;**

**import** java**.**sql**.**Connection**;**

**import** java**.**sql**.**DriverManager**;**

public class Conexion **{**

Connection con**;**

private static String db**=**"dbventas"**;**

private static String usuario**=**"root"**;**

private static String password**=**"Rmejia2021"**;**

private static String conexion**=**"jdbc:mysql://localhost:3306/"**+**db**;**

public Conexion**()** **{**

**try** **{**

Class**.**forName**(**"com.mysql.jdbc.Driver"**)**

con **=** DriverManager**.**getConnection**(**conexion**,**usuario**,**password**);**

System**.**out**.**println**(**"Conexion exitosa con la base de datos dbventas.....................................\n"**);**

**}** **catch** **(**Exception ex**)** **{**

System**.**err**.**println**(**"Error de conexion con la base de datos:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" **+** ex**.**getMessage**());**

**}**

**}**

public Connection getConection**()** **{**

**return** con**;**

**}**

**}**

Para acceder a los métodos de la clase *Conexión.java* tenemos un método publico llamado *getConection()* donde podemos acceder de manera rápida y sencilla con la base de datos *DBVENTAS* por medio del constructor de la clase Conexión.

## 5.1 Ejemplo de creación del CRUD MVC utilizando Servlets y JSP.

Para generar correctamente un CRUD utilizando Servlets y JSP es necesario considerar cuatro partes indispensables, en las cuales la generación componentes que realicen peticiones mediante solicitudes HTTP ya se de tipo Get o Post.

# 5.1.1Método Read(Lectura)

El método de lectura (Read) es el cual muestra la información que el usuario pueda visualizar e interactuar, para generar este método de extracción de información directamente desde la base de datos utilizando el patrón de diseño MVC es necesario definir el tipo de información que se requiere manipular, para ello se apoyara del modelo relacional que mencionamos en apartados anteriores, utilizaremos la extracción de la información de la tabla Usuarios para generar un apartado en el proyecto nombrado *Login*  el cual mediante credenciales de acceso nos permitirá redireccionar y extraer en primera instancia la información en forma de tabla. Tal navegación se mostrará en la imagen 1.4.6 y 1.4.7 correspondientes.

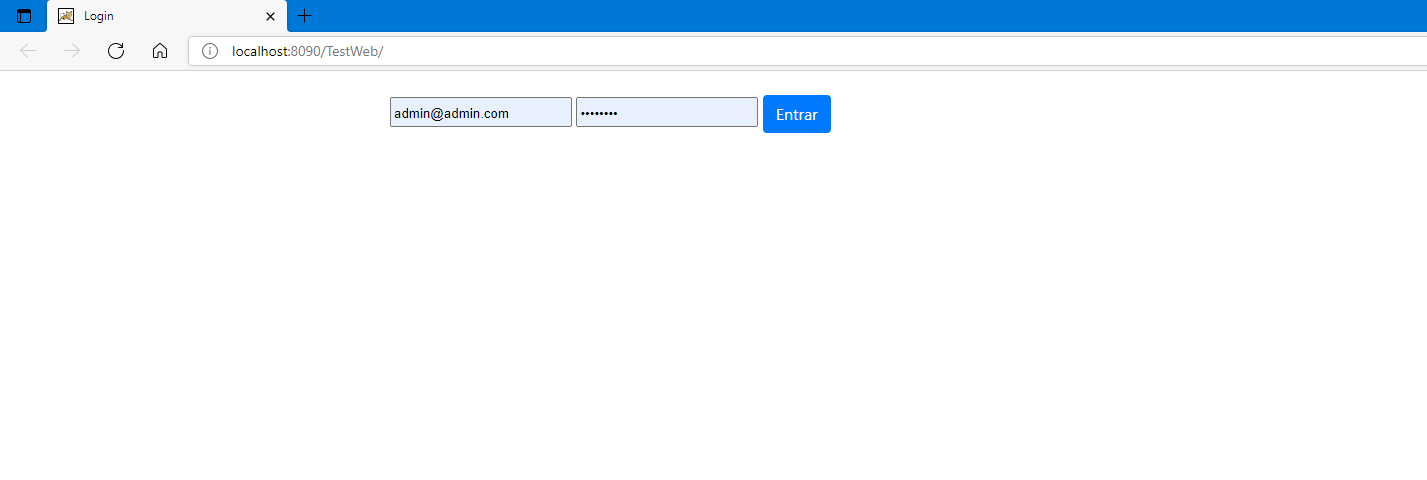


Imagen 1.4.6 Login de Usuarios.

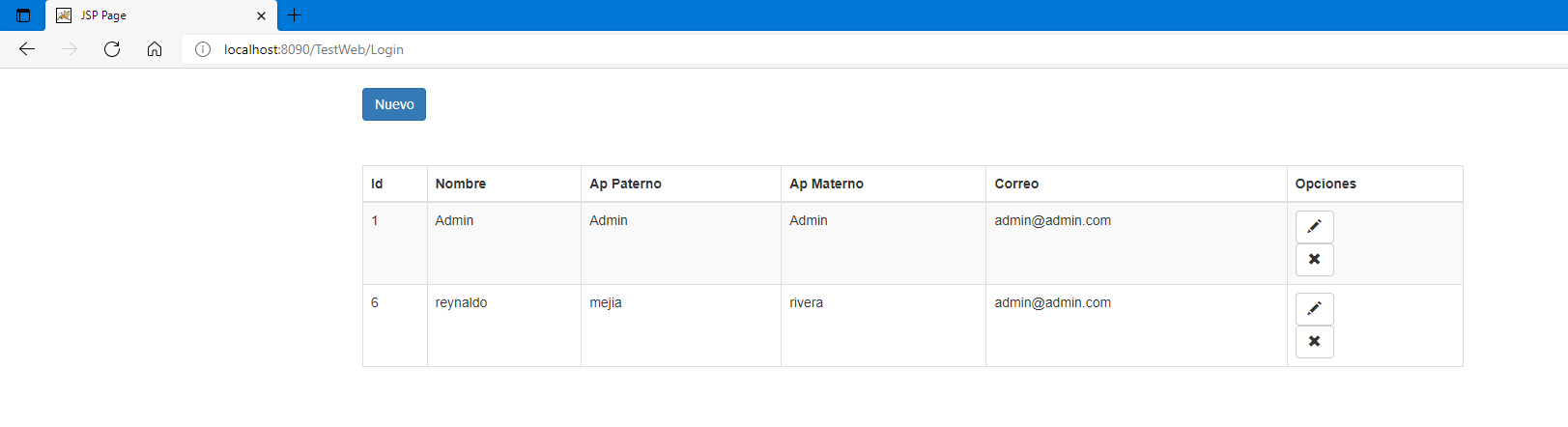


Imagen 1.4.7 Consulta de Usuarios.

En la imagen 1.4.6 utilizaremos información existente para utilizar un Login básico, para genera un Login básico es requerido utilizar el archivo *Index.jsp* el cual como primera instancia es necesario nombrar a este archivo el fichero de arranque del proyecto utilizado; dentro del archivo *Index.jsp* es necesario generar 3 componentes dentro de un formulario, como se muestra en el siguiente código.

**<**div class**=**"container"**>**

**<**form method**=**"post" action**=**"Login" **>**

**<**input type**=**"email" name**=**"correo"**/>**

**<**input type**=**"password" name**=**"password"**/>**

**<**input class**=**"btn btn-primary" type**=**"submit" value**=**"Entrar" **/>**

**</**form**>**

**</**div**>**

En el código anterior se utilizaron 3 compontes HTML los cuales nos permiten ingresar por orden de declaración un correo electronico, una contraseña y un botón que nos permitirá desde su tipo submit enviar la información hacia un componente Servlet con el nombre de Login, tal Servlet se creará dentro del paquete Controlador.

Tal controlador nos permitirá el acceso al sistema utilizando las credenciales correctas que están registradas con anterioridad en la base de datos, al mismo tiempo, si la respuesta de la existencia de las credenciales esta nos permitirá obtener todos los registros desde una consulta en base de datos desde la parte lógica, primeramente, utilizaremos el siguiente código que se incorporará en el Servlet Logica.java.

En caso contrario de que el acceso sea denegado nos mostrará un mensaje incrustado en la pantalla donde se generó la solicitud como se muestra en la imagen 1.4.8.

package Controlador**;**

**import** Logica**.**UsuarioManager**;**

**import** Modelo**.**Usuario**;**

**import** java**.**io**.**IOException**;**

**import** java**.**io**.**PrintWriter**;**

**import** jakarta**.**servlet**.**ServletException**;**

**import** jakarta**.**servlet**.**annotation**.**WebServlet**;**

**import** jakarta**.**servlet**.**http**.**HttpServlet**;**

**import** jakarta**.**servlet**.**http**.**HttpServletRequest**;**

**import** jakarta**.**servlet**.**http**.**HttpServletResponse**;**

**import** java**.**util**.**List**;**

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

public class Login **extends** HttpServlet **{**

protected void processRequest**(**HttpServletRequest request**,** HttpServletResponse response**)**

**throws** ServletException**,** IOException **{**

response**.**setContentType**(**"text/html;charset=UTF-8"**);**

**try** **(**PrintWriter out **=** response**.**getWriter**())** **{**

String correo **=** request**.**getParameter**(**"correo"**);**

String password **=** request**.**getParameter**(**"password"**);**

UsuarioManager manager **=** **new** UsuarioManager**();**

int login **=** manager**.**LoginUsuario**(**correo**,** password**);**

**if** **(**login **==** 1**)** **{**

UsuarioManager mn **=** **new** UsuarioManager**();**

List**<**Usuario**>** usuarios **=** mn**.**GetUsuarios**();**

request**.**setAttribute**(**"listaUsuariosRender"**,** usuarios**);**

request**.**getRequestDispatcher**(**"usuarios.jsp"**).**forward**(**request**,** response**);**

**}** **else** **{**

String error **=** "Contraseña incorrecta, intentalo de nuevo"**;**

request**.**setAttribute**(**"error"**,** error**);**

request**.**getRequestDispatcher**(**"index.jsp"**).**forward**(**request**,** response**);**

**}**

**}**

**}**

@Override

protected void doGet**(**HttpServletRequest request**,** HttpServletResponse response**)**

**throws** ServletException**,** IOException **{**

processRequest**(**request**,** response**);**

**}**

@Override

protected void doPost**(**HttpServletRequest request**,** HttpServletResponse response**)**

**throws** ServletException**,** IOException **{**

processRequest**(**request**,** response**);**

**}**

@Override

public String getServletInfo**()** **{**

**return** "Short description"**;**

**}**

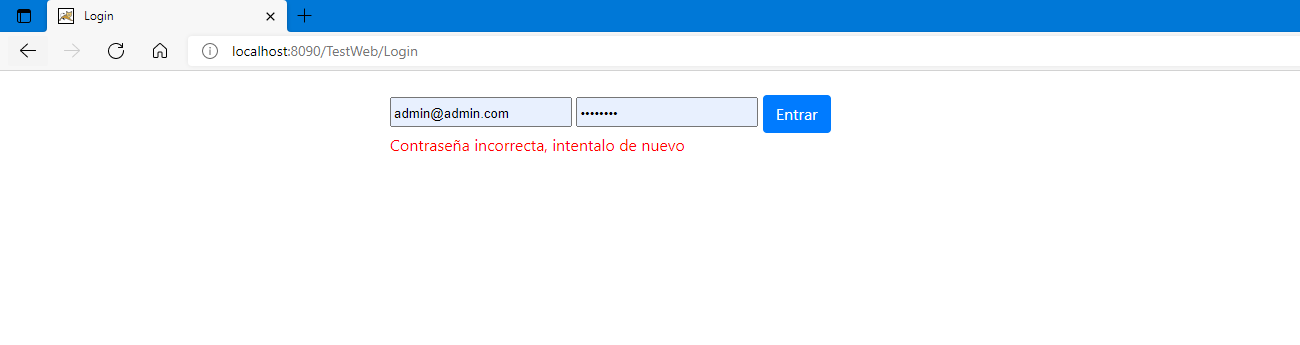


Imagen 1.4.8 Usuario Incorrecto.

Es necesario resaltar un apartado, el cual el envió de la información a través de un JSP el cual es utilizando el método request**.**getRequestDispatcher el cual nos permite redireccionarlo enviando el nombre de la ruta específica, para enviar la información se utilizará el tipo de componente request**.**setAttribute el cual recibe como parámetros el nombre del objecto y como tal el mismo objeto.

Para obtener los registros obtenidos en la consulta correspondiente es necesario generar un archivo .java en el paquete denominado Logica el cual se nombrará como UsuarioManager.java este archivo tendrá los métodos de obtención de usuarios, obtención de usuarios por identificador, registro de usuarios, actualización de usuarios y eliminación que se estarán especificando en apartados posteriores.

Todos los métodos correspondientes se muestran en el siguiente código.

package Logica**;**

**import** Modelo**.**Usuario**;**

**import** java**.**sql**.**Connection**;**

**import** java**.**sql**.**PreparedStatement**;**

**import** java**.**sql**.**ResultSet**;**

**import** java**.**sql**.**SQLException**;**

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**List**;**

/\*\*

\*

\* **@author** Acer ES 15

\*/

public class UsuarioManager **extends** Conexion **{**

public List**<**Usuario**>** GetUsuarios**()** **{**

List**<**Usuario**>** usuarios **=** **new** ArrayList**<**Usuario**>();**

**try** **{**

Connection conexion **=** getConection**();**

String query **=** "select \* from usuario"**;**

PreparedStatement st **=** conexion**.**prepareStatement**(**query**);**

ResultSet rs **=** st**.**executeQuery**();**

**while** **(**rs**.**next**())** **{**

Usuario **=** **new** Usuario**();**

usuario**.**setIdUsuario**(**rs**.**getInt**(**1**));**

usuario**.**setNombre**(**rs**.**getString**(**2**));**

usuario**.**setApPaterno**(**rs**.**getString**(**3**));**

usuario**.**setApMaterno**(**rs**.**getString**(**4**));**

usuario**.**setCorreo**(**rs**.**getString**(**5**));**

usuario**.**setPassword**(**rs**.**getString**(**6**));**

usuarios**.**add**(**usuario**);**

**}**

System**.**out**.**println**(**"Registros obtenidos correctamente"**);**

conexion**.**close**();**

**}** **catch** **(**SQLException E**)** **{**

System**.**out**.**println**(**"Error\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_:" **+** E**.**getMessage**());**

**return** **null;**

**}**

**return** usuarios**;**

**}**

public Usuario GetUsuario**(**int idUsuario**)** **{**

Usuario **=** **new** Usuario**();**

**try** **{**

Connection con **=** getConection**();**

String query **=** "select \* from usuario where idUsuario=?"**;**

PreparedStatement st **=** con**.**prepareStatement**(**query**);**

st**.**setInt**(**1**,** idUsuario**);**

ResultSet rs **=** st**.**executeQuery**();**

**while** **(**rs**.**next**())** **{**

usuario**.**setIdUsuario**(**rs**.**getInt**(**1**));**

usuario**.**setNombre**(**rs**.**getString**(**2**));**

usuario**.**setApPaterno**(**rs**.**getString**(**3**));**

usuario**.**setApMaterno**(**rs**.**getString**(**4**));**

usuario**.**setCorreo**(**rs**.**getString**(**5**));**

usuario**.**setPassword**(**rs**.**getString**(**6**));**

**}**

System**.**out**.**println**(**"Registro obtenido correctamente"**);**

con**.**close**();**

**}** **catch** **(**SQLException E**)** **{**

System**.**out**.**println**(**"Error:" **+** E**.**getMessage**());**

**return** **null;**

**}**

**return** usuario**;**

**}**

public int RegistraUsuario**(**Usuario u**)** **{**

int estatus **=** 0**;**

**try** **{**

Connection con **=** getConection**();**

String query **=** "insert into usuario(Nombre,ApPaterno,ApMaterno,correo,password)" **+** "values(?,?,?,?,?)"**;**

PreparedStatement st **=** con**.**prepareStatement**(**query**);**

st**.**setString**(**1**,** u**.**getNombre**());**

st**.**setString**(**2**,** u**.**getApPaterno**());**

st**.**setString**(**3**,** u**.**getApMaterno**());**

st**.**setString**(**4**,** u**.**getCorreo**());**

st**.**setString**(**5**,** u**.**getPassword**());**

// st.setInt(6, u.getIdcattipousuario());

estatus **=** st**.**executeUpdate**();**

System**.**out**.**println**(**"Registrado correctamente"**);**

con**.**close**();**

**return** estatus**;**

**}** **catch** **(**SQLException E**)** **{**

System**.**out**.**println**(**"Error:" **+** E**.**getMessage**());**

**return** 0**;**

**}**

**}**

public int ActualizaUsuario**(**Usuario u**)** **{**

int estatus **=** 0**;**

**try** **{**

Connection con **=** getConection**();**

String query **=** "update usuario set Nombre=?, ApPaterno=?,ApMaterno=?,correo=?,password=? where idUsuario=? "**;**

PreparedStatement st **=** con**.**prepareStatement**(**query**);**

st**.**setString**(**1**,** u**.**getNombre**());**

st**.**setString**(**2**,** u**.**getApPaterno**());**

st**.**setString**(**3**,** u**.**getApMaterno**());**

st**.**setString**(**4**,** u**.**getCorreo**());**

st**.**setString**(**5**,** u**.**getPassword**());**

st**.**setInt**(**6**,** u**.**getIdUsuario**());**

//st.setInt(6, u.getIdcattipousuario());

estatus **=** st**.**executeUpdate**();**

System**.**out**.**println**(**"Registrado correctamente"**);**

con**.**close**();**

**return** estatus**;**

**}** **catch** **(**SQLException E**)** **{**

System**.**out**.**println**(**"Error:" **+** E**.**getMessage**());**

**return** 0**;**

**}**

**}**

public int EliminaUsuario**(**int idUsuario**)** **{**

int estatus **=** 0**;**

**try** **{**

Connection con **=** getConection**();**

String query **=** "delete from usuario where idUsuario=? "**;**

PreparedStatement st **=** con**.**prepareStatement**(**query**);**

st**.**setInt**(**1**,** idUsuario**);**

estatus **=** st**.**executeUpdate**();**

System**.**out**.**println**(**"Eliminado correctamente"**);**

con**.**close**();**

**return** estatus**;**

**}** **catch** **(**SQLException E**)** **{**

System**.**out**.**println**(**"Error:" **+** E**.**getMessage**());**

**return** 0**;**

**}**

**}**

public int LoginUsuario**(**String correo**,** String password**)** **{**

int estatus **=** 0**;**

**try** **{**

Connection c **=** getConection**();**

String query **=** "select \* from usuario where correo=? and password=?"**;**

PreparedStatement st **=** c**.**prepareStatement**(**query**);**

st**.**setString**(**1**,** correo**);**

st**.**setString**(**2**,** password**);**

ResultSet rs **=** st**.**executeQuery**();**

**while** **(**rs**.**next**())** **{**

estatus**=**1**;**

**}**

System**.**out**.**println**(**"logeado correctamente"**);**

c**.**close**();**

**return** estatus**;**

**}** **catch** **(**SQLException E**)** **{**

System**.**out**.**println**(**"Error:" **+** E**.**getMessage**());**

**return** 0**;**

**}**

**}**

**}**

Para obtener la información de todos los usuarios registrados en la base de datos DBVENTAS es necesario utilizar el método GetUsuarios().

Para mostrar la información del método antes mencionado es necesario generar un archivo JSP con el nombre de usuarios.jsp el cual mostrará la tabla con la consulta generada con el método antes mencionado y para leer los elementos se muestra el siguiente código, este código permite indexar en una tabla todos los registros obtenidos.

**<%**@page **import=**"java.util.List" **%>**

**<%**@page **import=**"Modelo.Usuario" **%>**

**<%**@page **import=**"Logica.UsuarioManager" **%>**

**<%**@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**>**

**<**style**>**

table**,** th**,** td **{**

border**:** 1px solid black**;**

**}**

**</**style**>**

**<**br**/>**

**<**br**/>**

**<**br**/>**

**<**div class**=**"row col-lg-12"**>**

**<**div class**=**"container"**>**

**<%**

String error**=**""**;**

error**=(**String**)**request**.**getAttribute**(**"error"**);**

**if(**error**==null)**

error**=**""**;**

**%>**

**<**h4 style**=**"color:red"**><%=**error**%></**h4**>**

**</**div**>**

**</**div**>**

**<**br**/>**

**<**br**/>**

**<**div class**=**"row col-lg-12"**>**

**<**div class**=**"container"**>**

**<**table class**=**"table table-striped table-bordered dt-responsive nowrap" **>**

**<**thead**>**

**<**tr**>**

**<**th**>**Id**</**th**>**

**<**th**>**Nombre**</**th**>**

**<**th**>**Ap Paterno**</**th**>**

**<**th**>**Ap Materno**</**th**>**

**<**th**>**Correo**</**th**>**

**<**th**>**Opciones**</**th**>**

**</**tr**>**

**</**thead**>**

**<**tbody

**<%**

List**<**Usuario**>** usuarios**=(**List**<**Usuario**>)** request**.**getAttribute**(**"listaUsuariosRender"**);**

**for(**Usuario u**:** usuarios**){**

**%>**

**<**tr**>**

**<**td**>** **<%=** u**.**getIdUsuario**()%>** **</**td**>**

**<**td**>** **<%=** u**.**getNombre**()%>** **</**td**>**

**<**td**>** **<%=** u**.**getApPaterno**()%>** **</**td**>**

**<**td**>** **<%=** u**.**getApMaterno**()%>** **</**td**>**

**<**td**>** **<%=** u**.**getCorreo**()%>** **</**td**>**

**<%**

**}**

**%>**

**</**tr**>**

**</**tbody**>**

**</**table**>**

**</**div**>**

**</**div**>**

**</**body**>**

**</**html**>**

# 5.1.2Método Editar(Update)

Para comprender la realización de este método es necesario realizar modificaciones al código utilizado en el apartado anterior agregando una columna de opciones como se muestra en el siguiente apartado todo dentro de un formulario con acciones a la consulta de la información descrito por como *ConsultaUsuarioEditar* el cual permitirá la consulta de la información a detalle de un usuario en cuestión localizado por su identificador. Para ello es requerido crear un Servlet con el nombre anteriormente mencionado este.

La columna se está descrita con el siguiente código:

**<**td**>**

**<**form method**=**"post" action**=**"<%= path %>/ConsultaUsuarioEditar"**>**

**<**button type**=**"submit" class**=**"btn btn-default" aria**-**label**=**"Left Align" value**=**"<%=u.getIdUsuario()%>" name**=**"editar"**>**

**<**span class**=**"glyphicon glyphicon-pencil" aria**-**hidden**=**"true"**></**span**>**

**</**button**>**

**</**form**>**

**</**td**>**

La modificación anterior envía mediante un botón de tipo *submit el* identificador del usuario a consultar obtenido desde el objeto y pasándolo por parámetro al valor del botón en cuestión.

Es necesario mencionar que el Servlet *ConsultaUsuarioEditar* utilizará el método *actualizarUsuario* descrito en la parte lógica *UsuariosManager el* Servlet utilizado estará definido con el siguiente código:

package Controlador**;**

**import** Logica**.**UsuarioManager**;**

**import** Modelo**.**Usuario**;**

**import** java**.**io**.**IOException**;**

**import** java**.**io**.**PrintWriter**;**

**import** jakarta**.**servlet**.**ServletException**;**

**import** jakarta**.**servlet**.**annotation**.**WebServlet**;**

**import** jakarta**.**servlet**.**http**.**HttpServlet**;**

**import** jakarta**.**servlet**.**http**.**HttpServletRequest**;**

**import** jakarta**.**servlet**.**http**.**HttpServletResponse**;**

**import** java**.**util**.**List**;**

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

public class ConsultaUsuarioEditar **extends** HttpServlet **{**

protected void processRequest**(**HttpServletRequest request**,** HttpServletResponse response**)**

**throws** ServletException**,** IOException **{**

response**.**setContentType**(**"text/html;charset=UTF-8"**);**

**try** **(**PrintWriter out **=** response**.**getWriter**())** **{**

int valorSubmit **=** Integer**.**parseInt**(**request**.**getParameter**(**"editar"**));**

UsuarioManager manager **=** **new** UsuarioManager**();**

Usuario **=** manager**.**GetUsuario**(**valorSubmit**);**

request**.**setAttribute**(**"usuarioRender"**,** usuario**);**

request**.**getRequestDispatcher**(**"/Vistas/AccionesUsuario/EditarUsuario.jsp"**).**forward**(**request**,** response**);**

**}**

**}**

@Override

protected void doGet**(**HttpServletRequest request**,** HttpServletResponse response**)**

**throws** ServletException**,** IOException **{**

processRequest**(**request**,** response**);**

**}**

@Override

protected void doPost**(**HttpServletRequest request**,** HttpServletResponse response**)**

**throws** ServletException**,** IOException **{**

processRequest**(**request**,** response**);**

**}**

@Override

public String getServletInfo**()** **{**

**return** "Short description"**;**

**}**

**}**

Cabe resaltar que la información que se obtendrá se enviará a un nuevo archivo JSP *EditarUsuario.jsp descrito* por el siguiente código, el cual mediante la utilización de los métodos del objeto se envían los datos a los diferentes apartados, se obtendrá la siguiente imagen 1.4.9.

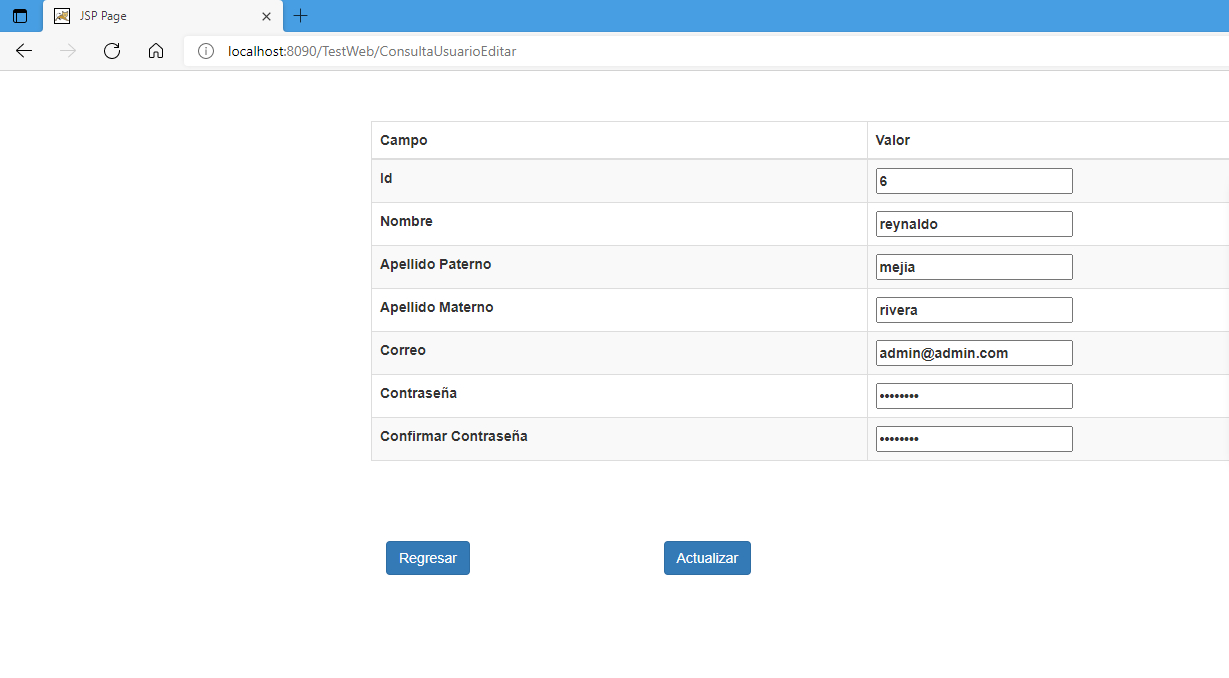


Imagen 1.4.9 Edición de usuarios.

**<%**@page **import=**"Controlador.ConsultaCatalogoTipoUsuarios" **%>**

**<%**@page **import=**"Modelo.Usuario" **%>**

**<%**@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**>**

**<**br**/>**

**<**br**/>**

**<**div class**=**"row col-lg-12"**>**

**<**div class**=**"container"**>**

**<%**

String error**=**""**;**

error**=(**String**)**request**.**getAttribute**(**"error"**);**

**if(**error**==null)**

error**=**""**;**

**%>**

**<**h4 style**=**"color:red"**><%=**error**%></**h4**>**

**</**div**>**

**</**div**>**

**<%** Usuario u**=(**Usuario**)**request**.**getAttribute**(**"usuarioRender"**);**

String uus**=**""**;**

**%>**

**<%** String path **=** request**.**getContextPath**();** **%>**

**<**div class**=**"row col-lg-12"**>**

**<**div class**=**"container"**>**

**<**form method**=**"post" action**=**"<%= path %>/EditarUsuario"**>**

**<**table class**=**"table table-striped table-bordered dt-responsive nowrap" **>**

**<**thead**>**

**<**tr**>**

**<**th**>**Campo**</**th**>**

**<**th**>**Valor**</**th**>**

**</**tr**>**

**</**thead**>**

**<**tbody**>**

**<**tr**>**

**<**th**>**Id**</**th**>**

**<**th**><**input type**=**"text" readonly**=**"true" name**=**"id" value**=**"<%= u.getIdUsuario()%>" **/></**th**>**

**</**tr**>**

**<**tr**>**

**<**th**>**Nombre**</**th**>**

**<**th**><**input type**=**"text" name**=**"nombre" value**=**"<%= u.getNombre()%>" **/></**th**>**

**</**tr**>**

**<**tr**>**

**<**th**>**Apellido Paterno**</**th**>**

**<**th**><**input type**=**"text" name**=**"apPaterno" value**=**"<%= u.getApPaterno()%>"**/></**th**>**

**</**tr**>**

**<**tr**>**

**<**th**>**Apellido Materno**</**th**>**

**<**th**><**input type**=**"text" name**=**"apMaterno" value**=**"<%= u.getApMaterno()%>"**/></**th**>**

**</**tr**>**

**<**tr**>**

**<**th**>**Correo**</**th**>**

**<**th**><**input type**=**"email" name**=**"correo" value**=**"<%= u.getCorreo()%>"**/></**th**>**

**</**tr**>**

**<**tr**>**

**<**th**>**Contraseña**</**th**>**

**<**th**><**input type**=**"password" name**=**"password" value**=**"<%= u.getPassword()%>"**/></**th**>**

**</**tr**>**

**<**tr**>**

**<**th**>**Confirmar Contraseña**</**th**>**

**<**th**><**input type**=**"password" name**=**"passwordConfirmar" value**=**"<%= u.getPassword()%>"**/></**th**>**

**</**tr**>**

**</**tbody**>**

**</**table>

**<**br**/>**

**<**br**/>**

**<**div class**=**"row col-md-12"**>**

**<**div class**=**"col-md-3"**>**

**<**button type**=**"submit" class**=**"btn btn-primary" aria**-**label**=**"Left Align" value**=**"1" name**=**"submitUsuario"**>**

Regresar

**</**button**>**

**</**div**>**

**<**div class**=**"col-md-3"**>**

**<**button type**=**"submit" class**=**"btn btn-primary" aria**-**label**=**"Left Align" value**=**"2" name**=**"submitUsuario"**>**

Actualizar

**</**button**>**

**</**div**>**

**</**div**>**

**</**form**>**

**</**div**>**

**</**div**>**

**</**body>

# 5.1.3Método Eliminar(Delete)

Para definir este método se requiere incorporar a la columna de opciones el código siguiente, el cual envía el identificador para utilizar el método el *EliminarUsuario* descrito en el archivo *UsuariosManager* establecido en el paquete *Logica* .

package Controlador**;**

**import** Logica**.**UsuarioManager**;**

**import** Modelo**.**Usuario**;**

**import** java**.**io**.**IOException**;**

**import** java**.**io**.**PrintWriter**;**

**import** jakarta**.**servlet**.**ServletException**;**

**import** jakarta**.**servlet**.**annotation**.**WebServlet**;**

**import** jakarta**.**servlet**.**http**.**HttpServlet**;**

**import** jakarta**.**servlet**.**http**.**HttpServletRequest**;**

**import** jakarta**.**servlet**.**http**.**HttpServletResponse**;**

**import** java**.**util**.**List**;**

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

public class EliminarUsuario **extends** HttpServlet **{**

protected void processRequest**(**HttpServletRequest request**,** HttpServletResponse response**)**

**throws** ServletException**,** IOException **{**

response**.**setContentType**(**"text/html;charset=UTF-8"**);**

**try** **(**PrintWriter out **=** response**.**getWriter**())** **{**

int valorSubmit **=** Integer**.**parseInt**(**request**.**getParameter**(**"eliminar"**));**

UsuarioManager manager **=** **new** UsuarioManager**();**

int estatus **=** manager**.**EliminaUsuario**(**valorSubmit**);**

**if** **(**estatus **==** 1**)** **{**

UsuarioManager mn3 **=** **new** UsuarioManager**();**

List**<**Usuario**>** usuarios **=** mn3**.**GetUsuarios**();**

request**.**setAttribute**(**"listaUsuariosRender"**,** usuarios**);**

request**.**getRequestDispatcher**(**"usuarios.jsp"**).**forward**(**request**,** response**);**

**}** **else** **{**

String error **=** "Error no se pudo eliminar el registro "**;**

request**.**setAttribute**(**"error"**,** error**);**

request**.**getRequestDispatcher**(**"/Vistas/AccionesUsuario/NuevoUsuario.jsp"**).**forward**(**request**,** response**);**

**}**

**}**

**}**

@Override

protected void doGet**(**HttpServletRequest request**,** HttpServletResponse response**)**

**throws** ServletException**,** IOException **{**

processRequest**(**request**,** response**);**

**}**

@Override

protected void doPost**(**HttpServletRequest request**,** HttpServletResponse response**)**

**throws** ServletException**,** IOException **{**

processRequest**(**request**,** response**);**

**}**

@Override

public String getServletInfo**()** **{**

**return** "Short description"**;**

**}**

**}**

Este Servlet elimina el usuario enviado y hace referencia y envía la actualización de usuarios a la pantalla principal descrita en el archivo *usuarios.jsp*  en caso de que la petición transcurra con éxito, en caso contrario envía un mensaje a la pantalla en la imagen 1.4.9

# 5.1.4Método Creación(Create)

Para generar este método es requerido incorporar el siguiente código al archivo *usuarios.jsp*  que nos permitirá crear elementos mediante un formulario incorporado en el archivo *NuevoUsuario.jsp.*

**<**div class**=**"row col-lg-12"**>**

**<**div class**=**"container"**>**

**<%** String path **=** request**.**getContextPath**();** **%>**

**<**button type**=**"button" class**=**"btn btn-primary" aria**-**label**=**"Left Align"

onclick**=**"location.href = '../TestWeb/Vistas/AccionesUsuario/NuevoUsuario.jsp'"**>**

Nuevo

**</**button**>**

**</**div**>**

**</**div**>**

La línea sombreada en el código anterior envía una solicitud de cambio de código entre JSP´s con la ruta absoluta de el JSP especifico donde se encuentra el formulario.

El siguiente código muestra el contenido del formulario de creación de usuarios establecido en el archivo *NuevoUsuario.jsp.*

**<%**@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**>**

**<**br**/>**

**<**br**/>**

**<**div class**=**"row col-lg-12"**>**

**<**div class**=**"container"**>**

**<%**

String error**=**""**;**

error**=(**String**)**request**.**getAttribute**(**"error"**);**

**if(**error**==null)**

error**=**""**;**

**%>**

**<**h4 style**=**"color:red"**><%=**error**%></**h4**>**

**</**div**>**

**</**div**>**

**<**br**/>**

**<**br**/>**

**<%** String path **=** request**.**getContextPath**();** **%>**

**<**div class**=**"row col-lg-12"**>**

**<**div class**=**"container"**>**

**<**form method**=**"post" action**=**"<%= path %>/NuevoUsuario"**>**

**<**br**/>**

**<**table class**=**"table table-striped table-bordered dt-responsive nowrap" **>**

**<**thead**>**

**<**tr**>**

**<**th**>**Campo**</**th**>**

**<**th**>**Valor**</**th**>**

**</**tr**>**

**</**thead**>**

**<**tbody**>**

**<**tr**>**

**<**th**>**Nombre**</**th**>**

**<**th**><**input type**=**"text" name**=**"nombre"**/></**th**>**

**</**tr**>**

**<**tr**>**

**<**th**>**Apellido Paterno**</**th**>**

**<**th**><**input type**=**"text" name**=**"apPaterno"**/></**th**>**

**</**tr**>**

**<**tr**>**

**<**th**>**Apellido Materno**</**th**>**

**<**th**><**input type**=**"text" name**=**"apMaterno"**/></**th**>**

**</**tr**>**

**<**tr**>**

**<**th**>**Correo**</**th**>**

**<**th**><**input type**=**"email" name**=**"correo"**/></**th**>**

**</**tr**>**

**<**tr**>**

**<**th**>**Contraseña**</**th**>**

**<**th**><**input type**=**"password" name**=**"password"**/></**th**>**

**</**tr**>**

**<**tr**>**

**<**th**>**Confirmar Contraseña**</**th**>**

**<**th**><**input type**=**"password" name**=**"passwordConfirmar"**/></**th**>**

**</**tr**>**

**</**tbody**>**

**</**table

**<**div class**=**"row col-md-12"**>**

**<**div class**=**"container"**>**

**<**div class**=**"col-md-3"**>**

**<**button type**=**"submit" class**=**"btn btn-primary" aria**-**label**=**"Left Align" value**=**"1" name**=**"submitUsuario"**>**

Regresar

**</**button**>**

**</**div**>**

**<**div class**=**"col-md-3"**>**

**<**input type**=**"reset" class**=**"btn btn-primary" value**=**"Limpiar"**>**

**</**div**>**

**<**div class**=**"col-md-3"**>**

**<**button type**=**"submit" class**=**"btn btn-primary" aria**-**label**=**"Left Align" value**=**"2" name**=**"submitUsuario"**>**

Guardar

**</**button**>**

**</**div**>**

**</**div**>**

**</**div**>**

**</**div**>**

**</**form**>**

**</**div**>**

**</**body**>**

**</**html**>**

El resultado del código anterior esta descrito en la imagen 1.5.0 que muestra el formulario donde incorpora los datos necesarios para generar un registro en la tabla *Usuarios* pera ello es necesario crear un nuevo archivo Servlet dentro del paquete *Controlador* descrito con el nombre *NuevoUsuario.java* el cual recibe los parámetros y crea un objeto de tipo *Usuarios.java* descrito con los métodos necesarios para llenar la información mediante *getters y setters.* La respuesta de la solicitud tiene como hemos visto la actualización de la tabla en *usuarios.jsp* descrito en el siguiente código.

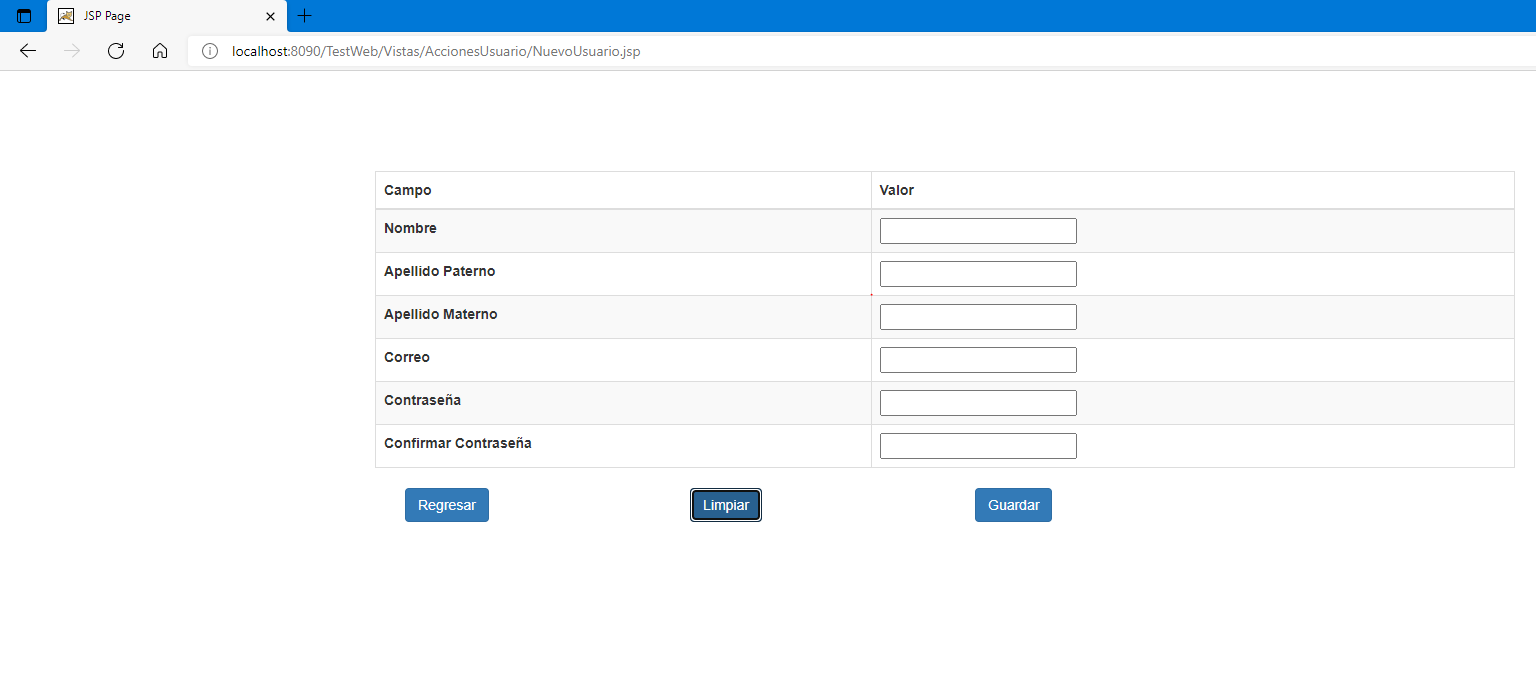


Imagen 1.5.0 Formulario de creación de usuarios.

package Controlador**;**

**import** Logica**.**UsuarioManager**;**

**import** Modelo**.**Usuario**;**

**import** java**.**io**.**IOException**;**

**import** java**.**io**.**PrintWriter**;**

**import** jakarta**.**servlet**.**ServletException**;**

**import** jakarta**.**servlet**.**annotation**.**WebServlet**;**

**import** jakarta**.**servlet**.**http**.**HttpServlet**;**

**import** jakarta**.**servlet**.**http**.**HttpServletRequest**;**

**import** jakarta**.**servlet**.**http**.**HttpServletResponse**;**

**import** java**.**util**.**List**;**

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

public class NuevoUsuario **extends** HttpServlet **{**

protected void processRequest**(**HttpServletRequest request**,** HttpServletResponse response**)**

**throws** ServletException**,** IOException **{**

response**.**setContentType**(**"text/html;charset=UTF-8"**);**

**try** **(**PrintWriter out **=** response**.**getWriter**())** **{**

int valorSubmit **=** Integer**.**parseInt**(**request**.**getParameter**(**"submitUsuario"**));**

**if** **(**valorSubmit **==** 2**)** **{**

String nombre **=** request**.**getParameter**(**"nombre"**);**

String app **=** request**.**getParameter**(**"apPaterno"**);**

String apm **=** request**.**getParameter**(**"apMaterno"**);**

String correo **=** request**.**getParameter**(**"correo"**);**

String password **=** request**.**getParameter**(**"password"**);**

String passwordConfirmar **=** request**.**getParameter**(**"passwordConfirmar"**);**

**if** **(**password**.**equals**(**passwordConfirmar**))** **{**

Usuario usuario **=** **new** Usuario**();**

usuario**.**setNombre**(**nombre**);**

usuario**.**setApPaterno**(**app**);**

usuario**.**setApMaterno**(**apm**);**

usuario**.**setCorreo**(**correo**);**

usuario**.**setPassword**(**password**);**

UsuarioManager manager **=** **new** UsuarioManager**();**

int registro **=** manager**.**RegistraUsuario**(**usuario**);**

**if** **(**registro **==** 1**)** **{**

UsuarioManager mn **=** **new** UsuarioManager**();**

List**<**Usuario**>** usuarios **=** mn**.**GetUsuarios**();**

request**.**setAttribute**(**"listaUsuariosRender"**,** usuarios**);**

request**.**getRequestDispatcher**(**"usuarios.jsp"**).**forward**(**request**,** response**);**

**}** **else** **{**

String error **=** "Error al generar el registro del usuario"**;**

request**.**setAttribute**(**"error"**,** error**);**

request**.**getRequestDispatcher**(**"/Vistas/AccionesUsuario/NuevoUsuario.jsp"**).**forward**(**request**,** response**);**

**}**

**}** **else** **{**

String error **=** "Error las contraseñas deben coincidir "**;**

request**.**setAttribute**(**"error"**,** error**);**

request**.**getRequestDispatcher**(**"/Vistas/AccionesUsuario/NuevoUsuario.jsp"**).**forward**(**request**,** response**);**

**}**

**}** **else** **{**

UsuarioManager mn3 **=** **new** UsuarioManager**();**

List**<**Usuario**>** usuarios **=** mn3**.**GetUsuarios**();**

request**.**setAttribute**(**"listaUsuariosRender"**,** usuarios**);**

request**.**getRequestDispatcher**(**"usuarios.jsp"**).**forward**(**request**,** response**);**

**}**

**}**

**}**

@Override

protected void doGet**(**HttpServletRequest request**,** HttpServletResponse response**)**

**throws** ServletException**,** IOException **{**

processRequest**(**request**,** response**);**

**}**

@Override

protected void doPost**(**HttpServletRequest request**,** HttpServletResponse response**)**

**throws** ServletException**,** IOException **{**

processRequest**(**request**,** response**);**

**}**

@Override

public String getServletInfo**()** **{**

**return** "Short description"**;**

**}**

**}**