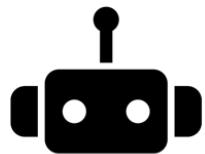




PUCP

SESIÓN DE LABORATORIO 1

Teoría de Aplicaciones Web



HORARIO 10M1

Empezaremos a las 7:10 p.m
Gracias!

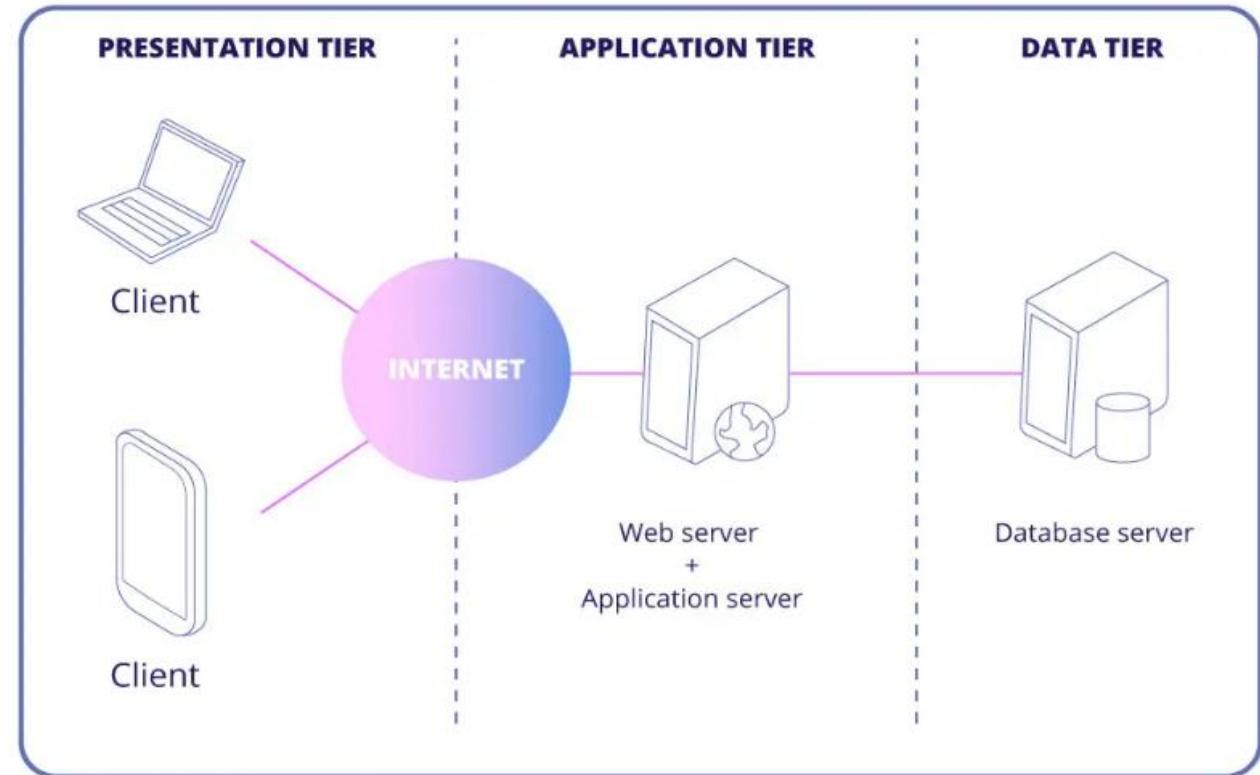


Teoría de aplicaciones web

Una aplicación web , es un software que corre en un servidor web, donde la data necesaria es procesada antes de ser mostrada

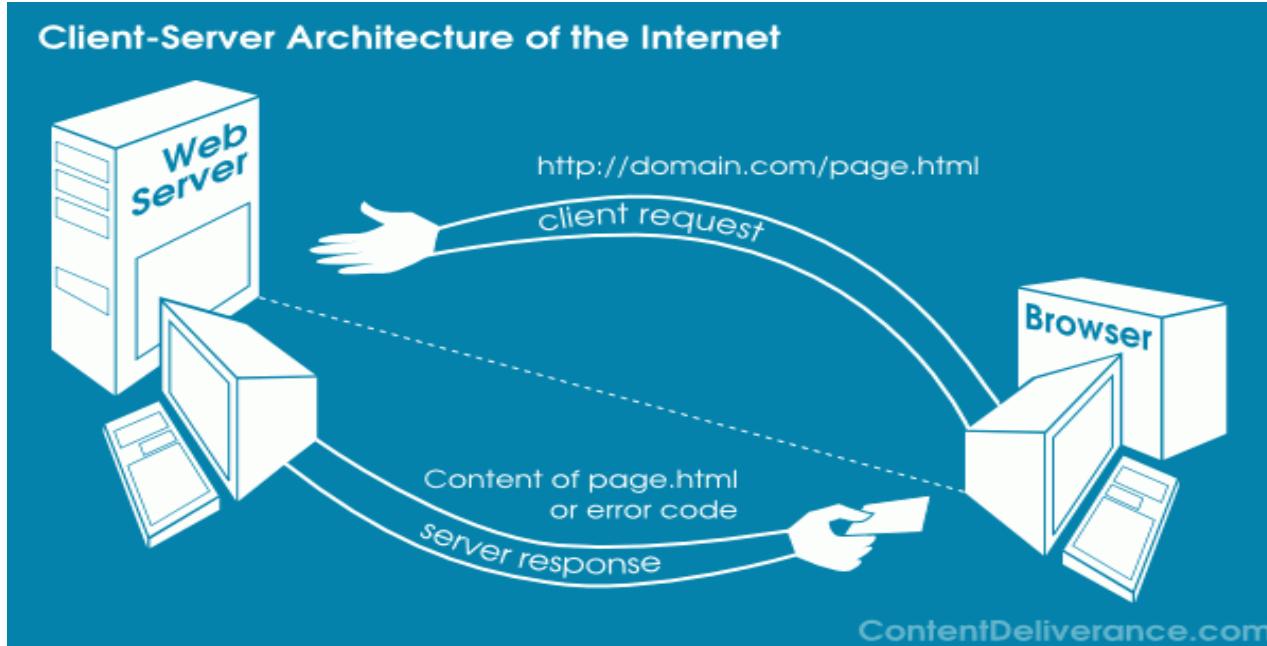
Ejemplos

- Google
- Correo electrónico
- Intranet PUCP
- Redes sociales
- Wikipedia
- etc...



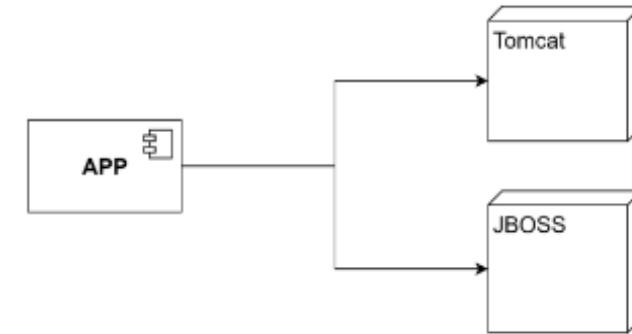
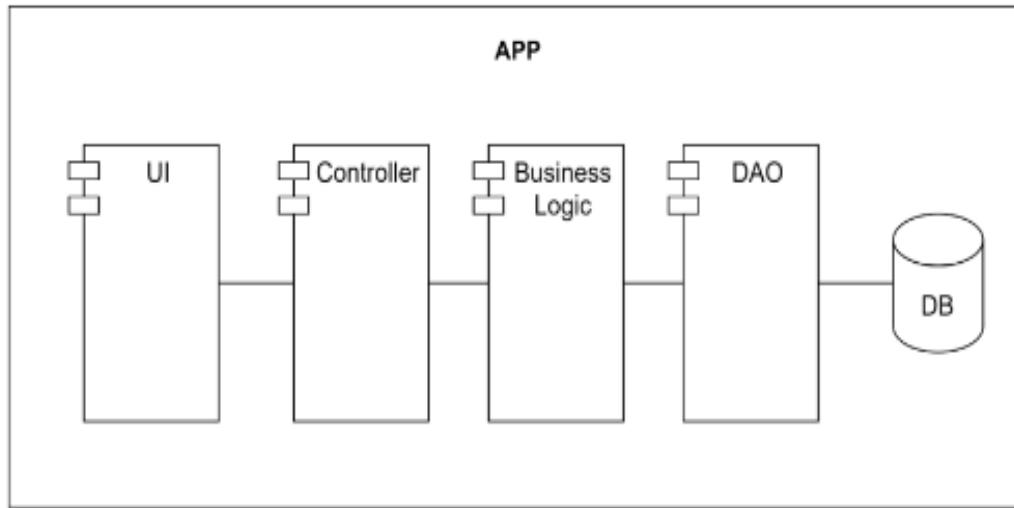
Teoría de aplicaciones web

Arquitectura Cliente - Servidor



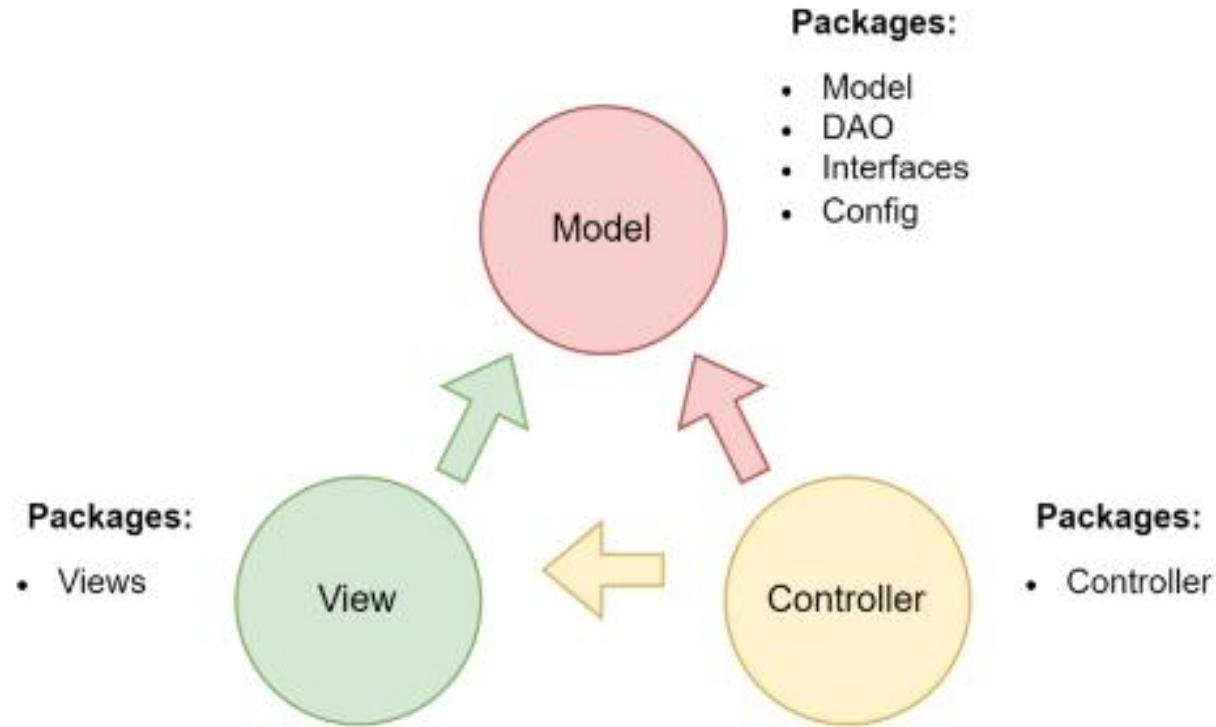
Teoría de aplicaciones web

Arquitectura monolítica



Teoría de aplicaciones web

Arquitectura por capas



Desarrollo de nuestra aplicación

Funcionalidades :

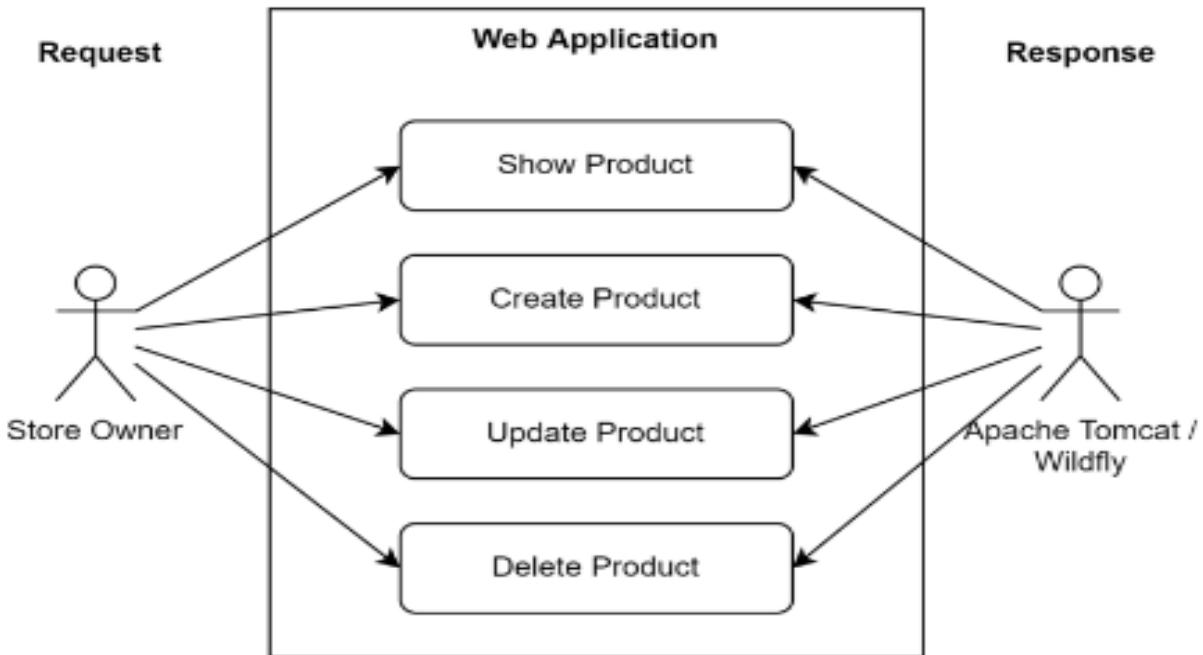
- Listado de productos
- Creación de productos
- Modificación de productos
- Eliminación de productos

Productos

Nuevo +

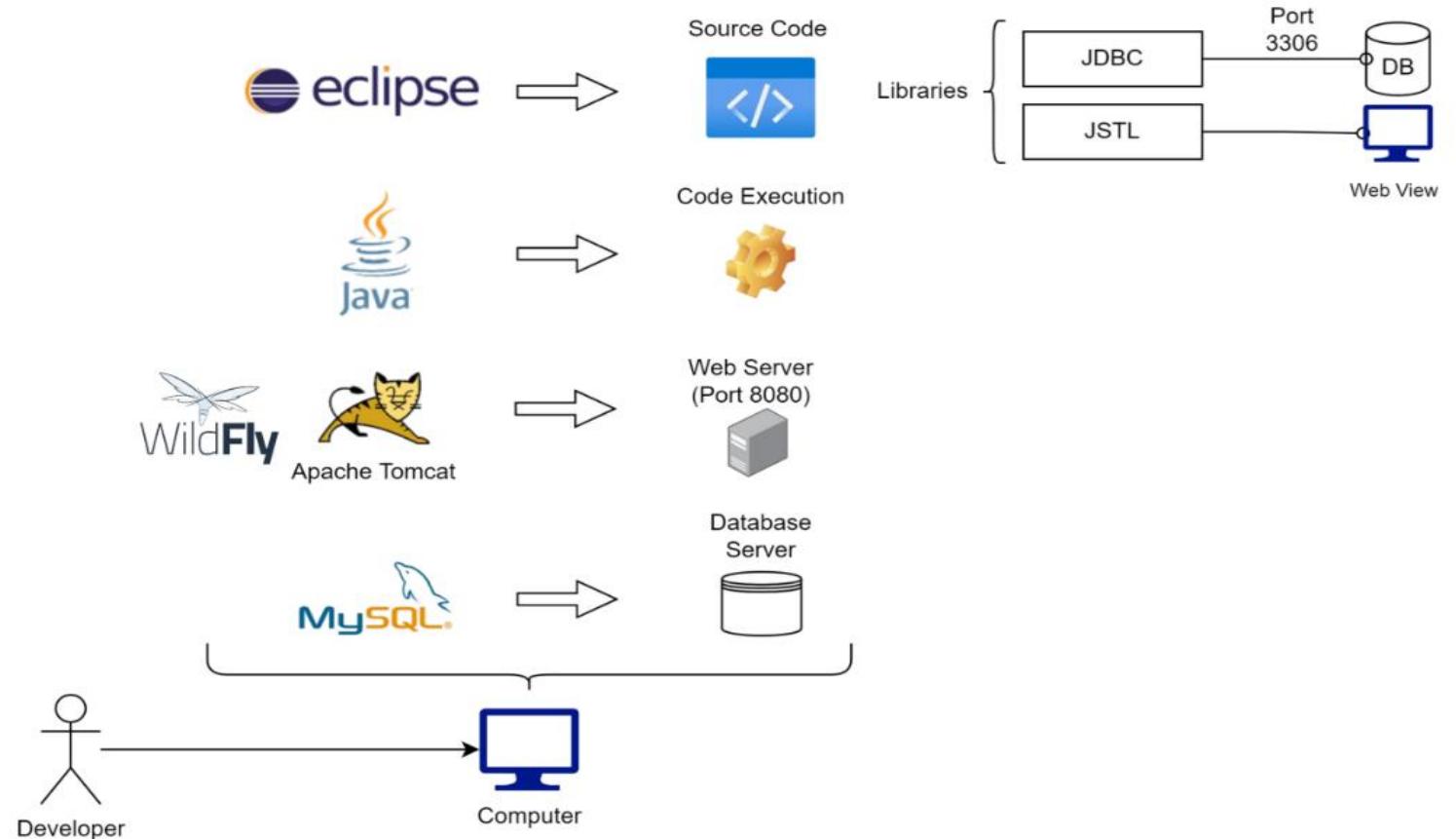
ID	Código	Descripción	Precio de compra	Precio de venta	Existencia	Editar	Eliminar
1	1	Galletas chokis	10.00	15.00	2.00		
2	2	Mermelada de fresa	65.00	80.00	97.00		
3	3	Aceite	18.00	20.00	100.00		
4	4	Palomitas de maíz	12.00	15.00	98.00		
5	5	Doritos	5.00	8.00	99.00		

CRUD



Requisitos

- Java SE 21
- Eclipse IDE
- Apache Tomcat 9
- WildFly (JBoss Community Version)
- MySQL Server
- MySQL Connector for Java 8



Instalaciones

Java SE 21 Instalación (1/4)

(Nota: Para poder descargar desde Oracle es necesario tener una cuenta)

El link pueden encontrarlo directamente en GitHub

Una vez accedido ,deben seleccionar el instalador para windows en el apartado de **Java SE Development Kit 21.0.8**

Java SE Development Kit 21.0.8

This software is licensed under the [Oracle No-Fee Terms and Conditions License](#).

Product / File Description	File Size	Download
Linux Arm 64 Compressed Archive	186.07 MB	https://download.oracle.com/java/21/archive/jdk-21.0.8_linux-aarch64_bin.tar.gz (sha256)
Linux Arm 64 RPM Package	185.76 MB	https://download.oracle.com/java/21/archive/jdk-21.0.8_linux-aarch64_bin.rpm (sha256) (OL 9 GPG Key)
Linux x64 Compressed Archive	187.89 MB	https://download.oracle.com/java/21/archive/jdk-21.0.8_linux-x64_bin.tar.gz (sha256)
Linux x64 Debian Package	159.73 MB	https://download.oracle.com/java/21/archive/jdk-21.0.8_linux-x64_bin.deb (sha256)
Linux x64 RPM Package	187.55 MB	https://download.oracle.com/java/21/archive/jdk-21.0.8_linux-x64_bin.rpm (sha256) (OL 9 GPG Key)

Instalaciones

Java SE 21 Instalación (2/4)

Deben aceptar los términos de licencia

Una vez hagan sign up, la descarga empezará

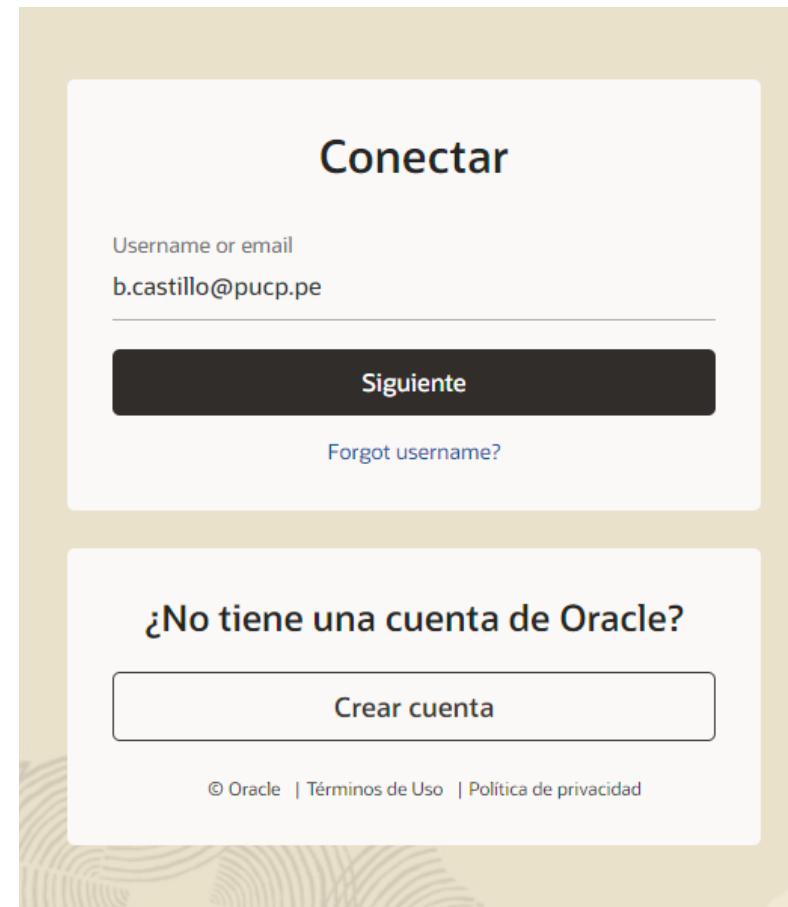
You must accept the [Oracle Technology Network License Agreement for Oracle Java SE](#) to download this software.

I reviewed and accept the Oracle Technology Network License Agreement for Oracle Java SE

Required

You will be redirected to the login screen in order to download the file.

[Download jdk-8u411-windows-x64.exe](#) 

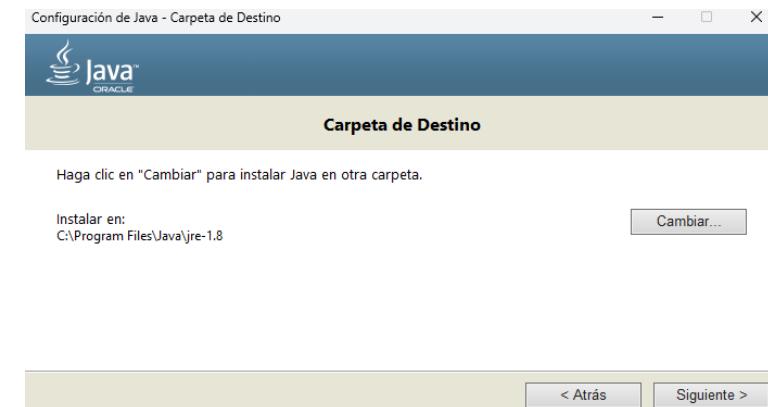
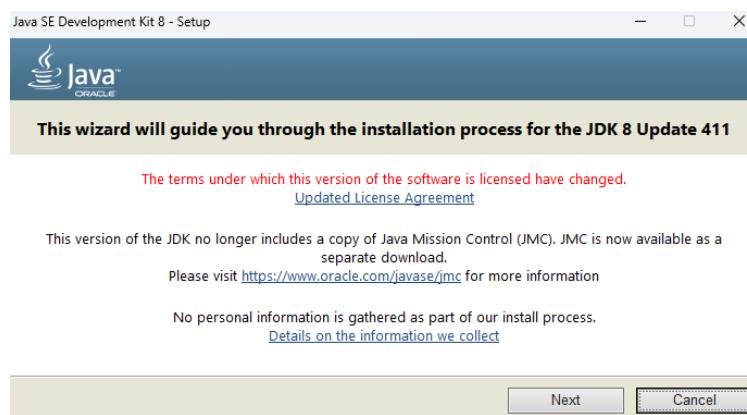
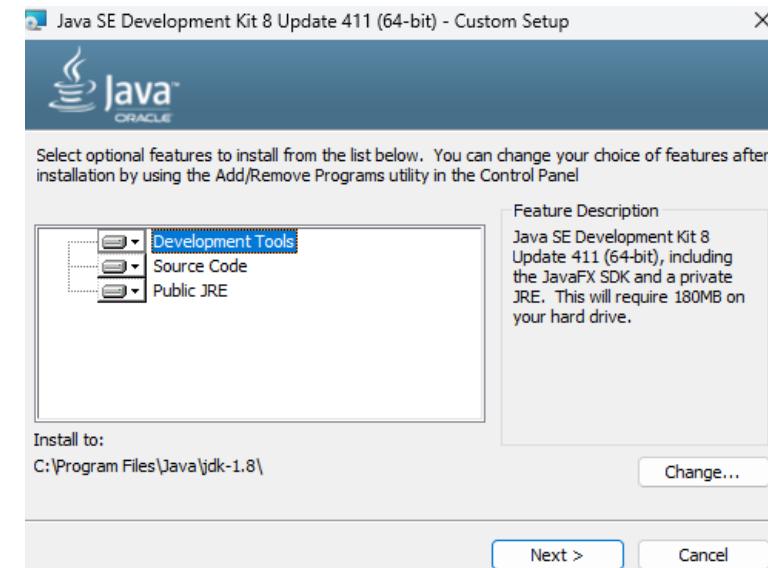


Instalaciones

Java SE 21 Instalación (3/4)

Luego deben de ejecutar el instalador

Deberán hacer click en siguiente en cada pantalla

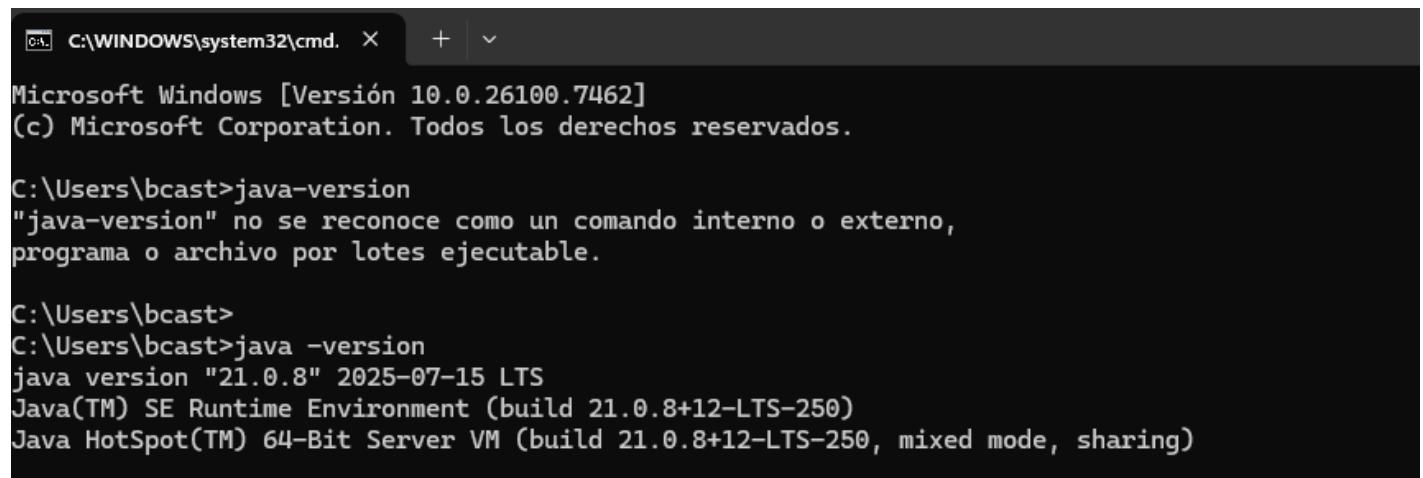


Instalaciones

Java SE 21 Instalación (4/4)

Para poder verificar que todo se ha instalado correctamente

Deben hacer Win + R →
CMD → java -version



```
C:\WINDOWS\system32\cmd. x + 
Microsoft Windows [Versión 10.0.26100.7462]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\bcast>java-version
"java-version" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

C:\Users\bcast>
C:\Users\bcast>java -version
java version "21.0.8" 2025-07-15 LTS
Java(TM) SE Runtime Environment (build 21.0.8+12-LTS-250)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.8+12-LTS-250, mixed mode, sharing)
```

Instalaciones

Eclipse IDE Instalación (1/2)

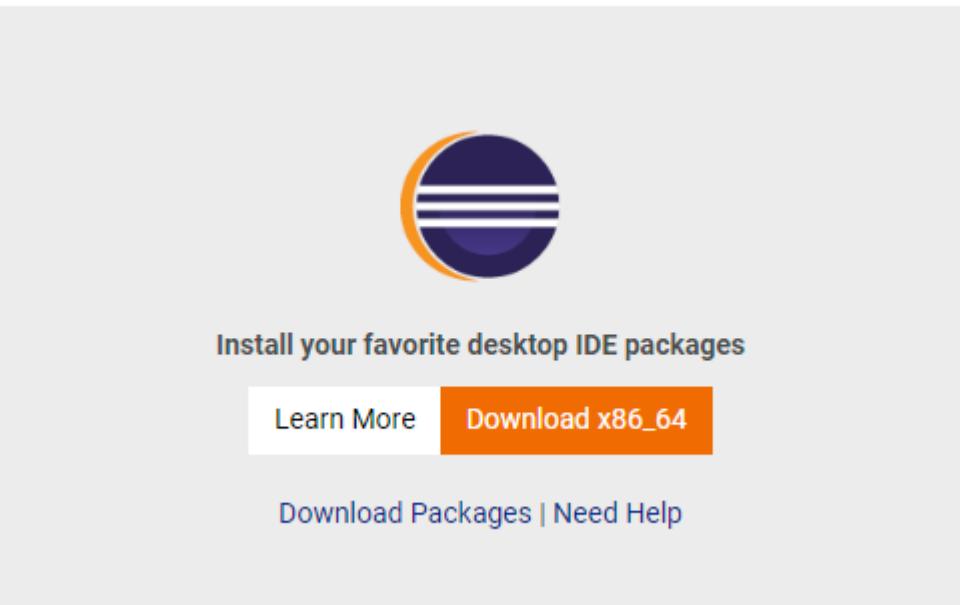
El link esta disponible en GitHub , simplemente deben darle clic y ejecutarlo cuando termine la descarga

 Download

Download from: United States - University of Maryland ([https](https://))

File: [eclipse-inst-jre-win64.exe](#) [SHA-512](#)

[>> Select Another Mirror](#)



Instalaciones

Eclipse IDE Instalación (2/2)

Al abrir el instalador , deberán seleccionar Eclipse IDE for Enterprise Java and Web Developers y seleccionar en instalar



Eclipse IDE for Enterprise Java and Web Developers

[details](#)

Tools for developers working with Java and Web applications, including a Java IDE, tools for JavaScript, TypeScript, JavaServer Pages and Faces, Yaml, Markdown, Web Services, JPA and Data Tools, Maven and Gradle, Git, and more.

Java 21+ VM JRE 21.0.9 - <https://download.eclipse.org/justj/jres/21/updates/release/latest> ▼

Installation Folder C:\Users\bcast\eclipse\jee-2025-122

[create start menu entry](#)

[create desktop shortcut](#)



Eclipse IDE for Enterprise Java and Web Developers

Tools for developers working with Java and Web applications, including a Java IDE, tools for JavaScript, TypeScript, JavaServer Pages and Faces, Yaml, Markdown, Web Services, JPA and Data Tools, Maven and...

[INSTALL](#)

Instalaciones

Apache Tomcat Instalación (1/1)

- El link de descarga esta accesible en el GitHub.
- Aquí deberemos usar la versión Zip del Binary Distributions Core y luego descomprimirlo y guardar la carpeta

11.0.15

Please see the [README](#) file for packaging information. It explains what every distribution contains.

Binary Distributions

- Core:
 - [zip \(pgp, sha512\)](#)
 - [tar.gz \(pgp, sha512\)](#)
 - [Windows zip \(pgp, sha512\)](#)
 - [Windows Service Installer \(pgp, sha512\)](#)
- Full documentation:
 - [tar.gz \(pgp, sha512\)](#)
- Deployer:
 - [zip \(pgp, sha512\)](#)
 - [tar.gz \(pgp, sha512\)](#)
- Embedded:
 - [tar.gz \(pgp, sha512\)](#)
 - [zip \(pgp, sha512\)](#)

Source Code Distributions

- [tar.gz \(pgp, sha512\)](#)
- [zip \(pgp, sha512\)](#)

Instalaciones

JSTL Instalación (1/1)

Es una librería para manejar tags en programas java

El link de descarga se encuentra en el GitHub

Aqui se deberán escoger las opciones que contienen Impl y Spec

Finalmente se deberán poner en el mismo folder que Apache

Standard-1.2.5

Source Code Distributions

- [Source README](#)
- [zip \(pgp, sha512\)](#)

Jar Files

- [Binary README](#)
- Impl:
 - [taglibs-standard-impl-1.2.5.jar \(pgp, sha512\)](#)
- Spec:
 - [taglibs-standard-spec-1.2.5.jar \(pgp, sha512\)](#)
- EL:
 - [taglibs-standard-jstl-el-1.2.5.jar \(pgp, sha512\)](#)
- Compat:
 - [taglibs-standard-compat-1.2.5.jar \(pgp, sha512\)](#)

Instalaciones

MySQL Server Instalación (1/12)

El link de descarga se encuentra en GitHub

Aquí se usará la versión 8.0.40

MySQL Installer 8.0.40



Note: MySQL 8.0 is the final series with MySQL Installer. As of MySQL 8.1, use a MySQL product's MSI or Zip archive for installation. MySQL Server 8.1 and higher also bundle MySQL Configurator, a tool that helps configure MySQL Server.

Select Version:

8.0.40

Select Operating System:

Microsoft Windows

[Windows \(x86, 32-bit\), MSI Installer](#)

(mysql-installer-web-community-8.0.40.0.msi)

8.0.40

2.1M

[Download](#)

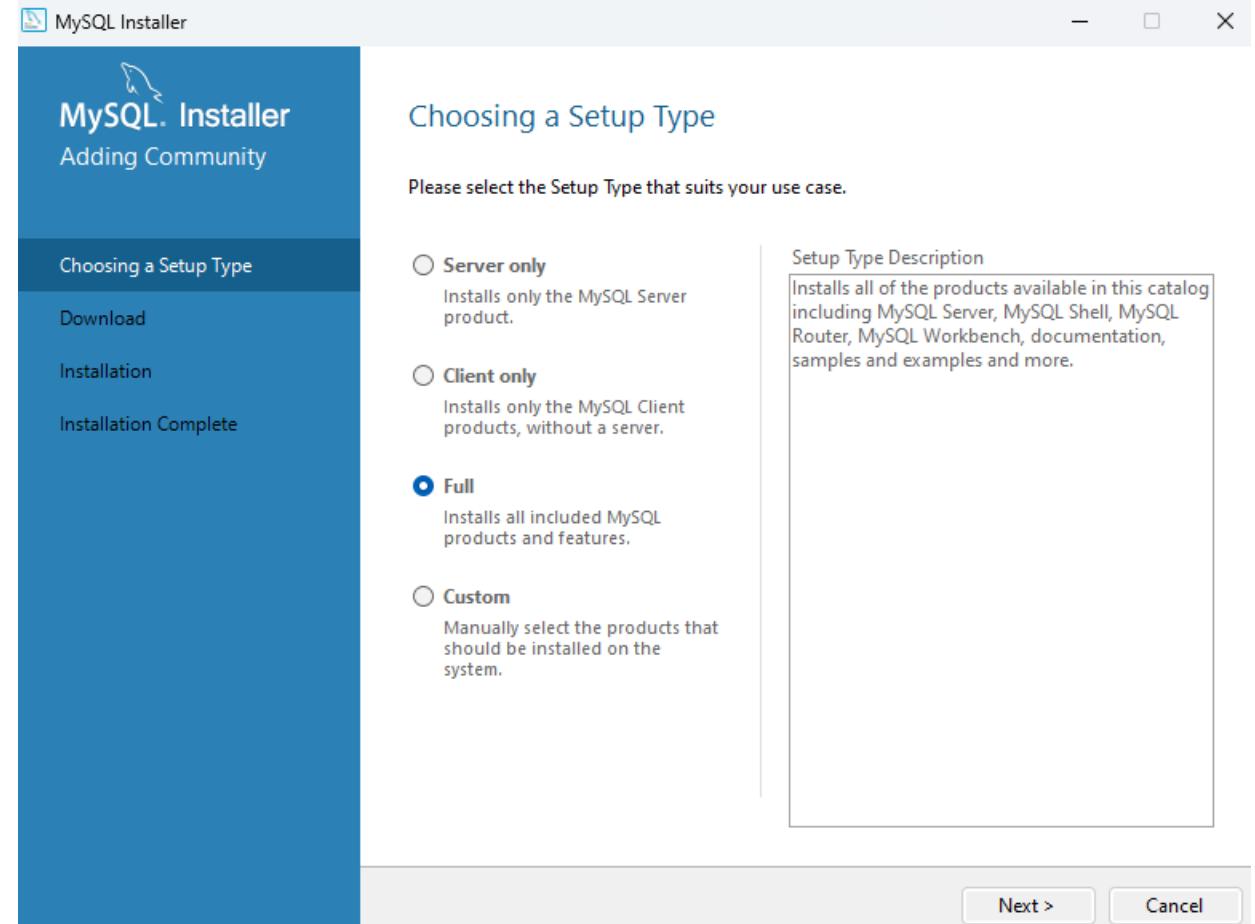
MD5: e3b10c3cd4be4bbdf4f8a23afe375917 | [Signature](#)

Instalaciones

MySQL Server Instalación (2/12)

Una vez se inicie la instalación
haremos clic en Full

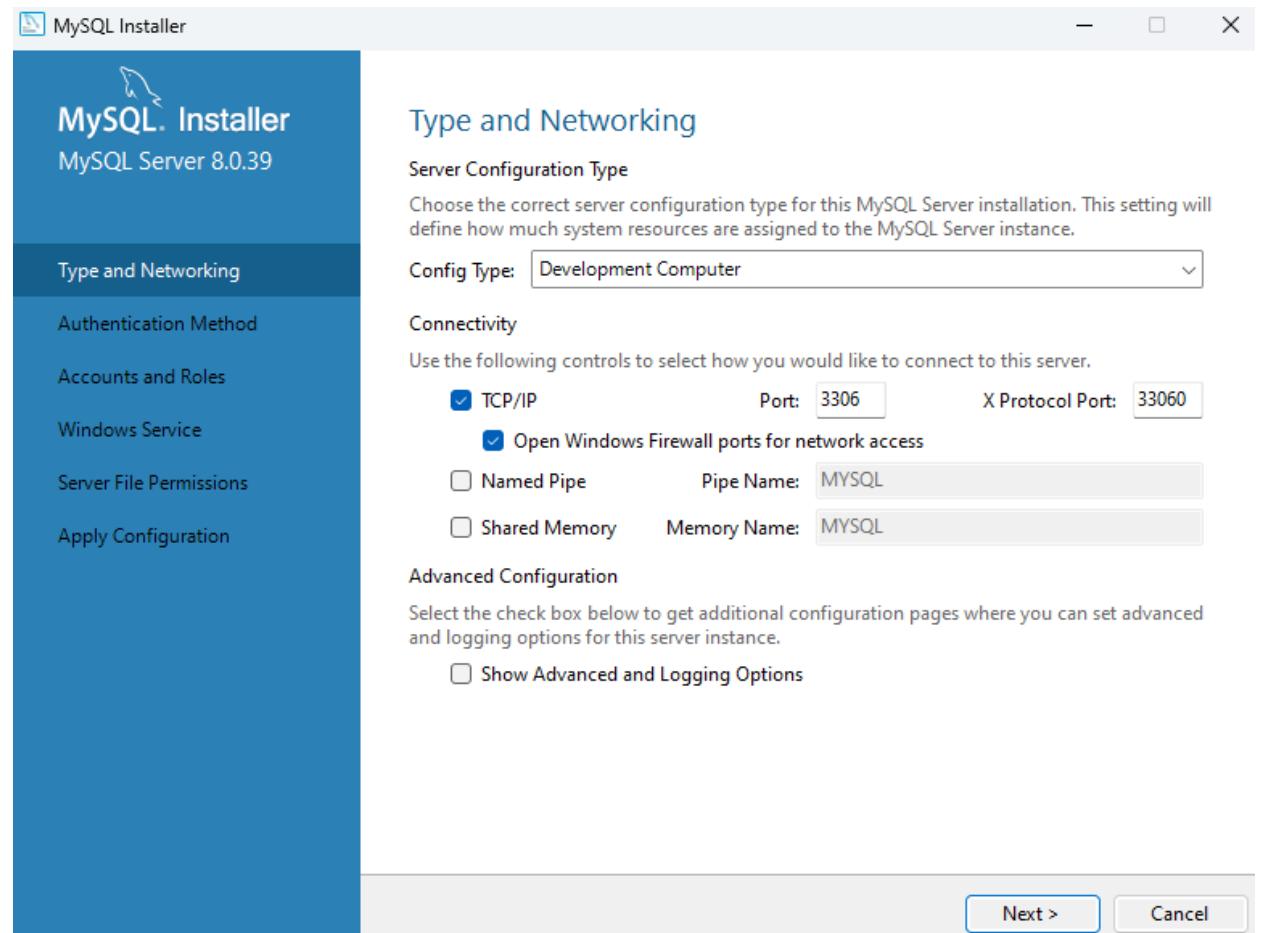
Luego haremos click en Next y en
Execute



Instalaciones

MySQL Server Instalación (3/12)

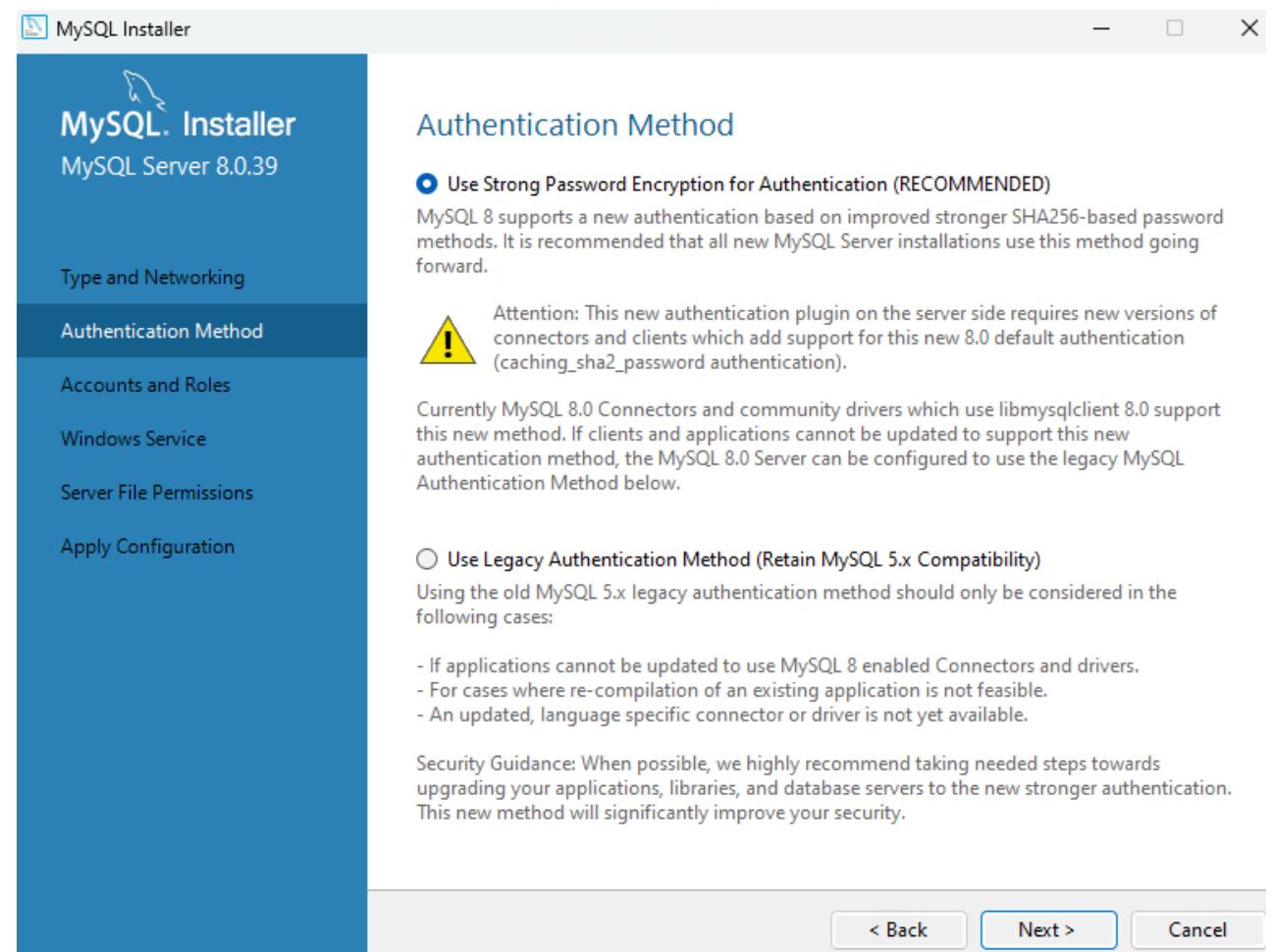
Una vez terminada debemos asegurarnos que el config Type está en Development Computer y apuntar los valores de Port : 3306 y X Protocol Port 33060



Instalaciones

MySQL Server Instalación (4/12)

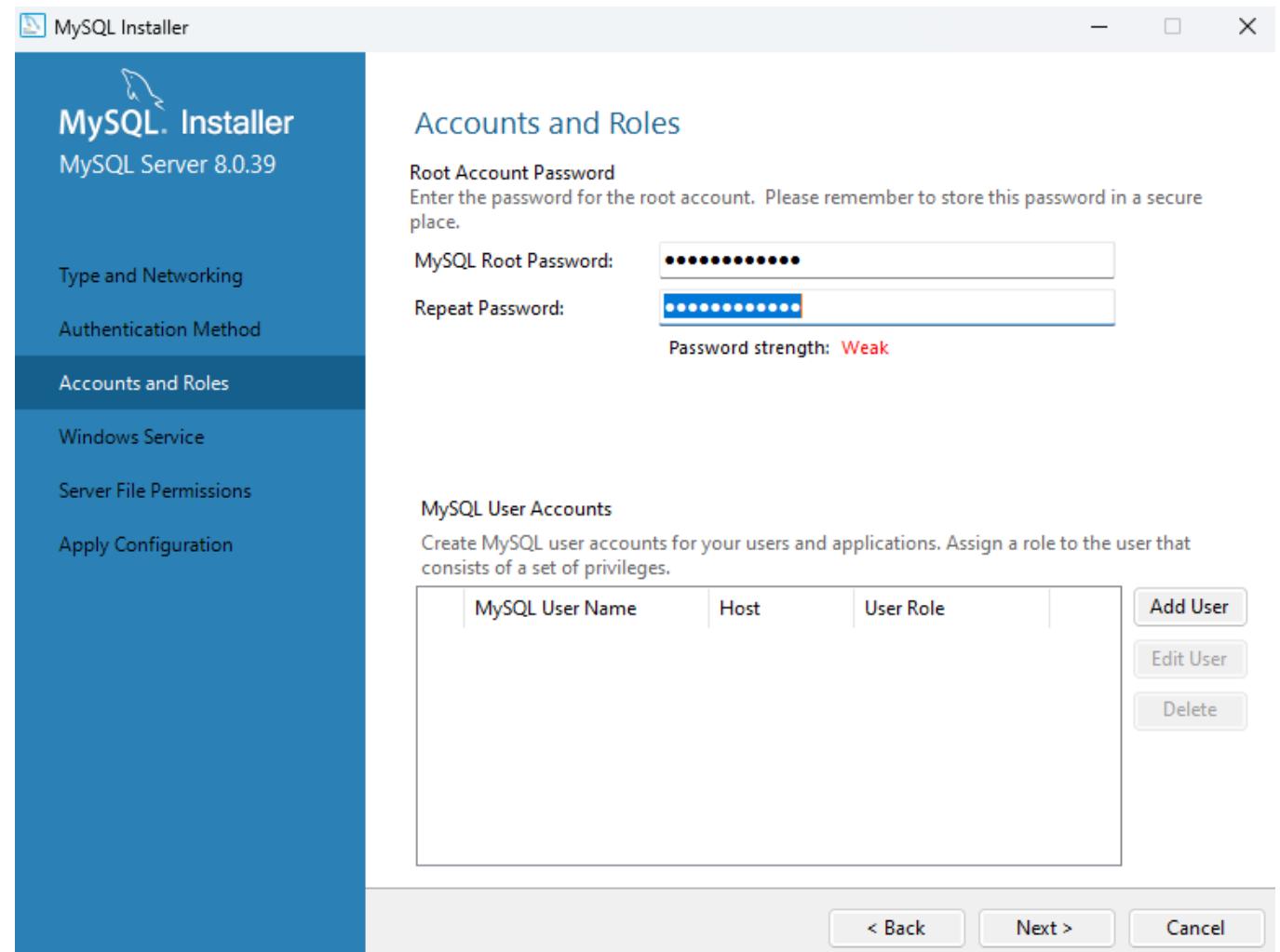
En el siguiente apartado se usará la opción recomendada para encriptación.



Instalaciones

MySQL Server Instalación (5/12)

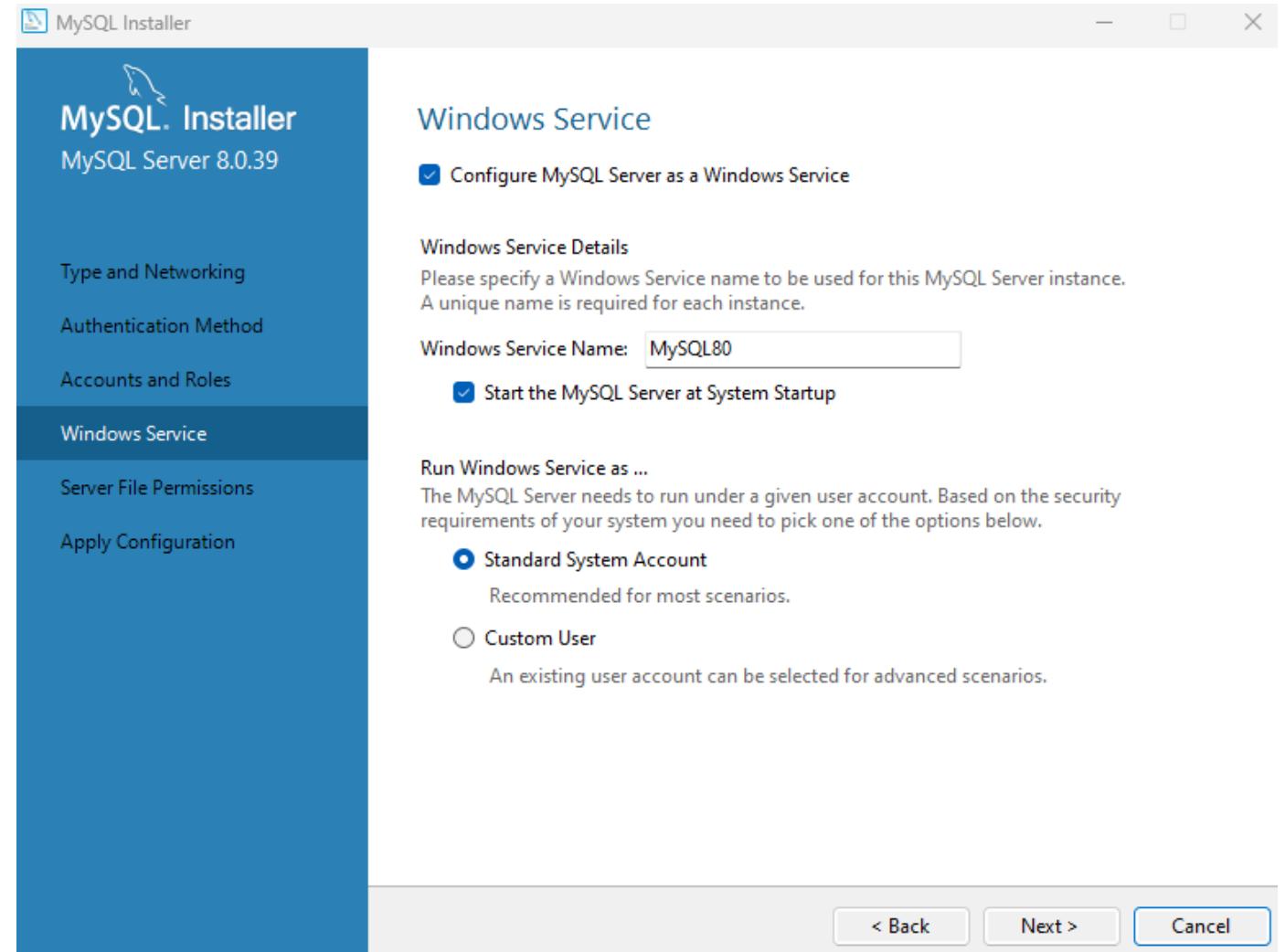
En el siguiente apartado debemos escoger una contraseña la cual debemos recordar dado que la usaremos para acceder a la base de datos



Instalaciones

MySQL Server Instalación (6/12)

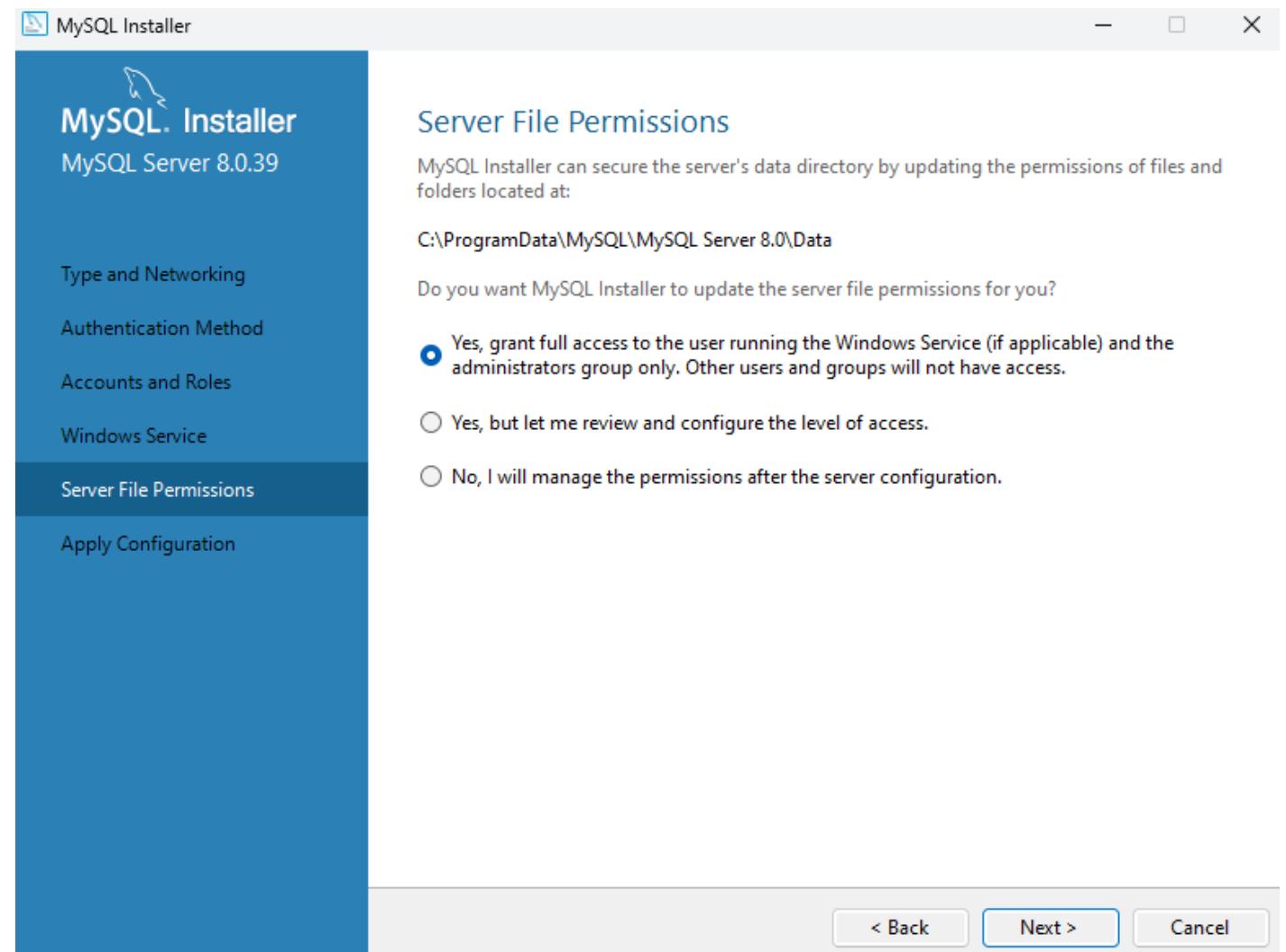
Luego deberemos aceptar la configuración por default de MySQL como servicio de Windows y aplicarlas



Instalaciones

MySQL Server Instalación (7/12)

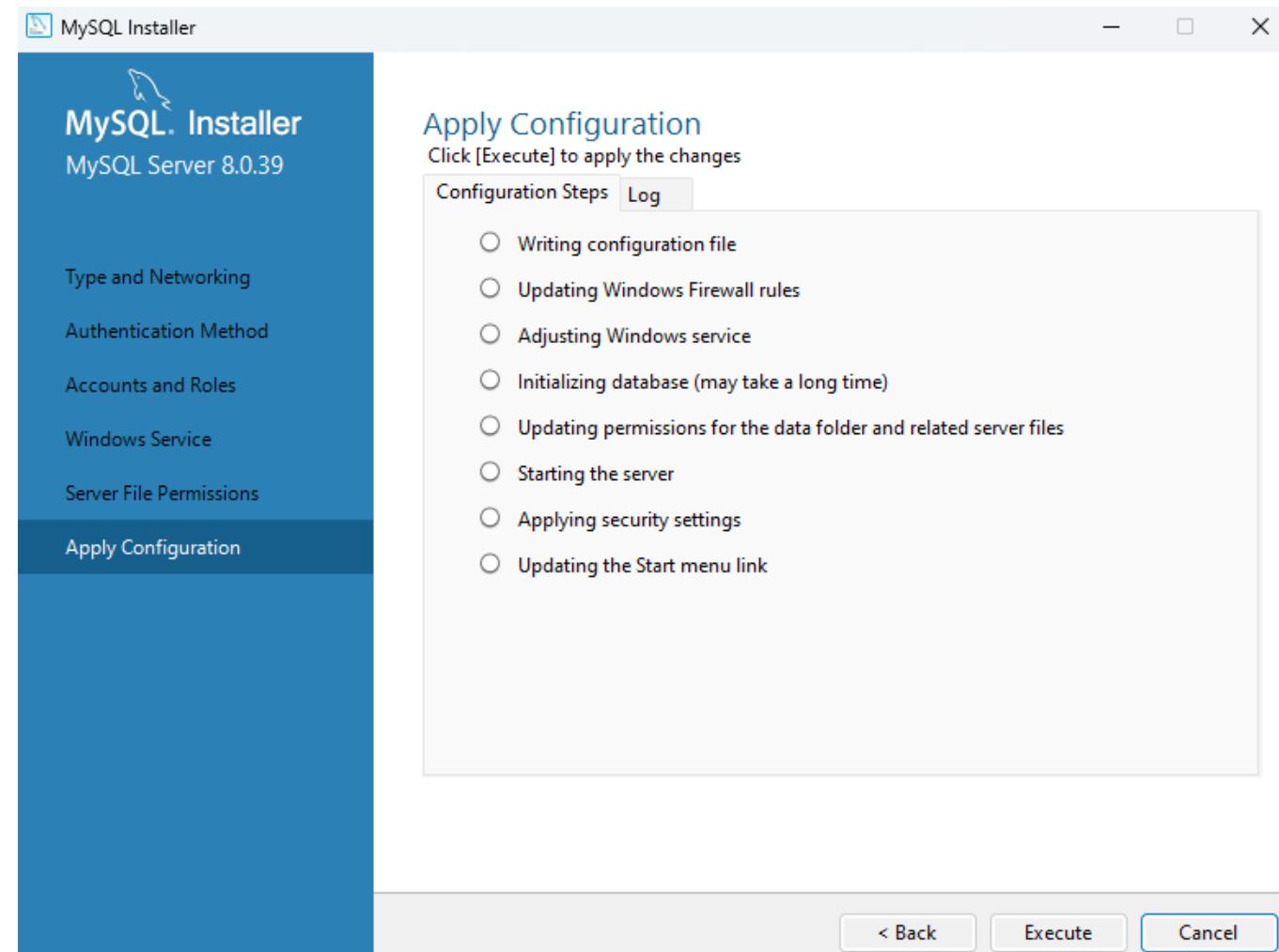
Luego debemos asignar permisos totales



Instalaciones

MySQL Server Instalación (8/12)

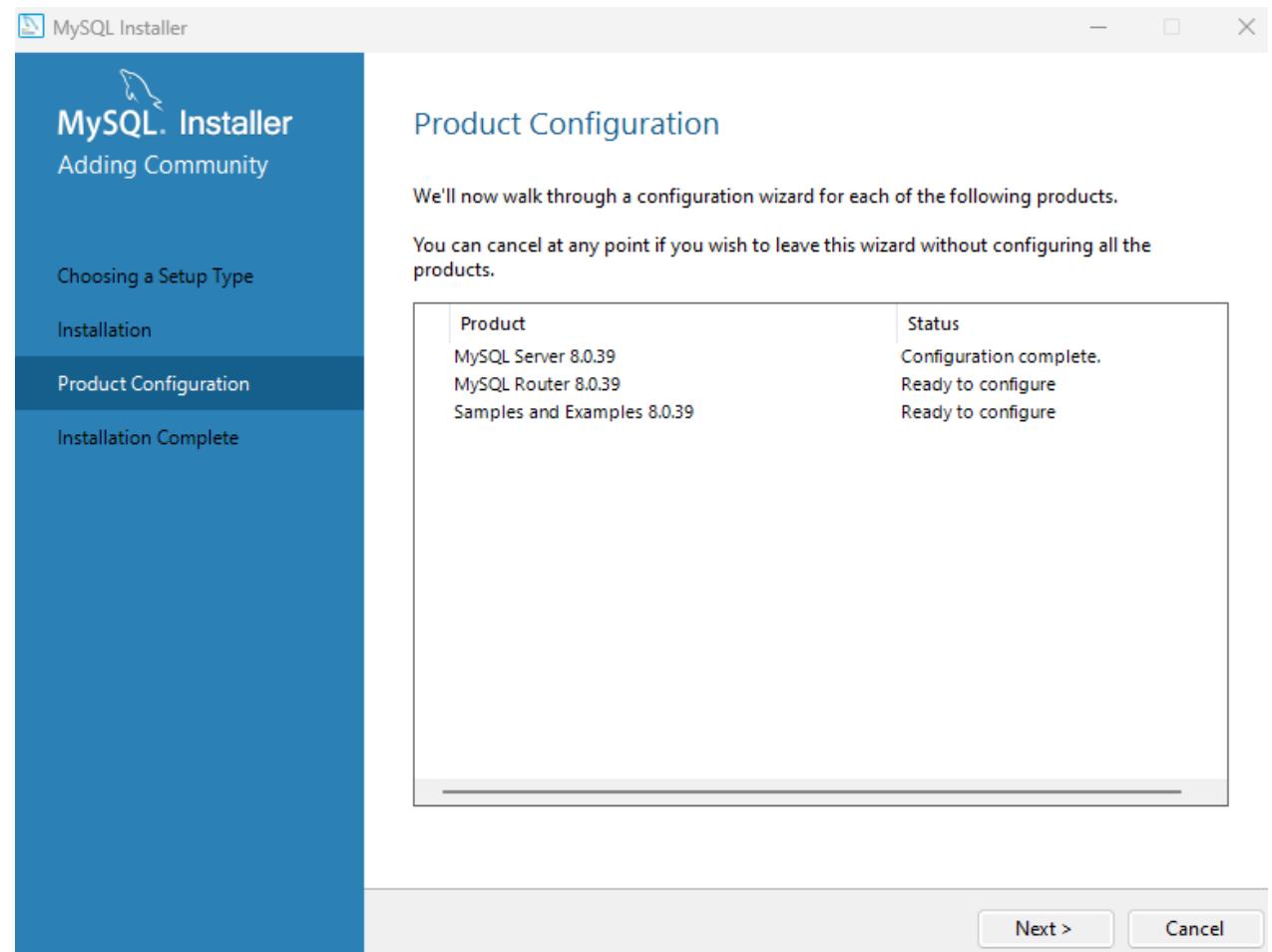
Finalmente se aplican las configuraciones dandole click en execute y finalmente en Finish



Instalaciones

MySQL Server Instalación (9/12)

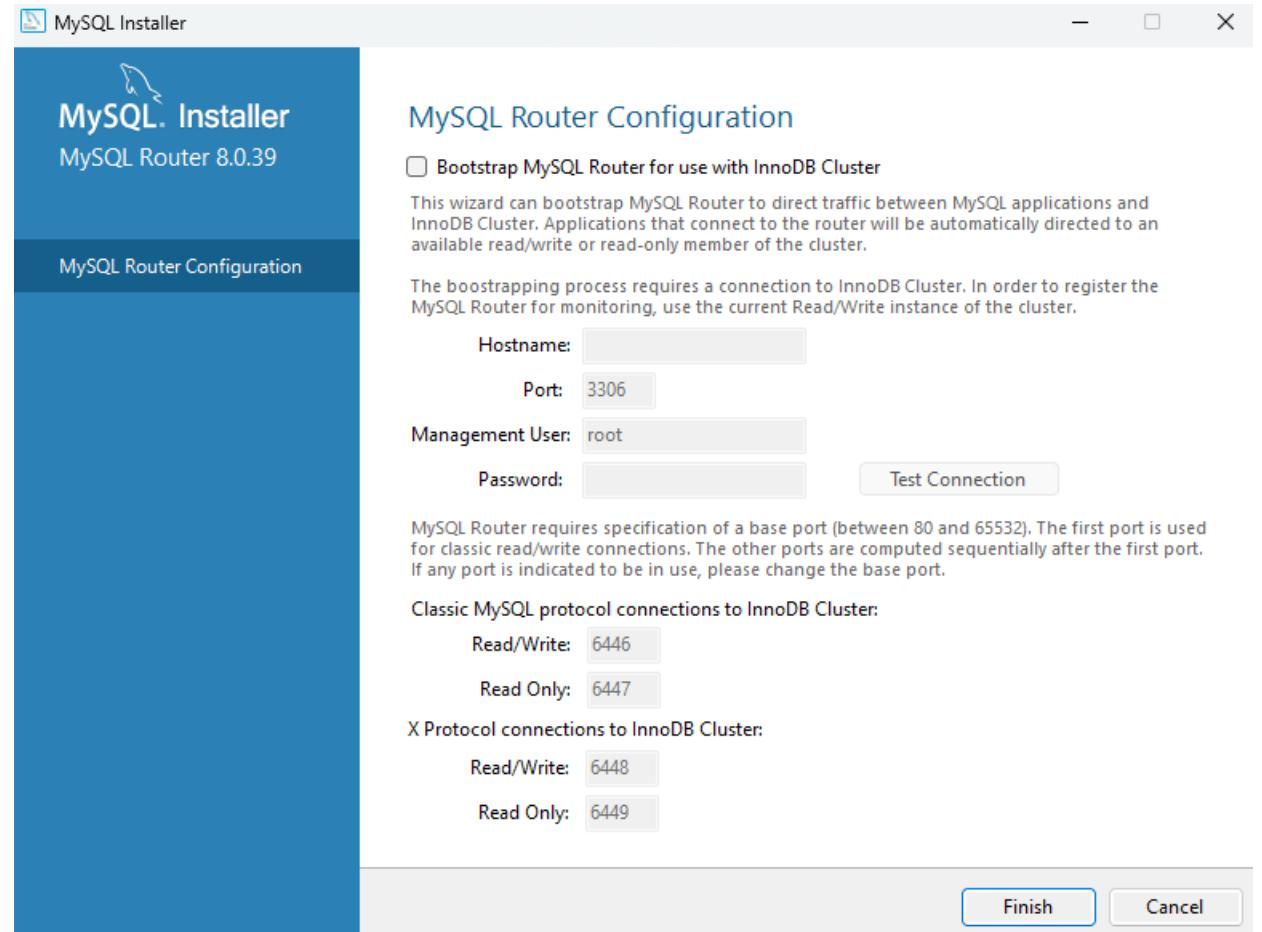
Continuando se deberá hacer la configuración de los otros productos



Instalaciones

MySQL Server Instalación (10/12)

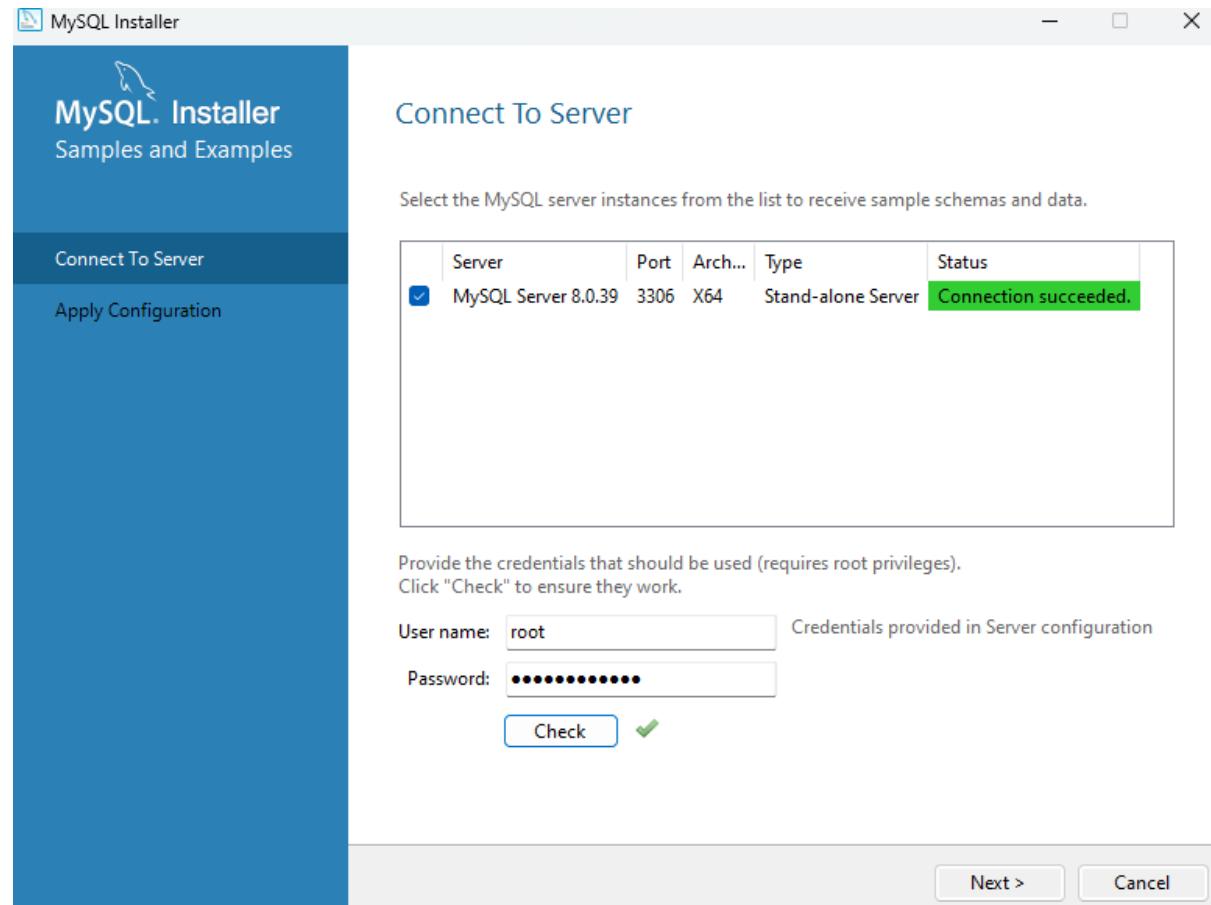
Respecto a la configuración de Router se debe mantener la configuración y solo dar en finish



Instalaciones

MySQL Server Instalación (11/12)

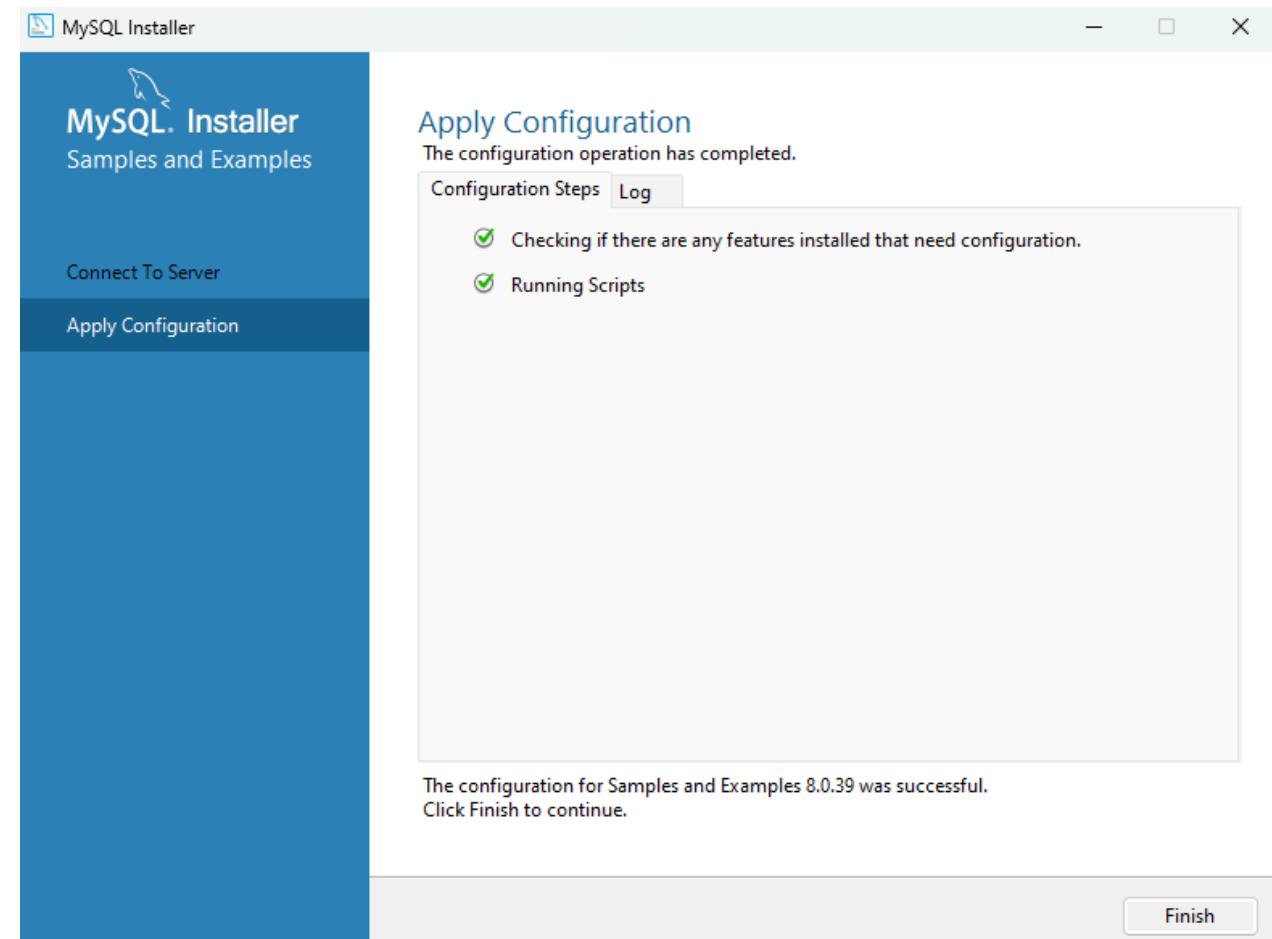
Finalmente se debe conectar al servidor y aplicar la configuración haciendo una prueba de conexión



Instalaciones

MySQL Server Instalación (12/12)

Finalmente se debe completar la instalación



Instalaciones

MySQL Connector Download

El link de descarga se encuentra en GitHub

Al ingresar al link se debe seleccionar Plataforma Independiente y descargar.

Luego se debe extraer el archivo ZIP y mantener solo el archivo jar

MySQL Community Downloads

Connector/J

The screenshot shows the MySQL Community Downloads page for the Connector/J 9.0.0 release. At the top, there are tabs for "General Availability (GA) Releases" (which is selected), "Archives", and a help icon. Below the tabs, it says "Connector/J 9.0.0" and "Select Operating System: Platform Independent". There are two download options listed:

File Type	Version	Size	Action
Platform Independent (Architecture Independent), Compressed TAR Archive (mysql-connector-j-9.0.0.tar.gz)	9.0.0	4.3M	Download
Platform Independent (Architecture Independent), ZIP Archive (mysql-connector-j-9.0.0.zip)	9.0.0	5.1M	Download

Below the download links, there is a note: "We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download." with a warning icon.

ORACLE © 2024 Oracle

[Privacy](#) | [Do Not Sell My Info](#) | [Terms of Use](#) | [Trademark Policy](#) |

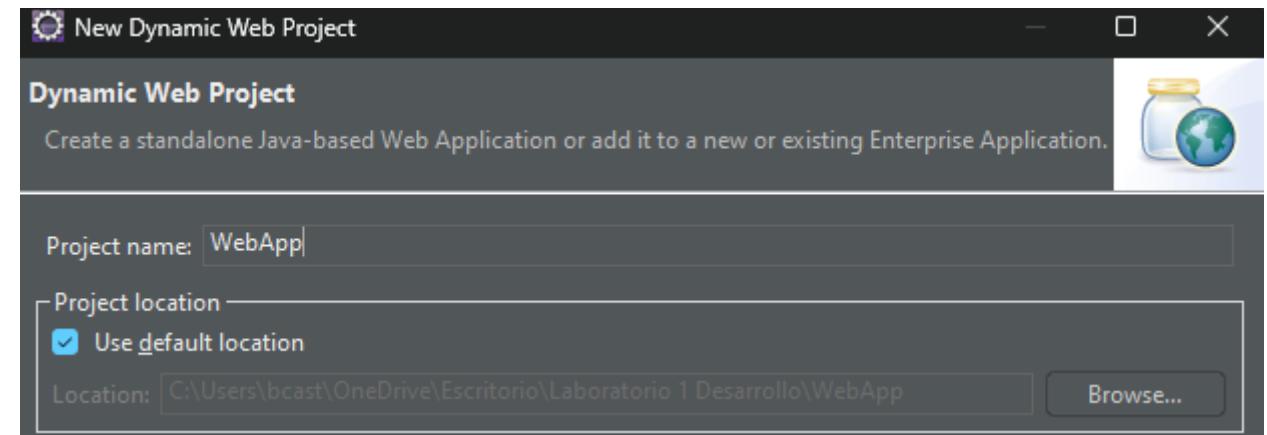
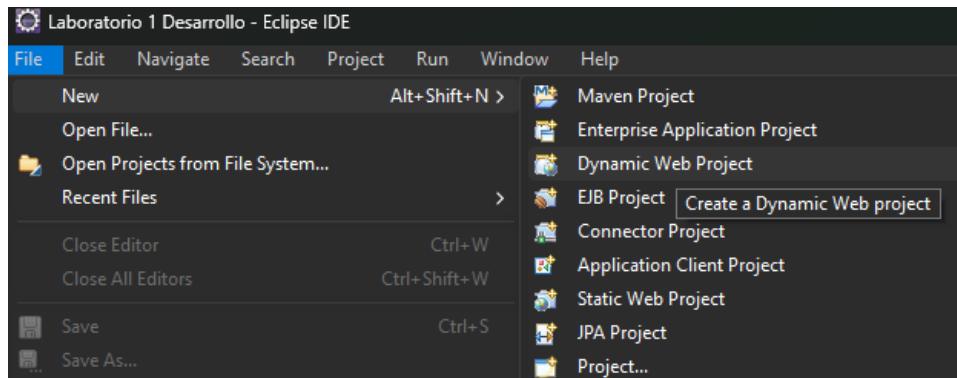
Configuración y creación del proyecto

Eclipse IDE con Tomcat y Java 8 (1/5)

File → New → Dynamic Web Project

Se debe dar un nombre al proyecto
(WebApp)

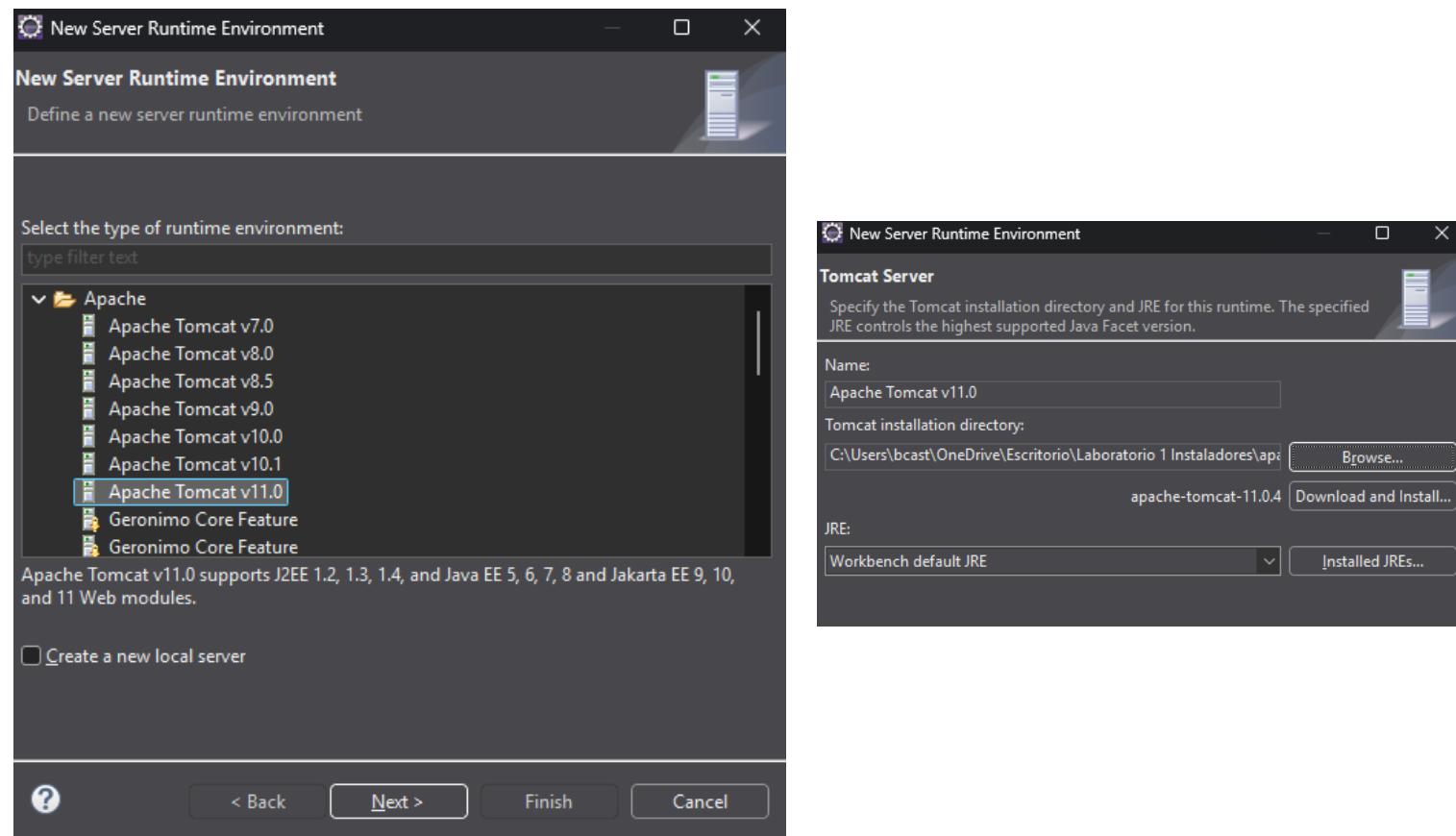
Se da click a New Runtime



Configuración y creación del proyecto

Configurando Eclipse IDE con Tomcat y Java 8 (2/5)

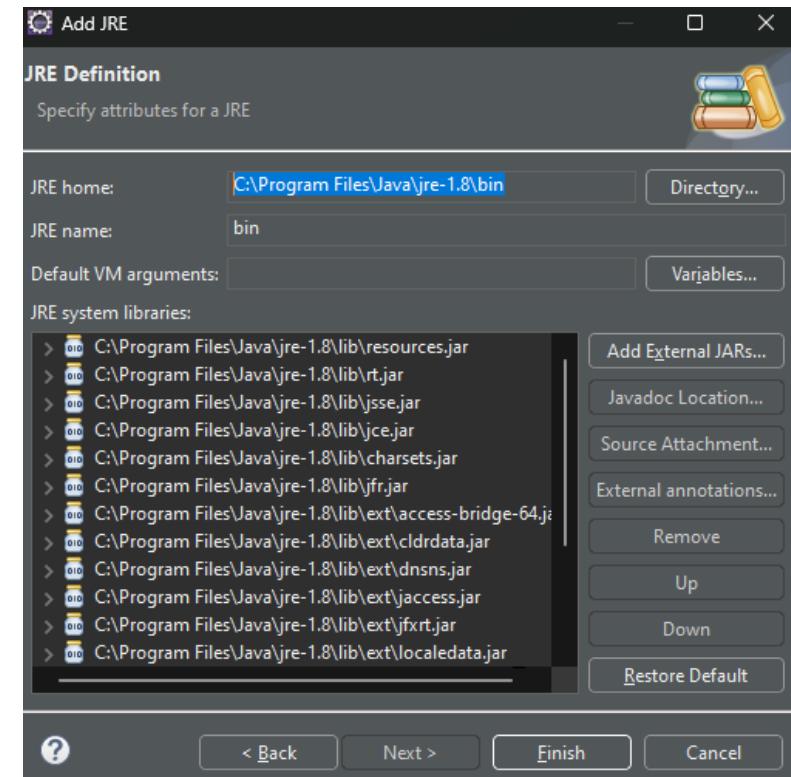
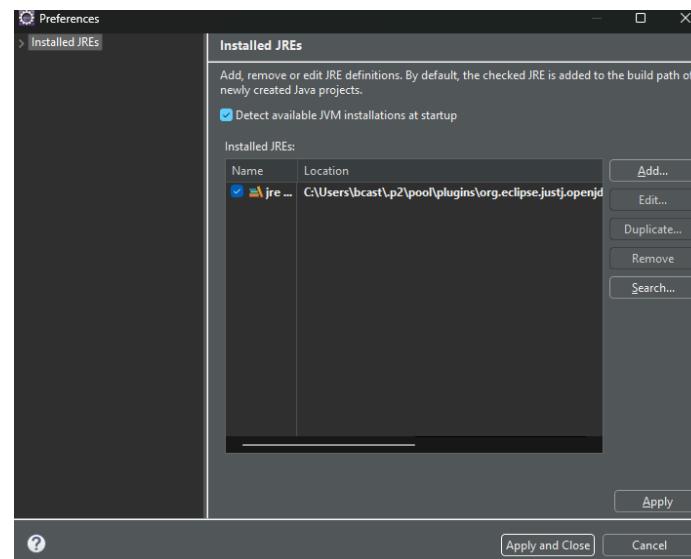
Para continuar debemos darle a new Runtime en Target Runtime y luego seleccionar Apache TomCat 11.0, le damos click en Next y luego seleccionaremos la carpeta donde descomprimimos el Tomcat



Configuración y creación del proyecto

Configurando Eclipse IDE con Tomcat y Java 8 (3/5)

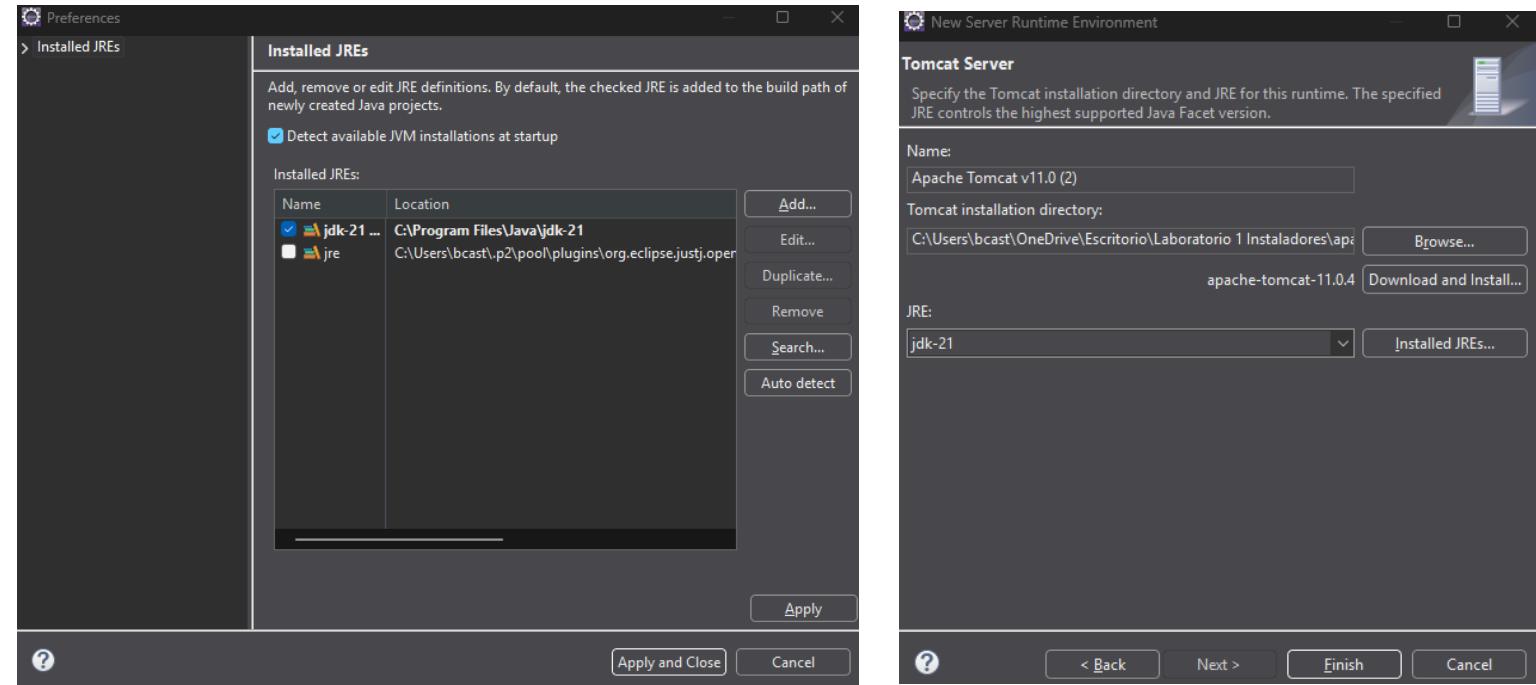
Luego debemos hacer click en Installed JREs y le damos en Add. Aquí haremos click en Standard VM y para JRE Home seleccionaremos “C:\Program Files\Java\jdk-21” y le damos a finish



Configuración y creación del proyecto

Configurando Eclipse IDE con Tomcat y Java 8 (4/5)

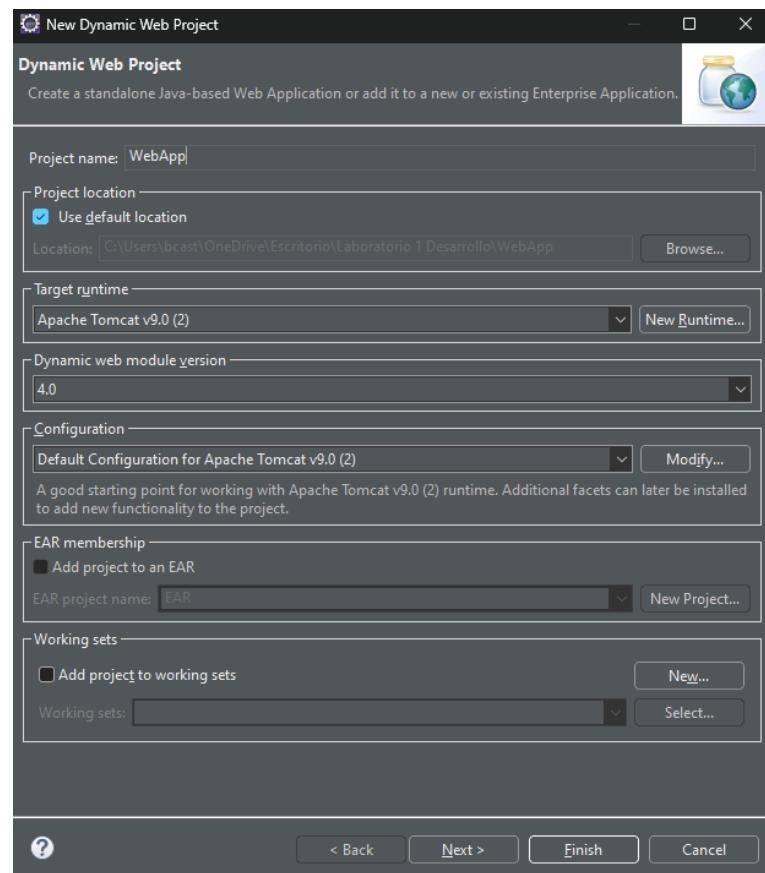
Una vez tengamos el nuevo bin, lo seleccionamos y le damos click en Apply and Close, Finalmente en JRE , Seleccionamos bin y le damos en Finish.



Configuración y creación del proyecto

Configurando Eclipse IDE con Tomcat y Java 8 (5/5)

Para finalizar con la configuración le damos click en Finish



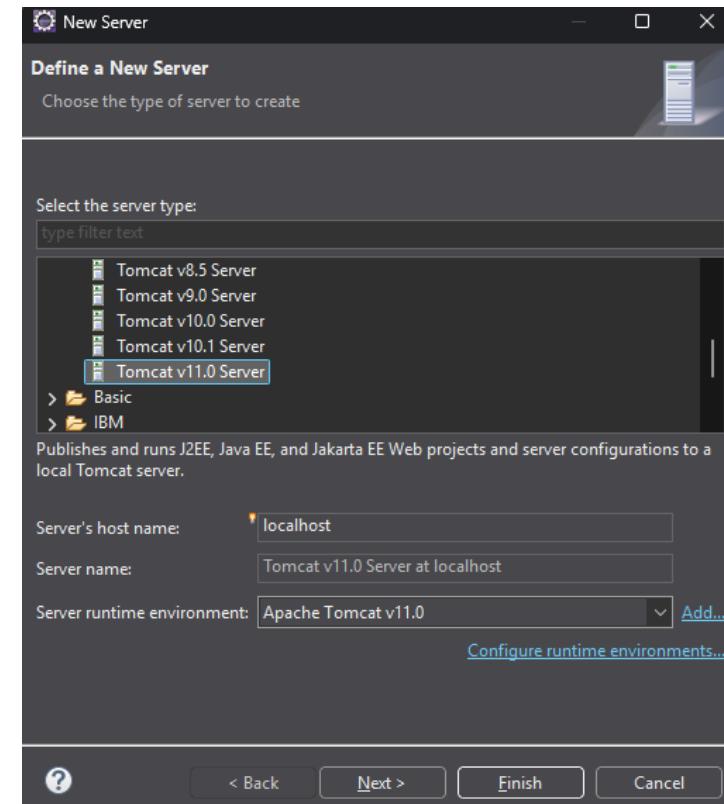
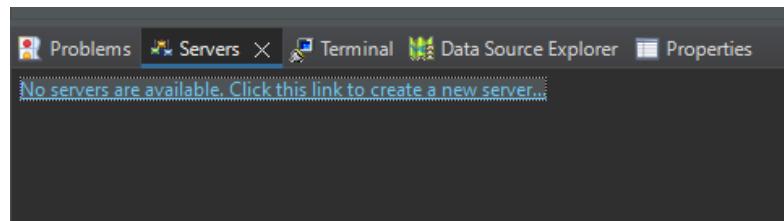
Configuración y creación del proyecto

Configuración del servidor (1/2)

En caso no tengan la vista servidores, debemos ir a window, show view, servers

Una vez hecho esto, la pestaña servidores será visible en la parte inferior del programa donde debemos agregar uno nuevo

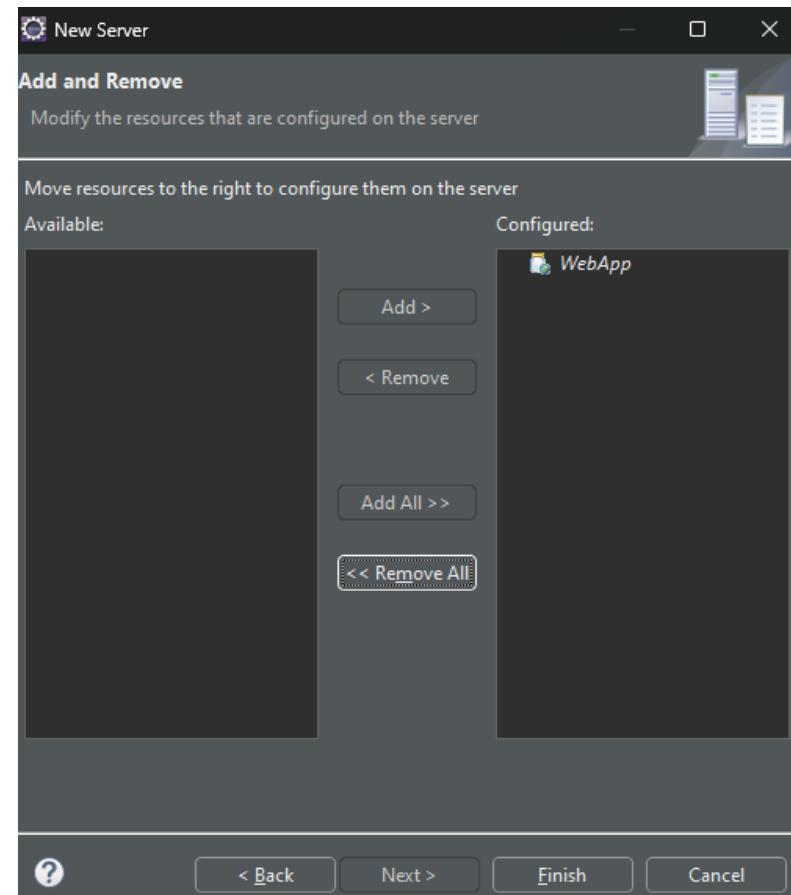
Aquí se deberá crear un nuevo servidor usando Tomcat V11.0 para luego dar click en Next



Configuración y creación del proyecto

Configuración del servidor (2/2)

Aquí finalmente se debe mover el proyecto a Configured y le daremos click en Finish



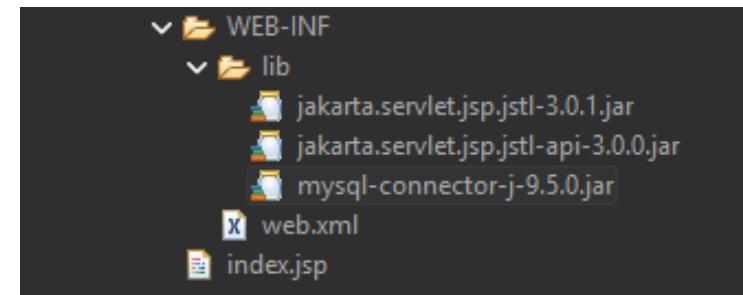
Configuración y creación del proyecto

Configuración JSTL

En el folder del proyecto vamos a ubicar la carpeta SRC

Aquí vamos a expandir de la siguiente forma : “src/main/webapp/WEB-INF/lib”.

Aquí vamos a copiar y a pegar los archivos “jakarta.servlet.jsp.jstl-api-3.0.0.jar”, “jakarta.servlet.jsp.jstl-3.0.1.jar” y “mysql-connector-j-9.5.0.jar”



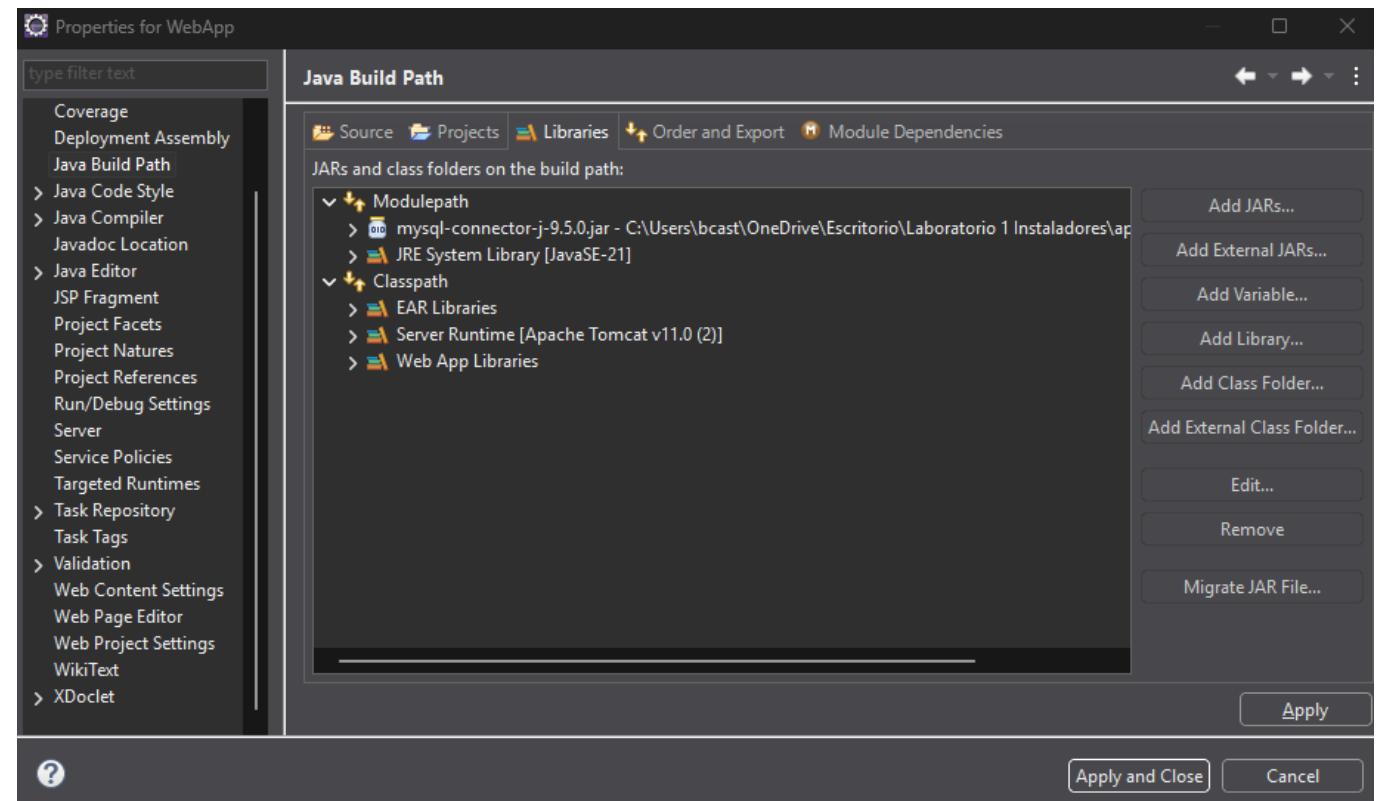
Configuración y creación del proyecto

Configuración de la base de datos con MySQL Connector

Para esto vamos darle click derecho al proyecto y darle a properties.

Aquí iremos a Java Build Path y seleccionaremos la pestaña libraries.

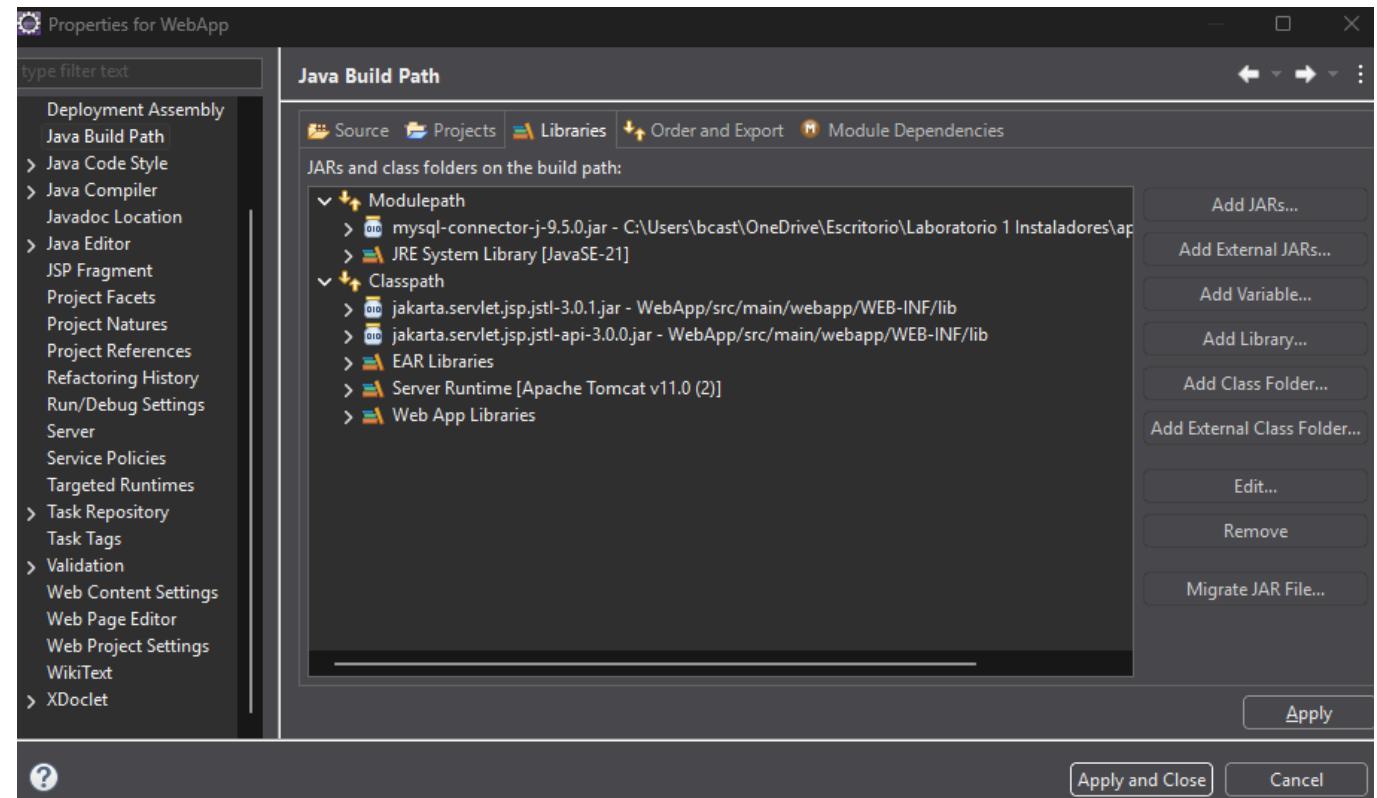
Finalmente le daremos click en Add External JARs y seleccionaremos el archivo previamente descargado.



Configuración y creación del proyecto

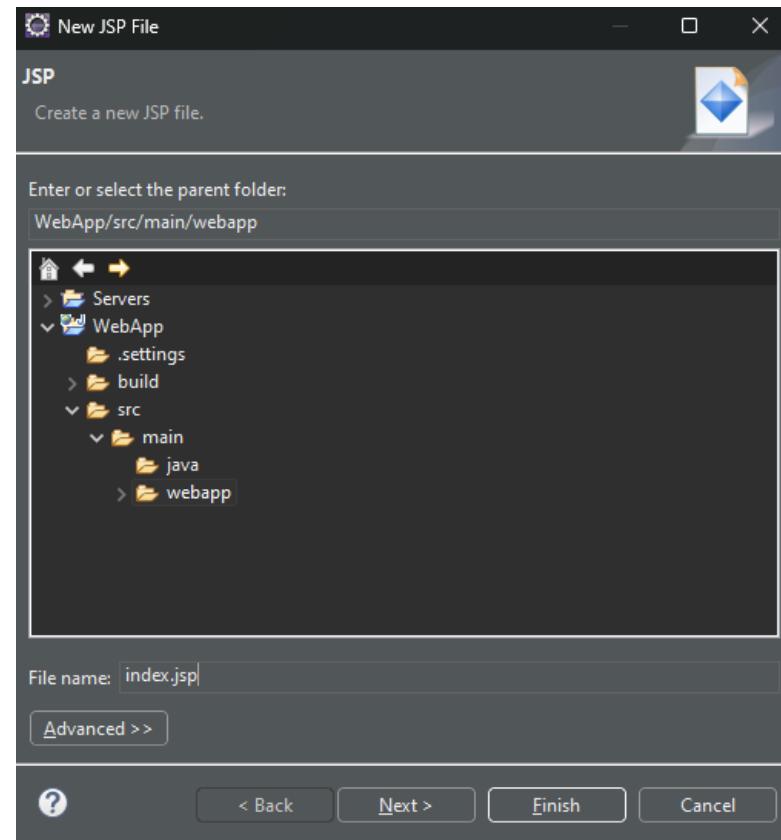
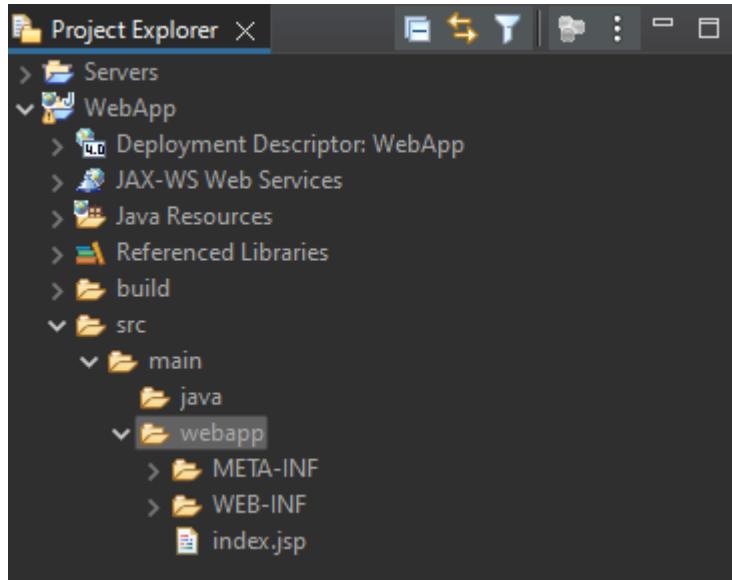
Adicionar JSTL

Adicional a esto en el apartado de Classpath, también adicionaremos los archivos API y IMPL , para esto daremos click en Add JAR y buscaremos la ruta de los archivos



Test de nuestro proyecto (1/2)

Para poder hacer el test , lo que vamos a hacer es crear un nuevo archivo JSP en src/main/webapp/ y lo nombraremos como index.jsp



Test de nuestro proyecto (2/2)

Esto nos abrirá un nuevo documento donde podremos hacer los cambios respectivos para que sea visible en nuestro test, en este caso:

```
<h1>Esta es nuestra página principal</h1>
```

Además de un poco de color con :

```
<style>
  h1 { color: #00A8F7 ; }
</style>
```

Y reemplazaremos el título por “Primera Prueba”

Finalmente grabaremos con Ctrl + S

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<style>
h1 { color: #00A8F7 ; }
</style>
<title>Primera Prueba</title>
</head>
<body>
<h1> Esta es nuestra pagina principal</h1>
</body>
</html>
```

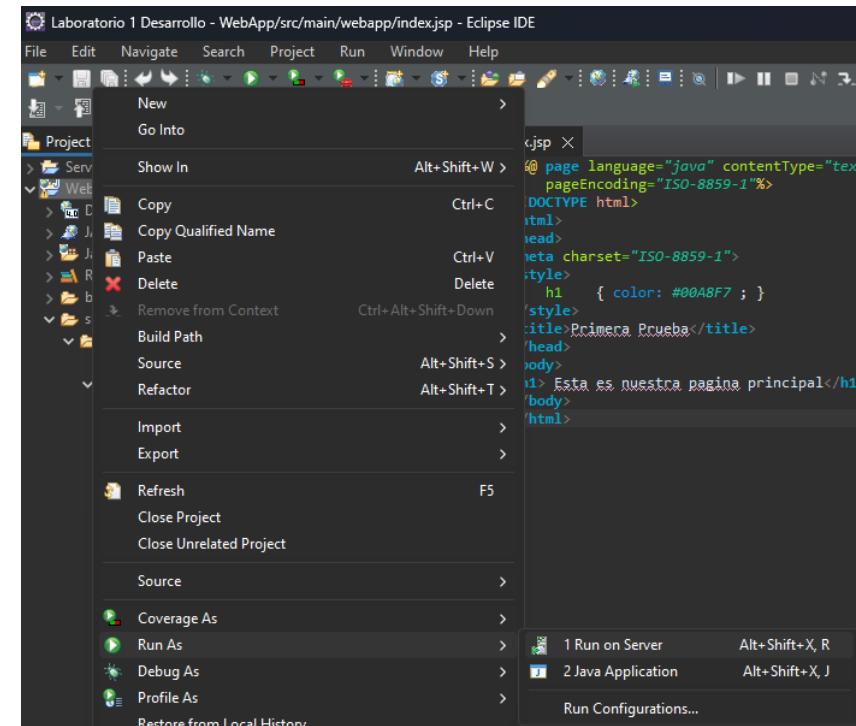
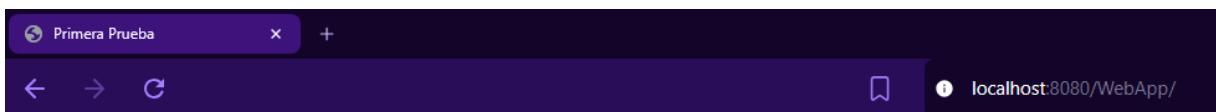
Visualizando nuestro primer test

Para poder visualizar nuestro test le daremos click derecho a nuestro aplicativo y le daremos a Run As , Run on Server

Aquí seleccionaremos nuestro servidor Tomcat 11.0 , Next y Finish

En caso de que nos salga una alerta de firewall , debemos dar click en permitir acceso.

Para poder acceder a nuestro test deberemos abrir una página en nuestro navegador y dirigirnos a
“<http://localhost:8080/WebApp/>



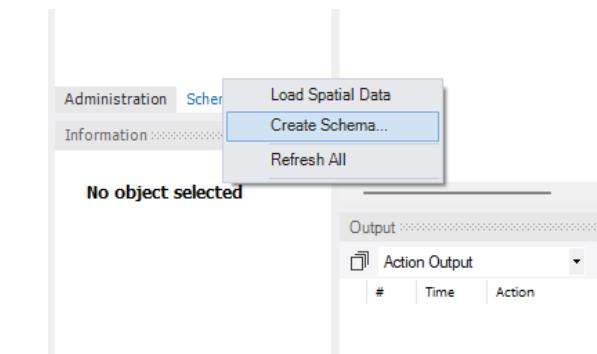
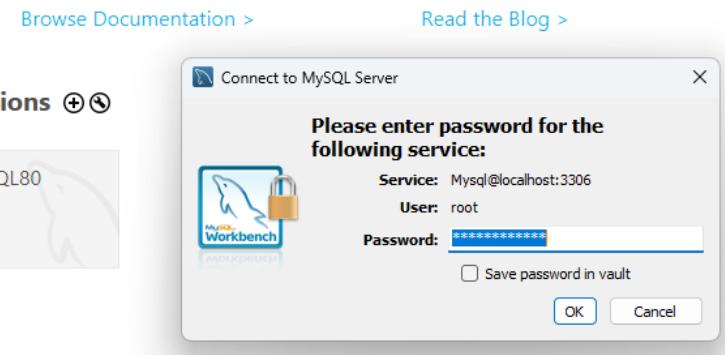
Esta es nuestra pagina principal

Implementación del proyecto

Inicialización de la base de datos (1/3)

Debemos abrir MySQL WorkBench
Luego hacemos click en Local Instance MySQL80 y ingresamos nuestras credenciales

Luego en el apartado Schemas le damos click derecho y seleccionamos create schema

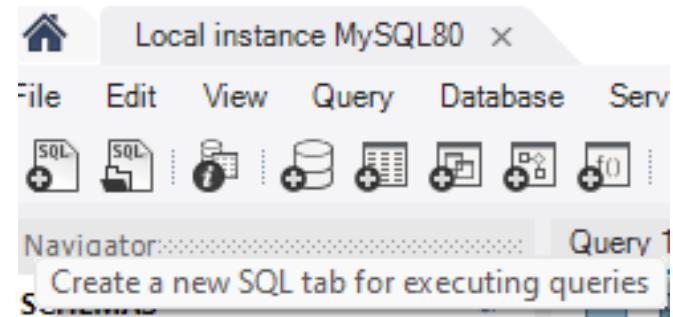
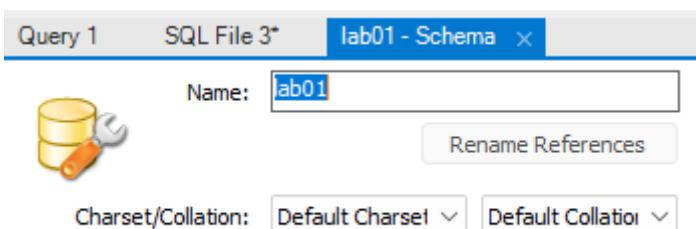


Implementación del proyecto

Inicialización de la base de datos (2/3)

Aquí generamos un nuevo schema al cual llamaremos lab01

Una vez se ha creado el schema generaremos un nuevo query con el botón +SQL y añadiremos el siguiente query



```
CREATE TABLE lab01.product(
    id INT NOT NULL AUTO_INCREMENT,
    name VARCHAR(45) NOT NULL,
    stock INT NOT NULL,
    PRIMARY KEY(id)
);
```

```
INSERT INTO lab01.product(name, stock)
VALUES ('Laptop', 1);
```

Implementación del proyecto

Inicialización de la base de datos (3/3)

Para poder correr cualquier query , es necesario darle click al rayito

Este query nos generará una nueva tabla llamada productos con un nuevo row

Para poder ver los valores de nuestra nueva tabla solo debemos hacer:

Select * from lab01.product

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' tree view is open, showing the 'lab01' schema with its tables, columns, indexes, foreign keys, triggers, views, stored procedures, and functions. The 'product' table is selected. In the center, the SQL editor contains the following code:

```
1 • CREATE TABLE lab01.product(
2     id INT NOT NULL AUTO_INCREMENT,
3     name VARCHAR(45) NOT NULL,
4     stock INT NOT NULL,
5     PRIMARY KEY(id)
6 );
7
8 • INSERT INTO lab01.product(name, stock)
9     VALUES ('Laptop', 1);
```

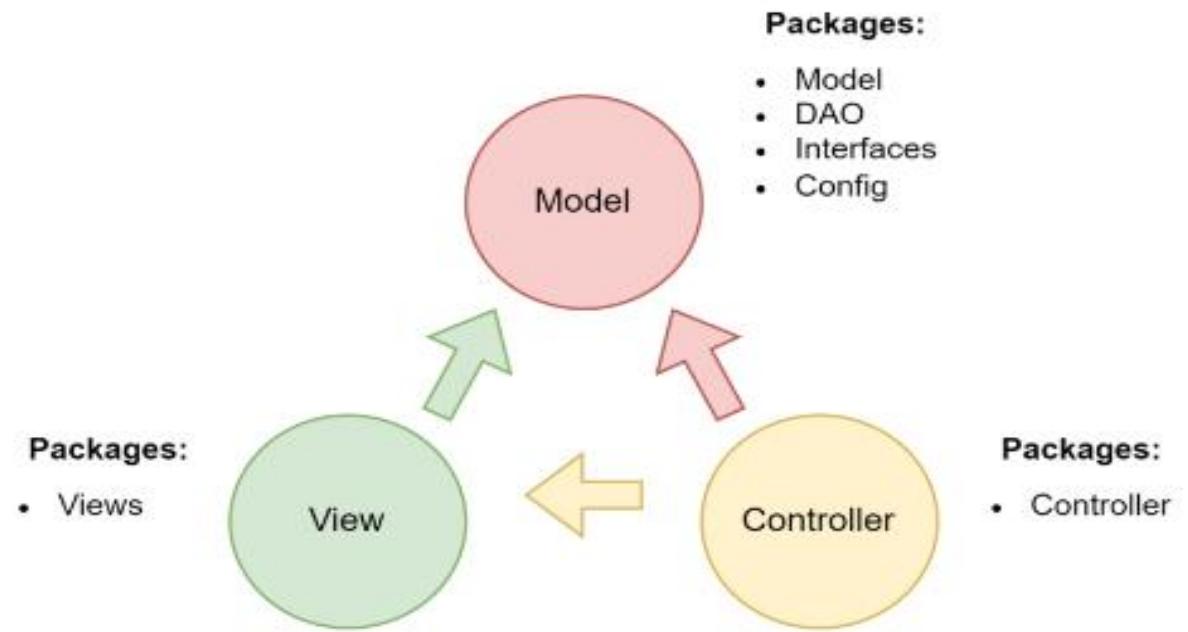
Below the SQL editor is the 'Result Grid' pane, which displays the data from the 'product' table:

	id	name	stock
▶	1	Laptop	1
*	NULL	NULL	NULL

Implementación del proyecto

Arquitectura en Capas (1/4)

Para seguir un patrón MVC(Modelo-Vista-Controlador), utilizaremos paquetes para aislar las capas de nuestra aplicación.



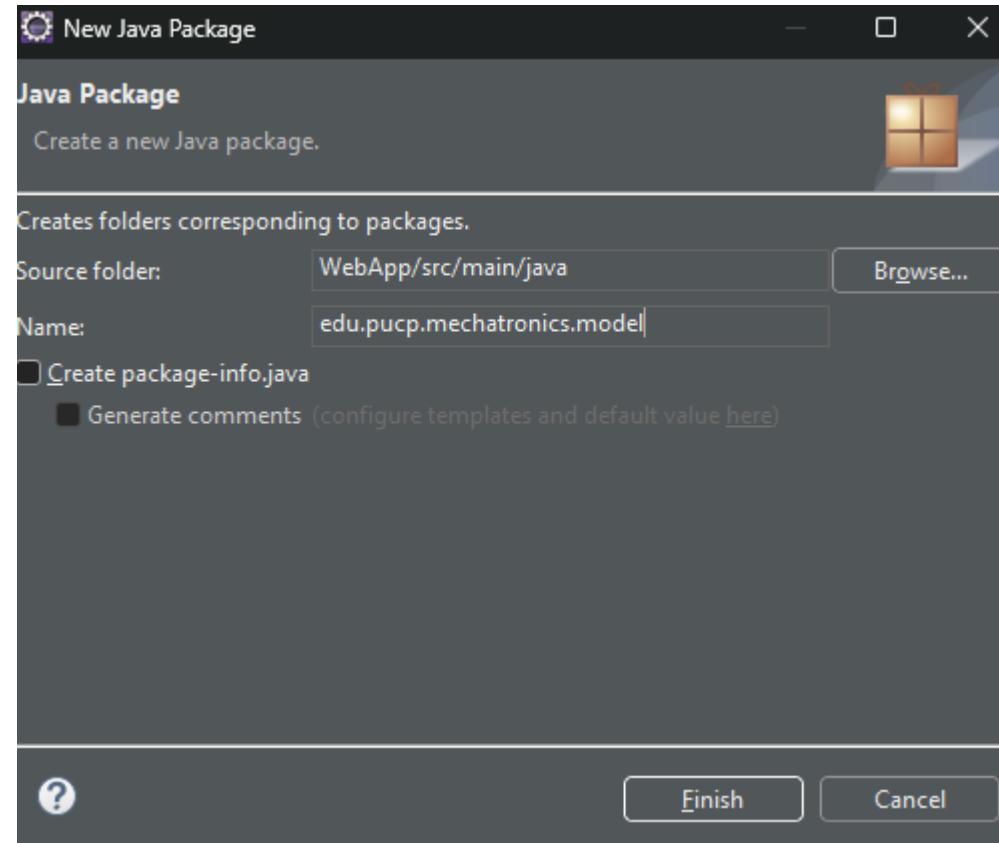
Implementación del proyecto

Arquitectura en Capas (2/4)

Para poder aplicar esto, en nuestro proyecto ya creado le daremos click derecho → new → paquete

El cual se llamará :

edu.pucp.mechatronics.model



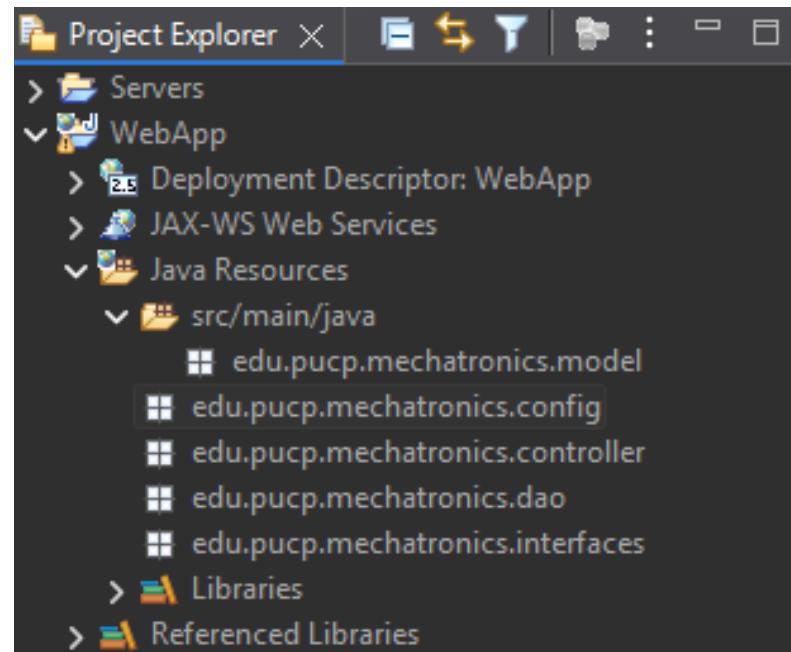
Implementación del proyecto

Arquitectura en Capas (3/4)

Una vez creado el nuevo paquete, crearemos otros nuevos paquetes

Los paquetes a crear serán:

edu.pucp.mechatronics.controller
edu.pucp.mechatronics.dao
edu.pucp.mechatronics.interfaces y
edu.pucp.mechatronics.config



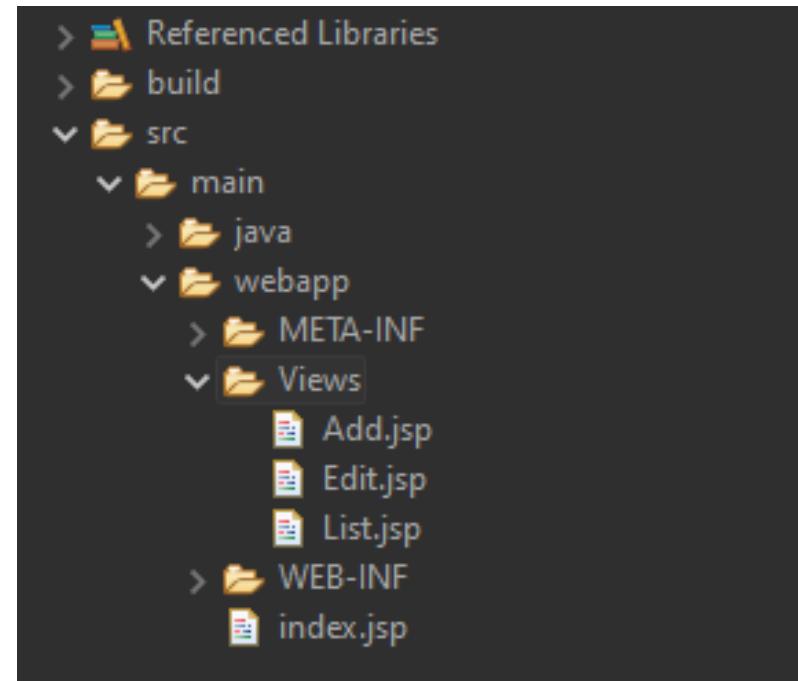
Implementación del proyecto

Arquitectura en Capas (4/4)

Para finalizar haremos nuestra Capa Vista la cual, estará compuesta de archivos .jsp

Por lo que dentro de nuestro WebApp Folder, crearemos un nuevo folder llamado views.

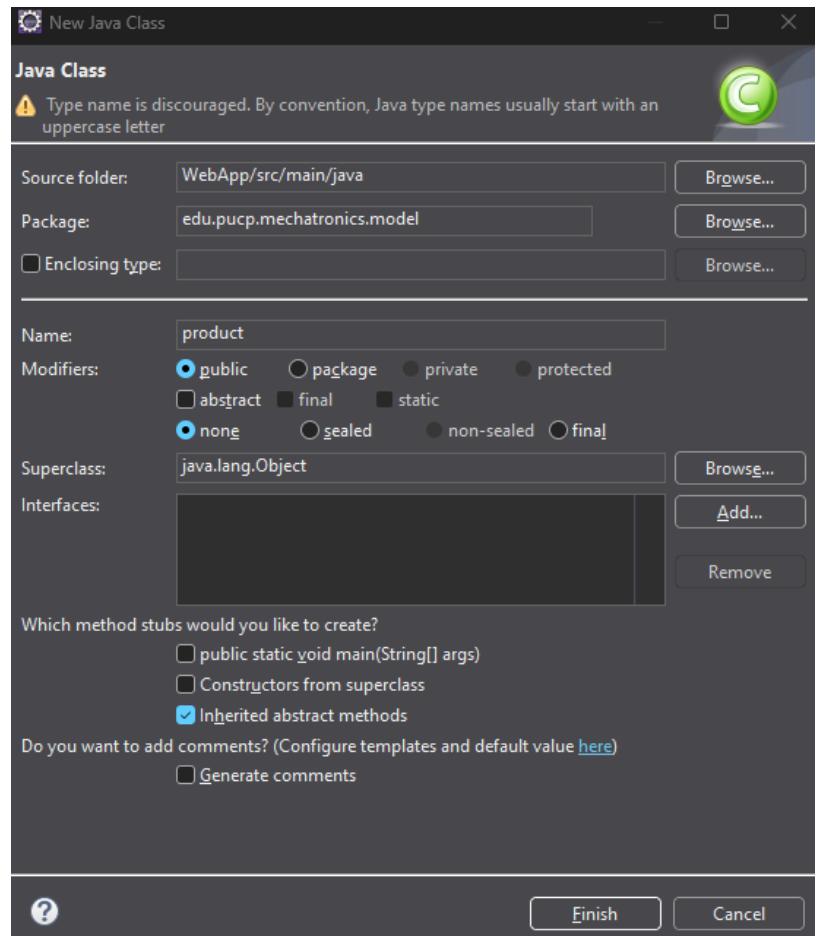
Dentro de este folder crearemos las vistas, add, edit y list



Implementación del proyecto

Paquete Model (1/4)

Aquí definiremos nuestra clase Producto, para eso crearemos una nueva clase dentro de nuestro paquete modelo , el cual se llamará product



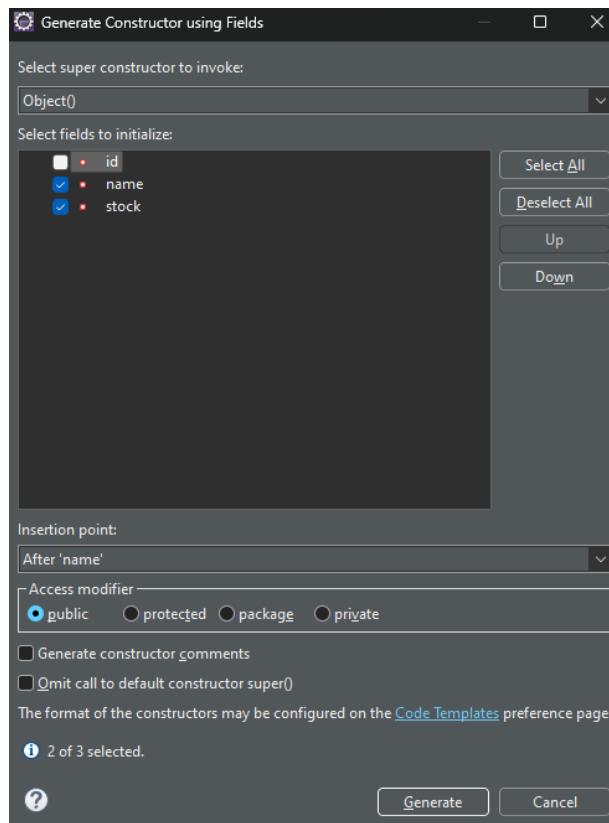
Implementación del proyecto

Paquete Model (2/4)

Una vez creado definiremos nuestra clase , la cual contendrá : id, name y stock

Una vez tengamos esto, podemos añadir los constructores dandole click derecho a nuestra clase → Source → Generate Constructor using Fields

```
1 package edu.pucp.mechatronics.model;
2
3 public class Product {
4     private int id;
5     private String name;
6     private int stock;
7 }
```

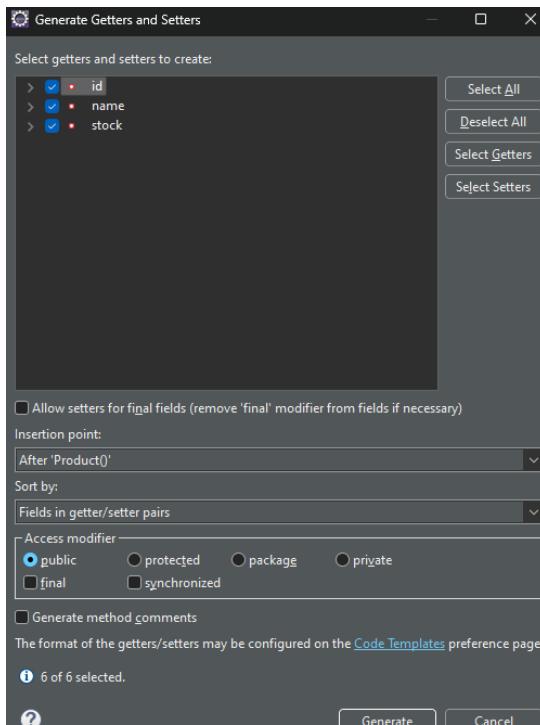


```
1 package edu.pucp.mechatronics.model;
2
3 public class Product {
4     private int id;
5     private String name;
6     private int stock;
7
8     public Product(String name, int stock) {
9         super();
10        this.name = name;
11        this.stock = stock;
12    }
13
14    public Product() {
15        super();
16    }
17
18 }
19
20 }
```

Implementación del proyecto

Paquete Model (4/4)

Finalmente Implementaremos los getters and setters de nuestra clase



```
1 package edu.pucp.mechatronics.model;
2
3 public class Product {
4     private int id;
5     private String name;
6     private int stock;
7
8     public Product(String name, int stock) {
9         super();
10        this.name = name;
11        this.stock = stock;
12    }
13
14    public Product() {
15        super();
16    }
17
18    public int getId() {
19        return id;
20    }
21
22    public void setId(int id) {
23        this.id = id;
24    }
25
26    public String getName() {
27        return name;
28    }
29
30    public void setName(String name) {
31        this.name = name;
32    }
33
34    public int getStock() {
35        return stock;
36    }
37
38    public void setStock(int stock) {
39        this.stock = stock;
40    }
41
42
43 }
```

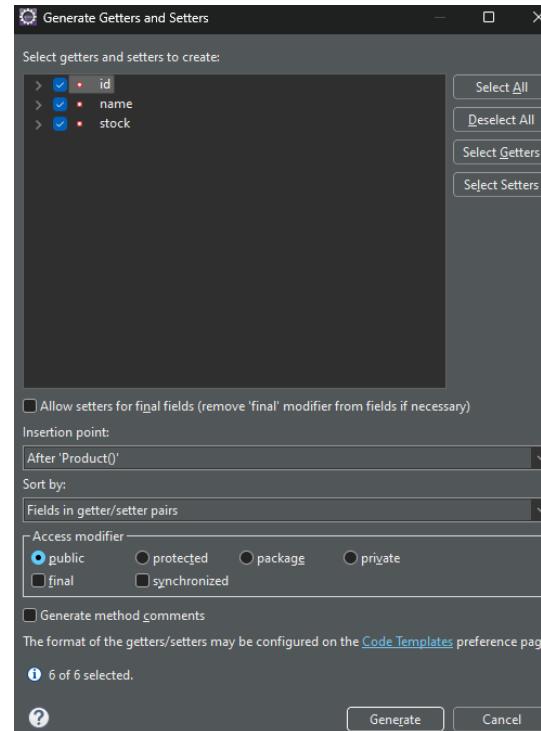
```
package edu.pucp.mechatronics.config;
```

Implementación del proyecto

Paquete Config

Este nos permite definir las conexiones con la base de datos así como también su interacción con este.

Para ello generamos una clase llamada DBConnection y insertamos el siguiente código



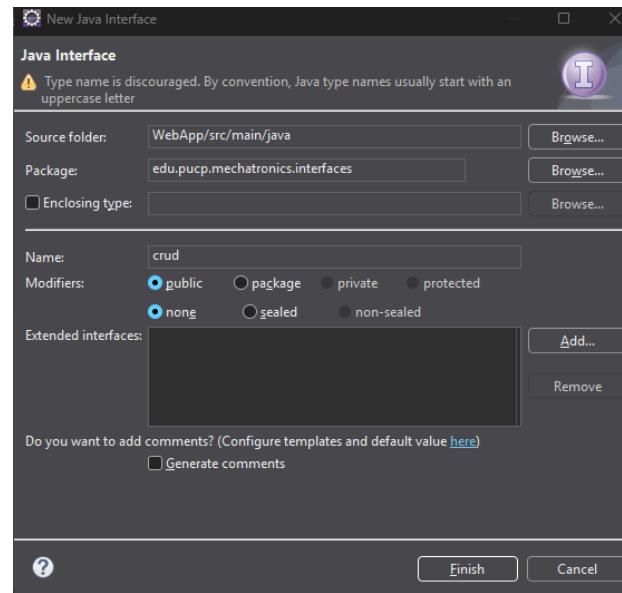
```
import java.sql.Connection;  
import java.sql.DriverManager;
```

```
public class dbconnection {  
    Connection con;  
    public dbconnection() {  
        try {  
            Class.forName("com.mysql.cj.jdbc.Driver");  
            //register driver  
            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/lab01","root","pucp2026");  
        }  
        catch (Exception e) {  
            System.err.println("Error: "+e);  
        }  
    }  
    public Connection getConnection() {  
        return con;  
    }  
}
```

Implementación del proyecto

Paquete interface

Este paquete define las actividades CRUD, las cuales son funciones que deben ser implementadas para realizar procedimientos como buscar, guardar, hacer update o eliminar. Para ello generaremos una Interfaz llamada crud e insertamos el siguiente código :



```
package edu.pucp.mechatronics.interfaces;
```

```
import java.util.List;
```

```
public interface CRUD<E> {  
    public List<E> findAll();  
    public E find(int id);  
    public boolean save(E p);  
    public boolean update(E p);  
    public boolean delete(int id);  
}
```

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;
```

```
import edu.pucp.mechatronics.config.dbconnection;
import edu.pucp.mechatronics.interfaces.crud;
import edu.pucp.mechatronics.model.product;
```

```
public class productdao implements crud<product>{
    DBConnection connDB = new DBConnection();
    Product currentProduct = new Product(); //product for details, updates or deletion
    @Override
    public List<product> findAll() {
        List<product> products = new ArrayList<>();
        String sql = "select * from product;" ;
        try {
            Connection con = connDB.getConnection();
            PreparedStatement statement = con.prepareStatement(sql);
            ResultSet rs = statement.executeQuery();
            while(rs.next()) {
                Product prod = new Product();
                prod.setId(rs.getInt("id"));
                prod.setName(rs.getString("name"));
                prod.setStock(rs.getInt("stock"));
                products.add(prod);
            }
        } catch (Exception e) {
            System.err.println("Error: "+e.getMessage());
        }
        return products;
    }
}
```

Implementación del proyecto

Paquete DAO (1/2)

Este paquete define como los objetos son leídos y escritos dentro de la base de datos.

Para esto generamos una clase llamada productdao la cual implementara la interfaz CRUD y ingresamos el siguiente código :

Implementación del proyecto

Paquete DAO (2/2)

```
@Override  
public Product find(int id) {  
    String sql = "select * from product where id=?;" ;  
    try {  
        Connection con = connDB.getConnection();  
        PreparedStatement statement = con.prepareStatement(sql);  
        statement.setInt(1, id);  
        ResultSet rs = statement.executeQuery();  
        while(rs.next()) {  
            currentProduct.setId(rs.getInt("id"));  
            currentProduct.setName(rs.getString("name"));  
            currentProduct.setStock(rs.getInt("stock"));  
        }  
    } catch (Exception e) {  
        System.err.println("Error: " + e.getMessage());  
    }  
    return currentProduct;  
}  
@Override  
public boolean save(Product p) {  
    String sql = "insert into product(name, stock) values(?,?)";  
    try {  
        Connection con = connDB.getConnection();  
        PreparedStatement statement = con.prepareStatement(sql);  
        statement.setString(1, p.getName());  
        statement.setInt(2, p.getStock());  
        statement.executeUpdate();  
    } catch (Exception e) {  
        System.out.println("Error: " + e.getMessage());  
    }  
    return false;  
}
```

```
@Override  
public boolean update(Product p) {  
    String sql = "update product set name=?,stock=? where id=?";  
    try {  
        Connection con = connDB.getConnection();  
        PreparedStatement statement = con.prepareStatement(sql);  
        statement.setString(1, p.getName());  
        statement.setInt(2, p.getStock());  
        statement.setInt(3, p.getId());  
        statement.executeUpdate();  
    } catch (Exception e) {  
        System.err.println("Error: " + e.getMessage());  
    }  
    return false;  
}  
@Override  
public boolean delete(int id) {  
    String sql = "delete from product where id=?";  
    try {  
        Connection con = connDB.getConnection();  
        PreparedStatement statement = con.prepareStatement(sql);  
        statement.setInt(1, id);  
        statement.executeUpdate();  
    } catch (Exception e) {  
        System.err.println("Error: " + e.getMessage());  
    }  
    return false;  
}
```

Implementación del proyecto

Paquete Controlador (1/2)

Este paquete define la clase controlador quien es el encargado de tomar acciones mandadas a traves de las vistas como transiciones, modificaciones , etc..

Para ello crearemos la clase controller y añadiremos el siguiente código :

```
package edu.pucp.mechatronics.controller;

import java.io.IOException;
import java.util.List;

import jakarta.servlet.RequestDispatcher;
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 edu.pucp.mechatronics.dao.ProductDAO;
import edu.pucp.mechatronics.model.Product;

@WebServlet("/Controller")
public class controller extends HttpServlet {
private static final long serialVersionUID = 1L;
String list = "views/list.jsp";
String edit = "views/edit.jsp";
String add = "views/add.jsp";
productdao dao = new productdao();
/**
 * @see HttpServlet#HttpServlet()
 */
public Controller() {
super();
// TODO Auto-generated constructor stub
}
```

Implementación del proyecto

Paquete Controlador (2/2)

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
String currentAcces = "";
String action = request.getParameter("action"); //parameter from "index.jsp"
if(action.equalsIgnoreCase("list")) {
currentAcces = list;
}
else if(action.equalsIgnoreCase("goAdd")){
currentAcces = add;
}
else if(action.equalsIgnoreCase("Add")){
//from add.jsp, submit button
String name = request.getParameter("txtName");
int stock = Integer.parseInt(request.getParameter("txtStock"));
product prod = new product(name, stock);
dao.save(prod);
currentAcces = list;
}
else if(action.equalsIgnoreCase("goEdit")){
request.setAttribute("idProd", request.getParameter("id"));
currentAcces = edit;
}
```

```
else if(action.equalsIgnoreCase("Update")){
int id = Integer.parseInt(request.getParameter("txtId"));
String name = request.getParameter("txtName");
int stock = Integer.parseInt(request.getParameter("txtStock"));
product prod = new product();
prod.setId(id);
prod.setName(name);
prod.setStock(stock);
dao.update(prod);
currentAcces = list;
}
else if (action.equalsIgnoreCase("Delete")) {
int id = Integer.parseInt(request.getParameter("id"));
dao.delete(id);
currentAcces= list;
}
List<product> products = dao.findAll();
request.setAttribute("products", products); //link object for jsp
RequestDispatcher view = request.getRequestDispatcher(currentAcces);
view.forward(request, response);
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse)
 */
protected void doPost(HttpServletRequest request
, HttpServletResponse response) throws ServletException, IOException {
// TODO Auto-generated method stub
doGet(request, response);
}
}
```

Implementación del proyecto

Vistas -- Add

Este apartado contiene todos los archivos JSP que permitirán ver y hacer update de los distintos objetos, este tipo de archivo permite que código java pueda ser insertado dentro de un código HTML , lo cual permite el uso de objetos variables y métodos.

```
<!DOCTYPE html>
<html lang="en">
<head><meta charset="UTF-8"><meta name="viewport"
content="width=device-width, initial-scale=1.0">
<title>Nuevo Producto</title>
<style>
body {font-family: Arial, sans-serif; margin: 0; padding: 0; background-
color: #f0f0f0;}
.container {max-width: 400px; margin: 50px auto; padding:
20px; background-color: #fff; border-radius: 8px; box-shadow: 0 0 10px
rgba(0, 0, 0, 0.1);}
h1 {text-align: center;}
form {text-align: center;}
```

```
input[type="text"], input[type="submit"] {
width: 100%; padding: 10px; margin: 5px 0;
box-sizing: border-box; border: 1px solid #ccc;
border-radius: 5px; transition: border-color 0.3s;}
input[type="text"]:focus {border-color: #007bff; outline: none;}
input[type="submit"] {
background-color: #007bff; color: #fff; cursor: pointer;}
input[type="submit"]:hover {
background-color: #0056b3;}
```

}

```
</style>
</head>
<body>
<div class="container">
<h1>Nuevo Producto</h1>
<form>
<label for="txtName">Producto:</label><br>
<input type="text" id="txtName" name="txtName"><br>
<label for="txtStock">Stock:</label><br>
<input type="text" id="txtStock" name="txtStock"><br>
<input type="submit" name="action" value="Add"><br>
</form>
</div>
</body>
</html>
```

Implementación del proyecto

Vistas – Edit

```
<%@page import="java.util.Enumeration"%>
<%@page import="edu.pucp.mechatronics.model.product"%>
<%@page import="edu.pucp.mechatronics.dao.productdao"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html><head><meta charset="UTF-8">
<title>Edit Product Information</title><style>
body {
    font-family: Arial, sans-serif; margin: 0; padding: 0;
    background-color: #f0f0f0;
}
.container {
    max-width: 400px; margin: 50px auto; padding: 20px;
    background-color: #fff; border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
h1 {text-align: center;}
form {text-align: center;}
input[type="text"], input[type="submit"] {
    width: 100%; padding: 10px; margin: 5px 0;
    box-sizing: border-box; border: 1px solid #ccc;
    border-radius: 5px; transition: border-color 0.3s;
}
```

```
input[type="text"]:focus {
    border-color: #007bff; outline: none;
}
input[type="submit"] {background-color: #007bff;
color: #fff; cursor: pointer;}
input[type="submit"]:hover {background-color: #0056b3;}
</style></head><body>
<div class="container">
<h1>Edit Product Information</h1>
<%>
Productdao dao = new productdao();
int id = Integer.parseInt((String)request.getAttribute("idProd"));
Product p = dao.find(id);
%>
<form action="Controller">
<label for="txtName">Product:</label><br>
<input type="text" id="txtName" name="txtName" value="<%= p.getName()%>"><br>
<label for="txtStock">Stock:</label><br>
<input type="text" id="txtStock" name="txtStock" value="<%= p.getStock()%>"><br>
<input type="hidden" name="txtId" value="<%= p.getId()%>">
<input type="submit" name="action" value="Update"><br>
</form>
</div>
</body>
</html>
```

Implementación del proyecto

Vistas – List

```
<%@ taglib prefix="c" uri="jakarta.tags.core" %>
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html><html><head><meta charset="UTF-8">
<title>Productos</title><style>
body {
    font-family: Arial, sans-serif; margin: 0; padding: 0;
    background-color: #f0f0f0;
}
.container {
    max-width: 800px; margin: 50px auto; padding: 20px;
    background-color: #fff; border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
h1 {text-align: center;}
table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 20px;
}
```

```
th, td {border: 1px solid #ccc; padding: 8px; text-align: left;}
th {background-color: #007bff; color: #fff;}
tr:nth-child(even) {background-color: #f2f2f2;}
tr:hover {background-color: #ddd;}
td a {margin-right: 5px; text-decoration: none;}
td a:hover {text-decoration: underline;}
.btn-container {display: flex; justify-content: space-between;
align-items: center;}
.back-btn, .create-btn {width: 150px; padding: 10px;
background-color: #007bff; color: #fff; text-align: center;
text-decoration: none; border-radius: 5px;}
.back-btn:hover, .create-btn:hover {
background-color: #0056b3;}
</style></head><body>


# Productos



Pagina principal
Crear un nuevo producto


</div><table><thead>
<tr><th>ID</th><th>Product</th><th>Stock</th><th>Actions</th></tr>
</thead><tbody><c:forEach var="element" items="${products}">
<tr><td>${element.id}</td><td>${element.name}</td><td>${element.stock}</td>
<td>
Editar
Eliminar
</td>
</tr>
</c:forEach>
</tbody></table>



1MTR15 – Temas de Ingeniería Mecatrónica D



 PUCP



62


```

Implementación del proyecto

Index

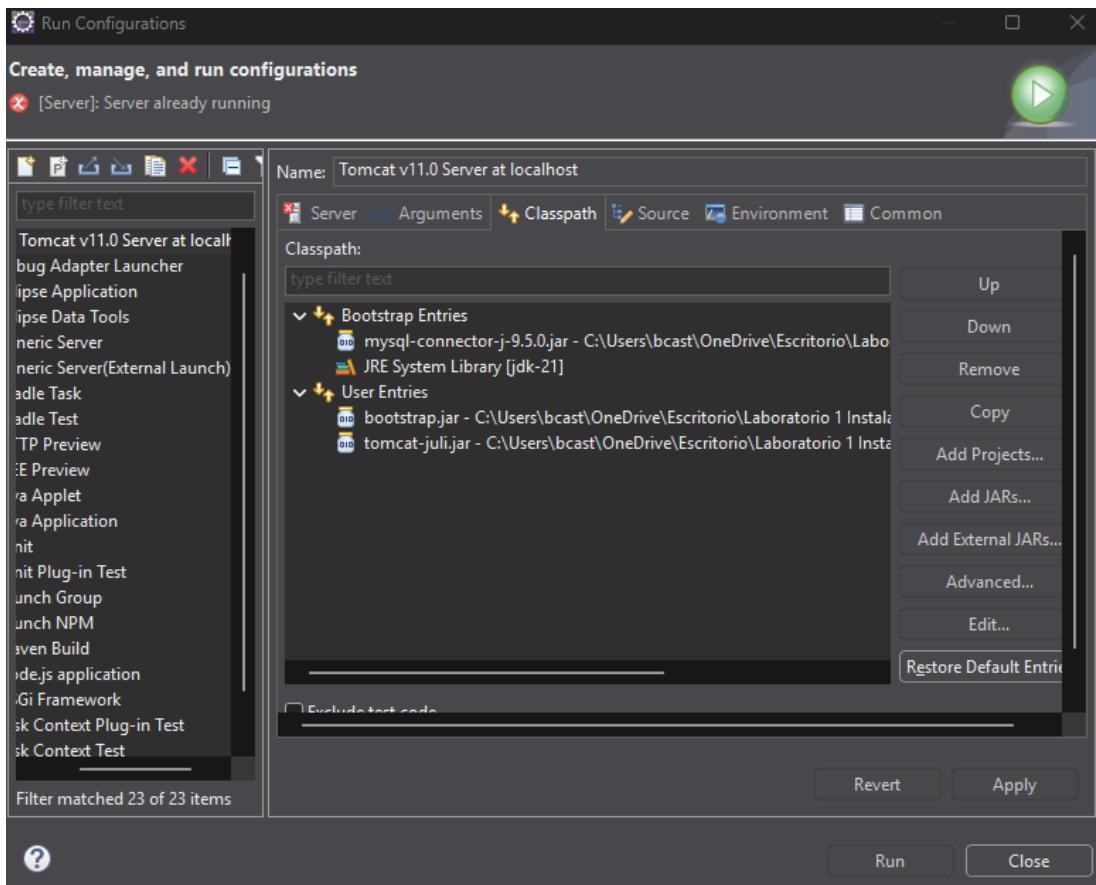
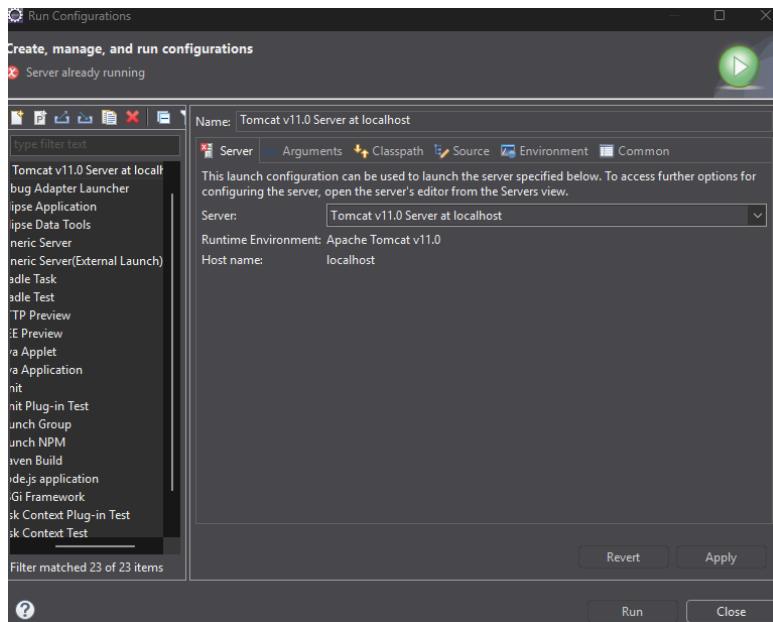
Así mismo , es necesario modificar el index , para que tengamos un punto de acceso en nuestra página

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html lang="es"><head><meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Prueba de Ingesta de Productos</title><style>
body {font-family: Arial, sans-serif; margin: 0; padding: 0;
background-color: #f8f9fa;}
.container {max-width: 800px; margin: 50px auto; padding: 20px;
background-color: #ffffff; border-radius: 10px; box-shadow: 0 0 20px
rgba(0, 0, 0, 0.1);}
h1 {color: #343a40; text-align: center;}
p {color: #6c757d; font-size: 18px; text-align: center;}
```

```
.button-container {text-align: center; margin-top: 30px;}
.button-container a {
display: inline-block; width: 200px; margin: 10px; padding: 15px;
background-color: #007bff; color: #ffffff;
font-size: 20px; text-align: center; text-decoration: none;
border-radius: 5px; transition: background-color 0.3s ease;}
.button-container a:hover {
background-color: #0056b3;
}
</style>
</head>
<body>
<div class="container">
<h1>Bienvenido a la Prueba de Ingesta de Productos</h1>
<p>Esta es la página principal de nuestro programa de ingestá de
productos. ¡Comencemos!</p>
<div class="button-container">
<a href="Controller?action=list">Mostrar productos</a>
</div>
</div>
</body>
</html>
```

Correr nuestro proyecto

Primero deberemos dar click derecho en nuestro proyecto → Run As → Run Configurations
Aquí seleccionaremos nuestro servidor TomCat y le daremos click on Classpath → Bootstrap Entries → Add External Jars
Aquí ingresamos nuestro connector Mysql en formato jar y le daremos en Apply



Correr nuestro proyecto

Para correr nuestro proyecto , es necesario que le demos click derecho → Run A → Run on Server

Aquí seleccionaremos nuestro servidor Tomcat 11.0 y le daremos en Finish

Nuestro Navegador nos dirigirá al url:
“[http://localhost:8080/WebApp/”](http://localhost:8080/WebApp/)

Bienvenido a la Prueba de Ingesta de Productos

Esta es la página principal de nuestro programa de ingestión de productos. ¡Comencemos!

[Mostrar productos](#)

Nuevo Producto

Producto:

Stock:

[Add](#)

Edit Product Information

Product:

Stock:

[Update](#)

Productos

[Pagina principal](#)

[Crear un nuevo producto](#)

ID	Product	Stock	Actions
1	Laptop	1	Editar Eliminar
2	Mouse	6	Editar Eliminar